



Ewers, Jan-Hendrik (2026) *Optimising wilderness missing person search through the application of machine learning for location probability prediction and mission planning*. PhD thesis.

<https://theses.gla.ac.uk/86000/>

Copyright and moral rights for this work are retained by the author

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

This work cannot be reproduced or quoted extensively from without first obtaining permission from the author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

Enlighten: Theses

<https://theses.gla.ac.uk>

research-enlighten@glasgow.ac.uk

Optimising Wilderness Missing Person Search Through the Application of Machine Learning for Location Probability Prediction and Mission Planning

Jan-Hendrik Ewers

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

School of Engineering
College of Science and Engineering
University of Glasgow



University
of Glasgow

22nd March 2026

Abstract

Wilderness search and rescue operations present a critical challenge where swift decision-making, and efficient resource allocation can significantly affect the likelihood of saving a lost person’s life. Traditional search planning methods rely heavily on manual coordination, and are constrained by limited human resources. However, in recent years, the expanding geospatial data, and advancing unmanned aerial vehicle capabilities have opened new possibilities for more adaptive, and data-driven search systems.

This work improves the search planning aspect of the wilderness search and rescue operations through the development of two components for an autonomous unmanned aerial vehicle. The first component is a probabilistic lost person behaviour model which leverages psychological profiles established from historical data to generate probability distribution maps through Monte-Carlo methods. The second component is a continuous-space reinforcement learning search planner which ingests the probabilistic location information over a given area to produce a search path subject to real-world operational constraints. By integrating behavioural modelling with heuristic-free path planning, this framework transforms raw search parameters into executable autonomous flight trajectories.

Results show that the probability distribution map generation algorithm closely matches historical datasets without requiring per-location retraining, which is the case with other methods from the literature. The search planning algorithm is then able to leverage the probability distribution map effectively whilst outperforming benchmark algorithms like the local hill climbing based algorithm LHC_GW_CONV by over 250%. Furthermore, by training the reinforcement learning policy in an environment with a mid-fidelity six degrees-of-freedom dynamic simulation controlled by a full navigation stack, it was shown that the resultant search paths were able to better leverage the search platform dynamics resulting in higher find rates than when a lower fidelity environment was used.

The proposed modular approach of generating a probability distribution map, which is then ingested by a search algorithm, has provided insights into the strengths and limitations of each part. Future work will focus on the combination of the two components into a full end-to-end algorithm capable of fully autonomous wilderness search and rescue using unmanned aerial vehicles.

This research provides a validated foundation for autonomous wilderness search and rescue systems, reducing operational burden and improving real-world rescue outcomes.

Contents

Abstract	i
Abbreviations And Acronyms	xiii
Acknowledgements	xv
1 Introduction	1
1.1 UAVs and Autonomy in WiSAR	1
1.2 The Wilderness Search Problem	3
1.3 Predictive Mapping	4
1.4 Search Path Planning	5
1.4.1 Knowledge Gaps	6
1.5 Research Aims and Objectives	7
1.6 Thesis Structure	7
1.7 Contributions To The Literature	8
1.7.1 Journal	8
1.7.2 Conference	8
1.7.3 Pre-print	9
1.7.4 Journal (Co-Author)	9
2 Literature Review	10
2.1 Probability Distribution Map Generation	10
2.1.1 Static	10
2.1.2 Agent-based	18
2.1.3 Discussion	23
2.2 Path Planning For Search And Rescue	26
2.2.1 Graph-based	26
2.2.2 Sampling-based	28
2.2.3 Coverage	29
2.2.4 Optimisation-based	30
2.2.5 Machine Learning	32
2.2.6 Discussion	34
2.3 Chapter Summary and Conclusions	37
3 Machine Learning Technical Background	38

3.1	Machine Learning Architectures	38
3.1.1	Fully Connected Network	38
3.1.2	Convolutional Neural Network	39
3.1.3	Recurrent Architectures	41
3.1.4	Attention	44
3.2	Auto-Encoder	47
3.2.1	Recurrent Auto-Encoder	47
3.3	Reinforcement Learning	49
3.3.1	Markov Decision Process	49
3.3.2	Proximal Policy Optimisation	50
3.3.3	Soft Actor-Critic	52
3.3.4	Chapter Summary and Conclusions	53
4	Behaviour-based Probability Distribution Map Generation	54
4.1	Section Overview	54
4.2	Introduction	55
4.3	Related Work	56
4.4	Methods	56
4.4.1	Path Generation	57
4.4.2	Sampling Of Paths	62
4.5	Design Of Experiment	64
4.6	Results and Discussion	66
4.6.1	Experimental Setup	67
4.6.2	Generated Paths	68
4.6.3	Mobility Distribution Sampling Effectiveness	70
4.6.4	Location Found Land Cover Categories	71
4.6.5	Computational Performance	72
4.7	Chapter Summary And Conclusion	73
4.8	Limitations and Future Work	73
5	Cubature Versus Sampling-based Integration Methods	75
5.1	Section Overview	75
5.2	Introduction	76
5.3	Method	78
5.3.1	Search Planning	78
5.3.2	Rover Model And Path Planning	79
5.3.3	Path Analysis	81
5.4	Results	84
5.5	Chapter Summary And Conclusion	87
6	Exploring The Efficacy Of Reinforcement Learning For Search Plan- ning	89

6.1	Section Overview	89
6.2	Introduction	89
6.3	Related Work	90
6.4	Method	92
6.4.1	Modelling	92
6.4.2	Reinforcement Learning	96
6.5	Results	99
6.5.1	Experimental setup	99
6.5.2	Percentage Of Detection	101
6.5.3	Distance To Find and Percentage Found	103
6.5.4	Area Efficiency	104
6.6	Conclusion	105
7	Recurrent Auto-Encoders For Enhancing Reinforcement Learning	
	Based Search Planning	106
7.1	Section Overview	106
7.2	Introduction	107
7.3	Related Work	108
7.4	Environment Modelling	109
7.4.1	Agent Dynamics	109
7.4.2	Probability Distribution Map	110
7.4.3	Rewards	111
7.4.4	Observation Processing	111
7.5	Model Architectures	112
7.6	Practical Implementation Details	114
7.6.1	Cubature	114
7.6.2	Further Architecture Details	115
7.7	Results	115
7.7.1	Experimental setup	115
7.7.2	Evaluation Metrics	116
7.7.3	Architecture	118
7.7.4	Ablation Study: Reinforcement Algorithm	120
7.7.5	Ablation Study: Path Feature Extraction	120
7.7.6	Benchmark	122
7.8	Discussion	122
7.9	Conclusion	123
8	Dynamically Aware Reinforcement Learning Search Planning	125
8.1	Section Overview	125
8.2	Introduction	126
8.3	Related Work	127
8.4	Environment Modelling	128

8.4.1	Six Degree-Of-Freedom UAV Model	128
8.4.2	Control	129
8.4.3	Trajectory Planning	130
8.4.4	Terrain Modelling	134
8.4.5	Component Integration	134
8.5	Auto-Encoders	136
8.5.1	Path History Recurrent Auto-Encoder	136
8.5.2	Probability Distribution Map And Terrain Convolution Neural Network Auto-Encoder	139
8.6	Search Planning	141
8.6.1	Architecture A: Recurrent Auto-Encoder	142
8.6.2	Architecture B: Convolution Neural Network	143
8.6.3	Action-space	145
8.6.4	Reward Function	146
8.6.5	Search Path Constraints	147
8.7	Results and Discussion	147
8.7.1	Parameter Configurations	147
8.7.2	Benchmark Algorithms	148
8.7.3	Dynamics and Static Analysis	150
8.7.4	Short-Flat Results	150
8.7.5	Long-Sloped Results	154
8.8	Section Summary and Conclusion	156
9	Conclusions, Limitations, and Future Work	158
9.1	Summary and Conclusions	158
9.2	Future Work and Limitations	160
9.2.1	Probability Distribution Map Generation	160
9.2.2	Search Planning	161
9.2.3	End-to-End System Integration	163
	References	164
A	Derivations	197
A.1	Derivative Of The Sigmoid Function	197
A.2	Derivative Of The Tanh Function	198
B	Values for 2D Cubature 2T Degree 13	199
C	LQR Tuning Using Genetic Algorithm	201
D	Additional Results From Ch. 8	204
D.1	Short, flat	204
D.2	Long, sloped	205

E	Further Machine Learning Technical Background	207
E.1	Backpropagation	207
E.1.1	Perturbation Learning	207
E.1.2	Hebbian Learning	208
E.1.3	The Backpropagation Method	208
E.1.4	The Importance Of Activation Functions	209
E.1.5	Backpropagation Example	209
E.2	Optimisation Algorithms	210
E.2.1	Stochastic Gradient Descent	210
E.2.2	Adam	212
E.2.3	Comparing Stochastic Gradient Descent and Adam	213

List of Tables

2.1	Horizontal distance in temperate mountainous terrain from PLS for two child-based categories	12
2.2	Dispersion angle (degrees) for a Hiker	13
2.3	Watershed number statistics	14
2.4	Excerpt of example percent of maximum travel time values for national land cover database cover classes	16
2.5	Best fit results for the 4D behaviour vector method	20
2.6	Best fit results for the 6D behaviour vector method	22
2.7	A quantitative comparison of various PDM generation methods.	24
2.8	Classification of search planning and its adjacent forms based on the presence of prior information and physical endurance limits.	34
2.9	A quantitative comparison of various path planning algorithms adjacent to search planning	35
4.1	Land cover category location found data for a <i>Hiker (solo)</i>	58
4.2	GIS maps used in this chapter.	58
4.3	Viewshed behaviour weights used for the building and tree behaviours	60
5.1	PDM sampling parameters	80
6.1	Definition of the five state observations.	96
6.2	SAC hyperparameters used for this chapter from empirical testing	101
6.3	Simulation parameters used for this chapter.	101
6.4	Mean Percentage of Detection	103
6.5	Mean distance to find	103

7.1	Simulation parameters used in this chapter.	110
7.2	Definition of the five state observations.	112
7.3	Definitions of architectures.	113
7.4	RAE parameters used in this chapter.	114
7.5	Reinforcement learning hyperparameters for the various architectures.	117
7.6	Aggregated architecture results over multiple metrics gathered at the end of a training run.	119
7.7	Aggregated algorithm (PPO, RPPO, SAC) results over multiple metrics gathered at the end of a training run.	120
7.8	Aggregated path feature extraction results over multiple metrics gathered at the end of a training run.	121
7.9	Mean probability efficiency from over 2000 generated paths by the respective architectures.	122
8.1	Definition of the six RAE architectures used within this chapter	137
8.2	RAE hyperparameters for the path history encoding	138
8.3	Number of parameters in the encoder and decoder.	138
8.4	Evaluation path generation method definitions	138
8.5	L2 norm construction loss for the RAE architectures trained using Adam	139
8.6	L2 norm construction loss for the RAE architectures trained using SGD	139
8.7	CAE hyperparameters for the PDM and terrain encoding	140
8.8	Number of parameters in the CAE encoder and decoder	141
8.9	L2 norm construction loss for the CAE architecture	141
8.10	Reward function parameters.	147
8.11	The permutation matrix of the parameter groups defined in Tbl. 8.12.	148
8.12	Parameters used to define the search path generation constraints and randomisation.	148
8.13	Simulation parameters	149
8.14	Reinforcement learning hyperparameters for CNN_SAC and HAE_SAC	149
8.15	$T_{\text{limit}} = 256 \text{ s}$, $U_{\text{limit}} = 20\,000$, $E_{\text{limit}} = 1 \times 10^5 \text{ J}$	151
8.16	$T_{\text{limit}} = 512 \text{ s}$, $U_{\text{limit}} = 40\,000$, $E_{\text{limit}} = 1 \times 10^5 \text{ J}$	153
B.1	Triangle cubature rule of degree 13 with 37 points	200

List of Figures

1.1	Total number of annual callouts and hours by SMR civilian rescue teams	2
-----	--	---

1.2	Top-level overview of the end-to-end data flow to get from the initial search parameters to the final search path.	7
2.1	The Euclidean Distance Circle is defined by four circles: the 25 %, 50 %, 75 % and 95 % distance rings. The radius of each ring is defined by the lost person (LP)'s statistical profile.	11
2.2	The Euclidean Distance Circle with inaccessible regions due to impassable features.	12
2.3	Distance Dispersion Circle example with an ID outside of the 95 % circle.	13
2.4	Example of a track offset measurement from the path going up <i>Goat Fell</i> on the Isle of Arran in Scotland.	14
2.5	A watershed is an area of land that channels precipitation to streams, and rivers into outflow points. The watershed boundary is defined by a continuous line of high-points.	14
2.6	Comparison of walking speed functions based on slope angle.	17
2.7	Definitions of the four behaviours from	20
2.8	Definitions of the two additional behaviours introduced in	21
2.9	Fitted functions to the Hiker profile data	22
2.10	SAR-A* architecture definition	27
2.11	Exploration-RRT sampling definitions	29
2.12	Example of a lawnmower pattern search pattern mission created in <i>QGroundControl</i>	30
2.13	The Hopf bifurcation spiral	30
3.1	An example FCN with four inputs, three outputs, and three hidden layers with five neurons each.	39
3.2	CNN simplified example	40
3.3	RNN unit internal structure	42
3.4	LSTM unit internal structure	43
3.5	GRU unit internal structure	44
3.6	CNN self-attention unit internal structure	47
3.7	Agent-environment interaction loop in reinforcement learning	49
3.8	Demonstration of the PPO reward clipping.	52
4.1	The process of extracting the weighted average angle from the nearest paths	62
4.2	Example path network	63
4.3	Discrete mobility time (hours) from two sources with a fitted log normal curve	63
4.4	Normalised heatmap derived from the raw SMR PLS data	64
4.5	The resultant heatmaps after applying conventional upsampling methods	65
4.6	Up-sampled heatmap using a GP	67

4.7	Up-sampled PLS locations	68
4.8	Subset of generated paths traversed by the agents for each behaviour .	69
4.9	Analysis of the path lengths by behaviour.	69
4.10	Paths traversed by agents using the four different behaviours	70
4.11	Distribution of the time points used	70
4.12	Results from J2, compared to J1 and the original data	71
4.13	Histogram of the real computation time per simulated behaviour. . . .	72
5.1	The 3D data maps used in this chapter centred around the Jezero Crater on Mars	80
5.2	Architectural diagram of the implementation of search planning using the multi-rover framework	81
5.3	Ray casting method for polygon intersection detection	82
5.4	Subdivision of the unit triangle into four smaller triangles.	84
5.5	Triangle cubature of rule degree 13 with 37 point.	85
5.6	Rover trajectories over the area given the mission planned by LHC_GW_CONV	85
5.7	The relative time as N increases, showing a trend towards an exponential increase as $N \rightarrow \infty$	87
5.8	The relative error as N increases, showing a trend towards 0% as $N \rightarrow \infty$.	87
5.9	Minimum N is required as a function of the sensor diameter.	88
6.1	An example MMBG PDM	93
6.2	Visualising a generalised subsection of search path H_2 with different θ .	95
6.3	Policy network using the 2D CNN for the frame-stacked path encoding	97
6.4	Ordinary least squares linear regression fitted to the $e_{p,D}$ results	98
6.5	Architectures of the three candidate path feature extractors	98
6.6	$e_{p,D}$ with the number of parameters within the policy network	99
6.7	p-value matrix from comparing the result distributions categorised by feature extraction method.	100
6.8	Probability efficiency analysis over all runs	102
6.9	Median distance to find	104
6.10	Area efficiency	105
7.1	Example PDM $p(x)$ with $N_{\text{gaussian}} = 3$ and constant covariance	111
7.2	RAE architecture using LSTM for encoding and decoding	112
7.3	The five proposed policy architectures for use with either PPO, RPPO, or SAC	113
7.4	Mean rollout episode reward over global step for all architectures high- lighting the training stability provided by the LSTMAE	119
7.5	Maximum probability efficiency during training for all architecture vari- ants with colour coded by RL algorithm.	121

8.1	Configuration of the UAV.	129
8.2	Look-ahead velocity relative definitions.	131
8.3	v_b over θ with different values for n	131
8.4	Demonstration of the normalised terrain randomisation	135
8.5	Hierarchical control architecture flowchart	135
8.6	Generic RAE architecture for both the vanilla and MHSA encoders, and the vanilla decoder	136
8.7	The architectures of the CNN, and CNN-Attn encoders with the shared CNN decoder	140
8.8	RAE and CAE based architecture layout diagram	142
8.9	Example rasterised path histories masks	144
8.10	CNN policy architecture layout	145
8.11	The magnitude of the action space over both axes	146
8.12	Final search path area for each algorithm using the <i>short-flat</i> parameter setup.	152
8.13	Final search path area efficiency for each algorithm using the <i>short-flat</i> parameter setup	152
8.14	Final search path probability efficiency for each algorithm using the <i>short-flat</i> parameter setup	154
8.15	Final search path accumulated probability per unit length for each al- gorithm using the <i>short-flat</i> parameter setup	154
8.16	Final search path area for each algorithm using the <i>long-sloped</i> para- meter setup	155
8.17	Final search path area efficiency for each algorithm using the <i>long-sloped</i> parameter setup	155
8.18	Final search path probability efficiency for each algorithm using the <i>long-sloped</i> parameter setup	156
8.19	Final search path accumulated probability per unit length for each al- gorithm using the <i>long-sloped</i> parameter setup	156
C.1	System velocity-altitude-heading control input response plotted for the three main state groupings — position, velocity, and euler angles — for the random (dotted), hand-tuned (dashed), and GA-tuned (solid) cases.	203
D.1	Final search path length for each algorithm using the <i>short-flat</i> para- meter setup.	204
D.2	Final area probability for each algorithm using the <i>short-flat</i> parameter setup.	204
D.3	Final search path accumulated probability per unit length for each al- gorithm using the <i>short-flat</i> parameter setup.	205
D.4	Final search path probability efficiency per unit length for each algorithm using the <i>short-flat</i> parameter setup.	205

D.5	Final search path accumulated probability per unit length for each algorithm using the <i>long-sloped</i> parameter setup.	205
D.6	Final area probability for each algorithm using the <i>long-sloped</i> parameter setup.	206
D.7	Final search path length for each algorithm using the <i>long-sloped</i> parameter setup.	206
D.8	Final search path probability efficiency per unit length for each algorithm using the <i>long-sloped</i> parameter setup.	206

List of Algorithms

5.1	Modified DCUTRI	86
6.1	Policy Network π_ϕ Input Process	96
8.1	Bisection method.	133
8.2	Bisection method with right expansion	133
E.1	Stochastic gradient descent.	211
E.2	Minibatch stochastic gradient descent.	211
E.3	Minibatch stochastic gradient descent with momentum.	212
E.4	Adaptive momentum estimation.	213

Abbreviations And Acronyms

Adam	Adaptive Moment Estimation
AE	Auto-Encoder
CAE	Convolution Auto-Encoder
CNN	Convolution Neural Network
CNN_SAC	Convolution Neural Network Soft Actor-Critic
CSA	Convolutional Self-Attention
DEM	Digital Elevation Map
FCN	Fully Connected Network
GA	Genetic Algorithm
GELU	Gaussian Error Linear Unit
GIS	Geographic Information System
GP	Gaussian Process
GRU	Gated Recurrent Unit
HAE_SAC	Hybrid Auto Encoder Soft Actor-Critic
ISRID	International Search And Rescue Incident Database
LHC	Local Hill Climb
LHC_GW_CONV	Local Hill Climbing Global Warming Convolution
LP	Lost Person
LQR	Linear Quadratic Regulator
LSTM	Long Short-Term Memory
LSTMAE	Long Short-Term Memory Auto-Encoder
Max_TSP	Maximum Positioning with the Travelling Salesman Problem

MDP	Markov Decision Process
MHSA	Multi-Headed Self-Attention
ML	Machine Learning
MMBG	Multi-Modal Bivariate Gaussian
MSAR	Marine Search And Rescue
PDM	Probability Distribution Map
PLS	Place Last Seen
PPO	Proximal Policy Optimisation
PSASU	Police Scotland Air Support Unit
RAE	Recurrent Auto-Encoder
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RPPO	Recurrent Proximal Policy Optimisation
RRT	Rapidly-exploring Random Tree
SAC	Soft Actor-Critic
SAC-FS-CNN	Soft Actor-Critic Frame Stacking Convolution Neural Network
SGD	Stochastic Gradient Descent
SGD-M	Stochastic Gradient Descent with Momentum
SMR	Scottish Mountain Rescue
TCNN	Transpose Convolution Neural Network
TTCSM	Travel Time Cost Surface Model
UAV	Unmanned Aerial Vehicle
USAR	Urban Search And Rescue
WiSAR	Wilderness Search And Rescue

Acknowledgements

My deepest appreciation goes to Dr. David Anderson for his unwavering support and invaluable guidance. His mentorship extends far beyond the past four years of this PhD, reaching back to a fourth-year design project in 2019 that remarkably evolved into a master's thesis and ultimately this doctoral work. I am also profoundly grateful to Dr. Douglas Thomson for his insightful advice and big-picture conversations, which consistently helped me stay grounded and calm during challenging times.

I gratefully acknowledge the Engineering and Physical Sciences Research Council for providing the essential funding that enabled this research. Furthermore, I extend my thanks to Dr. David Anderson, Dr. Euan McGookin, the University of Glasgow Autonomous Systems and Connectivity Research Division, and other contributors whose generous support allowed me to attend several incredible international conferences.

My sincere gratitude goes to everyone who made James Watt North a truly enjoyable place to work. A special thank you to Sarah Swinton for being an exceptional friend and colleague throughout my nine-year academic journey. I also appreciate Joe Gibbs and David Cormack for facilitating significant opportunities at Leonardo Electronics, and Stuart Neilson and Steven Preece from Police Scotland Air Support Unit for their invaluable support and advice.

To my parents, Christoph and Christine, thank you for instilling in me a profound sense of curiosity and for demonstrating, by example, the value of hard work. My uncle, Bernd, deserves special thanks for inspiring my path into aerospace engineering. And to Dom, Fiachra, Jack, my brothers Frederik and Jonathan, Duncan, Jo, Helena, Simon, and all my other friends not explicitly named, thank you for the craic along the way.

Finally, to my incredible wife, Laura, whom this endeavour would not have been possible without, thank you for your immense patience and understanding. We embarked on this PhD journey as a newly dating couple and conclude it as husband and wife; I am sure you are as excited as I am to see what comes next.

Chapter 1

Introduction

1.1 Unmanned Aerial Vehicles And Autonomy In Wilderness Search And Rescue

Wilderness search and rescue (WiSAR) operations in Scotland’s vast and often treacherous wilderness pose significant challenges for emergency responders. The Highlands and Islands, in particular, renowned for their stunning landscapes and remoteness, attract many keen mountaineers who are regularly caught out by the unpredictable conditions and unforgiving terrain [1]–[4].

The Cairngorm Plateau disaster of November 1971 [5], [6], which led to the death of six teenagers and is one of the most deadly mountaineering disasters in British history, is the epitome of how dangerous the Highlands can be. The group set off in the morning and reached the plateau when the weather quickly deteriorated and turned into blizzard conditions. A different group, who set off at the same time, struggled to get off the exposed plateau and were finally able to raise the alarm. Over 50 searchers were deployed to scour the landscape alongside a Royal Air Force helicopter. Unfortunately, by the time the group’s only survivor was spotted in the air, it was too late. An inquiry followed the disaster and led to reform, yet the Highlands remain dangerous and deadly today.

Even with access to more information about the weather and conditions, digital mapping, and better equipment, Scottish mountain rescue (SMR) deployment has drastically increased over recent years [7] with an almost doubling in callout counts from 2005 to 2024 (Fig. 1.1). Popular activities such as hill walking and fell running allow easy access to the Highlands with minimal equipment—however, this increase in accessibility results in more people being injured or lost.

The two leading organisations that handle missing persons search in Scotland are the SMR and Police Scotland air support unit (PSASU). Both provide different and vital services but suffer heavily from limited resources relative to the operational area. SMR relies on volunteers to carry out the WiSAR efforts and on helicopters from the Coast Guard and Royal Air Force to assist in search and medical evacuation procedures.

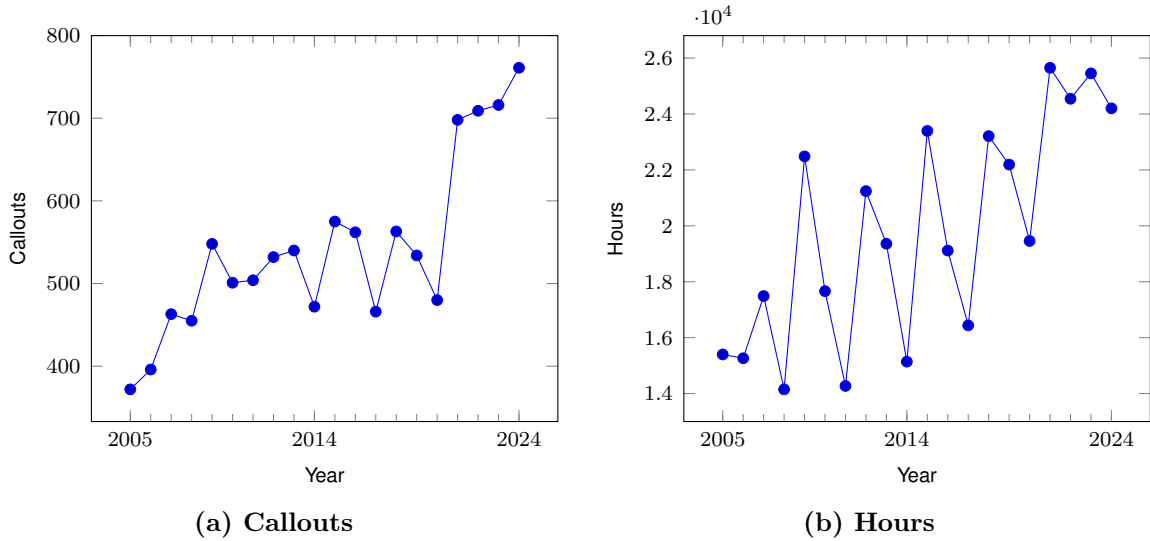


Figure 1.1: Total number of annual (a) callouts and (b) hours by SMR civilian rescue teams [7].

PSASU, on the other hand, is government-funded but only has a single operational Glasgow-based helicopter for the whole of Scotland. If, for example, an elderly dementia patient goes missing in Orkney (one of the most northern Islands in Scotland), it would take over 6 hours to arrive on the scene to assist in the search.

One approach both organisations are taking is using unmanned aerial vehicles (UAVs) during operations to augment current procedures [8], [9]. For SMR, this allows aerial search in scenarios where a helicopter is unavailable due to resource limitations or dangerous weather conditions. The UAV can be carried in a backpack into the operational area and deployed in tactically advantageous positions. Not only is the UAV used for horizontal search but it is also used for vertical search to locate climbers on cliff faces. Likewise, for PSASU, UAVs offer faster deployment in rural areas, which can give a head start to the search before the helicopter arrives. Police officers in key locations can be given training and equipment, which is comparatively cheap compared to the helicopters, to operate UAV hubs to reduce the time to deployment and thus the time to find.

Current UAV-based searches for the PSASU follow a three- or two-person pilot-observer model. In both configurations, the pilot manually manoeuvres the UAV into a position that they deem advantageous whilst someone maintains a visual line-of-sight to the UAV. In the three-person setup, a dedicated spotter analyses the video feed alongside the pilot, whereas the pilot does both the flying and spotting in the two-person scenario. Using autonomy, the three-person setup becomes redundant as the pilot can easily switch between flying and spotting.

Replacing the observer with vision models capable of detecting a LP is a keen topic of research [10]–[15]. However, the risk of a missed detection could lead to the loss of life. It, therefore, follows that the observer would be augmented and not replaced for the foreseeable future. The question is, if the spotter and observer are operational requirements, should the pilot be automated?

Koester *et al.* [16] found that the optimal search strategy for a foot-based searcher was the move-stop-look method. It was found that searchers who scanned their surroundings whilst navigating the terrain were likely to miss LP detections, whilst a searcher who paused moving periodically to scan their full surroundings performed better. This latter is the move-stop-look method and is visible in the trial flight test data presented in Ewers *et al.* [17].

The implication is that the cognitive load of simultaneously flying and searching leads to a loss in performance in both, which is an undesirable result. To mitigate this, the pilot should focus on a single task and offload the other to a dedicated system. In UAV search operations the two main tasks are observation (monitoring the sensor feed for detections) and navigation (optimising the flight path based on probability of area and platform constraints). Both can be automated: computer vision models can handle observation, while search planning algorithms can manage navigation. This thesis will focus on the navigation aspect.

1.2 The Wilderness Search Problem

Search and rescue can be applied to many domains, with the most prevalent being in the wilderness, maritime settings, and post-disaster urban areas. While the goal of locating a missing person is the same, the methods used and the problems faced can differ widely. Assumptions made and available equipment also vary greatly, which further widens the gap between each domain's search and rescue.

Marine search and rescue (MSAR) is unique because the search area changes dramatically over time primarily due to the environment and not the LP. Tidal and weather forecasting plays the predominant role in MSAR planning as the LP or naval vessel is typically unable to exert any control over its movement. The LP cannot swim for kilometres to self-rescue, and a stranded vessel will typically be without power. Therefore, the notion of a behavioural profile does not exist in MSAR, and the emphasis is placed on environmental models. For example, Wu *et al.* [18] and Ai *et al.* [19] apply reinforcement learning (RL) to the person-in-water scenario and only use physical environment properties (such as obstacles, currents, wind, and waves).

Urban search and rescue (USAR) is similarly primarily concerned with the present physical environment. However, it differs from MSAR in that human factors can be considered when estimating the locations of people prior to a disaster like an earthquake or hurricane. For example, a collapsed warehouse might have a lower priority than a populated apartment complex after an earthquake at night due to the likelihood of people being home at this time. Post-disaster, however, the LP is typically isolated and unable to move during the search efforts. USAR is also concerned with extremely unpredictable environments like collapsed buildings where methods like simultaneous localisation and mapping (also commonly known as SLAM) and frontier exploration path planning algorithms are important. Talha *et al.* [20] and Zuluaga *et al.* [21] are

examples where an internalised representation of the uncertain environment is built during the search phase without prior domain knowledge.

In WiSAR, unlike MSAR and USAR, the LP greatly influences their location and how they navigate the terrain. Whilst the LP is practically solely influenced by the elements in MSAR, isolated and unknowable decisions made by the LP drastically impact their location in WiSAR. This additional human contribution of uncertainty makes wilderness search planning particularly challenging.

The human element in WiSAR is often distilled into a probability distribution map (PDM), representing the probability of locating the LP at a given point. Subramanian *et al.* [22] treats the PDM as two dimensional heatmap whilst Macwan *et al.* [23] proposes the isoprobability PDM, which is a continuous formulation of the problem. No matter the form of the PDM, this layer of abstraction allows key decisions to be made based on models of the LP (PDM generation models are discussed in Sect. 2.1) alongside the physical landscape. In essence, the PDM becomes another map that the WiSAR searcher can use to inform their search and the level of accuracy dictates how closely it can be followed.

1.3 Before The Search Begins: Predictive Mapping

The search areas in WiSAR are vast by the problem definition itself. Koester [24] reports that 95% of hikers are found within 18.3km of the place last seen (PLS) and are mobile for 14h in temperate and mountainous terrain. Given the r^2 relation between distance covered and search area size, this would result in a naive worst-case estimation of an area over 1000 km² large, which must be fully searched. This poses the question for WiSAR teams: how can this be area be reduced to something searchable within a limited amount of time?

The answer to this question is the PDM; a heatmap over the search area, which describes the probability of finding the LP at a given position and can be generated through various models.

A PDM is a 2D heatmap over a given area describes the probability of finding the LP at a given position. The PDM is key in allocating search resources effectively as it provides insights into which areas are of a higher likelihood of finding the LP and thus more impactful. The most simple PDM is the euclidean distance circle [24]–[29] which uses the mobility data for a profile along with a time estimate of how long the person has been missing for. If the LP travels at a constant speed, then the radius grows linearly with time. This, however, ignores the most significant influence on any person in the wilderness, which is the landscape of the wilderness itself.

The clearest impact of the landscape on an LP is through physical interactions. This can take the form of rivers and cliffs creating natural barriers for movement, or deep snow reducing the speed of traversing the terrain, as well as in many other ways. Equally as impactful, however, is the manner in which an LP reasons about the terrain

whilst trying to self-rescue, or protect themselves. This effect can be modelled through psychological profiles [24], [30], [31].

A psychological profile is a statistically unique group that distinguishes the LP by markers such as age, condition, or activity from other groups. This profile then describes various behavioural properties of the LP such as where they are likely to be found, how far they are likely to travel, or their injury modes.

The hiker profile, for example, covers 1.62 times as much distance from the PLS as a child aged between 7 and 9 years. This is an intuitive result: the young child covers less area than the hiker. However, it is less intuitive that hunters stay mobile for 1.86 times as long as hikers, which could be the result of the physical terrain or a heightened drive to self-rescue. Being able to categorise a LP into one of the many groupings (34 for Koester [24] and 15 for Perkins *et al.* [30]) is invaluable for the searcher to fully understand the search mission at hand.

It is thus necessary to leverage the psychological profile of the LP, as well as the physical landscape of the search area at hand in order to effectively generate a PDM.

1.4 Search Path Planning

Typical path planning has a defined start and end state, which gives rigid boundaries to the possible solutions. Algorithms like rapidly-exploring random tree (RRT) [32] explore the problem space until a suitable solution is found based on a heuristic, like a Euclidean distance check. The applications for this form of path planning are virtually limitless, with real-world examples including lane-changing in autonomous driving [33], and helicopter autopilot systems [34].

In contrast, coverage planning represents a distinct, albeit less prevalent, category of path planning. Instead of focusing on reaching a target state, these algorithms prioritise the complete coverage of a designated area. The optimisation goals often involve minimising the total time to traverse the area or reducing overlapping coverage, effectively replacing the target end-state paradigm of traditional path planning. Applications of coverage planning tend to be more industrially oriented, finding utility in areas such as water quality monitoring [35], precision seed planting using UAVs [36], [37], and LiDAR surveying of urban structures [37], [38]. A key characteristic is the assumption of a finite and fully coverable area given a limited endurance.

Despite their differences in objectives, both traditional path planning and coverage planning share a common characteristic: a defined goal. Path planning aims to manoeuvre a system to a specific target state, while coverage planning strives to maximise the area covered. This fundamentally differs from the nature of search path planning (or search planning). Searching would be unnecessary in scenarios where the desired target state, such as the location of a LP, is known beforehand. Similarly, if a UAV could exhaustively cover an entire area within a single battery charge, the need for autonomous search strategies would be removed. This highlights that both traditional

path planning and coverage planning operate under conditions of complete information regarding the environment or the objective to be achieved.

Search planning distinguishes itself by operating within an environment that is known but cannot be entirely covered due to the limitations of the search platform’s endurance. The core challenge lies in efficiently exploring this environment, utilising a prior PDM to guide the search, with the primary objective of locating the LP as rapidly as possible. This implies that search planning is driven by uncertainty regarding the exact location of the target within a known, but partially unobservable, environment.

1.4.1 Knowledge Gaps

The current state-of-the-art in PDM generation, as reviewed in Sect. 2.1, leverages aggregated LP behavioural data and geographic information system (GIS)-based environmental modelling. However, existing methods such as the travel time cost surface model (TTCSM) [39] and T2net [40] are constrained by their reliance on comprehensive path network maps. Other approaches, like the 6D Behavioural Vector [41], [42], exhibit limited spatial transferability due to the need for location-specific retraining.

These limitations highlight a critical gap: the absence of a flexible and generalisable model capable of incorporating varied behavioural and environmental datasets without requiring extensive domain-specific calibration. Such a model would enhance PDM robustness across diverse terrains, particularly in data-sparse or previously unmapped regions. Although machine learning (ML) holds significant promise for addressing these challenges, its application to PDM generation remains largely unexplored. This is likely due to the scarcity of unified datasets and the complexity of integrating multimodal spatial data. This thesis aims to fill that gap through the development of a behaviour-informed, GIS-integrated framework for PDM generation.

In the domain of search path planning, prior work has demonstrated the potential of RL to outperform heuristic-based strategies, particularly in discretised observation and action spaces [18], [43]. However, these methods are often confined to simplified models with limited capacity for generalisation or emergent behaviour. Notably, current RL-based approaches do not account for continuous action spaces or platform dynamics, which are critical for UAV-based WiSAR operations in complex terrain.

There is therefore a gap in leveraging continuous RL techniques that incorporate high-fidelity motion models and advanced state representation methods (e.g., auto-encoders) for search planning. Addressing this gap could enable more realistic, adaptive, and efficient autonomous search behaviours. This thesis contributes a novel RL framework that integrates dynamic UAV models and continuous policy spaces, aiming to optimise real-world search efficiency and policy generalisability.

1.5 Research Aims and Objectives

This thesis aims to develop a system that generates UAV search paths to find lost people faster. The structure of this is outlined in Fig. 1.2, which shows the top-level flow from search parameters to a search path.

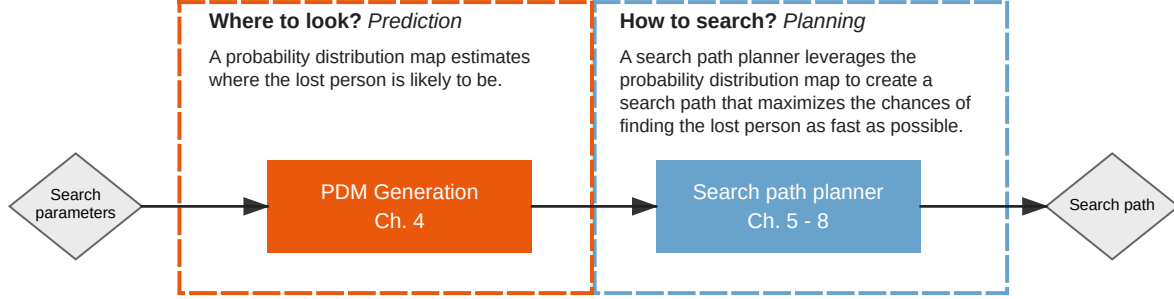


Figure 1.2: Top-level overview of the end-to-end data flow to get from the initial search parameters to the final search path.

In order to implement the structure from Fig. 1.2, the research objectives are to:

- develop a flexible PDM generation algorithm for WiSAR operations that are robust to varying geographical locations;
- introduce a framework for calculating the accumulated probability over the search path through cubature due to the use of continuous PDM and action spaces;
- create an RL algorithm for search planning that operates within a continuous action space;
- address issues in existing RL methods for search path planning, such as large model sizes and training overhead, by splitting out the feature extraction phase as an observation preprocessing step; and
- evaluate alternative methods to frame-stacking in RL, aiming to improve performance in training times, model size, and overall efficacy.

1.6 Thesis Structure

A literature review of the state-of-the-art in PDM generation and search planning is presented in Ch. 2. Following this is a summary of the ML methods and techniques used in this thesis in Ch. 3. This provides a standardised nomenclature for the reader, which is particularly important for ML due to the sheer vastness of the field. Furthermore, this section prevents duplication across the subsequent chapters.

The five technical chapters of this thesis then answer the four questions:

- *Where to look?* Ch. 4 — Create a generalisable, agent-based PDM generator that models real-world lost person behaviour without location-specific retraining.

- *How to search?* Ch. 5, and Ch. 6 — Show that a continuous-space RL agent is more capable for the search task than benchmark methods from the literature through cubature integration and a fully observable path history.
- *How to make it efficient?* Ch. 7 — Solve the path-memory problem by using recurrent architectures to compress the path history, creating a smaller and faster overall model.
- *How to make it realistic?* Ch. 8 — Develop a “dynamically-aware” planner through RL training with a six degree-of-freedom UAV model, 3D terrain, and energy constraints.

Finally, a conclusion is drawn in Sect. 9.1 with the limitations of this thesis and future work discussed in Sect. 9.2.

1.7 Contributions To The Literature

This section outlines the contributions to the literature made by the thesis author throughout the duration of their research.

1.7.1 Journal

- J.-H. Ewers *et al.*, ‘Predictive Probability Density Mapping for Search and Rescue Using an Agent-Based Approach With Sparse Data,’ *IEEE Access*, vol. 13, pp. 60 802–60 813, 2025, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2025.3557693 — Closely followed by Ch. 4.
- J.-H. Ewers *et al.*, ‘Deep reinforcement learning for time-critical wilderness search and rescue using drones,’ *Frontiers in Robotics and AI*, vol. 11, p. 1 527 095, Feb. 2025, ISSN: 2296-9144. DOI: 10.3389/frobt.2024.1527095 — Closely followed by Ch. 6.

1.7.2 Conference

- J.-H. Ewers *et al.*, ‘GIS Data Driven Probability Map Generation for Search and Rescue Using Agents,’ in *IFAC World Congress 2023*, 2023, pp. 1466–1471. DOI: 10.1016/j.ifacol.2023.10.1834 — Closely followed by Ch. 4.
- J.-H. Ewers *et al.*, ‘Enhancing Reinforcement Learning in Sensor Fusion: A Comparative Analysis of Cubature and Sampling-based Integration Methods for Rover Search Planning,’ in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Abu Dhabi, United Arab Emirates: IEEE, Oct. 2024, pp. 7825–7830, ISBN: 979-8-3503-7770-5. DOI: 10.1109/IROS58592.2024.10801978 — Closely followed by Ch. 5.

- J.-H. Ewers *et al.*, ‘Stone Soup Multi-Target Tracking Feature Extraction for Autonomous Search and Track in Deep Reinforcement Learning Environment,’ in *2025 28th International Conference on Information Fusion (FUSION)*, Rio de Janeiro, Brazil: IEEE, Jul. 2025, pp. 1–8, ISBN: 978-1-0370-5623-9. DOI: 10.23919/FUSION65864.2025.11123969
- J.-H. Ewers *et al.*, *Multi-Target Radar Search and Track Using Sequence-Capable Deep Reinforcement Learning*, Feb. 2025. DOI: 10.48550/arXiv.2502.13584. arXiv: 2502.13584 [cs] — Presented at RLDM 2025.

1.7.3 Pre-print

- J.-H. Ewers *et al.*, *Recurrent Auto-Encoders for Enhanced Deep Reinforcement Learning in Wilderness Search and Rescue Planning*, 2025. DOI: 10.48550/ARXIV.2502.19356 — Closely followed by Ch. 7.

1.7.4 Journal (Co-Author)

- S. Swinton *et al.*, ‘Autonomous mission planning for planetary surface exploration using a team of micro rovers,’ *Frontiers in Robotics and AI*, vol. 12, 2025, ISSN: 2296-9144. DOI: 10.3389/frobt.2025.1565173 — Informed aspects of Ch. 5.

Chapter 2

Literature Review

This chapter provides an overview of the academic literature pertinent to the research conducted in this work. Whilst not all concepts discussed will be directly relevant to this work, the ideas and influences across the literature will be.

The relevant literature has been split into two separate sections: PDM generation in Sect. 2.1 and search planning in Sect. 2.2. Finally, Sect. 2.3 provides an overview of the sections and insights into how they are related.

2.1 Probability Distribution Map Generation

PDM generation methods, at their most general, take a search region definition and output a two-dimensional heatmap of the probabilities of finding the LP at a given position. These methods typically incorporate multiple datasets from sources such as Koester [24] or Perkins *et al.* [30] and GIS maps to fine-tune the PDM to the search scenario at hand. Psychological profiles are used further to categorise the LP into an existing dataset to use the associated statistical information. Koester [24] uses International Search And Rescue Incident Database (ISRID) [52], the world’s largest WiSAR dataset, to identify over 30 unique profiles, which are most commonly used in the following sections. Likewise, Perkins *et al.* [30] outlines a similar database for WiSAR statistics in the United Kingdom.

The PDM generation methods are grouped into either *static* (Sect. 2.1.1) or *agent* (Sect. 2.1.2). The latter includes any model that simulates an agent traversing the terrain with the PDM resulting from a Monte-Carlo analysis, whereas the former is comprised of the remaining methods, which are typically analytical solutions. A final discussion of the PDM generation literature is presented in Sect. 2.1.3.

2.1.1 Static

This section will cover the *static* PDM generation methods from the literature. A static method does not generate the PDM through a Monte-Carlo manner with the use of an agent — that is a model that simulates a single LP traversing the landscape. Methods

like the euclidean distance circle (from Sect. 2.1.1.1) embody this label particularly well due to the simple analytical method used to generate the PDM.

2.1.1.1 Distance Circle

Euclidean Distance Circle The most basic form of PDM generation is the euclidean distance circle [24]–[29]. This method works on bounding a search area within a circle based on the straight line distance from statistics about how far away from the PLS a person might be. Backed by data from ISRID [52], Koester [24] sets a guideline for how to create the euclidean distance circle from statistical data:

1. Draw the 25 %, 50 %, 75 % and 95 % distance rings based on statistical crow’s-flight distance tables (as seen in Fig. 2.1).
2. Associate the aforementioned probability with each ring. The area outside of the euclidean distance circle is given the remaining 5 %.
3. Divide each probability by the area enclosed to give the probability per unit area.

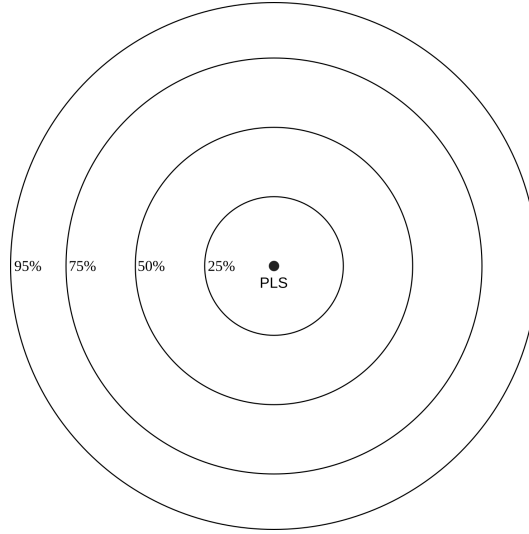


Figure 2.1: The Euclidean Distance Circle is defined by four circles: the 25 %, 50 %, 75 % and 95 % distance rings. The radius of each ring is defined by the LP’s statistical profile.

Whilst statistically sound, this method fails to incorporate the terrain of the search area. 75 % of WiSAR incidents happen in mountainous regions with many barriers to travel around [28]. Heth *et al.* [25] incorporates impassable features, such as rivers, by excluding areas with no direct access to them. This greatly reduces the search area as shown by Fig. 2.2.

As well as the inability to incorporate geographical features, the search area size radically changes depending on the LP profile. From Tbl. 2.1, the 95 % radius for *Child 13–15* is 5.65 times larger than the *Child 10–12* category. This makes it hard to focus resources in a WiSAR scenario based purely on the euclidean distance circle.

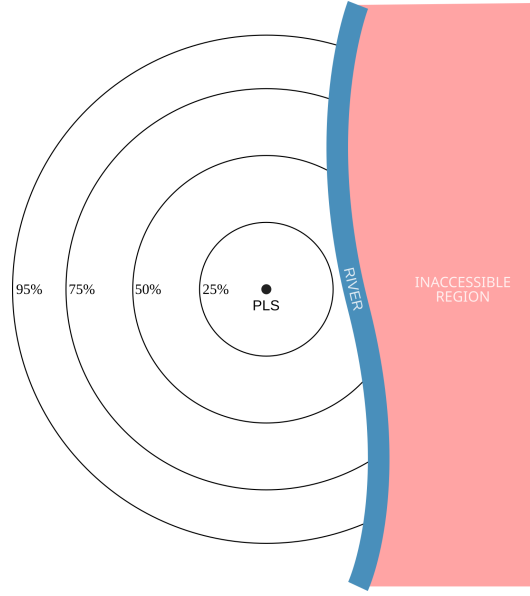


Figure 2.2: The Euclidean Distance Circle with inaccessible regions due to impassable features.

Table 2.1: Horizontal distance (km) found from PLS in temperate mountainous terrain for two child-based categories [24].

Percentage Found	Child 10–12	Child 13–15
25 %	0.8	0.8
50 %	1.6	2.1
75 %	3.2	4.8
95 %	9.0	21.4

The euclidean distance circle model is very fast to produce, and will most likely (95 % of the time) contain the LP. However, the search area grows proportional to r^2 , making it extraordinarily difficult to assign resources effectively.

Distance Dispersion Circle The previous euclidean distance circle assumes that a LP is equally likely to travel in all directions from the PLS. However, Koester [52] has data on the dispersion angle, which is the angle between the intended destination (or direction of travel) and the found location. This means that the WiSAR team must know where the LP was heading, which is not always the case. During a search, clues may lead to the direction of travel being found such as footsteps in the mud.

Jurecka *et al.* [53] proposes to couple the euclidean distance circle model with dispersion data in the distance dispersion circle model. A circle is drawn from the take-off location with a radius of the maximum flight time. This is then reduced down to a search area using the dispersion angle. From Tbl. 2.2, for a 75 % probability of enclosing the LPs position, a dispersion angle of 64° can be used. The area is then given an equal probability. However, an extension to this model may use a method as outlined in Sect. 2.1.1.3 for more granular probabilities. Fig. 2.3 shows how a large portion of the euclidean distance circle is superfluous using the distance dispersion circle model.

If the take-off location and the PLS were the same, this would be around 64.4% of the euclidean distance circle being discarded.

Table 2.2: Dispersion angle (degrees) for a Hiker [24]. The dispersion angle is the angle between the intended destination (or direction of travel) and the find location with respect to the PLS.

Percentage Found	Temperate	Dry
25 %	2	20
50 %	23	47
75 %	64	124
95 %	132	175

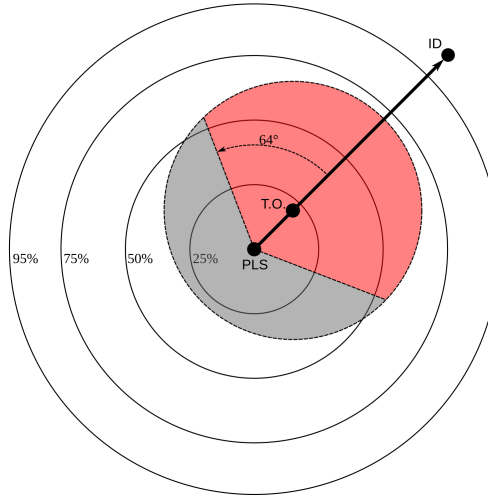


Figure 2.3: Distance Dispersion Circle example with an ID outside of the 95 % circle.

2.1.1.2 Track Offset Model

The track offset is the shortest distance from a track or path to the find location [24], as seen in Fig. 2.4. Doke [29] found that the *hiker* category, within Yosemite National Park, has a median track offset of 58m, whereas Koester [52] places this figure closer 100m. It is worth noting that the ISRID data shows a larger track offset, as it also contains data for flatter terrain where travel away from a path is generally easier.

2.1.1.3 Watershed

A watershed, shown in Fig. 2.5, is defined by the high ground between two drainage basins [54] and provides a natural resistance to travel. This is the theory behind the watershed model [24], [28], [29], in that the LP is more likely to be found within the watershed that contains the PLS.

Watersheds are given an integer number based on how many watershed divides are between them and the 0^{th} — the one that contains the PLS — watershed to construct

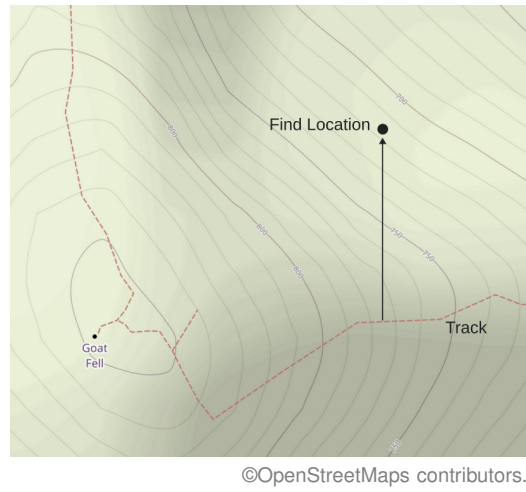


Figure 2.4: Example of a track offset measurement from the path going up *Goat Fell* on the Isle of Arran in Scotland.

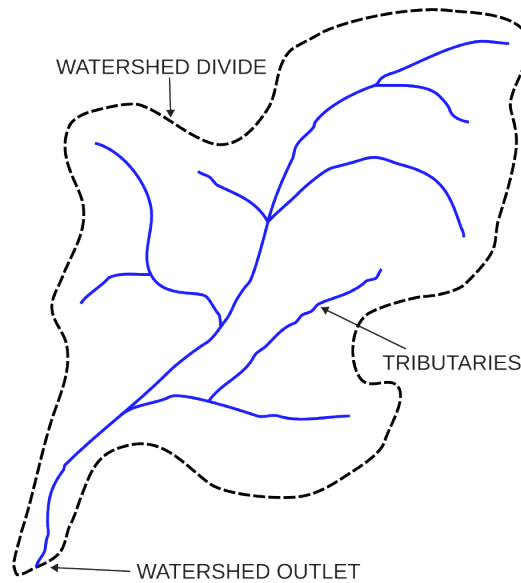


Figure 2.5: A watershed is an area of land that channels precipitation to streams, and rivers into outflow points. The watershed boundary is defined by a continuous line of high-points.

the model. Doke [29] found that in Yosemite National Park, 48 % of LPs were found within the 0^{th} watershed, and 38 % were found in the adjacent watersheds. Sava *et al.* [28] found similar results, presented in Tbl. 2.3.

Table 2.3: Statistics of LP find location by watershed number categorised by US state [28].

US State	Watershed Number			
	0	1	2	3+
Arizona	47 %	38 %	6 %	10 %
New York	70 %	24 %	3 %	4 %
Yosemite	50 %	43 %	6 %	2 %

Historical statistics are assigned to the watersheds of that number to assign a prob-

ability to each watershed. For example, all areas one watershed away in Arizona will be assigned 38% as per Tbl. 2.3. This probability is then divided by the total area of all watersheds at that distance to give each number's probability density.

As with other methods, the difficulty of using this method is the access to accurate GIS data. The USA has high fidelity free, and open source maps of hydrological units [55], classified into size units: regions, sub-regions, accounting units, cataloguing units, watershed, and sub-watershed. Sava *et al.* [28] used the sub-watershed model with 10,000–40,000 acres of size for the most granular results. However, tools like SAGA [56], [57] coupled with QGIS [58] allow extrapolation of watersheds from digital elevation map (DEM) and other open-source data sets.

Distance watershed Sava *et al.* [28] builds on the watershed model and proposes the distance watershed model, which is a combined model using the euclidean distance circle and watershed models. Combining models is a common technique in WiSAR as described by Mattson [59], which uses a consensus among searchers to inform where to begin the search objectively. Surowiecki [60] has a broader result applicable to any model.

The distance watershed is constructed by dissecting the watersheds from Sect. 2.1.1.3 using the rings from Sect. 2.1.1.1. This splits a watershed intersecting with a ring into at least two new watersheds. To then assign the probability to the sub-areas, the probability from a lookup table is divided by the total area of all polygons assigned to that distance-watershed region.

Using Sava *et al.* [61] over a 25×25 km area, the euclidean distance circle model had a mean score of 0.780 and the watershed model 0.611. The DW model out-scored both with 0.805 by simply combining the terrain information from the watershed model and the distance information from the euclidean distance circle model.

2.1.1.4 Mobility Model

Whilst the watershed, distance watershed, and Track-Offset models incorporate some information about the terrain, it is limited to 1 layer of data — watershed or path network. Using more models, or data, is beneficial in creating more nuanced models, as discussed in Surowiecki [60]. This also gives the benefit of generating more customised PDMs for a single search area. The distance dispersion circle model, for example, uses historical data, which is sparse at best, giving highly generalised results. However, a benefit of simpler models is that less computational or human effort is required to generate a PDM. A searcher in the hills may not have access to a powerful computer to calculate specific PDMs or the time to wait for results. However, if the PDMs were generated at a base station on the route to the incident scene and sent to the searcher, this problem would be mitigated.

Mobility models [27], [62] try to estimate how far the LP may have travelled in all directions from the PLS. This is done by combining simulations of walking speed with

environmental effects that may affect their motion. For example, walking through a dense forest will be slower than over a grassy field. Another approach is considering the path of least resistance to a given coordinate from the PLS. This can be done by calculating the cost of passing through a cell, then applying a path planning algorithm like RRT [32], IRRT [63], or RRT* [64]. Other models may use Accumulated Cost Surface [65] algorithms as they tend to be included in GIS software.

TTCSM In 2015, the USA National Park Service created the TTCSM [39]. This model uses the notion of *percent of maximum travel time* to evaluate the cost of travel through a cell. The percent of maximum travel time is assigned by a location expert and can vary for different search areas as seen in Tbl. 2.4. Limits are assigned to the area through the percent of maximum travel time by giving it a value of 0, such as for the middle of a lake or a cliff *.

Table 2.4: Excerpt of example percent of maximum travel time values for national land cover database [67] cover classes from Frakes *et al.* [39].

National Land Cover Database Cover Class	Percent of Maximum Travel Time (% of normal)
Open Water	0
Developed, Open Space	90
Mixed Forest	75
Woody Wetlands	20
Cultivated Crops	80

A major factor affecting the walking speed of the LP is the slope angle, which is extrapolated from the DEM. TTCSM uses the tobler hiking function [68] model for walking speed,

$$w = 6e^{-3.5|m+0.05|}, \quad (2.1)$$

which provides a continuous estimation based on gradient $m = \tan(s)$, where s is the slope angle.

Another walking speed estimation method is the Naismith approximation with Langmuir correction [69], [70], given by

$$w = w_0 / \begin{cases} 1 + \alpha m & s > 0^\circ, \\ 1 - 0.55p_0 |m| & -12^\circ < s < -5^\circ, \\ 1 + 0.55p_0 |m| & s < -12^\circ, \\ 1 & else \end{cases}, \quad (2.2)$$

where $\alpha = 7.92$ is called the *Naismith's number* [71] and relates 7.92 units of horizontal to 1 unit of climb, p_0 is the base walking speed at 0° inclination. Campbell *et al.* [72]

*Cliffs are defined as any slope steeper than 31° [66]. What a LP may perceive as traversable greatly depends on their experience, fitness levels, and equipment.

leverages modern open-source datasets and fits the Lorentz function

$$w = \frac{3.6c}{b\pi \left(1 + \left(\frac{s-a}{b}\right)^2\right)} + d + se, \quad (2.3)$$

where $a, b, c, d, e \in \mathcal{R}$ are the learned parameters. Campbell *et al.* [72] suggests using $a = -1.527, b = 14.041, c = 36.813, d = 0.320, e = 0.00273$ for walking and provides parameters for faster-paced activities like running.

Fig. 2.6 shows how tobler hiking function smoothly decreases walking speed from a maximum of 6km h^{-1} as the slope goes into the negative. On the other hand, Naismith abruptly drops around -15° , contrary to data from Kay [73]. Like the tobler hiking function, Campbell *et al.* has a negative x -axis offset, highlighting that walking at a slightly negative gradient improves speeds until it becomes too much effort to break the impact.

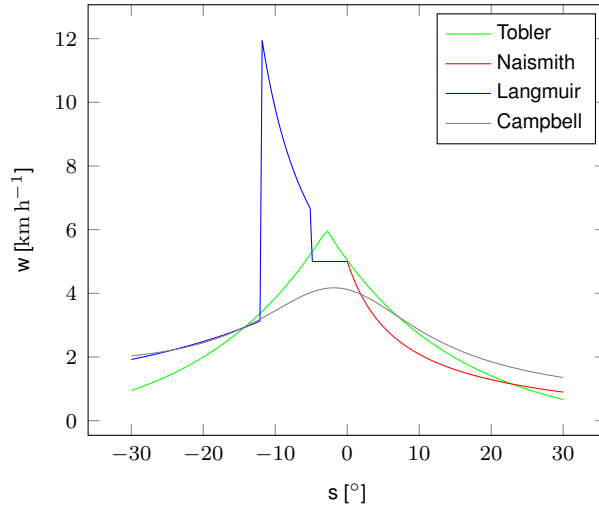


Figure 2.6: Comparison of walking speed functions based on slope angle.

T2Net T2Net [40] takes inspiration from TTCSM and focuses on trail networks. In this model, network theory is used rather than the rasterised method from before. Network theory is prevalent in many disciplines, such as electrical engineering [74] and biology [75], and can increase performance [76] drastically. Drexel *et al.* [40] define the network’s edges as a path from the GIS layer.

Every edge is given a cost associated with the time taken to traverse. This cost is derived from Eqn. 2.1 and a constant driving speed if the edge is a road. The cost is then inverted to give time per unit length for every edge to calculate the time necessary to travel from one node to the next via the edge.

An evident drawback of this model is the requirement for a detailed model of the trails and roads of a search area, and for the search area to have sufficient trails. The latter is significant as this model would not work as well in the rural Scottish Highlands as it would in the English Lake District. Nonetheless, the statistical likelihood from Koester [52] of a person being found close to the trail may be deemed sufficient to

employ this model, or like the distance dispersion circle model, it may be used to inform other models further.

2.1.1.5 Linear Regression with Transfer Learning

Šerić *et al.* [77] proposes a linear regression-based model trained on GPS data from WiSAR searchers with transfer learning [78] to adapt to how a LP would walk across a landscape. Linear regression has been previously used in Keijsers *et al.* [79] to predict walking speed, and Feng *et al.* [80] used a latent terrain model to predict the traversal path of an agent. Wang *et al.* [81] employed transfer learning to predict urban crowd movement patterns.

Šerić *et al.* [77] used the European CORINE land cover [82] dataset and a European DEM [83]. The GPS data was collected by the Croatian Mountain Rescue Service from 1999 to 2020, resulting in 1908 trails. Combined with the CORINE land cover and DEM datasets, this allows a linear regression equation with six variables: CORINE land cover ID, length of the segment, slope, the difference in elevation in the segment, elevation above mean sea level, and total distance travelled. The equation is

$$y = q_0 + \sum_{i=1}^n q_i x_i, \quad (2.4)$$

where y is the prediction, and q_0 is a bias. q_i are training constants, and x_i are variable inputs.

Once the model had been sufficiently trained on the large dataset, it was adapted to a smaller sample of $n = 20$ Croatian WiSAR cases via transfer learning. 50 % of LPs were found within 1km of the PLS, and 75 % are found within 2km. Thus, a total simulation time of 4h was selected. If a cell can be reached from multiple cells at each time step, it is given the lowest time to reach the PLS.

2.1.2 Agent-based

An agent-based model is one where the PDM is generated using a Monte-Carlo analysis to create an aggregate dataset [84]. For each data point in the dataset a model — the agent — is applied in a sequential manner. It is important that each data point is generated independently to simulate an individual LP.

2.1.2.1 Isoprobability curves

Macwan *et al.* [23] proposes a modular target-behaviour-prediction method. Unlike the distance dispersion circle model, this model assumes that the LP can travel in any direction from the PLS with equal likelihood. The agent is then propagated outwards in a straight line at a target speed. Similar to TTCSM, this model uses a walking speed factor based on slope steepness. However, for the implementation in Macwan *et*

al. [23], empirical data from Wojtyra [85] and Bohannon [86] was used to give

$$q(s) = \begin{cases} m_1 s + b_1 & s_{\max, dec} \leq s \leq 0^\circ \\ m_2 s + b_2 & 0^\circ \leq s \leq s_{\max, inc} \\ 0 & else \end{cases} \quad (2.5)$$

where q is the speed scaling factor, m_i, a_i and b_i are parameter constants fitted to the data with $i \in [1, 2]$, and s is the slope angle. Macwan *et al.* [23] also implements basic object avoidance by traversing the object's edge with the shortest path to get back to the ray.

To then create the PDM, a normal distribution is fitted along the ray using data, such as Perkins *et al.* [30], with increments to create probability contours. The more rays are cast from the PLS, the higher fidelity the PDM becomes, with the drawback of increasing computational time. A simulated search had a 45 % success rate with three rays, and this increased to 97 % with 25 rays.

Macwan *et al.* [87] created a multi-robot path planning algorithm to use the iso-probability PDM and found that 68.4 % of searches resulted in a success compared to 20.1 % with a predetermined path that did not use a PDM. Furthermore, the probabilistic method had an average search time of 1271s, which was 540s faster than the deterministic method.

However, the method fails to incorporate more information about the area than a DEM. The model would create the same PDM for woody wetlands as it would for developed open space if the slopes were identical.

2.1.2.2 Behaviour Vectors

4D Behaviour Vector The model proposed in Hashimoto *et al.* [88] uses four behaviours — random walk, line following, straight line, and staying put — which are defined in Fig. 2.7. At every time step k , the agent randomly picks a behaviour based on a likelihood factor encoded in a 4D vector, emulating a lost person's core behaviours. Based on the selected behaviour, the agent then picks one of the nine adjacent cells (including the one currently occupied). For line follow, a Canny edge detection filter [89] was used on satellite imagery. Random walk picks any cell, and straight line goes in the same direction as the previous step. Staying put selects the currently occupied cell.

The likelihood of picking a behaviour is adjusted to match statistics gathered for a category in Koester [24]. Every possibility (with a granularity of 0.1) is generated with a total of 266 unique layouts and run as a Monte Carlo simulation. The most statistically similar is then selected as the representative agent via the Kullback-Leibler divergence [90]. Results varied significantly for different initial conditions, as seen in Tbl. 2.5, which implies that a more granular model is required.

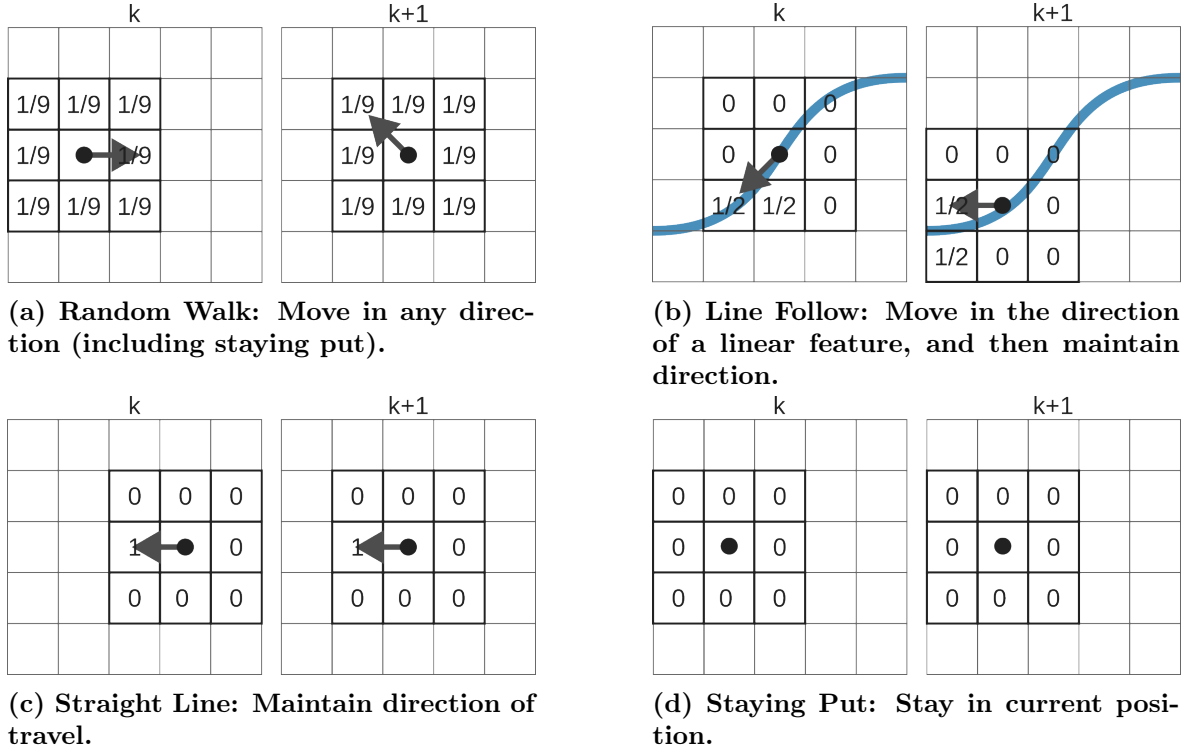


Figure 2.7: Definitions of the four behaviours from Hashimoto *et al.* [88]. The agent’s possible moves are defined by a 3×3 grid centred around the current position. Each cell is assigned a probability of being selected based on the current behaviour.

Table 2.5: Best fit results from Hashimoto *et al.* [88] where the 4D behaviour vector is defined as [random walk, line follow, straight line, staying put].

LP Category	Initial Position 1	Initial Position 2
Autism	[0.0, 0.9, 0.0, 0.1]	[0.1, 0.5, 0.2, 0.2]
ATV	[0.2, 0.2, 0.5, 0.1]	[0.1, 0.1, 0.5, 0.3]
Child	[0.0, 0.9, 0.0, 0.1]	[0.1, 0.5, 0.2, 0.2]
Dementia	[0.0, 0.9, 0.1, 0.0]	[0.2, 0.0, 0.3, 0.5]
Vehicle	[0.3, 0.1, 0.6, 0.0]	[0.2, 0.6, 0.0, 0.2]

Force-based Agent Cangan *et al.* [91] and Cangan [92] build on the 4D Behaviour Vector by incorporating the behaviours into a force-based agent model. The dynamic equation is given by

$$m\ddot{x} + (a||\dot{x}|| - b)\dot{x} = \alpha F^G(x_i) + \beta F^R, \quad (2.6)$$

where x is the agent position, m is the inertia, a is a friction coefficient, b is self-acceleration, $F^G(x_i)$ is an environmental force acting on the agent, F^R is a stochastic process, and α, β are constant tuning parameters.

This model encodes the environmental impact on the LPs movement in $F^G(x_i)$, where the force may be resistance to movement across the terrain or the slope tending the agent to move downhill. F^R is where the behavioural data is incorporated, allowing expansion on [88] by including other behaviours.

To generate the PDM, a Monte-Carlo simulation of the agent is undertaken to generate a heatmap. This is a very computationally expensive process, and run-time information is based on [24] to give the final location.

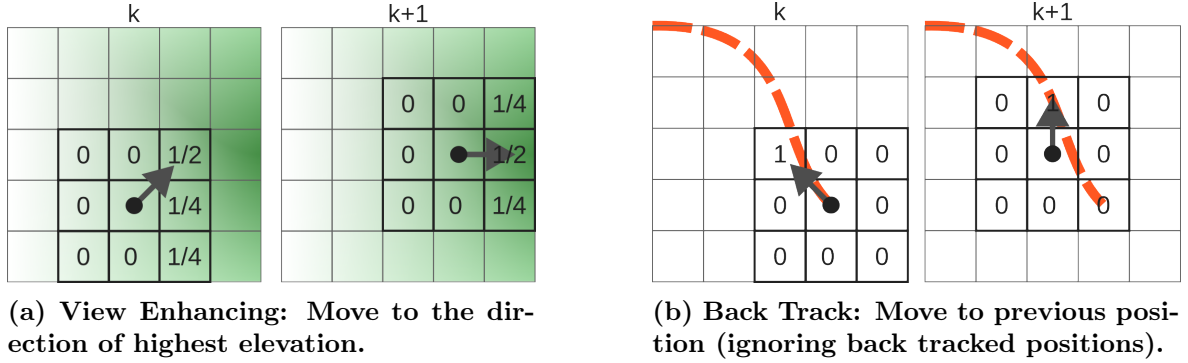


Figure 2.8: Definitions of the two additional behaviours introduced in Heintzman *et al.* [41]. The remaining four behaviours can be seen in Fig. 2.7. The agent's possible moves are defined by a 3×3 grid centred around the current position. Each cell is assigned a probability of being selected based on the current behaviour.

6D Behaviour Vector Like the Force-based Agent, Heintzman *et al.* [41] builds on 4D Behaviour Vector by implementing two new strategies — view enhancing (Fig. 2.8a), and back tracking (Fig. 2.8b) — to capture the behaviour of an LP in a 6D Behaviour Vector. View enhancing, is a local hill climb [93] at time-step k on the 8 surrounding cells, and back tracking moves to the cell occupied at $k - 1$.

At each time step t , the velocity is calculated as $v(t) = x(t + 1) - x(t)$, with a smoothing factor of α in the previous velocity to make the trajectory more realistic. The updated position for the next time step then becomes

$$x(t + 1) = (2 - \alpha)x(t) + (\alpha - 1)x(t - 1) + \alpha v(t). \quad (2.7)$$

The agent is run in a Monte Carlo simulation to generate the PDM. This makes having a dynamic simulation length important based on the LP profiles mobility data from Koester [52]. T is sampled from the cumulative distribution function which is fitted to the Weibull distribution, defined as

$$f(x, a, b)_{\text{weibull}} = 1 - \exp \left(- \left(\frac{x}{a} \right)^b \right), \quad (2.8)$$

where a and b are the fitting parameters. However, when using the Kullback-Leibler symmetric divergence [94] for this study (see Ch. 4), it was found that the log normal (also known as a Galton) function [95], defined as

$$f(x, \mu, \sigma, c)_{\text{log normal}} = \frac{1}{2} \left(c + \operatorname{erf} \left(\frac{\ln x - \mu}{\sigma \sqrt{2}} \right) \right), \quad (2.9)$$

was more representative of the original data with a Kullback-Leibler symmetric diver-

gence value (lower is better) of 0.0161 compared to 2.8711 from the original distribution. However, the Weibull distribution from Eqn. 2.8 is notably without a modifiable offset term, which can be seen in Fig. 2.9 and Eqn. 2.8 can be modified to include the offset term c to become

$$f(x, a, b, c)_{\text{weibull}} = c - \exp\left(-\left(\frac{x}{a}\right)^b\right), \quad (2.10)$$

which has a much improved divergence value of 0.0494, putting it inline with the log normal of Eqn. 2.9.

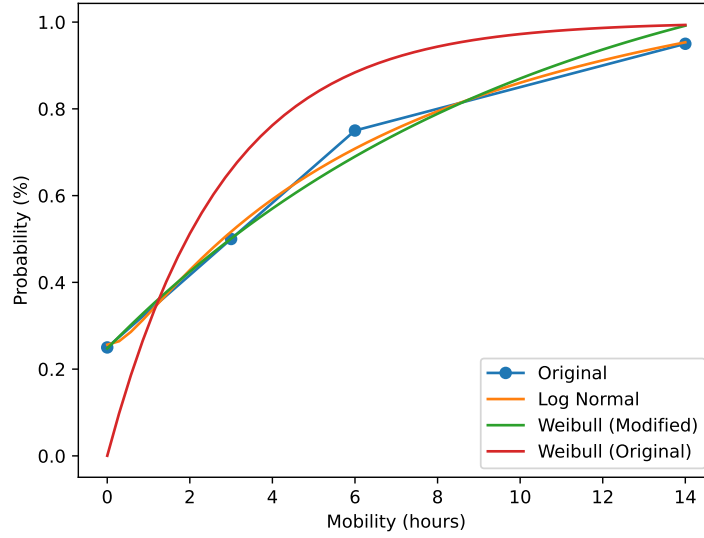


Figure 2.9: Fitted functions to the Hiker profile data from Koester [24].

As with the 4D variant, the results varied greatly depending on the location. Tbl. 2.6 shows how the behavioural vector model in an area with hills and lots of linear features is most likely to back track. In the other location — one that is flatter with fewer roads — the preferred behaviour is straight line. However, it should be noted that this does not mean that a LP with that behavioural profile *will* behave like that, but rather that the model must use these strategies to behave like the statistical data.

Table 2.6: Best fit results from Heintzman *et al.* [41] for the 6D behaviour vector.

Area Description	Random Walk	Line Follow	Straight Line	Staying Put	View Enhancing	Back Tracking
Varied foothills with linear features and bodies of water	1/6	1/6	0	0	1/6	1/2
Flatter with fewer roads	0	0	2/3	1/6	0	1/6

2.1.2.3 State Change Bayesian Network

Lin *et al.* [96] proposes a method employing Bayesian Networks to predict which adjacent hexagonal cell [97] an agent will move to based on state changes. Bayesian Networks have been used in path planning for multiple coordinated UAVs to maximise accumulated probability in Bourgault *et al.* [98], and to fit human behaviour data [99]. Along with Markov Chain Monte Carlo [100] simulations, a PDM can be generated based on statistical data.

The model from Lin *et al.* [96] incorporated three types of state:

- *topography*: lake, plains or hill;
- *vegetation*: sparse, medium, or dense; and
- *elevation*.

Due to the lack of data, the probability distributions associated with a transition from one state to another (i.e. a person is more likely to transition from deep forest to open field for vegetation than the other way around) were estimated by area experts. In order to incorporate impassable boundaries, another state could be added with a 0% chance of transition to it.

In a Bayesian χ^2 test of goodness-of-fit [101], Lin *et al.* [96] found that the model fits a synthetic dataset well.

2.1.3 Discussion

The analytical comparison of various PDM generation algorithms is incredibly challenging. One approach is to run a Monte-Carlo analysis over many PLS and compare the new aggregate dataset to input datasets from the likes of Koester [24], and Perkins *et al.* [30]. This, however, does not encapsulate nuance in specific areas that may lead to various biases.

Sava *et al.* [28] and Sava *et al.* [61] therefore proposed *MapScore*[†] which was intended to be the benchmark for this field. This software is based on a metric proposed by Rossmo [102] to compare probability maps for crime forecasting. Nevertheless, whilst this is a universal PDM benchmark, it has not seen widespread adoption across the literature. PDM generation algorithms must, therefore, be compared using alternative features.

Tbl. 2.7 shows a quantitative comparison of the algorithms outlined in Sect. 2.1.1 and Sect. 2.1.2. This table highlights that euclidean distance circle, the most widely used algorithm as described by Koester [24], does not consider the environment at hand. Euclidean distance circle works by drawing a circle with the radius being equal to the time the LP has been lost multiplied by the expected movement speed, which can

[†]*MapScore* [61] is unavailable (<http://mapscore.sarbayes.org/>) and unmaintained at the time of writing.

Table 2.7: A quantitative comparison of various PDM generation methods.

Citation	Algorithm	Agent-based	Environment Features	Behavioural Data	Models Endurance
[24]–[29]	Euclidean Distance Circle			Mobility	✓
[53]	Distance Dispersion Circle			Mobility, Dispersion Angle	✓
[29]	Track Offset			Mobility, Track Offset	✓
[24], [28], [29]	Watershed		DEM	†	
[28]	Distance Watershed		DEM	†	✓
[39]	TTCSM		Roads, Paths, Flowing Water, Stagnant Water, DEM, Land Cover	Mobility, ‡	✓
[40]	T2Net		Paths, DEM	Mobility, ‡	✓
[77]	Linear Regression with Transfer Learning		DEM, Land Cover	†	✓
[23], [87]	Isoprobability Curves	✓	DEM	Mobility	✓
[88]	4D Behaviour Vector	✓	DEM	Mobility, Distance From PLS	✓
[91], [92]	Force-based Agent	✓	DEM	Mobility	✓
[41], [42]	6D Behaviour Vector	✓	DEM, Roads, Paths, Water Features, Power Lines	Mobility, Distance From PLS	✓
[96]	State Change Bayesian Network	✓	DEM, Land Cover	Mobility, ‡	✓

† This work used generic behavioural data. However, the dataset the model was based on will likely have been biased towards the *Hiker* profile [24].

‡ A dataset specific to this work was used, which is not a part of Koester [24] or Perkins *et al.* [30].

be done on a paper map with only a compass at hand. This is simple yet effective, so Quigley *et al.* [103] encodes this method within their path planning algorithm. However, this method benefits from simple improvements such as the distance dispersion circle or track offset model, effectively overcoming the r^2 search area growth of the euclidean distance circle.

More complex methods, like the watershed or distance watershed methods, work on the same principle as the euclidean distance circle but incorporate the terrain at hand through GIS datasets. This method is much more complex than drawing circles but ensures that the corresponding PDM indicates the physical landscape.

TTCSM and T2Net take this approach one step further by incorporating the local network of roads and paths alongside multiple other GIS datasets. However, these models assume that the LP is on a path. This assumption is sound in popular areas but less so in remote regions. Like the aforementioned algorithms, both methods incorporate the mobility time and terrain impedance values similar to Doherty *et al.* [27]. This approach of using scores associated with the difficulty level of traversing a type of terrain effectively affects the results based on the local geology.

Incorporating terrain impedance values is something that the 4D Behaviour Vector, Force-based Agent, and 6D Behaviour Vector do not do. However, these methods model the impact of the slope on the LP, which is similar to the walking speed models discussed in Sect. 2.1.1.4. The 6D Behaviour Vector method, like TTCSM and T2Net, does, however, include the notion of linear feature following (roads, paths, water features, and power lines from Tbl. 2.7). This latter method then indirectly incorporates the *location found* dataset from Koester [24].

With ISRID, the most complete WiSAR database, only having around 145 000 entries [24], [52], [104], [105], it is sometimes impossible to statistically model certain behaviour. Lin *et al.* [96], for example, uses the transition probabilities of an LP moving from one land cover type to another. However, these values were created by a *subject matter expert* lacking rigour. Linear Regression with Transfer Learning approaches this problem from an ML perspective and uses transfer learning. Šerić *et al.* [77] trains a linear regression on a larger dataset of WiSAR expert's GPS data and then retrain the regression on a much smaller dataset of LP data. This could be an effective approach for future ML-based models given the data-constrained environment that this field lies in.

One dataset that seems near to universally used across all models from Tbl. 2.7 is the notion of endurance through the mobility dataset. The constraints invoked by physics and statistics are intuitive and rigorous. No similar hard limitation exists for the other datasets, which is why there is no consensus on which ones are vital to the problem of PDM generation. For this reason, the importance of terrain cover on the model, like in TTCSM and T2Net, should be further explored.

2.2 Path Planning For Search And Rescue

Traditional path planning is concerned with getting from position A to position B. Constraints such as obstacle avoidance [106]–[114], and optimality [107], [108], [114]–[120] are commonly seen in the literature and reflect the real-world requirements of vehicle path planning. This is most commonly in the form of point-to-point path planning, where the system needs to get from one defined state to another. For a UAV path planner this could be the tuple (x, y, z, ψ) with the associated velocities or an address for an in-car navigation system. The path planner must then derive the path from the starting state to the final state.

This clearly poses a fundamental issue for search or prioritised coverage planning in that no final state is definable. In this path planning formulation, the target state is unimportant; rather, how the system traverses the search region is paramount.

This section will discuss various forms of path planning—from traditional point-to-point to exploration to search planning—in the literature. Whilst some path planning formulations are not applicable to search planning, they provide valuable insights into different heuristics and approaches through their application. A final discussion of the literature is presented in Sect. 2.2.6.

2.2.1 Graph-based

One of the earliest and best-known path-planning algorithms is Dijkstra’s algorithm, which was first introduced by Dijkstra in 1959 [121] and has been successfully applied in the field of robotics [119], [120], [122], [123]. This graph algorithm finds the *shortest* route from one node to another based on the cost of traversing the associated edge. Typically, this cost refers to the total distance travelled but can also refer to the total time or control input taken and other metrics. Furthermore, the notion of an edge does not mean that the route from one node to another is a straight line; it is just that there is no opportunity to branch to another node along the path associated with the edge. Whilst Dijkstra’s is a graph algorithm, it can easily be applied to a grid-based problem by simply making each cell a node in the graph that can be entered from any of the four immediate or eight surrounding cells.

Dijkstra’s core pitfall is that the cost of traversing the optimal path to every node is calculated and quickly becomes intractable. Hart *et al.* [116] proposed A*, a heuristic-based extension to Dijkstra’s algorithm. In Dijkstra’s algorithm, the next node to be evaluated is the one with the lowest score; in A*, the next node to be evaluated is based on a heuristic — typically the Euclidean or Manhattan distance. The inclusion of this heuristic approach makes A* substantially cheaper to compute than Dijkstra’s for large search spaces [124].

Similar algorithms to A* have been proposed, such as D* by Stentz [107], which effectively inverts A* by starting from the destination node, resulting in better exploratory behaviour. Further alternatives such as Focused D* [125], D* Lite [126], and

Theta* [127] add additional incremental improvements.

A typical pitfall of graph-based methods for search planning is the requirement to have a target node. Cai *et al.* [128] therefore proposes SAR-A*, which selects the next waypoint based on multiple criteria to promote maximising gathering information. However, this approach relies heavily on the waypoint selection process and does not benefit substantially from applying A*. This dependency on the waypoint selection process is clearly seen from Fig. 2.10, where the set of waypoints identified, and used in the resultant path are the same.

A search mission transitions into rescue when the LP is found, making graph-based algorithms more applicable to this stage of WiSAR. Du [129], for example, applies a modified A* to the cooperative UAV path planning problem in a wilderness scenario where the drones could feasibly carry equipment to multiple LP once detected.

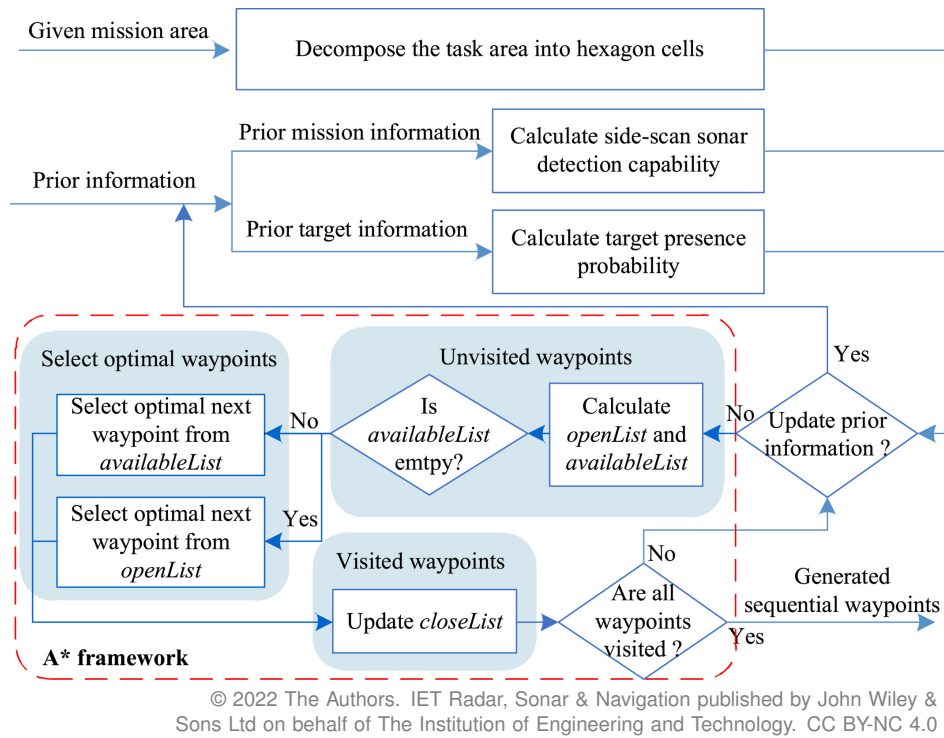


Figure 2.10: SAR-A* is a graph-based algorithm that builds a sequential set of waypoints using the A* framework multiple times. It visits every identified waypoint using a notion of priority. Source: [128].

Another downside of graph-based methods is the requirement to deal with a discretised search space, which introduces limitations in scalability and solution quality [130]. WiSAR is typically concerned with vast areas in the tens to hundreds of squared kilometres, making the use of discretised spaces problematic. Bergman *et al.* [131] addresses this problem for path planning by using a low-quality solution from a graph-based algorithm as the initial solution for the subsequent optimisation path planning routine.

Graph-based path-planning algorithms, like Dijkstra's and A*, excel at finding optimal routes by calculating edge traversal costs. However, they are ill-suited for search planning, especially in WiSAR, primarily because they require a predefined target node, which is unknown in search missions. Additionally, these methods necessitate

a discretised search space, proving computationally expensive and limiting solution quality across the vast areas common in WiSAR operations.

2.2.2 Sampling-based

Sampling-based methods are an effective way to avoid the *curse of dimensionality* [132]. This approach goes beyond simply escaping the discretised two-dimensional grid; it can further encode dynamics constraints.

LaValle *et al.* [32] proposed the RRT algorithm for non-holonomic vehicle path planning. This approach randomly samples the state space, and if a connection can be made to this new point from a known safe state, then the connection is added to the tree. This is powerful in that not every possible trajectory from state to state needs to be explored, but rather, only a small subset of samples is ever attempted. The final path is then found by backtracking through the tree.

There are a multitude RRT variants such as IRRT [63], RRT* [64], RRT-Connect [133] and RRT+ [134], which build on this sampling-based approach. Brunner *et al.* [135] and Palmieri *et al.* [136] propose A*-RRT and Theta*-RRT respectively where a hierarchical approach is used. First, a graph-based algorithm is applied on a rough grid followed by a RRT variant to fine-tune the path. This greatly accelerates the performance over standard RRT whilst not succumbing to the large areas that graph-based methods struggle with.

Similar to the graph-based methods from Sect. 2.2.1, RRT-based approaches are most commonly designed for point-to-point path planning and are unsuitable for search planning. Lindqvist *et al.* [137] attempts to overcome this with Exploration-RRT. By applying a multi-goal framework with RRT-based methods, Exploration-RRT successfully explores an unknown area effectively. However, using the multi-goal means that this approach still requires targets to be generated — as seen in Fig. 2.11 — and is highly sensitive to how these targets are generated.

Lin *et al.* [138] takes a similar approach to Exploration-RRT by proposing hierarchical heuristics using the *mode goodness* metric based on Gaussian mixture models. This approach prioritises regions of interest for search based on a two-dimensional cost function and then searches them in order. Yao *et al.* [139] further builds on this approach using Gaussian mixture model with model predictive control for cooperative multi-UAV search.

Zhang *et al.* [140] proposes Shrinking-POMCP, which removes the need for a hierarchical framework through the use of the partially observable Markov decision process and Monte-Carlo tree search. This method builds an internal representation of the environment with A* as the rollout policy. Shrinking-POMCP outperformed its benchmark methods but requires expensive operations during policy unrolling. In order to reduce the complexity, the method is constrained to giving commands to the UAV in only the four cardinal directions, which massively reduces the action space, limiting performance.

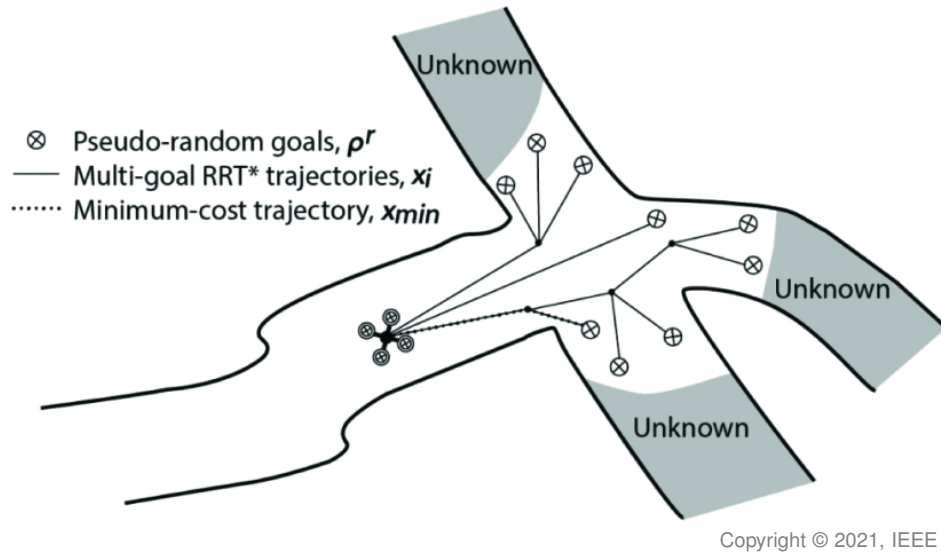


Figure 2.11: Exploration-RRT is an iterative algorithm, which generates multiple potential goal states to which multi-goal RRT* generates trajectories. These are then refined before the minimum cost trajectory is selected and traversed. Source: [137].

2.2.3 Coverage

A common approach to searching an unknown area is to aim for complete coverage. This indiscriminate search can be effective when no a priori knowledge is available, and a uniform probability distribution of detecting the LP can be assumed [141].

The simplest form of complete coverage is the lawnmower pattern (also known as parallel swaths or zig-zag) [16], [37], [141]–[143], which can be seen in Fig. 2.12. This approach decomposes the defined search area into a set of straight search segments with a short path to move on to the next one. The union of these search areas is then the greater search area and covers 100% of the defined region.

Quigley *et al.* [103] and Goodrich *et al.* [141] propose an alternative to the lawnmower pattern in the form of a parametrised spiral pattern; the Hopf bifurcation spiral seen in Fig. 2.13. This is an effective pattern when using fixed-wing UAVs instead of multi-copters, as the former platform handles sharp corners of the lawnmower pattern extremely poorly, requiring additional interpolation techniques such as Dubin paths [144] for smooth traversal.

The Boustrophedon Cellular Decomposition proposed by Choset *et al.* [145] is an alternative to the spiral and lawnmower patterns. The two aforementioned methods handle non-convex search regions or obstacles poorly. The Boustrophedon Cellular Decomposition subdivides the search region into primitive subregions whose union provides complete coverage. The subregions are then traversed using any coverage path planning method. However, the order in which this is done is critical. In order to solve this efficiently, Bähmann *et al.* [146] formulates this as the Equality Generalised Travelling Salesman Problem [147]. This formulation of the Travelling Salesman Problem is applied as each cell has multiple directions in which the lawnmower pattern

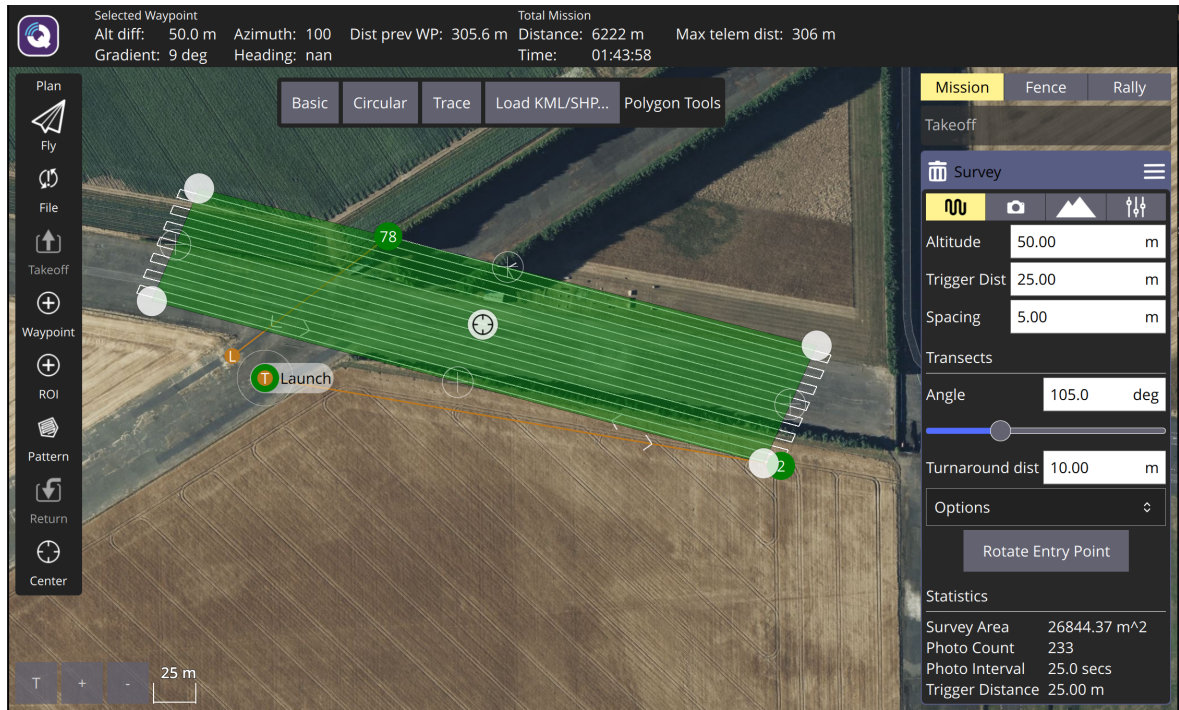


Figure 2.12: Example of a lawnmower pattern search pattern mission created in *QGroundControl*.

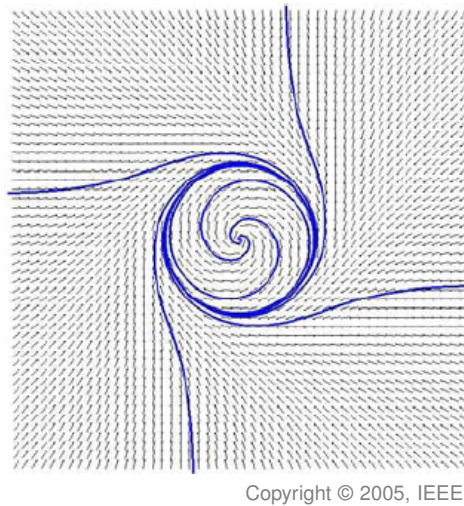


Figure 2.13: The Hopf bifurcation spiral ensures that any start position on the plane will converge on the limit circle. This would have the limits from the euclidean distance circle model with the central start position being the PLS to cover the viable search area. Source: [103].

can be oriented (north to south, west to east, north-east to south-west, north-west to south-east, and the reverse) resulting in different performances for the same cell depending on entry and exit cells.

2.2.4 Optimisation-based

Optimisation techniques are often used in path planning when multiple competing objectives, difficult-to-define boundaries, or complex nonlinear systems are at hand. Methods such as model predictive control have found wide-spread use in the context

of UAV path planning [148]–[152]. The ability to encode the desired target state in terms of optimising cost functions whilst respecting constraints, such as the multi-UAV method proposed by Ge *et al.* [153], which incorporates collision avoidance and turn-rate constraints, make this family of path-planning methods highly suitable to the search context.

Local hill climb (LHC) is one of the simplest optimisation-based path planning algorithm methods. LHC is the opposite of gradient descent because it takes a step in the direction of the positive gradient of the cost function instead of in the negative direction. Waharte *et al.* [154] finds that this family of methods is highly prone to converging on local minima and requires modification to be useful in search planning. Similar conclusions are drawn by Zhang *et al.* [140], where it was found that standard LHC performed only as well as the lawnmower pattern or worse in a search context. Lin *et al.* [155] therefore introduces and applies a method labelled *global warming* to combat LHC’s propensity to get stuck in local modes. The resultant algorithm, local hill climbing global warming convolution (LHC_GW_CONV), escapes local maxima by sequentially raising the probability floor, which forces the LHC algorithm to explore alternative routes. Lin *et al.* [138] finds that this method is superior to the LHC-esque approach from Bourgault *et al.* [98], Moreover, in some instances, it is also superior to a more complex hierarchical approach.

A commonly used improvement on LHC is simulated annealing [156]–[159]. This method uses probabilistic transitions to neighbouring states to prevent the LHC pitfall of getting stuck in local modes. Worrall [158] applies simulated annealing to an urban exploration problem, which shows vastly improved results over simple LHC.

Worrall [158] further proposes the use of the genetic algorithm (GA) [159]–[162] for urban exploration planning. GA is an evolutionary algorithm that introduces random mutations and population crossover at each step, which is particularly suited to encourage the exploration of highly non-linear problem spaces. However, Worrall [158] found the GA approach to be in line with the results produced by simulated annealing in the urban context. A similar approach is implemented by Morin *et al.* [163], which uses ant colony optimisation for search planning. Ant colony optimisation mimics an ant colony through the notion of pheromones to encourage exploration. However, it was found that performance for ant colony optimisation varied significantly based on the hyperparameters of the pheromone system. The wall time to solve the problem also increased from 1 min for the smallest time horizon to 30 min for the largest. This highlights a fundamental issue with sampling-based optimisation techniques like GA and ant colony optimisation, which require large amounts of interactions with the environment to effectively converge on a solution.

Time budgets are a common problem for optimisation problems, and designing an efficient optimisation scheme is paramount. Subramanian *et al.* [22] proposed the LRH-B algorithm to optimise a search path. LRH-B is compared to the interior point method and was found to converge on a solution substantially faster. Likewise, performance can

vary significantly between algorithms. For this reason, Wang *et al.* [164] proposes the Hyper-Heuristic Method, which combines multiple standard optimisation algorithms, such as GA, and selects from the population of solutions using a fitness function. This approach was found to outperform individual optimisers, with outstanding performance being seen in large search areas.

2.2.5 Machine Learning

ML in the form of RL is widely used in path planning with a successful application for UAVs [18], [165]–[172], marine vessels [19], [173]–[176], robotic manipulators [177]–[179], hypersonic gliding vehicles [180], unmanned ground vehicles [109], [181], [182]. The underlying principle of the Markov decision process (MDP) (see Sect. 3.3.1) makes RL particularly suited for the field of control. In comparison, supervised and unsupervised learning, the most prevalent ML methods, require vast datasets for optimal operations. This is a problem for path planning because to get this data the problem must be solved using alternative means, which negates the need for the dataset. This dataset would not be wasted, however, as the transfer of a costly expert policy, perhaps using optimisation methods, to a real-time student policy is possible. For this application imitation learning algorithms such as Behaviour Cloning [183], Generative Adversarial Imitation Learning [184], [185], or DAgger [186] would be more suitable than RL. Similarly, Laperrière-Robillard *et al.* [187] proposes the use of multiple supervised learning methods as surrogate models to estimate the probability of success of a lawnmower pattern (Sect. 2.2.3) in various configurations in order to speed up optimisation methods without the need of simulations. Yao *et al.* [188], on the other hand, uses a fully connected network (FCN) to prioritise which stretches of a river to search.

Exploration planning is a common application for RL due to the complex interactions of the unknown environment with the agent dynamics. Niroui *et al.* [189] proposes the use of SLAM alongside RL for real-time exploration in cluttered environments. Similarly, Talha *et al.* [20] applies RL with environment discretisation for simultaneous exploration and search without a priori knowledge of the environment. Zuluaga *et al.* [21] incorporates frontier exploration [190] into the search task where the UAV-based agent gathers information on the environment over time. These methods outperform benchmarks through the ability of RL to learn emergent behaviour through its interactions with the environment.

A different methodology to the holistic single policy approach is used by Peake *et al.* [191] who uses a hierarchical RL scheme, which uses Recurrent-DDQN for target search and A2C for region exploration with a single UAV without prior information. Likewise, Peake *et al.* [192] applies a dual-policy RL (DDPG and recurrent A2C) to WiSAR, handling exploration and trajectory planning separately but using a single policy for top-level exploration planning. A dual-policy design is also incorporated by Talha *et al.* [20] where separate policies are used for navigation and exploration. Ebrahimi

et al. [173] uses RL to localise LPs via radio signal strength indexes without prior knowledge of the search domain map. Whilst performant, these approaches require a substantial amount of additional training overhead. Updating independent policies simultaneously can also result in unstable training, as shown by Talha *et al.* [20] where both the navigation and exploration planner see total performance loss during training before recovering. Peake *et al.* [191] avoids this by training the lower-level navigation policy first, followed by the exploration policy, avoiding a shifting observation or action spaces.

Whilst effective at exploration planning, none of the aforementioned path planning approaches consider a priori information to prioritise search. This makes them akin to the coverage planning methods discussed in Sect. 2.2.3. These methods typically aim to efficiently explore as much of the unknown area given a set budget (time, energy, control action, and other metrics) to maximise the finding of a uniformly placed LP. This makes them mostly applicable to the field of urban or mining exploration, where highly uncertain environments make predictions of LP locations near impossible.

Without prioritised regions, the exploration problem is akin to the standard coverage problem. However, adding prior information within the coverage path planning algorithm allows for prioritising areas of interest. Ai *et al.* [19] and Wu *et al.* [18] both explore this problem in a maritime setting. A PDM can be easily generated using drift models, which is then passed to the RL algorithm as an observation. Since the area is covered entirely by the nature of this approach, emphasis is put on the speed of detection, with both works showing improved results by using RL. Benalaya *et al.* [193] applies this approach to a scenario using UAVs and a randomised PDM and was able to outperform a benchmark method using mixed integer linear programming from Benalaya *et al.* [194]. However, the action space of all three approaches is only in the cardinal directions, and the state space is on a discrete map. This is lacking in fidelity and misses out on the ability for RL to learn emergent behaviour by constraining the problem. This further exacerbates small search regions up to only 6×6 in size, which further dramatically reduces the solution space.

The search planning problem can be viewed as the prioritised coverage path planning problem with path length constraints. These constraints can take the form of time, distance travelled, energy usage, or any other quantitative metric that prevents the search path from covering the entire area. This problem is applied to a underwater unmanned vehicle scenario using RL in Manjanna *et al.* [195] and Manjanna *et al.* [43]. The Policy Gradient method, with a discrete action and state space, uses a multi-resolution feature aggregation observation space [196]. Manjanna *et al.* [197] extends this approach to the multi-agent scenario for prioritised data collection. Both single underwater unmanned vehicle implementations were able to outperform the benchmark lawnmower (referred to as the *boustrophedonic* path [145] in this work) coverage pattern, whilst the multi vehicle implementation outperformed DARP [198], which is analogous to a multi-agent lawnmower pattern. However, similar to the previously

discussed prioritised coverage algorithm, the action space is limited to the eight surrounding cells, which restricts the potential of the RL policy. Likewise, the state space is constrained to a discretised map, but unlike the prioritised coverage algorithms, this work uses a much larger 30×30 search area.

2.2.6 Discussion

While search planning shares elements with various path planning methods, it is uniquely defined by the intersection of two criteria:

- *Prioritised planning*: utilising prior information to rank task urgency; and
- *Endurance constraints*: adhering to the physical limits of the system.

As illustrated in Tbl. 2.8, search planning occupies a specific niche. If the prioritised planning criterion is removed, the objective shifts to exploration planning, where the agent seeks information without pre-defined weights. Conversely, if the endurance constraint is ignored, the method simplifies to prioritised coverage planning. In the absence of both criteria, the system defaults to simple coverage planning.

	Respects Endurance	Ignores Endurance
Prioritised Planning	Search Planning	Prioritised Coverage Planning
Non-Prioritised Planning	Exploration Planning	Simple Coverage Planning

Table 2.8: Classification of search planning and its adjacent forms based on the presence of prior information and physical endurance limits.

For this reason, a quantitative assessment of a subset of the previously discussed work is carried out to determine which methods can be classified as search planning. The results from this can be seen in Tbl. 2.9 where methods that implement prioritised planning or limited endurance or both are grouped into rows. Additional information is presented (dynamic awareness, continuous action space, grid world, and intended application) to assist the discussion but does not impact the method’s classification. Furthermore, this is not an exhaustive list of work in this field, as highlighted by the presence of only three coverage methods and the same number of exploration planning methods. However, the table becomes significantly more representative of the literature when it comes to search planning, which is an underdeveloped field.

An interesting result from Tbl. 2.9 is that the coverage methods do not adhere to a grid-world formulation and implement a continuous action space. In contrast, the prioritised coverage and search planning methods almost exclusively do the opposite. This is due to the significant increase in search space complexity, as is evident from the prioritised coverage methods from Ai *et al.* [19] and Wu *et al.* [18]. In this work, the grid world was kept below a 10×10 area, which is too small to be representative of a real WiSAR scenario. However, For research purposes, this dramatically improves computational speeds. Interestingly, Worrall [158] is the only optimisation-based method to

Table 2.9: A quantitative comparison of various path planning algorithms that incorporate elements of true search planning from the literature review. The path planning categorisation from Sect. 2.2.6 is highlighted on the left of the table. The *planner type* column corresponds to the sections from Sect. 2.2 in which the path planner is first mentioned.

Citation	Planner Type	Prioritised Planning	Limited Endurance	Dynamic Awareness	Continuous Action Space	Grid World	Intended Application
Standard Coverage	[145] Coverage				✓		Generic
	[103] Coverage				✓		Wilderness
	[146] Coverage				✓		Urban
Exploration	[137] Sampling		✓		✓	✓	Subterranean
	[20] ML		✓	✓		‡	Wilderness
	[158] Optimisation	†	✓	✓	✓		Urban
Prioritised Coverage	[19] ML	✓		✓		✓	Marine
	[18] ML	✓				✓	Marine
Search	[128] Graph	✓	✓			✓	Submarine
	[138] Sampling	✓	✓			✓	Wilderness
	[139] Sampling	✓	✓	✓		✓	Urban
	[140] Sampling	✓	✓			✓	Urban
	[87] Optimisation	✓	✓		✓		Wilderness
	[155] Optimisation	✓	✓			✓	Wilderness
	[163] Optimisation	✓	✓			✓	Generic
	[22] Optimisation	✓	✓	✓	✓	‡	Wilderness
	[164] Optimisation	✓	✓			✓	Wilderness
	[43] ML	✓	✓	✓		✓	Submarine
	[197] ML	✓	✓	✓		✓	Submarine

† Prioritised planning is used in this work only when the vehicle gets within a threshold range of the LP, at which point it aims to detect the LP's temperature. Outwith a temperature gradient detection, which is contained to a limited radial zone around the LP, the algorithm uniformly explores the environment.

‡ The search space exists as a grid world whilst the agent has a continuous state space. This distinction is important given that it carries out prioritised planning.

not use a grid-world. This is due to the application of GA, which creates solutions by sampling the environment, which is much less costly than applying a gradient-based method. In the context of graph- or sampling-based methods, like Cai *et al.* [128] and [138], using a grid world is fundamental to the algorithm, which cannot work in a continuous state space.

The inclusion of dynamic awareness across the categories also poses an interesting debate. RRT from Sect. 2.2.2 inherently facilitates this, however Exploration-RRT from Lindqvist *et al.* [137] does not leverage this. Yao *et al.* [139] is the only sampling-based approach that has dynamic awareness during path planning. However, this is handled by a model predictive controller within the algorithm which uses optimisation methods. The only methods that, therefore, implement dynamic awareness from Tbl. 2.9 are then optimisation or ML-based. This makes sense as the dynamics are typically encoded in either the constraints for optimisation methods or the cost function for the ML methods in order to maximise the utility from the limited endurance. Coverage paths could feasibly be traversed over multiple missions, whereas search paths attempt to minimise the time it takes to find the objective.

Drawing the focus onto the search planning categorisation, it is evident that the grid-world and non-continuous action space configuration is dominant. Subramanian *et al.* [22] is the standout work against the paradigm. In this work, the path is in a continuous space, whilst the PDM is represented as a grid-world. The path is terminated when an estimate of the available information left to gather is below a threshold value, which puts this algorithm in a grey area between prioritised coverage and search planning. However, The inclusion of dynamic awareness emphasises efficient search paths, which is in the spirit of efficient search planning. The standout search planning work that uses realistic search areas is seen in Manjanna *et al.* [43] and Manjanna *et al.* [197] which applies ML in the form of RL to the submarine search problem. These search paths are evaluated using physical submersibles in field tests, which show the robot's adherence to its physical limitations. The use of the RL method *policy gradient* enables path generation through a learned policy vector learned from simulated interactions with the environment. However, like optimisation methods, compromises are made to reduce the search space size in the form of a significantly reduced action space (the policy can select one of the eight surrounding cells).

The gap in the literature is then as follows: a true search planning method using continuous action and state spaces with dynamic awareness does not exist. This has most likely not been attempted yet due to the associated computational complexity of this approach. However, the real world is continuous, and algorithms should reflect this to bridge the simulation-to-reality gap.

2.3 Chapter Summary and Conclusions

This chapter reviewed the academic literature relevant to this research, with a particular focus on WiSAR using UAVs. In general, it is not easy to compare PDM generation or search planning methods using benchmark results. The methods from the two aforementioned categories were, therefore, quantitatively evaluated in Sect. 2.1.3 and Sect. 2.2.6 respectively. This does not allow a like-for-like comparison but does highlight trends in the literature.

Within the PDM generation literature, there is a trend towards incorporating additional information to improve models. More complex methods use five or more environment features and two or more behavioural data points. In comparison, simpler methods typically use one environmental feature and one or two behaviour data points. Across the literature, however, the DEM is extensively used for the environment feature and the mobility behaviour dataset.

Across the search planning literature, an apparent abundance of methods that work on a grid world model with a discrete action space exist. Optimisation and ML methods are also favoured when incorporating dynamic awareness. These latter two methods are well suited to applying bounds on the resultant path.

There are few works, however, that span both disciplines. Heintzman *et al.* [41] applies the 6D Behaviour Vector method to generate the PDM and then applies RRT and trajectory refinement with predefined sector entry and exit points. Macwan *et al.* [23] and Macwan *et al.* [87] generates a mobility-based PDM, which the path is generated from using the notion of isoprobability curves (the contours of the probabilities). The former work proposes a detailed PDM generation method but then does not apply a search planning algorithm (see Sect. 2.2.6 for the definition of search planning). The latter applies a simple DEM and mobility model to generate the PDM and then implements an optimisation-based search planning method.

In summary, it is challenging to incorporate PDM generation, and search planning and a modularised approach should be taken. This gives flexibility in the selection or design of the PDM generation method, which can output an image. This gives further flexibility in the search planning, which is already a complex task. This is simplified by incorporating the generated PDM during path planning instead of explicitly encoding the LP behaviour.

Chapter 3

Machine Learning Technical Background

This chapter serves as a generalised introduction of the ML methods used within this thesis. Concepts like the FCN (Sect. 3.1.1) are widely used within the literature to be common knowledge, but others like the convolutional self-attention (CSA) (Sect. 3.1.4.2) or recurrent auto-encoder (RAE) (Sect. 3.2.1) are relatively new at the time of writing. Similarly, ML as a field has a surplus of terminology and syntax for the same concepts, which can be ambiguous and lead to confusion. It is therefore important for the reader’s understanding of this thesis to have a single source of truth to be able to refer back to.

All relevant ML base architectures, like the FCN and convolution neural network (CNN), are introduced in Sect. 3.1. Finally, Sect. 3.2 describes the auto-encoder (AE) and its relevant variants whilst Sect. 3.3 introduces the two RL algorithms used in Ch. 6, Ch. 7, and Ch. 8. Additional concepts and techniques — such as backpropagation and optimisation algorithms — are available in Appendix E.

3.1 Machine Learning Architectures

3.1.1 Fully Connected Network

The FCN (also known as the Multi-Layer Perceptron, Feedforward Neural Network, or Artificial Neural Network) is the basis of most machine learning models. At its core, the FCN is made up of many individual neurons.

The basis of the neuron is the perceptron, which attempt to model the human brain [199]. A layer of perceptrons can be defined by

$$\mathbf{y} = H(\mathbf{W} \cdot \mathbf{x} + b),$$
$$H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}, \quad (3.1)$$

where H is the Heaviside step function, W are the weights, and b is the bias. $\mathbf{x}$ and $\mathbf{W}^T$ are of the shape $(1, m)$ and m depends on the architecture.

With backpropagation becoming the de facto learning method for ML, this approach became outdated due to the derivative of the Heaviside step function being the dirac delta function

$$\frac{\partial}{\partial x} H(x) = \delta(x). \quad (3.2)$$

This would result in zero gradients for all inputs other than $W \cdot x + b = 0$, at which point the gradient is infinite resulting in impossible updates.

Replacing $H(x)$ with a general function $f(x)$, that has a more meaningful derivative $f'(x)$, is therefore required. This modifies Eqn. 3.1 to become

$$\mathbf{y} = f(\mathbf{W} \cdot \mathbf{x} + b), \quad (3.3)$$

where W and b are the same as in Eqn. 3.1. $f(x)$ is more commonly known as the activation function and can take many forms (see Sect. E.1.4).

Stacking multiple layers of neurons results in the FCN, which exhibits powerful non-linearity. The layers between the input and output layer are referred to as hidden layers and are typically more numerous than either the input or output. Fig. 3.1 shows a small FCN but highlights the interconnectivity between each sequential layer.

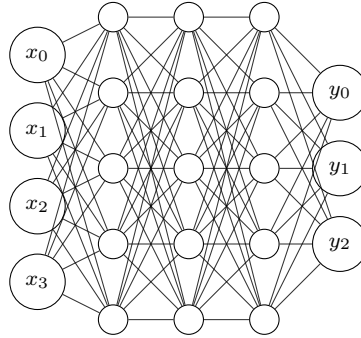


Figure 3.1: An example FCN with four inputs, three outputs, and three hidden layers with five neurons each.

3.1.2 Convolutional Neural Network

There are many data types within machine learning that can be used as inputs to the system with one of the most prevalent being an image. An image is typically defined to have the shape $1 \times W \times H$ for greyscale, or $3 \times W \times H$ for RGB. It is therefore easy to see that this input then rapidly explodes in size as larger inputs are used. For example, a simple 100×100 greyscale image would require 100 000 input neurons for an FCN structure, which is impractical at best. The CNN was therefore introduced [200], [201], which leverages multidimensional convolution to drastically reduce the required amount of learnable parameters.

The CNN applies a two dimensional kernel over a single channel of the input image

with

$$G(x, y) = W \star f(x, y) = \sum_{\delta x=0}^M \sum_{\delta y=0}^N W(\delta x, \delta y) \cdot f(x + \delta x, y + \delta y), \quad (3.4)$$

where f is the input data of shape $W \times H$, W is the convolution kernel of shape $M \times N$, and G is the output image of shape $(W - M + 1) \times (H - N + 1)$. This can be applied multiple times to each channel with different learnable kernels to increase the total amount of channels which can be helpful during the feature extraction stage. Eqn. 3.4 can be modified to account for C_{in} input channels and C_{out} output channels, as well as the addition of a bias term W_{bias} , to become

$$G(x, y, c_{\text{out}}) = \sum_{c_{\text{in}}=0}^{C_{\text{in}}} \left[W_{\text{bias}, c_{\text{in}}, c_{\text{out}}} + \sum_{\delta x=0}^M \sum_{\delta y=0}^N W_{c_{\text{in}}, c_{\text{out}}}(\delta x, \delta y) \cdot f(x + \delta x, y + \delta y) \right], \quad (3.5)$$

where $c_{\text{out}} \in [0, \dots, C_{\text{out}}]$.

What is important to note is that having multiple input and output channels results in $C_{\text{in}}C_{\text{out}}$ kernels. Fig. 3.2 shows a scenario with $C_{\text{in}} = 3$ and $C_{\text{out}} = 2$ resulting in a total of 6 kernels.

As previously alluded to, the CNN has substantially fewer parameters than an FCN for the same image input. Eqn. 3.5 also shows that the CNN has the added benefit of being able to easily change the number of channels at a linear cost. A common use for this is to take an RGB image ($C_{\text{in}} = 3$) and increase the number of channels to $C_{\text{out}} = 64$. With a small kernel of $M = N = 3$, this results in a total of 1728 learnable parameters which is orders of magnitude lower than if the single input and output FCN were to be used.

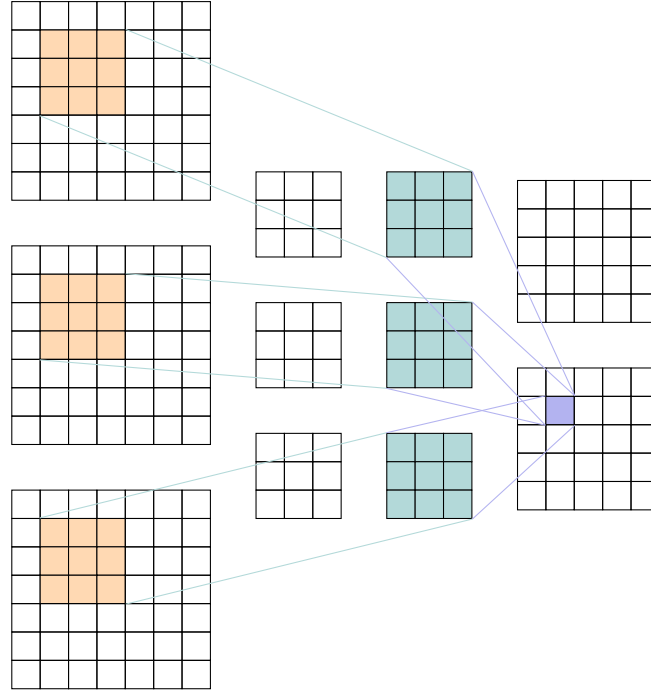


Figure 3.2: CNN simplified example with $C_{\text{in}} = 3$, $C_{\text{out}} = 2$, $M = N = 3$, and $W = H = 7$. This results in 6 kernels and an output of shape 5×5 .

Additional modifications can be made to Eqn. 3.5 to further customise its interaction with the input data. Stride, dilation, and padding are common additions, which primarily relate to how the kernel is applied to f .

3.1.3 Recurrent Architectures

The recurrent network is a class of neural networks that enables handling of sequential data such as text [202], [201], image captioning [203] and timeseries [204]. A traditional FCN assumes that there is no dependency between successive data points which is unequivocally false for many tasks. Recurrent networks therefore store information about the previous data point in the hidden state h_{t-1} which is used alongside the network input x_t . This then means that every output is dependent on the previous calculations building up a notion of *memory*.

3.1.3.1 Recurrent Neural Network

The main recurrent neural network (RNN) formulation is also known as the Elman network [205], which is one of the simplest and most prevalent recurrent architectures. For each element in the input sequence, each recurrent unit computes the function

$$h_t = \tanh(x_t W_{xh}^T + h_{t-1} W_{hh}^T + b_h), \quad (3.6)$$

where:

- h_t is the hidden state at time t ;
- x_t is the input at time t ;
- h_{t-1} is the hidden state of the previous layer at time $t - 1$ or the initial hidden state at time $t = 0$; and
- W and b are parameter matrices and vectors (weights and biases).

The recurrent unit internal structure can be seen in Fig. 3.3. It then follows that the RNN is constructed of $N_{\text{RNN}} \in \mathcal{N}$ recurrent units where the hidden state is propagated through the network with y_t and h_t coming from the final unit.

3.1.3.2 Long Short-Term Memory

The long short-term memory (LSTM) [206] approaches the recurrent problem in a similar manner to the RNN but with a substantial improvement. Both architectures propagate the hidden state through each subsequent layer whereas only the LSTM gives each recurrent unit its own hidden state (known as the cell state). Along with the more computationally effective architecture, the LSTM overcomes the well-known *vanishing gradient problem* [207] that RNNs suffer from. This problem is a direct consequence of using gradient-based backpropagation (Sect. E.1.4) because the internal structure

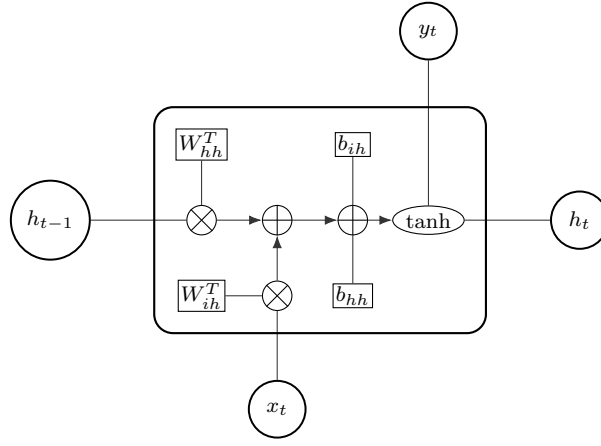


Figure 3.3: The RNN unit internal structure from Eqn. 3.6.

has to *unroll* for training during a process called back propagation through time. The computationally effectiveness of the LSTM, and the name, therefore comes from its ability to store information for a long time through the use of the cell states.

For each element in the input sequence, each layer computes the functions

$$\begin{aligned}
 i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \\
 f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \\
 g_t &= \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g), \\
 o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \\
 c_t &= f_t \odot c_{t-1} + i_t \odot g_t, \\
 h_t &= o_t \odot \tanh(c_t),
 \end{aligned} \tag{3.7}$$

where:

- h_t is the hidden state at time t ;
- c_t is the cell state at time t ;
- x_t is the input at time t ;
- h_{t-1} is the hidden state of the previous layer at time $t - 1$ or the initial hidden state at time $t = 0$;
- i_t , f_t , g_t , and o_t are the input, forget, cell, and output gates respectively;
- σ is the sigmoid function;
- $\odot$ is the hadamard product; and
- W and b are parameter matrices and vectors (weights and biases).

The LSTM unit internal structure can be seen in Fig. 3.4.

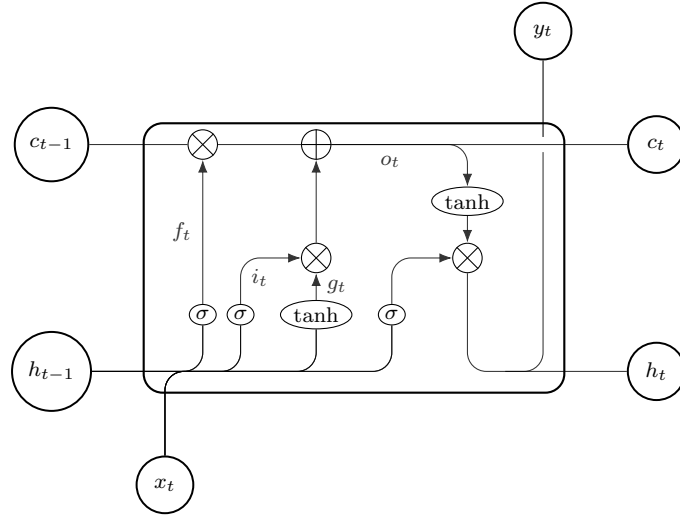


Figure 3.4: The LSTM unit internal structure from Eqn. 3.7.

3.1.3.3 Gated Recurrent Unit

Whilst the LSTM architecture effectively deals with the vanishing gradient problem, and is well suited to capture long-term dependencies, it is very slow to train and expensive to inference. The LSTM has eight weight matrices, four bias terms, eleven matrix multiplications, one memory cell and four gating units. Cho *et al.* [208] identified this and proposed the much simpler gated recurrent unit (GRU) which still manages to outperform the LSTM whilst being much more lightweight. This is corroborated by Cahuantzi *et al.* [209] who found that the GRU was faster to train and outperformed LSTM for lower-complexity strings. In contrast to this result it was also found that the LSTM outperformed the GRU for higher-complexity strings and were faster to train. It is therefore evident that the LSTM and GRU are suited for different applications and must be carefully assessed before deciding which recurrent architecture to select.

For each element in the input sequence, each layer computes the functions

$$\begin{aligned}
 r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r), \\
 z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z), \\
 n_t &= \tanh(W_{xn}x_t + b_{xn} + r_t \odot (W_{hn}h_{t-1} + b_{hn})), \\
 h_t &= (1 - z_t) \odot n_t + z_t \odot h_{t-1},
 \end{aligned} \tag{3.8}$$

where:

- h_t is the hidden state at time t ;
- c_t is the cell state at time t ;
- x_t is the input at time t ;
- h_{t-1} is the hidden state of the previous layer at time $t - 1$ or the initial hidden state at time $t = 0$;

- r_t , z_t , and n_t are the reset, update, and new gates respectively;
- σ is the sigmoid function;
- $\odot$ is the hadamard product; and
- W and b are parameter matrices and vectors (weights and biases).

The *gated* aspect of the GRU comes from the update gate z_t in Eqn. 3.8, which selects whether to update the hidden state h_t , whilst the reset gate r_t decides whether the previous hidden state is ignored. These gates are visible in Fig. 3.5, which shows the internal structure of the GRU.

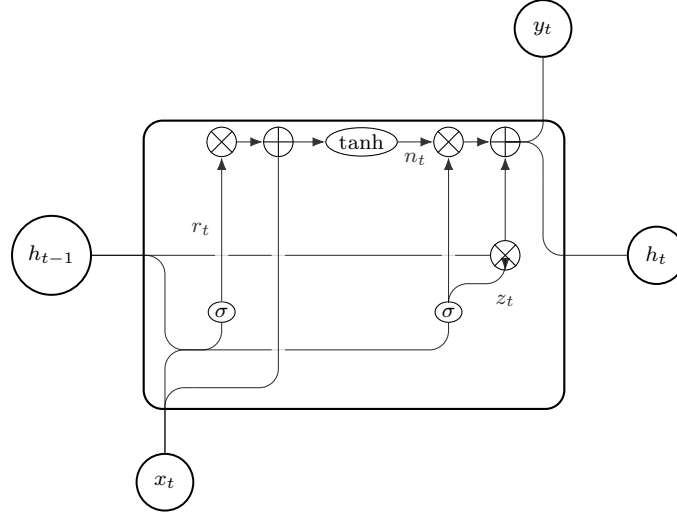


Figure 3.5: The GRU unit internal structure from Eqn. 3.8.

3.1.4 Attention

Attention is a useful machine learning technique that assigns importance to inputs. Typically the weights within a model are constant during the forward pass until updated during training leading to the notion of persistent weights. Through the use of attention this is changed by applying soft weights which are weights that change with every forward pass. This architecture is useful for sequential data types where it is unclear which entry is important to the understanding the context. This line of reasoning was the foundation for the sequence-to-sequence models.

Sequence-to-sequence models were developed for the use in language translation where the semantic differences make it difficult to encode which words are critical for understanding. To this effect the additive attention was introduced by Bahdanau *et al.* [210], which, when combined with a GRU, showed better performance than non-attention methods. The additive attention matrix is calculated by

$$\text{Attention}(Q, K, V) = \text{softmax}(\tanh(W_Q Q + W_K K)) V, \quad (3.9)$$

where W_Q and W_K are learnable matrices of parameters.

The use of the softmax function,

$$\text{softmax}(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_{j=0}^K \exp(x_j)}, \quad (3.10)$$

is imperative to ensure data integrity when applying the attention matrix. softmax converts a vector of K numbers into a probability distribution of K possible outcomes. That is, it bounds each entry in the attention matrix to the range $[0, 1]$.

The multiplicative attention was introduced by Luong *et al.* [211] and is given by

$$\text{Attention}(Q, K, V) = \text{softmax}(QW_QK^T)V, \quad (3.11)$$

where W_Q is a learnable matrix of parameters. This formulation is similar to that of Eqn. 3.9, however it benefits greatly from highly optimised matrix multiplication code and is more space-efficient in practice. The multiplicative attention was coupled with the LSTM and also showed performance increases in the field of language translation.

Whilst multiplicative attention has performance benefits over additive attention, Britz *et al.* [212] found that the former had better inference performance. Vaswani *et al.* [213] overcame this by introducing the scaled dot-product attention, which applies a scaling term $\frac{1}{\sqrt{d_k}}$ to give

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (3.12)$$

This prevents the dot-product from growing large in magnitude and moving the softmax into regions with extremely small gradients. Unlike Eqn. 3.9 or Eqn. 3.11 the scaled dot-product attention does not have any learnable parameters. This is intentional as Vaswani *et al.* [213] further introduced the multi-headed attention, which is the application of Eqn. 3.12 a h amount of times giving

$$\begin{aligned} \text{MultiHeadedAttention}(Q, K, V) &= \text{concat}(\text{head}_1, \dots, \text{head}_h)W_O, \\ \text{head}_i &= \text{Attention}(QW_{Qi}, KW_{Ki}, VW_{Vi}), \end{aligned} \quad (3.13)$$

where W_{Qi} , W_{Ki} , W_{Vi} , and W_O are learnable matrices of parameters. The multi-head attention was then coupled with the novel transformer architecture, introduced in the same work, resulting in massive gains in terms of evaluation performance, and training and inference speeds.

3.1.4.1 Self-Attention

A special case of attention is self- or intra-attention [214], [215]. This is where the query, key, and value vectors are the same input vector giving

$$\text{SelfAttention}(V) = \text{Attention}(V, V, V). \quad (3.14)$$

Similar, and crucial for the encoder of the transformer architecture from Vaswani *et al.* [213], is the multi-headed self-attention (MHSA) given by

$$\text{MultiHeadedSelfAttention}(V) = \text{MultiHeadedAttention}(V, V, V). \quad (3.15)$$

3.1.4.2 Convolutional Self-Attention

A slightly different approach must be taken for image data types due to the weight matrices in the aforementioned attention models being directly related to the input size. Xu *et al.* [216] uses Eqn. 3.9 on the text embeddings over the input sequence to improve the performance of the text-to-image model. This, whilst effective, only avoids the large matrix problem and does not directly address it. Wang *et al.* [217] therefore proposes the CSA which is computationally efficient for large batched inputs with many channels. In this work CSA is applied to extract kinematic features over temporal images (a video stream). Zhang *et al.* [218] and Parmar *et al.* [219] corroborate this by applying CSA to image generation.

CSA is defined as follows:

$$\begin{aligned} f_{\kappa}(x) &= W_{\kappa}x, \\ s_{ij} &= f_f(x_i)^T f_g(x_j), \\ \beta_{j,i} &= \frac{\exp(s_{ij})}{\sum_{i=0}^N \exp(s_{ij})}, \\ o_j &= f_v \left(\sum_{i=0}^N \beta_{j,i} f_h(x_i) \right), \\ y_i &= \gamma o_i + x_i, \end{aligned} \quad (3.16)$$

where:

- $x \in \mathbb{R}^{C \times W \times H}$;
- $W_g \in \mathbb{R}^{\bar{C} \times C}$;
- $W_f \in \mathbb{R}^{\bar{C} \times C}$;
- $W_h \in \mathbb{R}^{\bar{C} \times C}$;
- $W_v \in \mathbb{R}^{C \times \bar{C}}$;
- $o = (o_1, o_2, \dots, o_N) \in \mathbb{R}^{C \times W \times H}$;
- $\bar{C} = C/k$ where $k \in \mathbb{N}$; and
- γ is a learnable parameter initialised as 0.

The internal structure of the CSA is given in Fig. 3.6

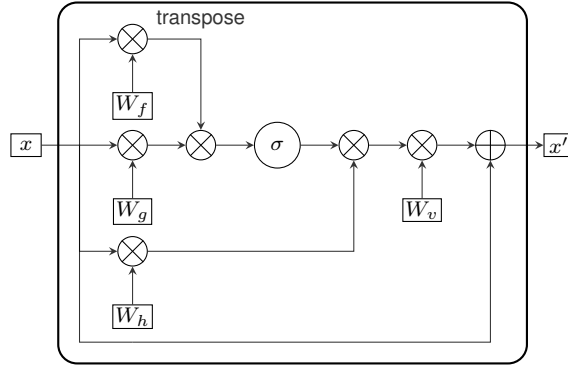


Figure 3.6: The CNN self-attention unit internal structure.

3.2 Auto-Encoder

The AE is a type of machine learning architecture that learns efficient projections of multidimensional data to a latent representation. This projection is achieved through the encoder-decoder architecture where the encoder applies the transformation $E_\phi : x_t \mapsto z_t$, and the decoder estimates the reconstruction through $D_\theta : z_t \mapsto \hat{x}_t$. The encoder, parameterised by ϕ , and the decoder, parameterised by θ , are then jointly learned through the reconstruction loss. Work from Zhou *et al.* [220] showed that this joint learning method outperforms greedy layer-wise training. It further benefits the training process by removing any supervised learning elements making this a purely self-supervised technique.

The reconstruction loss is the L2 norm (also known as the euclidean distance) given by

$$\mathcal{L}(x, \hat{x}) = ||\hat{x} - x||_2. \quad (3.17)$$

However, alternative loss functions like the mean-square error or cross-entropy loss are also useable here. The choice of loss function is highly dependent on the application.

To find the optimal AE Eqn. 3.17 is minimised through

$$\phi^*, \theta^* = \arg \min_{\phi, \theta} \mathbb{E}_{x \sim \mathcal{D}} [\mathcal{L}(x, D_\theta(E_\phi(x)))], \quad (3.18)$$

where $\mathcal{D}$ is the training dataset and $x \sim \mathcal{D}$ represents random sampling from said dataset giving ϕ^* and θ^* .

3.2.1 Recurrent Auto-Encoder

Whilst standard applications of AEs employ CNNs (Sect. 3.1.2) or FCNs (Sect. 3.1.1), it is often desirable to use recurrent architectures like the RNN (Sect. 3.1.3.1), LSTM (Sect. 3.1.3.2), or GRU (Sect. 3.1.3.3) for sequence data types.

Recurrent architectures are characterised by the inclusion of information (the hidden state) from the previous set of data. This can be applied to the AE in two ways to get a RAE:

1. *Full Sequence*: In this formulation the entire input sequence is passed to the RAE without any hidden state information. This is how the language translation from Bahdanau *et al.* [210] and Luong *et al.* [211] is handled. This approach allows preprocessing on the entire input using methods like CNNs or attention. However, the computational cost of recurrent auto-encoders grows with $\mathcal{O}(n)$ making it intractable for large datasets in a real-time setting.
2. *Partial Sequence*: This approach inputs the partial sequence, such as the information gathered since the last pass, as well as the previous hidden state [221]. A drawback of this approach is that preprocessing of the data is much more limited due to the lack of a complete dataset. However, this runs in $\mathcal{O}(1)$ meaning that this formulation is more suitable for real-time applications.

Practically these two methods behave identically if the above considerations are taken into account.

This *RNN Encoder-Decoder* architecture was first proposed by for language translation [208] and differs slightly from the RAE. Firstly, language translation is a supervised learning problem where the input language has a label which corresponds to the translated sentence in a different language. Secondly, given the semantic differences in languages it must be assumed that the translated sentence is not necessarily of the same length. Both of these considerations require complex solutions within language translation, the latter requires complex alignment models, but are a moot point for the RAE formulation. The estimation of the input sequence by the decoder overcomes the requirement for labels (RAE is a self-supervised technique) and the need for alignment models because x and $\hat{x}$ must be the same size.

In the encoder, each input x_k at step k in the series is fed into the encoder to get the latent representation

$$z_k = E_\phi(x_k, h_{k-1}^\phi), \quad (3.19)$$

where $z_k = h_k^\phi$ and h_{k-1}^ϕ is the hidden state of the recurrent encoder at the previous step. The decoder then reconstructs the *entire* series up to k through

$$\hat{x} = \{D_\theta(z_k, h_{i-1}^\theta) : i \in [0, k]\}. \quad (3.20)$$

Eqn. 3.17 is updated to account for the multi-step reconstruction:

$$\mathcal{L}_k(x, \hat{x}) = \frac{1}{k} \sum_{i=0}^k \|\hat{x}_i - x_i\|_2. \quad (3.21)$$

Eqn. 3.18 is likewise modified to reflect the new series formulation:

$$\begin{aligned}\phi^*, \theta^* &= \arg \min_{\phi, \theta} \mathbb{E}_{x \sim \mathcal{D}, k \sim (0, \dim(x))} [\mathcal{L}_k(x_{0:k}, \hat{x}_{0:k})], \\ z_{0:k} &= \left\{ E_{\phi}(x_i, h_{i-1}^{\phi}) : i \in [0, k] \right\}, \\ \hat{x}_{0:k} &= \left\{ D_{\theta}(z_k, h_{i-1}^{\theta}) : i \in [0, k] \right\}.\end{aligned}\tag{3.22}$$

The order of reconstruction within the decoder is unimportant and can either be forward [221] or backward [208]. This is because the entire dataset from 0 to k must be reconstructed regardless of order meaning that the latent representation must contain all the information required to do this effectively.

3.3 Reinforcement Learning

RL is a subset of ML concerned with learning an optimal policy, also referred to as the agent, to make sequences of decisions by interacting with an environment. The agent observes the current state of the environment, then selects and executes an action, and finally receives a scalar reward that correlates to the efficacy of the action taken. This loop continues over time, which is the basis of the RL framework.

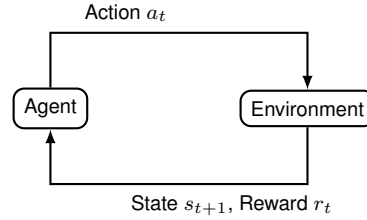


Figure 3.7: Agent-environment interaction loop in reinforcement learning. At each time step, the agent receives a state from the environment, takes an action, and receives a reward and new state in return.

As seen in Fig. 3.7, the interaction between agent and environment is cyclical: the agent selects an action $a_t \in \mathcal{A}$ based on the current state $s_t \in \mathcal{S}$, receives a reward $r_t \in \mathbb{R}$, and transitions to a new state s_{t+1} according to the environment dynamics.

This section introduces fundamental concepts in RL, including MDP, followed by detailed descriptions of two prominent RL algorithms that are used in this thesis: proximal policy optimisation (PPO) [222] and soft actor-critic (SAC) [223].

3.3.1 Markov Decision Process

The MDP is a mathematical construct where the next state depends only on the current state and action, not the history of past states. This formulation enables the modelling of interactions between an agent and a stochastic environment [224], [225]. This MDP framework is used to formalise decision-making in environments where outcomes are

partly random and partly under the control of an agent. A MDP is a tuple defined as $\langle S, A, p, r, \gamma \rangle$ where:

- S is the set of all valid states;
- A is the set of all valid actions;
- $p(s'|s, a) \in [0, 1]$ is the probability of transitioning into state s_{t+1} from current state s_t and by taking action a_t ;
- $r : S \times A \rightarrow \mathbb{R}$ is the reward emitted by the environment on each transition; and
- $\gamma \in [0, 1)$ is the discount factor controlling the present value of future rewards.

The objective in RL is to learn a policy $\pi(a|s)$, the distribution over actions given a state which maps states to actions either deterministically or probabilistically, that maximises the expected cumulative discounted reward

$$J(\pi) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \quad (3.23)$$

Following conventions from Haarnoja *et al.* [223], $\rho_{\pi}(s_t)$ and $\rho_{\pi}(s_t, a_t)$ describe the state and state-action marginals of the trajectory distribution induced by a policy $\pi(a_t|s_t)$.

3.3.2 Proximal Policy Optimisation

PPO is an on-policy policy gradient method [222] that has been widely used successfully in many different applications from playing video games [226] to flying racing drones in real time [171]. PPO attempts to estimate the maximum safe update step when learning the policy through a clipped objective. This helps training by allowing the algorithm to safely explore the parameter space whilst avoiding a policy collapse.

In PPO, the advantage function $A^{\pi}(s_t, a_t)$ plays a crucial role by quantifying how much better a specific action is compared to the average action in a given state. It is defined as the difference between the action-value function $Q^{\pi}(s_t, a_t)$ and the state-value function $V^{\pi}(s_t)$:

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t). \quad (3.24)$$

This gives a measure of how good taking action a_t from state s_t is and then following policy π forever relative to sampling an action from $\pi(\cdot|s_t)$ and following that forever. The advantage function helps reduce variance during policy updates by centring the scale of updates around zero, emphasising actions that outperform the baseline $V^{\pi}(s_t)$. This centring stabilises training, allowing the policy to focus on meaningful action differences rather than absolute rewards

PPO implements the clipping of the update step through the clipped surrogate objective. This is defined as

$$\mathcal{L}_{\theta_k, s_t, a_t}^{\text{CLIP}}(\theta) = \mathbb{E}_{s, a \sim \pi_{\theta_k}} [\min(r_t(\theta) A^{\pi_{\theta_k}}(s_t, a_t), \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A^{\pi_{\theta_k}}(s_t, a_t))], \quad (3.25)$$

where ϵ is a hyperparameter. The first term, $r_t(\theta) A^{\pi_{\theta_k}}(s_t, a_t)$, refers to conservative policy iteration from Kakade *et al.* [227], which aims to improve the policy in a more uniform manner over the state-space whilst performing a conservative policy update. However, without a constraint this would lead to a large policy update. The second term, $\text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A^{\pi_{\theta_k}}(s_t, a_t)$, thus modifies the surrogate objective to prevent $r_t(\theta)$ from moving outside of the interval $[1 - \epsilon, 1 + \epsilon]$. Taking the minimum of the two terms results in a pessimistic bound of the unclipped objective. This means that the probability ratio,

$$r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, \quad (3.26)$$

is only included when it would make the objective worse. If $A^{\pi} > 0$ then $\mathcal{L}^{\text{CLIP}}$ is increasing until $\pi_{\theta}(a_t|s_t) > (1 + \epsilon)\pi_{\theta_k}(a_t|s_t)$ because the action becomes more likely. After this, $\mathcal{L}^{\text{CLIP}}$ has a maximum value of $(1 + \epsilon)A^{\pi_{\theta_k}}(s_t, a_t)$ meaning the policy does not gain anything by going too far away from the previous policy. Conversely, if $A^{\pi} < 0$ then $\mathcal{L}^{\text{CLIP}}$ is increasing until $\pi_{\theta}(a_t|s_t) < (1 - \epsilon)\pi_{\theta_k}(a_t|s_t)$ because the action becomes less likely. $\mathcal{L}^{\text{CLIP}}$ decreases until it reaches $(1 - \epsilon)A^{\pi_{\theta_k}}(s_t, a_t)$ and thus the policy, again, does not benefit from deviating too far from the previous policy. This behaviour is evident from Fig. 3.8 showing that $\mathcal{L}^{\text{CLIP}}$ is increasing until it reaches the saturation threshold.

While the clipped surrogate objective is a key component of PPO, the full algorithm combines Eqn. 3.25 with a value function loss and an entropy bonus to promote exploration. The complete PPO objective function is give by

$$\mathcal{L}_{\theta_k, s_t, a_t}^{\text{CLIP}+\text{VF}+\text{S}}(\theta) = \mathbb{E}_{s, a \sim \pi_{\theta_k}} [\mathcal{L}_{\theta_k, s_t, a_t}^{\text{CLIP}}(\theta) - c_1(V_{\theta}(s_t) - V_{\text{target}})^2 + c_2 S[\pi_{\theta}](s_t)], \quad (3.27)$$

where c_1 and c_2 are coefficients.

PPO applies generalised advantage estimation [228] in order to approximate the advantage from Eqn. 3.24 using the sampled trajectories. Generalised advantage estimation is defined as the infinite-horizon exponentially-weighted average of k -step estimators:

$$\begin{aligned} \hat{A}_t^{\text{GAE}(\gamma, \lambda)} &= \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}^V, \\ \delta_t^V &= r_t + \gamma V(s_{t+1}) - V(s_t). \end{aligned} \quad (3.28)$$

In practice, however, this is impractical as the number of steps taken in an epoch are

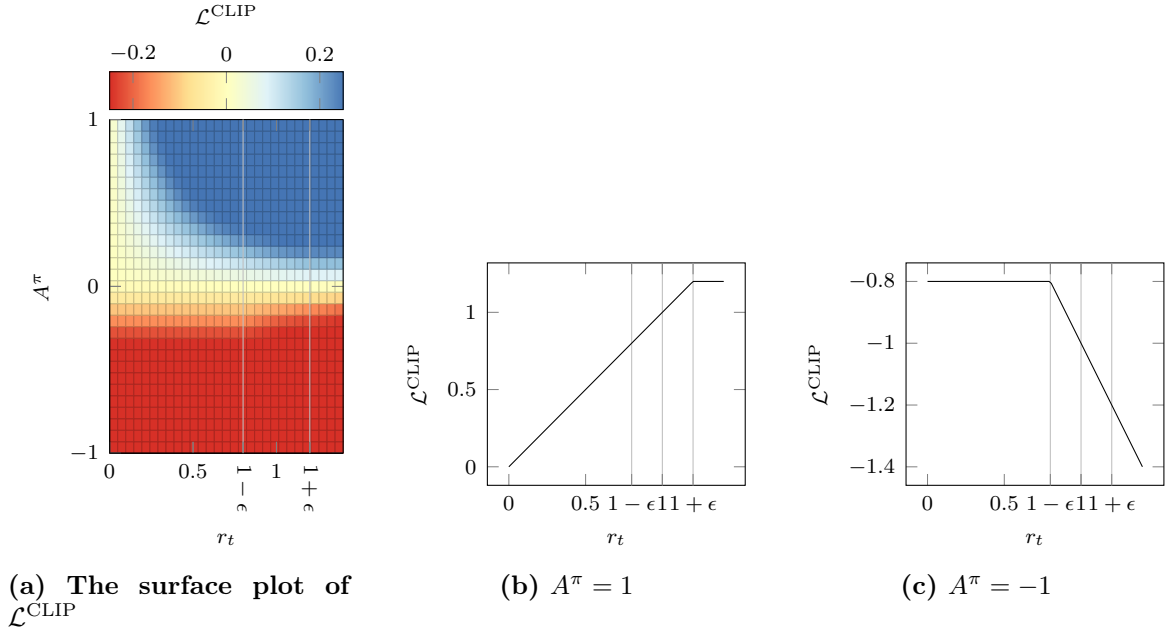


Figure 3.8: $\mathcal{L}^{\text{CLIP}}$ for a single timestep with $\epsilon = 0.2$ as a function of: A^π and r_t (a), and r_t (b) (c). The optimisation starts at $r = 1$ and is clipped in the bounds $[1 - \epsilon, 1 + \epsilon]$.

finite. The truncated version of the generalised advantage estimate is then

$$\hat{A}_t^{\text{GAE}(\gamma, \lambda)} = \sum_{l=0}^T (\gamma \lambda)^l \delta_{t+l}^V. \quad (3.29)$$

A from Eqn. 3.25 is then replaced with $\hat{A}$ to denote that it is an approximation.

3.3.3 Soft Actor-Critic

SAC is an off-policy RL algorithm that introduces the notion of *entropy maximisation* to learn a stochastic policy [223]. This maximises the trade-off between expected return and randomness in the policy. The intent of this is to efficiently handle the exploration-exploitation trade-off that is prevalent in RL to prevent policies from collapsing on local maxima.

Classical RL maximises the expected sum of rewards whereas SAC augments this objective function with the expected entropy of the policy,

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}} [-V_\phi(s_t)], \quad (3.30)$$

with

$$V_\phi(s_t) = \mathbb{E}_{a_t \sim \pi} [Q_\theta(s_t, a_t) - \alpha \log(\pi_\phi(a_t|s_t))]. \quad (3.31)$$

The soft Q-function parameters are updated using the soft Bellman residual

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}} \left[\frac{1}{2} Q_\theta(s_t, a_t) - \left(r(s_t, a_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p} [V_\theta(\mathbf{s}_{t+1})] \right)^2 \right], \quad (3.32)$$

where the value function is implicitly parameterised through the soft Q-function parameters through Eqn. 3.31. Haarnoja *et al.* [223] introduced a function approximator for the value function but Haarnoja *et al.* [229] found this to be unnecessary. Mnih *et al.* [230] highlights the benefits of using exponentially moving average of the soft Q-function weights, which lead to stabilised training, which replaces θ with $\bar{\theta}$ in Eqn. 3.32.

Haarnoja *et al.* [229] further uses two Q-functions within SAC to prevent the positive bias in the update steps, which is known to reduce performance [231], [232]. They are independently updated through Eqn. 3.31 and the lower of the two values is used such that

$$Q_{\theta}(s_t, a_t) = \min_{i \in \{1,2\}} Q_{\theta,i}(s_t, a_t). \quad (3.33)$$

3.3.3.1 Temperature Parameter

The temperature parameter, α , in the original implementation of SAC from Haarnoja *et al.* [223] is constant. However, Haarnoja *et al.* [229] proposes that α is a learnable parameter in order to automatically regulate the entropy. This is important as the optimal α is different for every policy and since the policy changes over time it is incredibly hard to find a constant α that is sufficient. The gradient of α is computed using the objective function

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi_t} [-\alpha \log \pi_t(a_t | s_t) - \alpha \bar{\mathcal{H}}], \quad (3.34)$$

where $\bar{\mathcal{H}}$ is the target entropy and is set to $\dim(\mathbf{a})$.

By automatically tuning α , SAC can maintain an appropriate balance between exploration and exploitation throughout the learning process, adapting to the complexity of the task and the stage of learning.

3.3.4 Chapter Summary and Conclusions

This chapter serves as comprehensive technical foundation for the ML architectures and methods utilised throughout this thesis. Through the introduction of core models, like the FCN and CNN, as well as specialised versions, like the CSA, a standard nomenclature has been established. Furthermore, the introduction of RL algorithms in Sect. 3.3 provides the necessary background for the search path planning frameworks developed in subsequent chapters.

Chapter 4

Behaviour-based Probability Distribution Map Generation

4.1 Section Overview

Predicting where a LP could be found is crucial to understand the search portion of a WiSAR scenario. To improve the precision and efficiency of these predictions, simulated agents can be created to emulate the behaviour of the LP. Within this chapter, an agent-based model is introduced, which is designed to replicate diverse psychological profiles of LPs, allowing these agents to navigate real-world landscapes while making decisions autonomously without needing location-specific training. The PDM depicting the potential location of the LP emerges through a combination of Monte Carlo simulations and mobility-time-based sampling. The model is validated using real-world WiSAR data to train a Gaussian process (GP) model. This allows the generalisation of the data to sample initial starting points for the agents during validation. Comparative analysis with historical data showcases promising outcomes relative to alternative methods. This work introduces a flexible agent that can be employed in WiSAR operations, offering adaptability across various geographical locations.

This chapter closely follows the work of the thesis author in the peer-reviewed journal paper ‘Predictive Probability Density Mapping for Search and Rescue Using an Agent-Based Approach With Sparse Data’ [44], as well as including information, also by the thesis author, from the peer-reviewed conference paper ‘GIS Data Driven Probability Map Generation for Search and Rescue Using Agents’ [46]. This is further clarified in Sect. 4.6.1.2.

In this chapter related PDM generation algorithms are discussed in Sect. 4.3, followed by the methodology used in this chapter in Sect. 4.4. The design of the experiment is introduced in Sect. 4.5. Results are presented and discussed in Sect. 4.6. Finally, Sect. 4.7 concludes this chapter and discusses future work.

4.2 Introduction

WiSAR of vulnerable people is unfortunately a common task for PSASU and SMR. Organizations like the Centre for Search and Rescue and the Grampian Police carry out research and training in areas related to WiSAR, and while their training and published papers offer a valuable resource to the people responsible for finding a LP, they primarily focus on land-based search [30], [31]. This is a slow and methodical process that would undoubtedly benefit from the assistance of autonomous UAV. As a result of rapid advances in this sector over the last decades, UAVs capable of carrying high-powered sensor payloads have become cheaper and more accessible than ever before [8], [9]. Consequently, several concept evaluation trials for measuring the efficacy of incorporating UAVs into the search process have recently been undertaken in Scotland [9].

The mission profile for UAVs in a WiSAR setting is solely for search. This requires finding the LP as quickly as possible and relaying this information back to the search leader so that they can organise the rescue operations. The search further breaks down into prediction, flying, and sensing. Flying and sensing are done in real-time [233], but the prediction of the LP location can be done prior to deployment. This prediction is called the PDM and would then be used by the search leader to allocate their available resources to where the LP is the most *likely* to be found.

One key piece of information in the prediction process is using data to inform and validate the models. Historical data collected from WiSAR cases agree that significant behavioural profiles exist [24], [26], [30], [105], [234]. This means that a solo hiker behaves differently from an older adult with dementia when moving across a landscape, which ultimately affects the location found. Sect. 2.1 discussed how PDMs can be highly customised on a per-location and per-person basis through the use of this a priori information along with other location data, resulting in better PDMs.

A crucial constraint is the lack of available data. Even the largest WiSAR database ISRID [52] has only 145,000 data points [105], with many not including geospatial information. PDMs need large amounts of data that cannot be gathered effectively through simple data logging by various bodies. This drives the need for models to generate the data so that the likes of SMR and PSASU always have readily available PDMs.

To generate the PDM, Heintzman *et al.* [41] implements a 6D behavioural vector model, where each strategy emulates a self-rescue technique. These are then randomly selected based on a learned weighting vector at every time step. This was then further developed by Hashimoto *et al.* [42] using a leave-one-out analysis to further improve the accuracy of the resultant behaviour vector compared to the real-world data from Koester [24]. However, this vector has to be learned for every new location as results differed substantially between trials.

It is therefore hypothesised that by using a similar set of data as from Hashimoto

et al. [42], as well as the location found data, a more generalised model can be created that does not need training. The find location data from Perkins *et al.* [30] are used to drive four different behaviours, each with the intent of arriving at a unique target location. The artificially generated data is then compared to historical data to analyse the results. Furthermore, this chapter presents a supervised machine-learning approach to the model validation stage by up-sampling sparse historical data. By leveraging real-world data, the new proposed approach allows for a more robust assessment of the models' effectiveness, potentially leading to significant improvements in real-world applications.

4.3 Related Work

As outlined in Sect. 2.1.2.2, Hashimoto *et al.* [88] and Heintzman *et al.* [41] developed a 6D behaviour vector algorithm to simulate an agent navigating a 2D grid. Each time step involves selecting a behaviour through a learned weighting vector. This chosen behaviour assigns probabilities to the agent's current cell and its eight surrounding cells. For example, *staying put* would set the surrounding cells to 0 and the current cell to 1, whereas *random walk* sets all cells to $\frac{1}{9}$. The agent subsequently picks one of the nine possible cells based on these probabilities and moves to it. This iterative process is then repeated. The weighting vector is constructed by evaluating all permutations (in steps of $\frac{1}{6}$) and comparing the model data against the statistics from Koester [24]. This was then further developed in Hashimoto *et al.* [42] using a leave-one-out analysis to further improve the accuracy of the resultant behaviour weighting vector compared to the real-world data from Koester [24]. However, this vector has to be learned for every new location as results differed substantially between trials.

Drexel *et al.* [40] takes a different approach, as described in Sect. 2.1.1.4, which focuses on the network of paths and roads within the search area. By applying network theory, Tobler's mobility function — introduced in Eqn. 2.1 — captures the LP's mobility. This method, however, fails to encapsulate the behaviour of the LP if they leave the trail or are never on one.

Thus, the method outlined in this chapter combines the behaviour-based approach from Hashimoto *et al.* [42], and a modification of the network-based approach from Drexel *et al.* [40] to further incorporate the physical information about the landscape. Unlike Hashimoto *et al.* [42], however, behaviours are segmented by final find location rather than by movement types, which aims to prevent the per-location training.

4.4 Methods

The method outlined in this section, has two parts. In Sect. 4.4.1, the Monte Carlo path generation using the LP surrogate is introduced. The sampling of the paths to generate locations found is then developed in Sect. 4.4.2.

The data relating to lost person behaviour is primarily sourced from Perkins *et al.* [30] with missing data being taken from Koester [24]. Both sources consider multiple categories of LP, such as *Child 10–12* or *Climber*. However, one of the largest datasets reported by both belongs to the *Hiker (solo)* profile; defined as an adult hiking alone. All parameters used in this chapter are based on this category.

4.4.1 Path Generation

At its core, every simulation run is a behaviour traversing the landscape from a starting point until the path length goes above the maximum path length $D_{\max}$. At this stage, the agent does not tire. Whilst this is an unrealistic assumption by itself, the LP movement times are introduced during the sampling stage defined in Sect. 4.4.2.

The LP is modelled as an agent navigating a 2D grid with square cells of shape $5\text{ m} \times 5\text{ m}$. Its viewpoint sits at a constant 1.6 m above the surface, allowing it to see over smaller obstacles like rocks or long grass. At each time step, the agent moves to the next cell, and the total distance increases accordingly until it reaches the termination distance $D_{\max}$. The simulation will then terminate at the next distance check.

The starting points are then sampled from the bivariate Gaussian probability distribution function $PDF_{start}(\mathbf{x})$, which is defined as

$$PDF_{start}(\mathbf{x}) = \frac{\exp\left\{\left[-\frac{1}{2}(\mathbf{x} - \mu)^T \sigma^{-1}(\mathbf{x} - \mu)\right]\right\}}{\sqrt{4\pi^2 \det(\sigma)}},$$

$$\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{bmatrix},$$

$$\mu = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix},$$
(4.1)

centred around a given start location μ . In a real WiSAR mission, this would be the PLS. The purpose of PDF_{start} is to model the uncertainty, through σ , of the reported PLS. SMR reports a PLS using the 6-digit Ordnance Survey National Grid reference, which reduces the British Isles into $100 \times 100\text{m}$ cells. A PLS described by this system has a $\pm 100\text{m}$ error in x and y , with a total magnitude of the error being $\pm 141.42\text{m}$. This would result in a variance of $\sigma_{xx} = \sigma_{yy} = 10,000$. To validate this model, Sect. 4.5 outlines how μ is generated based on real-world data.

4.4.1.1 Lost Person Behaviour

To simulate the behaviour of an LP, four behaviours were created. Each behaviour corresponds to a land cover category the agent will attempt to travel to. Throughout the simulation, the agent keeps the same behaviour. The intent of this is to emulate an LP over a large number of data points. From the data shown in Tbl. 4.1, a behaviour can be selected $P\%$ of the time. *Building*, *trees* and *water* were given individual be-

haviours, but *linear feature* and *travel aid* were merged under Head2Paths. This was done because *linear feature* is defined as being a *stream/ditch* or *wall/fenced line*, and these features often follow paths and roads. As well as this, a *stream* is already being handled by the *water* behaviour (as further outlined below). Finally, Hyp. 4.1 (with later empirical evaluation) was employed to generalise the *open-ground* find data.

Table 4.1: Land cover category location found data for a *Hiker (solo)* from Perkins *et al.* [30] and the associated behaviours.

Land cover category	%	Behaviour
Open Ground	40.8	n/a
Travel Aid	25.4	Head2Paths
Building	23.1	Head2Buildings
Linear Feature	6.9	Head2Paths
Trees	3.1	Head2Trees
Water	0.8	Head2Water

Hypothesis 4.1. *The open-ground land cover category location found data naturally results from an LP trying to navigate to the other possible locations.,*

Table 4.2: GIS maps used in this chapter.

Map	Description	Source
Digital Elevation	LiDAR composite Digital Terrain Model	[235]
Land Cover ID	Land cover ID map	[236]
Cumulative Catchment Area	The area draining through a point	[237]
Water Surface Type	Most hydrologically significant surface type	[237]
Water Outflow Direction	Shows the direction of the overland flow	[237]
Road Network	Road and path network	[238]

The GIS maps utilised in this study are presented in Tbl. 4.2. It is crucial to clarify that, although land cover ID and land cover category may seem similar, they differ. The former is exclusively employed within the viewshed-based behaviours, as outlined in Sect. 4.4.1.1, and is sourced from Morton *et al.* [236]. On the other hand, the latter influences the meta-behaviour of the LP and originates from Koester [24] and Perkins *et al.* [30].

Head2Water The methodology for the Head2Water behaviour was left unchanged from Ewers *et al.* [46] with the usage of a vector field following the model. The GIS data outlined in Tbl. 4.2 shows that three of the six maps (cumulative catchment area, water surface type, and water outflow direction) contain hydrological information, with the water outflow direction map used as a direct input by the vector field follower. To increase the fidelity of the model, the cumulative catchment area and the water surface type can further be used to force the agent to navigate around large bodies of water.

This was done by evaluating the position of the agent in the next time step, and if the water surface type at the future position was either *lake*, *sea*, or *river*, then a scaled probability of carrying out this step is calculated. This is achieved through

$$p(a) = 1 - \begin{cases} 1, & a \geq b \\ \frac{a}{b}, & \text{else} \end{cases}, \quad (4.2)$$

where $v \in [0, \infty)$ is the cumulative catchment area value, which describes the total number of cells that drain into that cell, in the next time step. The upper bounds of the input a is $b = 8000$ for this chapter and was manually tuned. Thus $p(a)$ is the scaled percentage chance for the agent to step into the body of water. If the check failed, then, the agent would walk around the obstacle by turning left or right with equal probability. In a real-world scenario, this equates to a hiker stepping over a small stream or encountering a lake and walking along its edges.

Head2Buildings and Head2Trees Using the land cover ID and digital elevation maps from Tbl. 4.2, the agent could *see* the landscape and act accordingly. Using a viewshed algorithm means that expensive ray-casting can be mitigated. Viewshed algorithms are popular in GIS [239] for applications such as radio tower positioning. The resultant map is a mask that outlines which areas within a given radius are visible to the observer. Any map can be easily analysed using an element-wise AND operation from this mask. The visible cells can then be analysed further. To differentiate between building- and tree-seeking behaviour, every possible land cover type is given a weight, and the maximum weights are the only ones considered in the map, which allows multiple map values to be treated as equals.

Then, the mean angle, $\bar{\theta}(\mathbf{p})$, to every c_i in the set of visible cells $\mathbf{c}$ is selected as the command direction, as seen below in Eqn. 4.3.

$$\bar{\theta}(\mathbf{p}) = \frac{1}{N_{vis}} \sum_{i=0}^{N_{vis}} \tan^{-1} \left(\frac{p_y - c_{i,y}}{p_x - c_{i,x}} \right), \quad (4.3)$$

where $\mathbf{p}$ is the current position, and N_{vis} is the number of visible cells.

The weights from Tbl. 4.3 were based on the impedance values from Doherty *et al.* [27], Frakes *et al.* [39], and the associated behaviour's intended land cover category location found. This is done to reflect the decision-making process that an LP might undergo whilst being lost. The core behaviour is encoded within the weights, nevertheless, subtle preferences between land cover IDs are also included. For example, the LP will avoid a bog at all costs whilst aiming for the arable and horticulture (i.e. farmland) if the core land cover ID target is not visible. Given more future data, these weightings can be created for individual profiles.

Table 4.3: The viewshed behaviour weights used in this chapter for the building and tree behaviour. Head2Water was included to show how this behaviour could be done without the data from Morris *et al.* [237].

Land cover ID	Head2Buildings	Head2Trees	Head2Water
Acid grassland	0.05	0.05	0.05
Arable and horticulture	0.25	0.25	0.25
Bog	0.01	0.01	0.01
Calcareous grassland	0.05	0.05	0.05
Fen, March, Swamp	0.03	0.03	0.03
Heather	0.06	0.06	0.06
Heather grassland	0.05	0.05	0.05
Improved grassland	0.05	0.05	0.05
Neutral grassland	0.10	0.10	0.10
Rock	0.05	0.05	0.05
Saltmarsh	0.05	0.05	0.05
Urban	0.40	0.20	0.20
Water	0.20	0.20	0.40
Woodland	0.20	0.40	0.20

Head2Paths The Head2Paths behaviour has two stages: *hierarchical nearest path search*, and *path network exploration with memory*. Both use a path graph where V is the set of nodes, and E is the set of edges such that $E \subseteq \{\{v_1, v_2\} | v_1, v_2 \in V \text{ and } v_1 \neq v_2\}$. Every edge in E is a representation of a physical path from v_i to v_j , where $v_i, v_j \in V$ (shortened to $e_{i,j}$ for brevity), that can be traversed both ways, making it an undirected graph. Likewise, the nodes $V \in \{v_0, v_1, \dots\}$ represents the junctions or ends of paths. Every edge $e_{i,j}$ has an associated physical path $path(e_{i,j})$ and a path type $type(e_{i,j})$. The path type values come from the road network map in Tbl. 4.2 which correlates to the size of the path. This path type can then be used to map an edge to an integer score

$$score(e_{i,j}) = \begin{cases} 10, & type(e_{i,j}) = \text{Major road} \\ 5, & type(e_{i,j}) = \text{Trunk road} \\ 2, & type(e_{i,j}) = \text{Path} \\ 4, & \text{else} \end{cases}. \quad (4.4)$$

Using the above-defined path graph, the hierarchical nearest path search then uses the geospatial information of the path; $path(e_{i,j})$ and $score(e_{i,j})$. On an evenly spaced grid $\mathbf{M}$ of points covering the search area, every $\mathbf{m} \in \mathbb{R}^2$ has k distinct nearest edges $E^{k-\text{nearest}} \subset E$. This subset is found by finding the Euclidean distance from $\mathbf{m}$ to the single closest location on every edge's path $point(e_{i,j}) \in \mathbb{R}^2$ and then selecting the k

closest ones. The weighted average angle $\bar{\theta}$ that the agent will move in is then

$$d_n = ||\mathbf{m} - \text{point}(E_n^{k-\text{nearest}})|| \quad (4.5)$$

$$c_n = \frac{\text{score}(E_n^{k-\text{nearest}})}{d_n} \quad (4.6)$$

$$\hat{u}_n = \frac{\text{point}(E_n^{k-\text{nearest}}) - \mathbf{m}}{d_n} \quad (4.7)$$

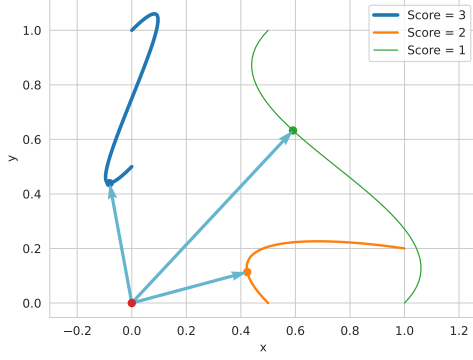
$$\bar{\theta} = \arctan \left(\sum_{n=0}^k c_n \hat{u}_n \right) \quad (4.8)$$

where c_n is the scaling term associated to $E_n^{k-\text{nearest}}$, and $\hat{u}_n$ is the unit vector in the direction from $\mathbf{m}$ to $\text{point}(E_n^{k-\text{nearest}})$. This process can be seen in Fig. 4.1, showing the influence that the scaling introduced by Eqn. 4.4 and Eqn. 4.6 has on the weighted average angle. As the agent traverses the landscape looking for a path, they follow the angle associated with their current position. The weighted nature of the paths in Eqn. 4.8 represents the agent being able to prioritise a major road over a small dirt path, but still giving precedence over a smaller, closer, walking path via the $\frac{1}{d}$ scaling in Eqn. 4.6.

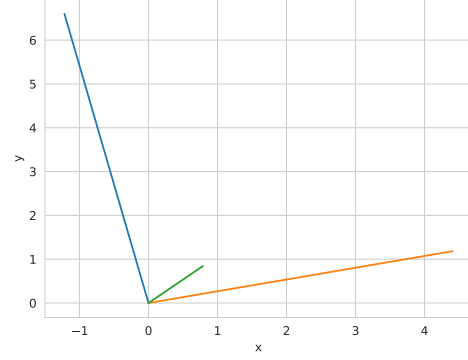
Once the agent reaches a path, the behaviour switches to traversing the path network, as seen in Fig. 4.2. At this switching position, $\mathbf{x}$, the nearest path is searched for, as outlined above, but with $k = 1$. Then, the point $\mathbf{x}$ is inserted into V as a new node v_{new} and two new edges, $e_{-}^{\text{nearest}} = \{v_{\text{new}}, v_i\}$ and $e_{+}^{\text{nearest}} = \{v_{\text{new}}, v_j\}$, are inserted into E and the associated physical paths undergo a similar process. The original path $\text{path}(e^{\text{nearest}})$ is cut at $\mathbf{x}$ into two pieces. One piece is reversed to start at $\mathbf{x}$, and the other is left as is, but a straight line from $\mathbf{x}$ to the path is inserted for completeness. These new edges are marked as non-traversable once the agent moves along them. The agent then begins traversing the path network from the node v_{new} by selecting randomly from e_{-}^{nearest} and e_{+}^{nearest} with equal probability. This physically represents a person coming across a path and either turning left or right.

At every subsequent step, the agent moves along the whole physical path $\text{path}(e_{i,j})$ entirely before the next maximum distance check is done. As the upper time limit is observed during the sampling stage, this does not impact the final result and improves performance by not requiring costly spatial computations on the paths. Once it reaches the next node, the agent randomly selects the next edge from the node's adjacency list, which is a finite graph representing the set of neighbours of that particular node. Every edge in the adjacency list is assigned a score of being selected. It is given a score of 0.1 if it has been visited in the last λ_{max} steps, or 1 if it has not. This vector of scores is normalised to give a probability vector, which is then used to select an edge randomly to traverse. This discourages the agent from backtracking, but does not altogether forbid it, which is crucial to prevent the agent from becoming isolated in a sub-graph of the network.

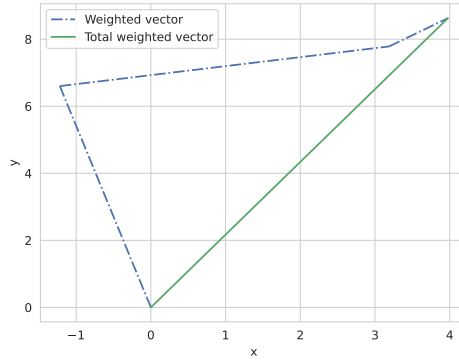
It should be highlighted that it is rare for a LP to come across a road, more likely



(a) Nearest points on 3 paths from (0,0) (red) where each path has an associated score representing $score(E^{k-\text{nearest}})$ where $k = 3$.



(b) Scaled vectors based on Eqn. 4.6 and Fig. 4.1a.



(c) Tip-to-tail stacked scaled vectors from Fig. 4.1b resulting in weighted average vector from which the angle is calculated.

Figure 4.1: The process of extracting the weighted average angle $\bar{\theta}$ from $E^{k-\text{nearest}}$ for a given position m on the grid.

for a less permanent path or a LP with dementia [24], and then to ignore it. This decision-making process is modelled by the fact that the other behaviours completely ignore paths while Head2Paths always follow them.

4.4.2 Sampling Of Paths

The simulation length has a termination distance of $D_{\text{max}} = 10\,000\text{ m}$, which gives a simulated time of 2.58 h, and accounts for over 95 % of LP scenarios when using the Perkins *et al.* [30] dataset. The endurance, or fatigue, of the LP can be modelled using the data from Fig. 4.3. This shows the two mobility time distributions found from historical data, which measure the amount of time the LP was moving [24], [30].

To fit the mobility time distribution, two functions were evaluated: the Weibull from Hashimoto *et al.* [88], modified to include an offset term to become

$$f(x, a, b, c)_{\text{weibull}} = c - e^{-\left(\frac{x}{a}\right)^b}, \quad (4.9)$$

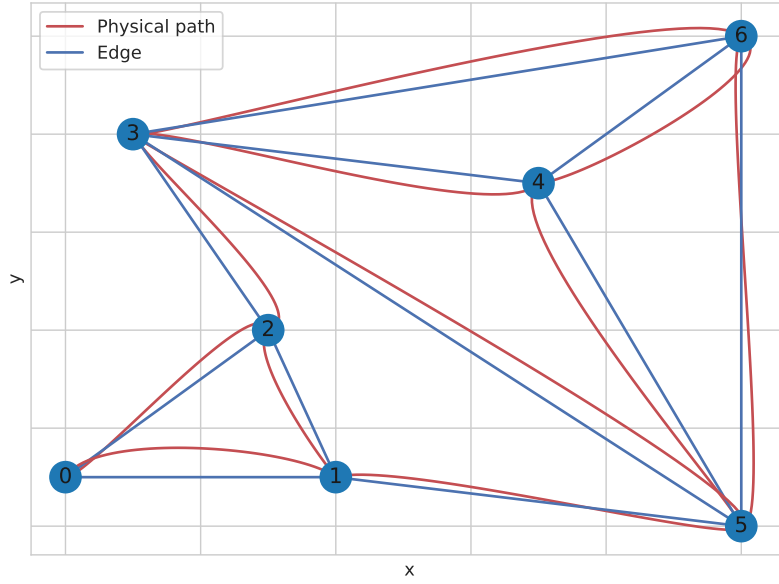


Figure 4.2: Example path network. If the agent moves from 0 to 2, the edge (0, 2) will have a score of 0.1 assigned to it whilst both (2, 3) and (2, 1) will have a score of 1. After normalising the scores, this gives a probability of 4.76 % to backtrack to 0.

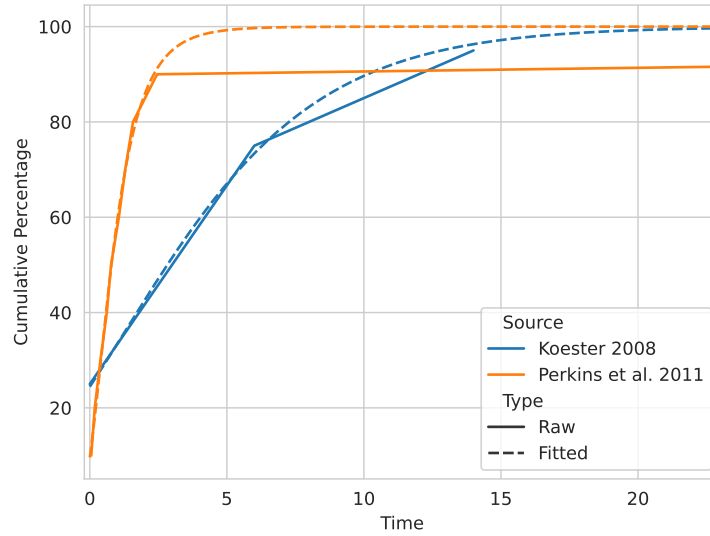


Figure 4.3: Discrete mobility time (hours) from both Koester [24] (n=232) and Perkins *et al.* [30] (n=132) along with their respective continuous log normal curve. Perkins *et al.* [30] provides distance from PLS data in kilometres, and was scaled by 3.87 km h^{-1} from Gast *et al.* [240].

and the log normal cumulative distribution function [95]

$$f(x, \mu, \sigma, c)_{\log \text{ normal}} = \frac{1}{2} \left(c + \operatorname{erf} \left(\frac{\ln x - \mu}{\sigma \sqrt{2}} \right) \right). \quad (4.10)$$

It was found that, by using the symmetrical Kullback-Leibler divergence function [94],

defined as

$$SKL(P||Q) = \sum p_i(x) \log \left(\frac{p_i(x)}{q_i(x)} \right) + \sum q_i(x) \log \left(\frac{q_i(x)}{p_i(x)} \right), \quad (4.11)$$

as a metric, the log normal function was the best fitting with a score of 0.0161 compared to 0.0494 for the Weibull.

To sample the location found from the N_{gen} generated paths from Sect. 4.4.1.1, M time samples were taken from Eqn. 4.10 for every path. A constant speed of 3.87 h km^{-1} , which is the average preferred walking speed of a hiker over rough terrain [240], is used to convert time to distance. A given path is then sampled at the given distances along it to generate the locations found. This is repeated for all N_{gen} paths to generate $N_{gen} \cdot M$ samples.

4.5 Design Of Experiment

The data supplied by Perkins *et al.* [30], Koester [24], and Grampian Police [31] combines generalised data from hundreds of LP cases to draw broad conclusions about the various LP profiles. Therefore, to draw similar conclusions from the methods outlined in Sect. 4.4, an adequate distinct number of PLS locations must be used for the data generation.

Whilst SMR provided a sample dataset for the Isle of Arran of historical rescue data, with ≈ 300 PLS locations, this position data is in the form of a 6-digit Ordnance Survey National Grid reference. This has a worst-case spatial aliasing of $\pm 141.42\text{m}$ due to its $100 \times 100\text{m}$ sized cells. Consequently, the resulting heatmap generated from this data has a cell size of 100 m , as seen in Fig. 4.4, with a primary hotspot where the mountain *Goatfell* on the Isle of Arran in Scotland is located. Any attempt to over-sample from this distribution to generate a larger, synthetic dataset for analysis would fail to produce meaningful spatial variety; the model would simply replicate existing grid-aligned clusters rather than simulating a continuous probability distribution.

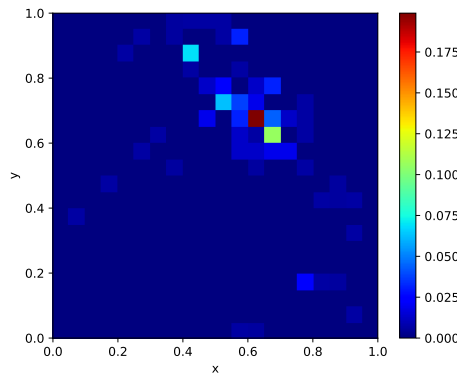


Figure 4.4: Normalised (x -, y - and z -axis) heatmap derived from the raw SMR PLS data.

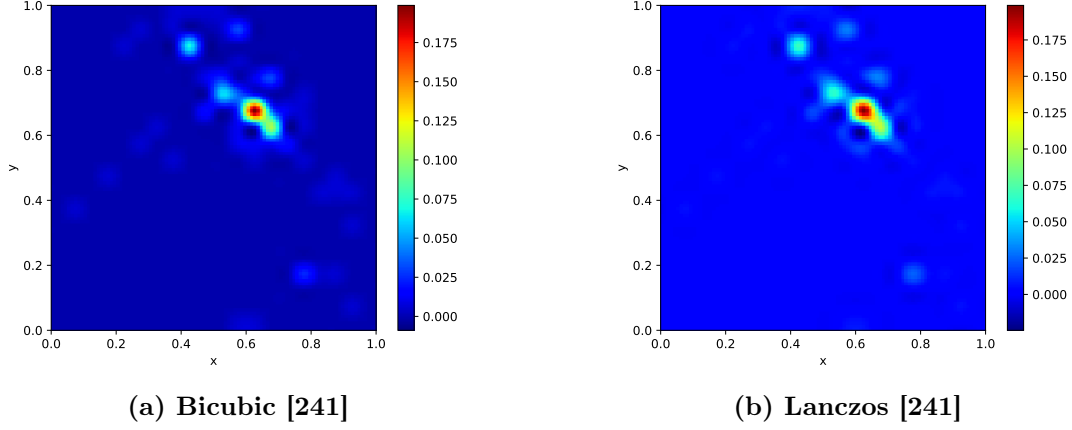


Figure 4.5: The resultant heatmaps after applying conventional upsampling methods to Fig. 4.4.

To draw generalised conclusions from the sparse PLS data, it is crucial to up-sample the heatmap and generate unique PLS locations. While traditional linear interpolation methods such as bicubic or Lanczos interpolation [241] can increase resolution, they often preserve the disconnected modes present in the original heatmap, as illustrated in Fig. 4.5. These methods, although computationally efficient, fail to fully capture the underlying spatial correlations and uncertainties inherent in WiSAR data. To overcome these limitations, a GP model is employed, a supervised machine learning approach well-suited for noisy and uncertain systems [242]. GPs offer a probabilistic framework for modelling spatial correlations and naturally handle uncertainty, both of which are crucial in WiSAR scenarios where data is often sparse and noisy [243]. Furthermore, GPs can incorporate prior knowledge about the smoothness and variability of the PLS distribution through kernel selection and hyperparameter optimisation. While GP interpolation can be more computationally intensive than linear methods, the improved representation of uncertainty and spatial correlation is paramount for the PLS up-sampling. The results of our GP-based approach are discussed in detail in Sect. 4.5, demonstrating its effectiveness in generating a continuous, smooth representation of the PLS probability distribution.

For the GP to work efficiently, the data must be formatted correctly. The original heatmap z^- is normalised by

$$z^+ = \frac{z^- - z_{\min}^-}{z_{\max}^- - z_{\min}^-}, \quad (4.12)$$

to give $z^+ \in [0, 1]$. The x and y values are normalised in the same manner. The normalised 2D heatmap is then unravelled in the row-major form such that every unique (x, y) pair maps to a single z^+ value in a 1D array.

The GP is then based primarily on the Matern kernel [242], which computes a

covariance matrix between inputs $\mathbf{a}$ and $\mathbf{b}$:

$$k_{\text{Matern}}(\mathbf{a}, \mathbf{b}) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}d \right)^\nu K_\nu \left(\sqrt{2\nu}d \right), \quad (4.13)$$

$$d = (\mathbf{a} - \mathbf{b})^\top \Theta^{-2} (\mathbf{a} - \mathbf{b}),$$

where d is the distance between $\mathbf{a}$ and $\mathbf{b}$, scaled by the length scale parameter Θ , ν is a smoothness parameter [244], and K_ν is the modified Bessel function [245]. k_ν can be simplified at half-integer steps of ν ($\nu = p + \frac{1}{2} \forall p \in \mathbb{N}$) with the most commonly used values being $\nu = 1.5$ and $\nu = 2.5$. This is due to $\nu = 0.5$ generally giving noisy outputs, and $\nu \geq 3.5$ being hard to distinguish between [242]. As such, a value of $\nu = 2.5$ [246] was selected, which simplifies the modified Bessel function to

$$k_{\nu=2.5} = \exp \left(1 + \frac{\sqrt{3}r}{l} + \frac{5r^2}{3l^2} \right) \left(\exp - \frac{\sqrt{5}r}{l} \right). \quad (4.14)$$

The Matern kernel is then scaled using a scale kernel to become

$$K_{\text{scaled}} = \theta_{\text{scale}} K_{\text{orig}}. \quad (4.15)$$

where, θ_{scale} is the output scale. As the dataset is image-based, and has a large amount of data points as a result, the KISS-GP [247] approximation for a given kernel is then applied. Given a base kernel K , the covariance $k(\mathbf{a}, \mathbf{b})$ is approximated by using a grid of regularly spaced *inducing points*:

$$k(\mathbf{a}, \mathbf{b}) = \mathbf{w}_{\mathbf{a}}^\top K_{U,U} \mathbf{w}_{\mathbf{b}}, \quad (4.16)$$

where U is the set of gridded inducing points, $K_{U,U}$ is the kernel matrix between the inducing points, $\mathbf{w}_{\mathbf{a}}$ and $\mathbf{w}_{\mathbf{b}}$ are sparse vectors based on $\mathbf{a}$ and $\mathbf{b}$ that apply cubic interpolation.

This covariance is then combined with a zero mean, in a multivariate normal distribution, and quantified by an exact marginal log-likelihood. This allows the measurement of the probability of generating the observed sample from a prior [244]. Finally, this is optimised using the Adam optimisation algorithm [248] with a learning rate of 0.1 over 250 epochs.

4.6 Results and Discussion

This section presents the results from the PDM generation method described in this chapter. Results from prior published peer-reviewed work are used as an indication of progression and are outlined in Sect. 4.6.1.2. This prior method is referred to as J1 from now on, while the method from this chapter is referred to as J2.

4.6.1 Experimental Setup

4.6.1.1 Sampling Place Last Seen Locations

A key element in deciding which location to analyse was the availability of the historical PLS data outlined in Sect. 4.5. The availability of that data, coupled with the availability of a high-quality LiDAR digital elevation map [235], meant that the Isle of Arran, Scotland, was therefore chosen.

The purpose of the process outlined in Sect. 4.5 was to estimate the raw data provided by SMR, and sample PLS locations from this model to further drive the experiments such that generalised statistics could be drawn from it. Fig. 4.6 shows the up-sampled heatmap, which retains the geospatial information of the PLS locations whilst increasing the number of sample points infinitely through the GP model.

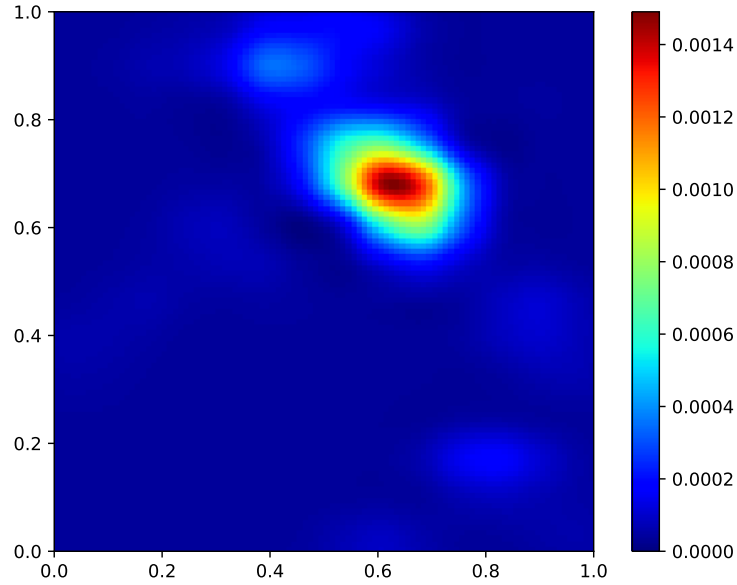
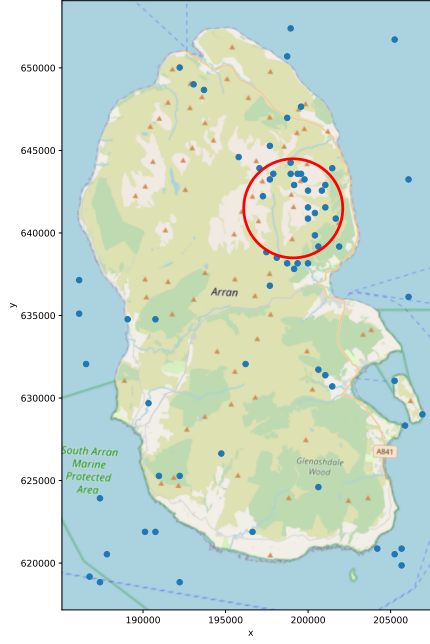


Figure 4.6: Up-sampled heatmap from Fig. 4.4 using a GP. Every grid cell now represents a $20\text{ m} \times 20\text{ m}$ area resulting in more possible sample locations.

Fig. 4.7 shows a small sample of $n = 70$ points (n was kept low for better figures) taken directly from the GP model. The nature of the GP model ignores the geospatial constraints of hikers not getting lost in the middle of the sea. Dealing with this is a task for the PDM generation algorithm.

The start locations were then sampled from the trained GP model, like in Fig. 4.7, and then further sampled from Eqn. 4.1 with $\sigma_{xx} = \sigma_{yy} = 10000$ to model the uncertainty in the reported PLS. In total 270 starting points were evaluated, resulting in 50,000 generated paths. The final number of locations found was 41,000,000 after the sampling stage. This is a sufficiently large number of data points for statistical analysis since Koester [24] and Perkins *et al.* [30] reported $n = 3,800$ and $n = 130$ respectively.



©OpenStreetMaps contributors.

Figure 4.7: $n = 70$ sampled PLS locations from Fig. 4.6 with the normalisation of Eqn. 4.12 reversed. Note the cluster of PLS locations around the hotspots (circled in red) from the raw data seen in Fig. 4.4. These sampled PLS locations are used as the start location for each PDM.

4.6.1.2 Benchmark Algorithm

An earlier iteration of this PDM generation algorithm was presented in ‘*GIS Data Driven Probability Map Generation for Search and Rescue Using Agents*’ in 2023 [46]. This algorithm is the same as the one outlined in Sect. 4.4 with the key difference being that the Head2Paths behaviour is much simpler.

Within the Head2Paths behaviour, this prior work uses the same staged approach. Similar to Sect. 4.4.1.1, the first stage is the *nearest path search*, and the second is *random path network exploration*. The former replaces Eqn. 4.8 with

$$\cos \theta = \frac{\mathbf{p}_{E,\min} \cdot \mathbf{p}}{|\mathbf{p}_{E,\min}| |\mathbf{p}|}, \quad (4.17)$$

where $\mathbf{p}_{E,\min}$ is the closest point on any road and $\mathbf{p}$ is the current position vector. Similarly, the latter behaviour sets $\lambda_{\max} = 0$ from Sect. 4.4.1.1, which negates the memory component resulting in a random search.

4.6.2 Generated Paths

Fig. 4.8 shows the paths created by various behaviours. As expected, there are substantially fewer paths for Head2Water than there are for Head2Paths. From Fig. 4.8b,

it can be seen that due to the lack of buildings in the north of the Isle of Arran, the paths there converge on a couple of points but tend to converge on different locations in the south. Furthermore, this behaviour shows the effect of the sampled points from Sect. 4.5.

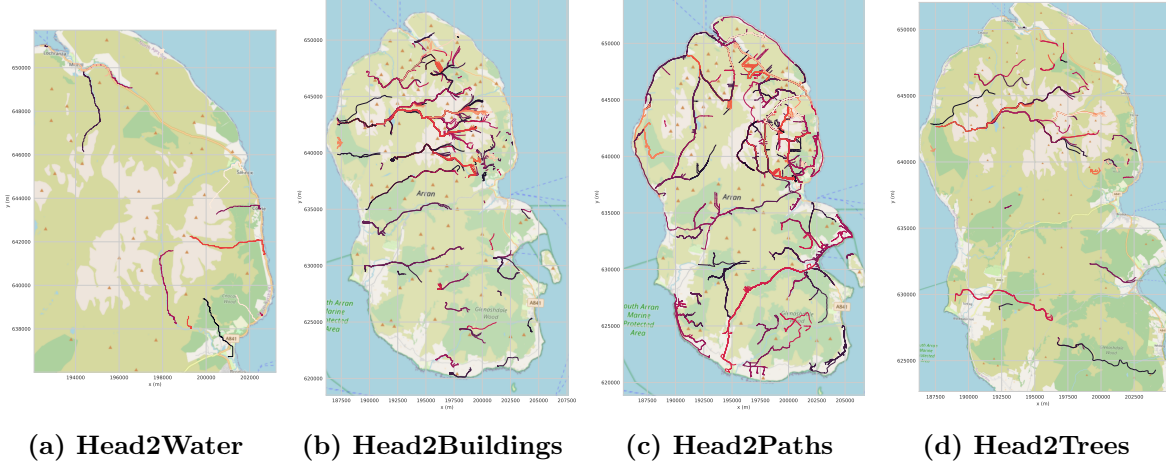


Figure 4.8: Subset ($n = 1,000$) of generated paths traversed by agents using the four different behaviours. The colour is used for a contrast between paths.

Base maps: ©OpenStreetMaps contributors.

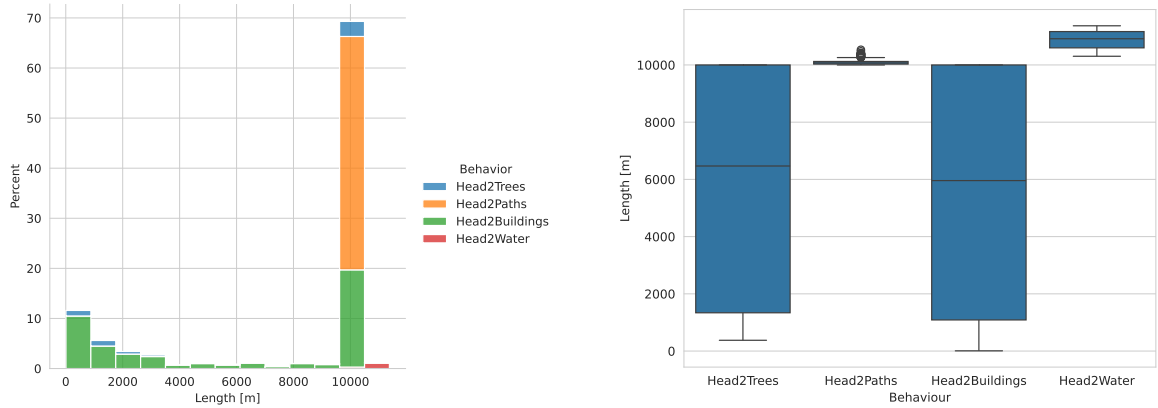
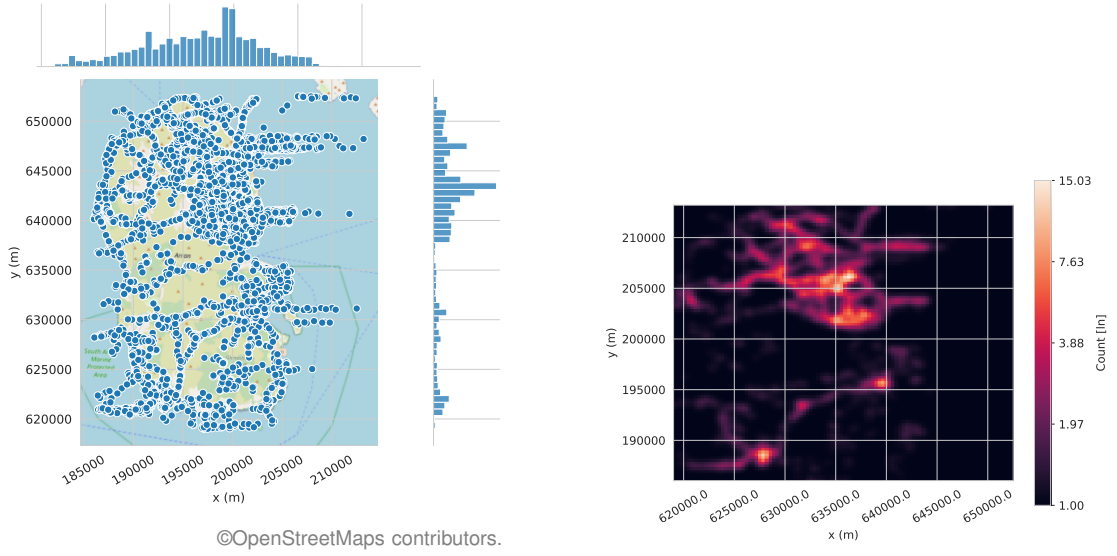


Figure 4.9: Analysis of the path lengths by behaviour.

As the simulation termination distance was 10 000 m, as described in Sect. 4.4.2, it is no surprise that over 60 % of paths (Fig. 4.9a) terminated at this criterion. However, as seen in Fig. 4.9b, it is evident that the two viewshed behaviours (Head2Buildings and Head2Trees) are terminating early. Sect. 4.4.1.1 explains this behaviour, and future work should explore this further.

Fig. 4.10 gives the first glimpse at the generated PDM. However, these are the aggregated results for multiple PLS locations and, as such, would look different for an individual PDM. Nonetheless, it reflects the sampled PLS by having more locations



(a) Locations found showing the distribution of points in the x-y axis.

(b) Logarithmic PDM of all PLS locations showing more activity in the north than the south.

Figure 4.10: Paths traversed by agents using the four different behaviours with $\sigma_{xx} = \sigma_{yy} = 10,000$, $\rho = 0$.

found in the north of the island with a slanted band of low probability. Insight like this would influence the decision-making during a search mission by not looking in that area, for example.

4.6.3 Mobility Distribution Sampling Effectiveness

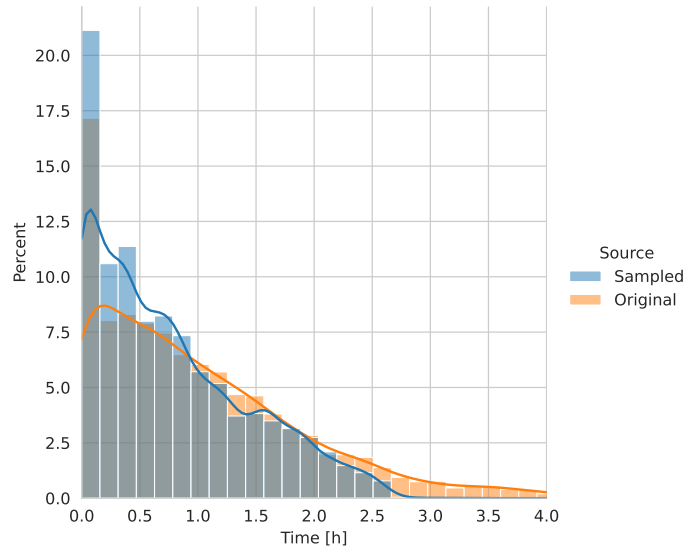


Figure 4.11: Distribution of the used time points for the sampled points (blue) and the original distribution (orange) showing the filtering in action.

The shortened paths from early termination of behaviours result in samples not being taken at distances longer than the path in question. Fig. 4.11 shows the times at which a point was sampled, which is skewed heavily to the left, with the original

distribution more spread out along the time axis. The sampled points have a mean time of 0.75 h and a standard deviation of 0.66, conversely, the original distribution has a mean of 1.06 h with a standard deviation of 1.01.

This difference in means shows that the upper limit of time travelled correlates to the total time an LP might be missing, with the distribution becoming closer to matching the historical data as $D_{\max} \rightarrow \infty$. Therefore, $D_{\max}$ can be further used to inform the generated PDM for the scenario at hand.

4.6.4 Location Found Land Cover Categories

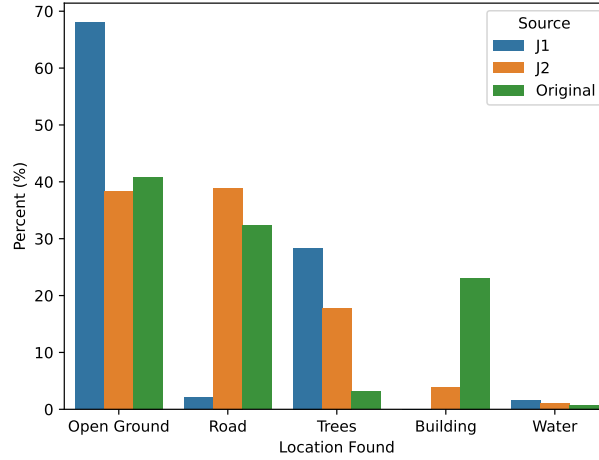


Figure 4.12: Results from this chapter (J2), a previous implementation (J1) [46], and the original data [30].

The resultant sampled locations of the agents are the most important metric to gauge the effectiveness of this model. This will be compared to the original data [30], and J1, a previous version of the algorithm.

As can be seen from Fig. 4.12, an LP was found in the *open ground* land cover category 38.36% of the time. This is only a 2.44% difference to the original data (compared to a 27.22% difference for J1), empirically supporting Hyp. 4.1. Therefore, LPs end up naturally in the open-ground land cover category without having a dedicated behaviour.

The next land cover category is *road*, at which 38.95% of LPs were found. Similarly to *open ground*, this is only a difference of 6.65% to the original data whilst J1 has a 30.20% difference. Such a large improvement in the *road* land cover category is a result of an improved Head2Paths behaviour algorithm from Sect. 4.4.1.1. However, the viewshed-based algorithms, Head2Trees, and Head2Buildings have improved but are still substantially off the mark compared to the source data. This result may occur by the way that the sampling is handled. Fig. 4.12 shows that even though 46.20% of paths are due to the Head2Buildings behaviour, only 33.07% of sampled points resulted from the aforementioned paths. This is due to Head2Buildings having a low

mean length of 5.62 km compared to the largest of 11.04 km as seen in Fig. 4.9b.

The last land cover category, *water*, remains close to the source data, as was the result for J1, showing that the strategy of following the water vector map remains valid.

Overall, when using Eqn. 4.11, the algorithm proposed in this paper had a score of 61.56 whilst J1 has a score of 306.02. This is a substantial improvement when compared to a randomly generated distribution with a score of 159.16 (with $n = 1 \times 10^7$). This shows a significant improvement in matching the source data, as is the target.

4.6.5 Computational Performance

Assessing how long it takes to generate a PDM for a given scenario is paramount to evaluating the feasibility of deployment for J2. Fig. 4.13 shows the histogram of the time taken per behaviour simulation from 163 000 data points. A single run took 13.70 s on average with a maximum of computational time of 76.91 s. With the distribution from Fig. 4.3, sampling each simulated path 10 000 times resulted in an additional 1.40 ms of runtime. Thus, in order to generate a PDM by running 10 000 behaviour simulations with 10 000 samples each would take 38.06 h and result in 1×10^8 unique points. Whilst this is in the orders of magnitude above an acceptable amount of time to wait for a single PDM, there are simple solutions to overcome this. Firstly, This type of workload is particularly suited for parallel processing, given that each simulation is completely independent of others. Multiple CPU cores or even multiple CPUs can be used to accelerate the workload. Using a 64-core CPU would theoretically then result in the total runtime decreasing to 0.59 h assuming linear scaling which is unrealistic but does highlight the potential improvements. Secondly, the work in this project was implemented using vanilla Python, which made it slow by default. This can be easily overcome by transitioning to faster languages like C/C++ or Julia, leveraging just-in-time compilation, or using CUDA [249]–[251].

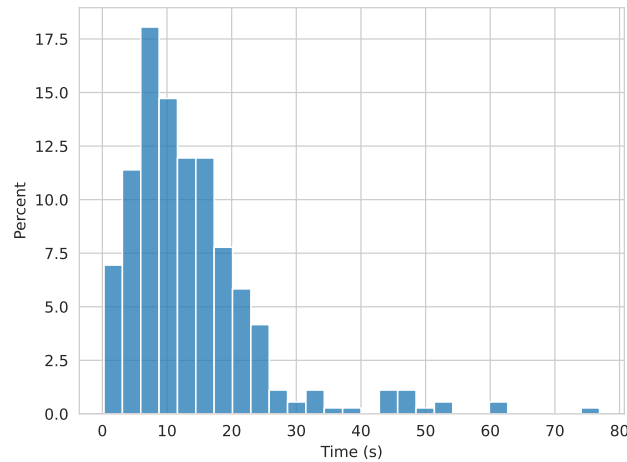


Figure 4.13: Histogram of the real computation time per simulated behaviour.

4.7 Chapter Summary And Conclusion

This chapter explored a psychological profile-based PDM generation algorithm, which emulates the movement of an LP over a landscape. Through GP-driven analysis, it is clear that the sparse data problem experienced in WiSAR can be overcome, leading to the tangible result of saving lives in future operations.

By characterising the profile into four distinct behaviours, the model can be adjusted to match the land cover category location found in the description from local datasets. These distinct, simpler, behaviours are their own model and traverse the landscape as if they were an LP with a single goal in mind. The data shows that running each behaviour a percentage of times would produce a distribution of land cover category locations and descriptions that match the original dataset. However, there were discrepancies between the tree and building categories. As both rely on viewshed-based behaviours, it follows that this is the root cause of the discrepancy as further explained in Sect. 4.6.3. The introduction of the GP-based design of the experiment also benefited the analysis by generalising the PLS locations. This is something that was not done in J1, which further decreased the effectiveness of the analysis with a $n = 2$.

Overall, the method of using the land cover category location found statistics directly from the source data in J2 is valid with a symmetric Kullback-Leibler (Eqn. 4.11) score of 61.56, whilst a truly random distribution has a score of 159.16. This means that per-location training is not required, and a generalised model has been created. However, further effort needs to be put into improving the viewshed-based algorithms.

4.8 Limitations and Future Work

A lack of usable location data was one of the key limitations of this chapter, as touched upon in Sect. 4.5. This chapter used the top-level location found data from Koester [24] which was used to analyse the algorithm's performance. However, Sava *et al.* [28] describes a method that evaluates the PDM using matched PLS and find coordinates. This, however, requires a substantial amount of data that has both sets of information. Hashimoto *et al.* [42] uses a similar approach and defines an energy metric based on the model's output and the find coordinates. Future work should consider collecting this data since the dataset from Sect. 4.5 has many data points but only two, with both the PLS and find coordinate which is insufficient for this kind of analysis. Similarly, having this data for other locations outside of the Isle of Arran would result in a broader assessment of the model's generalisability.

Future work should also consider additional proposed lost person behaviour such as moving uphill [252] and direction travelling [24], which could add a further layer of fidelity to the model. Likewise, visibility should be considered within the model because it is one of the leading causes of becoming lost [253]. Dudchenko [253] highlights that

self-reorientation is typically done in a reduced visibility environment.

Chapter 5

Case Study: Cubature Versus Sampling-based Integration Methods

5.1 Section Overview

This section investigates the computational speed and accuracy of two numerical integration methods — cubature and sampling-based — for integrating an integrand over a 2D polygon. This topic is important to this thesis, as it helps inform the implementation details of the continuous-space RL search algorithm developed in the chapters to come. The integration methods are compared in terms of their computational performance and accuracy to better understand their suitability for use in search planning applications. RL requires accurate reward calculations, and answering this question before using it in search planning is key.

A group of rovers searching the Martian surface with a limited sensor footprint is used as a test bed. Analysing the results of multiple rovers traversing the search path gives a highly irregular search area, making it a challenging benchmark for either method.

This work was carried out as part of a collaborative effort by the thesis author and colleagues, resulting in two peer-reviewed papers: ‘Enhancing Reinforcement Learning in Sensor Fusion’ Ewers *et al.* [47] and ‘Autonomous Mission Planning for Planetary Surface Exploration Using a Team of Micro Rovers’ Swinton *et al.* [51]. The multi-rover simulation described in Sect. 5.3.2 was developed by Sarah Swinton from the University of Glasgow, and the section was rewritten for this thesis. Changing from UAVs to rovers is unimportant as the search path and trajectories would be similar, giving the same conclusions.

The results show that the sampling-based approach exhibits a 14.75% deviation in relative error compared to cubature when it matches the computational performance at 100%. Furthermore, achieving a relative error below 1% necessitates a 10000% increase in relative time to calculate due to the $\mathcal{O}(N^2)$ complexity of the sampling-based method. It is concluded that for enhancing reinforcement learning capabilities and other high-iteration algorithms, the cubature method is preferred over the sampling-

based method.

The top-level search planning and PDM generation is introduced in Sect. 5.3.1, with modelling and path planning in Sect. 5.3.2. Sampling-based and cubature integration is defined in Sect. 5.3.3.1 and Sect. 5.3.3.2 respectively. The results are presented and discussed in Sect. 5.4 with a chapter conclusion being drawn in Sect. 5.5.

5.2 Introduction

ML has gained massive popularity in the last decade [254] driven by both the ongoing research into learning algorithms and cheap computational performance [255]. Phenomenon such as the success of generative artificial intelligence [256] have put ML more into the eye of the public than ever before. However, behind the scenes, researchers spend a vast amount of time preparing data for learning. All three major forms of ML require high-quality data for the best performance: supervised learning needs well-labelled data points, unsupervised learning must have noise-free data for labelling, and reinforcement learning has to have access to a good representation of the environment. Studies on the impact of non-systematic noise on training have shown massively degraded performance [257], [258].

While significant progress has been made in developing robust ML algorithms [259], [260], the impact of noise in reinforcement learning remains a critical challenge [261]. In the realm of reinforcement learning, noise can be beneficial. Action noise, which is the addition of randomness to the action selected by the algorithm, can be desirable to encourage exploration of the action space. Using Gaussian or Ornstein-Uhlenbeck noise has been shown to significantly improve training across popular reinforcement algorithms such as Soft Actor-Critic [223] or Deep Deterministic Policy Gradient [262] when coupled with scheduled noise reduction [263]. However, this scheduling shows that noise must be minimised to fine-tune the model. In comparison, observation noise in reinforcement learning, akin to poor-quality labelled data for supervised learning, is known to make training unstable [261]. Noise in the reward channel for reinforcement learning is particularly undesirable as it can lead to actions being misinterpreted. Everitt *et al.* [264] highlights multiple scenarios in which the agent may take advantage of a reward function leading to *reward corruption* and a stagnation in learning on a local maxima. Whilst a solution to this was proposed by Everitt *et al.* [264], a much easier approach is to minimise the chances of this happening in the first place by implementing a robust method with as little noise as possible. Overall noise in ML, when not controllable, is undesirable as it increases the difficulty in successfully learning the policy.

A famously noisy task is numerical integration. From Euler integration to complex modern algorithms, errors are inevitable compared to continuous methods as numerical methods are just approximations. However, reducing these errors is paramount to decreasing noise. It is very common to have simulation step updates within a re-

inforcement learning environment, which require integration in multiple forms. This might be double integrating acceleration to get the position, or integrating a function within a 2D shape. This chapter focuses on the latter. This problem takes the form

$$I(f, H) = \oint_H f(\mathbf{x}) dH, \quad (5.1)$$

where $\mathbf{x} \in \mathbb{R}^2$, f is the integrand, and H is the collection of s -simplices. In search path planning, the integrand is commonly a PDM, and H is the path buffered by the sensor footprint r . A widely used method of integration is a sampling-based approach which calculates the integration result through grid sampling of the PDM and then calculating the summation of pixels within H . A different approach uses cubature [265] to achieve a similar result. This relies on the transformation of H into a sum of computable simplices such as triangles or rectangles. *cubpack++* by Cools *et al.* [266], implement methods for 15 primitive regions that any simplex can be dissected into, like the aforementioned triangle and rectangle. The integration result is then the sum of cubature results for each simplex. Another approach by Robinson *et al.* [267], *r2d2lri*, uses a sequence of embedded lattice rules and claims to result in little computational overhead. Whilst Hill *et al.* [268] found *r2d2lri* to be faster during integration over an infinite domain, *cubpack++* proved to be more adaptable to different scenarios without a massive speed penalty. For the rest of this chapter, *cubpack++* will be used for its adaptability and adaptive properties.

The task of numerically integrating an integrand over a 2D shape within search planning can be thought of as “*How much probability has been seen by executing this mission?*” In this application, the integrand is a PDM that defines the probability of detection at a given point, and the 2D shape is the search path buffered by some quantity to get a polygon. In order to apply reinforcement learning to this problem, the newly observed probability could be used to calculate the reward function. As previously outlined, this requires a fast and noiseless numerical integration method for good training characteristics.

Search planning is typically applied in situations where exhaustive search (also known as coverage planning) is infeasible due to mission constraints such as limited resources. One such situation where a reduced vehicle lifespan is the limiting factor is searching the Martian surface for items of scientific interest based on a PDM. In 2020, NASA’s Mars Mission utilised two robots: the Perseverance rover and the Ingenuity helicopter — opening the door to the use of collaborative robotic teams for planetary exploration [269]. The use of multi-rover teams for planetary exploration missions has been considered in terms of both surface [270] and cave [271] exploration. By using a team of planetary exploration rovers, the data collection capabilities of a mission can be increased due to the extended sensor footprint. Whilst this study was focused on exploratory Mars rovers, search planning has applications within other domains, such as in the wilderness [155] or marine [272] search and rescue.

In this study, the two aforementioned integration methods are benchmarked against each other to determine the performance differences between them. This will inform future studies on which method is appropriate for their application. A group of rovers searching the Martian landscape with reduced sensor footprints is used as a benchmark. Due to collision avoidance and inter-rover communications, the generated paths are highly irregular, creating a challenging 2D polygon to integrate. The PDM for this scenario is analogous to the probability distribution of a meteor that has crashed on the surface and that the rovers are now searching for.

5.3 Method

This section introduces the search planning, rover modelling and low-level path planning, and path analysis in Sect. 5.3.1, Sect. 5.3.2, and Sect. 5.3.3 respectively. In the latter, the two integration methods — sampling-based and cubature — are introduced.

5.3.1 Search Planning

As outlined in Sect. 2.2, search planning requires the notion of prioritised coverage and limited endurance. For this reason, LHC_GW_CONV, first proposed by Lin *et al.* [155], is used to generate the search path. This chapter compares the two integration methods — cubature and sampling-based — not the search planning itself. Since LHC_GW_CONV is cheap to compute and has modest search performance over a PDM, this algorithm can quickly generate many paths for analysis.

LHC_GW_CONV is used to generate $N_{\text{wp}} = 64$ mission waypoints with a search diameter of $d = 5$ m; corresponding to five rovers sweeping side-by-side with a search footprint diameter of 1 m each. The area is defined as a rectangle with width and height being a multiple of d . In this chapter a search area of $150 \text{ m} \times 150 \text{ m}$ is used giving a search grid of 30×30 cells.

LHC_GW_CONV is, at its core, a greedy algorithm by employing the LHC optimisation method. This evaluates all eight surrounding cells and moves to the highest-scoring one. If there is a tie, a convolution kernel ω centred around the offending cells is evaluated and the cell with the highest score,

$$p^{\text{conv}}(\mathbf{x}) = \omega * p(\mathbf{x}) = \sum_{i=-1}^1 \sum_{j=-1}^1 \omega(i, j) p\left(\mathbf{x} - \begin{bmatrix} i \\ j \end{bmatrix}\right), \quad (5.2)$$

is selected. A 3×3 normalised box blur kernel is typically used and is defined as

$$\omega = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \quad (5.3)$$

Furthermore, LHC does not handle multi-modal problems well as it attempts to

cover one mode completely before being able to move to the next. This often leads to LHC ignoring much more rewarding modes. To solve this, global warming is introduced. This works by subtracting C from the PDM a l number of times, where $C = p_{\max}/l$ and $p_{\max}$ is the global maxima. This then gives the updated PDM,

$$p'(\mathbf{x}) = \begin{cases} p(\mathbf{x}) - C, & p(\mathbf{x}) > C \\ 0, & \text{else} \end{cases}, \quad (5.4)$$

which the LHC uses to generate a new path. After all l global warming steps are evaluated, they are scored using the original PDM $p(\mathbf{x})$ and the path with the highest accumulated probability is selected.

Finally, each visited cell is marked as *seen*, and subsequent evaluations of $p(\mathbf{x})$ at this cell will return 0. This discourages revisiting a position but does not entirely forbid it.

The continuous PDM is constructed of four multi-modal bivariate Gaussians (MMBGs) as Yao *et al.* [139] and Yao *et al.* [188] showed that MMBGs are effective at approximating 3D PDMs. The random PDM that describes the probability $p(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^2$ is thus given by

$$p(\mathbf{x}) = \frac{1}{N_{\text{gaussian}}} \sum_{i=0}^{N_{\text{gaussian}}} \frac{\exp\left\{-\frac{1}{2}(\mathbf{x} - \mu_i)^T \sigma_i^{-1}(\mathbf{x} - \mu_i)\right\}}{\sqrt{4\pi^2 \det(\sigma_i)}}, \quad (5.5)$$

where the mean μ and covariance matrix σ are instantiated by

$$\begin{aligned} \sigma_i &= \begin{bmatrix} \sigma_{i,x}^2 & \rho_i \sigma_{i,x} \sigma_{i,y} \\ \rho_i \sigma_{i,x} \sigma_{i,y} & \sigma_{i,y}^2 \end{bmatrix} \\ \mu_i &\sim \mathcal{U}([x_{\min}, x_{\max}], [y_{\min}, y_{\max}]) \\ \sigma_{i,xy} &\sim \mathcal{U}(\sigma_{\min}, \sigma_{\max}) \\ \rho_i &\sim \mathcal{U}(0, \rho_{\max}) \\ \forall i &\in [1, \dots, N_{\text{gaussian}}] \\ N_{\text{gaussian}} &\sim \mathcal{U}(N_{\text{gaussian,min}}, N_{\text{gaussian,max}}) \end{aligned}, \quad (5.6)$$

where $N_{\text{gaussian,min/max}} \in \mathcal{N}$. The PDM parameters given in Tbl. 5.1 are sampled at the beginning of the simulated search mission. This PDM formulation can be easily converted into the representation required for LHC_GW_CONV by rasterising the PDM into a grid of pixels and likewise for the sampling-based integration.

5.3.2 Rover Model And Path Planning

The individual rover is modelled as the six-degree-of-freedom differential-drive six-wheeled buggy from Swinton *et al.* [51]. This model has a small form factor, which when combined with the search parameters from Sect. 5.3.1, gives a coordinated group

Table 5.1: The parameters used for sampling the PDM from Tbl. 5.1.

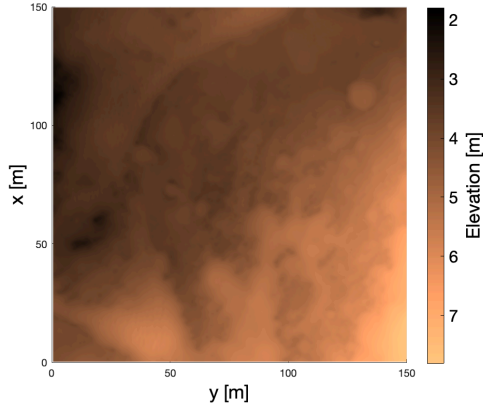
Parameter	Value	Unit
$N_{\text{gaussian,min}}, N_{\text{gaussian,max}}$	4	
$x_{\text{min}}, y_{\text{min}}$	0	m
$x_{\text{max}}, y_{\text{max}}$	150	m
$\sigma_{\text{min}}, y_{\text{max}}$	500	
ρ_{max}	0	

of five rovers a 5 m effective search footprint.

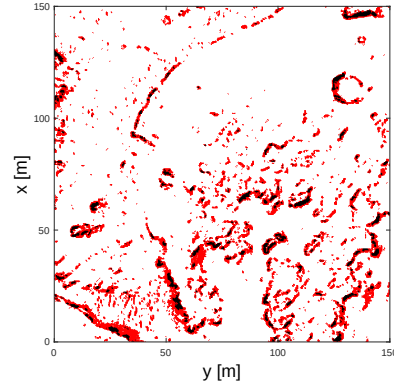
A 3D terrain DEM is constructed using data from the high-resolution imaging science experiment [273], [274] to model slip. This map, seen in Fig. 5.1a, covers a $150\text{ m} \times 150\text{ m}$ area and provides insights into the traversability of the area. Three categories are established which influence the rover path planning:

$$\text{TraversabilityCategory}(\theta) = \begin{cases} \text{Traversable,} & \theta < 10^\circ \\ \text{High - Risk,} & \theta < 15^\circ . \\ \text{Impassable,} & \text{else} \end{cases} \quad (5.7)$$

These categories can be seen in Fig. 5.1b.



(a) 3D terrain model of the selected mission site from the High-Resolution Imaging Science Experiment.



(b) Traversability analysis of the selected mission site. Traversable terrain is shown in white, high-risk terrain is shown in red, and impassable terrain is shown in black.

Figure 5.1: The 3D data maps used in this chapter centred around the Jezero Crater on Mars. This $150\text{ m} \times 150\text{ m}$ area was selected due to its challenging topology without having large impassable regions.

This terrain is searched with the search path from Sect. 5.3.1 by dissecting it into smaller waypoint path planning problems. The RRT*-based planner from Swinton *et al.* [275] is applied with offset start- and end-points based on the rover index, search footprint, and path normals. During the multi-rover path planning, an iterative process of prioritised safe coordination is applied to ensure traversability and avoid collisions between rovers. Each rover has a dedicated low-level guidance and control module to

follow the bespoke path. A line-of-sight trajectory controller carries out the guidance, which provides velocity and yaw error inputs to the motor controller. This hierarchical procedure can be seen in Fig. 5.2.

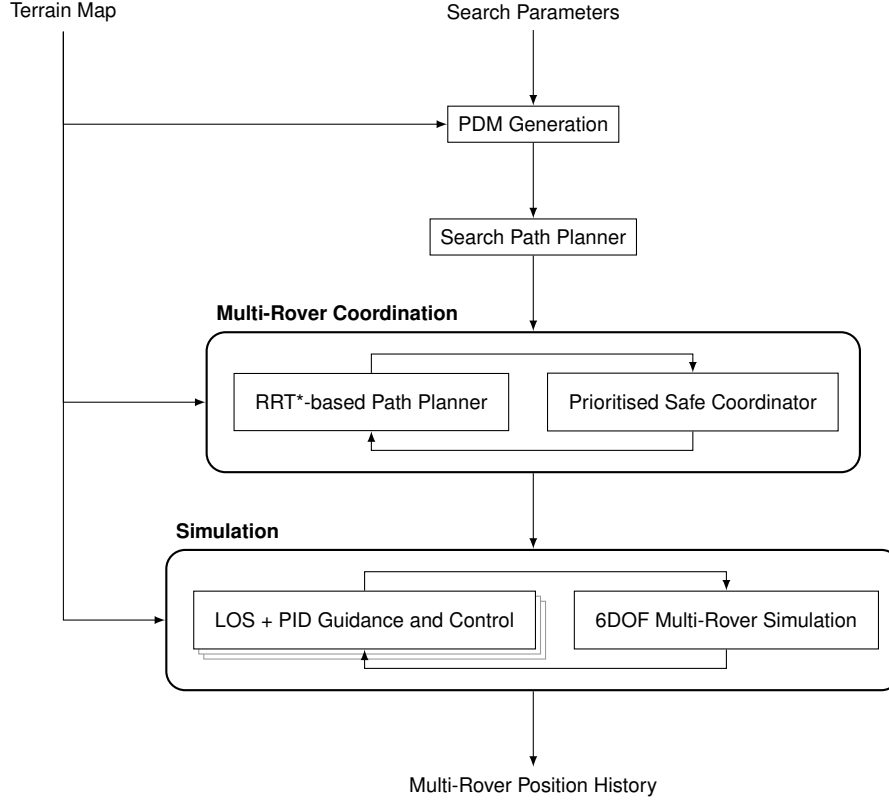


Figure 5.2: Block diagram of the search planning for multi-rover implementation using a hierarchical path planning framework. The search planning is undertaken by LHC_GW_CONV [155], which is transformed into a safe multi-rover path. This path is traversed within the six degrees of freedom multi-rover simulation [51], [275] using the line-of-sight waypoint guidance and control modules.

5.3.3 Path Analysis

Each rover generates a unique path as it executes the mission, which is buffered by the search footprint diameter d_{rover} . The total search area H_{total} is then the union of the set of buffered paths H such that

$$H_{total} = \bigcup_{h \in H} h. \quad (5.8)$$

This can be extended to be a function of time t by

$$H_{total}(t) = \bigcup_{h \in H(t)} h. \quad (5.9)$$

where $H(t)$ is the set of buffered paths covered by the rovers up until ts . If t_{max} is the time taken for the entire mission to be completed, it then follows that $H_{total} = H_{total}(t_{max})$ and $H_{total} \neq H_{total}(t) \forall t < t_{max}$.

Thus, the total probability contained by the search path is found by integrating the probabilities defined by the PDM over the search area $H(t)$ using Eqn. 5.1. Sect. 5.3.3.1, and Sect. 5.3.3.2 will outline two methods to calculate this value; sampling-based and cubature, respectively.

5.3.3.1 Sampling-based Integration

Accessing the value of $p(x, y)$ at a given coordinate is trivial, and is as computationally expensive as the definition of $p(x, y)$ itself. A more computationally challenging task is determining if the point at (x, y) is within $H_{total}(t)$.

A popular method is ray casting along a straight line from a point [276]. If this ray intersects $H_{total}(t)$ an odd amount of times, then the buffered search paths contain the point. This is demonstrated in Fig. 5.3 where the ray originating from the point (geometry B) intersects the polygon (geometry A) three times indicating that the polygon contains the point.

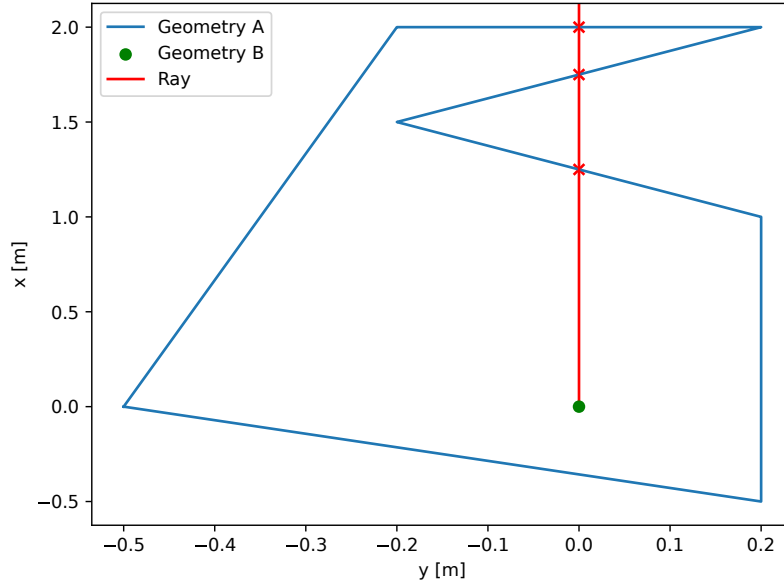


Figure 5.3: Ray casting method to check if the non-convex polygon (geometry A) contains a given point (geometry B). If the ray originating from B intersects A an odd amount of times (three, in this example), then A contains B.

Another popular method uses the Dimensionally Extended Nine Intersection Matrix [277]. This is calculated using

$$DE - 9IM(A, B) = \begin{bmatrix} D(A^I \cap B^I) & D(A^I \cap B^B) & D(A^I \cap B^E) \\ D(A^B \cap B^I) & D(A^B \cap B^B) & D(A^B \cap B^E) \\ D(A^E \cap B^I) & D(A^E \cap B^B) & D(A^E \cap B^E) \end{bmatrix}. \quad (5.10)$$

where $D(x)$ is the dimension of x ($D(x) \in (0, 1, 2)$ if $x \neq \emptyset$, and $D(x) = -1$ if $x = \emptyset$), and the superscripts I , B , and E denote the interior, boundary, and exterior regions, respectively. This matrix can be used to calculate 47 different predicates for

A and B to be in. The predicate of interest here is the topological *within*, which is described by the pattern matrix

$$within = \begin{bmatrix} T & * & F \\ * & * & F \\ * & * & * \end{bmatrix}, \quad (5.11)$$

where an entry in the matrix is marked T if $D(x) \in (0, 1, 2)$, F if $D(x) = -1$, and $*$ if $D(x) \in (-1, 0, 1, 2)$.

The dimensionally extended nine intersection matrix approach is 10.05% faster than the ray-casting method when dealing with the high-vertex polygons generated by the buffered paths H and is the method used for the rest of this study.

As touched on in Sect. 5.2, the sampling method is the sum of pixels within the polygon written as

$$I(f, H) = \frac{A}{NM} \sum_{n=0}^N \sum_{m=0}^M \begin{cases} 0, & (n, m) \text{ not within } H \\ f(n, m), & \text{else} \end{cases}, \quad (5.12)$$

where N and M are the dimensions of the rasterisation, A is the area enclosed by the rasterisation, $f(n, m)$ is the integrand (akin to $f(\mathbf{x})$ from Eqn. 5.1) which is a function accessing the pixel value at (n, m) , and H is a single polygon. This method can be fast with low values of N and M , and accurate with high values alike. However, this method highly suffers from aliasing around the borders of H . Methods to get around the aliasing problem have been tried in previous work [17] with major performance penalties. Nonetheless, due to its simplicity, it is prominently used in the literature [22], [138], [162], [163], [173].

5.3.3.2 Cubature Integration

The problem of integrating a continuous function $f(\mathbf{x})$ in two dimensions over the domain H can be approximated using the cubature rule

$$I(f, h) \approx Q(f, h) = \frac{1}{A_h} \sum_{i=0}^L w_i f(\mathbf{x}_i), \quad (5.13)$$

which evaluates $f(\mathbf{x})$ at a number of L set points and normalises by the area of h . This general approach can be applied to many shapes, with Cools *et al.* [266] identifying fifteen primitive shapes that any other shape can be subdivided into, which includes the triangle. As H is likely to have many holes and unlikely to be convex, the constrained Delaunay triangulation [278] is used to get the set of triangles T . Cools *et al.* [266] employs the DCUTRI algorithm from Berntsen *et al.* [279] with minor modifications to calculate the integral over a triangle as defined in Alg. 5.1. The triangle subdivision scheme applied in this algorithm can be seen in Fig. 5.4.

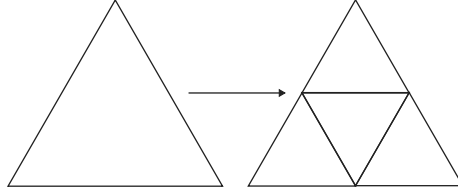


Figure 5.4: Subdivision of the unit triangle into four smaller triangles.

This approach is iterative until either the error constraint $\hat{E} \leq \max(\epsilon_a, |\epsilon_r \hat{Q}|)$ is met, or the total number of iterations breaches a threshold. The null rule is defined by

$$N(f, H) = Q(f, H) - Q'(f, H) = \sum_{i=0}^L (w_i - w'_i) f(\mathbf{x}_i), \quad (5.14)$$

which takes the difference between the cubature result of degree d and one of a lower degree d' at the same points [280]. This indicates the *quality* of the result and gives an error estimate. Berntsen *et al.* [279] expands on this by evaluating multiple lower degrees for a better estimate of the error.

Cools *et al.* [266] uses the triangle cubature rule of degree 13 with 37 points to estimate the integral. The position of these points and their respective weights can be seen in Fig. 5.5a, which shows their definition with respect to the unit triangle. Given a target triangle h , all points on the unit triangle can be affinely transformed to lie within h . This linear transform is defined by

$$T \underbrace{\begin{bmatrix} x_{1,1} & x_{2,1} & x_{3,1} \\ y_{1,1} & y_{2,1} & y_{3,1} \\ 1 & 1 & 1 \end{bmatrix}}_A = \underbrace{\begin{bmatrix} x_{1,2} & x_{2,2} & x_{3,2} \\ y_{1,2} & y_{2,2} & y_{3,2} \\ 1 & 1 & 1 \end{bmatrix}}_B, \quad (5.15)$$

where the transformation matrix T can be found with

$$T = BA^{-1}. \quad (5.16)$$

An example of this transformation can be seen in Fig. 5.5b which highlights how the weights and points stay within h after the fact for evaluations of $f(\mathbf{x})$ in Eqn. 5.13. It should also be noted that the unit triangle can take other forms, as discussed in Lyness *et al.* [265].

5.4 Results

As discussed in Sect. 5.3.1, LHC is a greedy algorithm that prioritises local modes. This can be seen in Fig. 5.6 at (80, 80)m where the mission abruptly leaves the saddle between the modes to ascend the local probability mode. The rovers attempt to avoid collision and obstacles, creating small holes in the buffered search path as well as

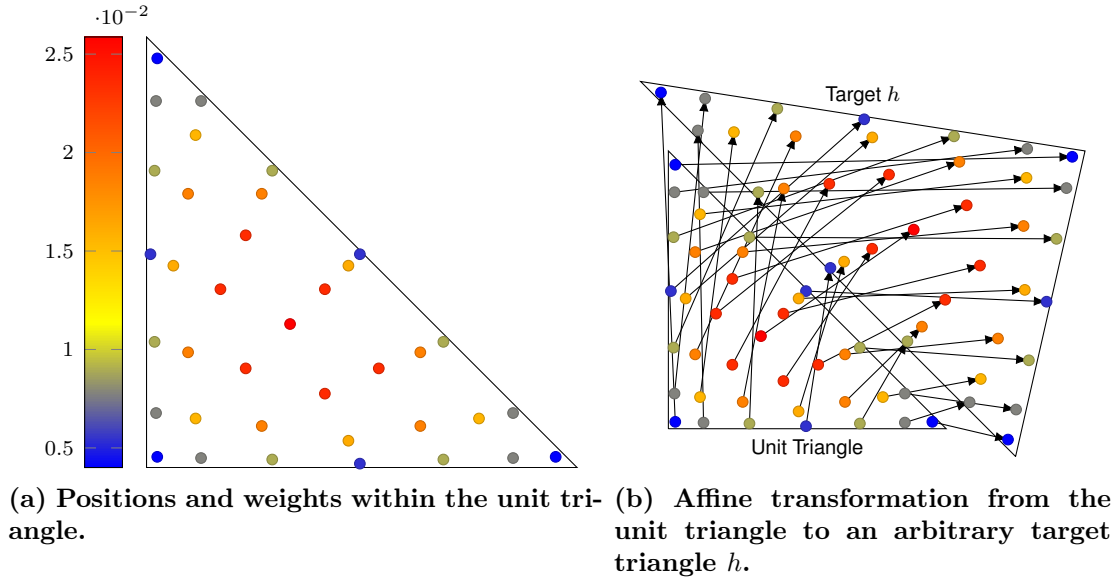


Figure 5.5: Triangle cubature rule of degree 13 with 37 points from Cools *et al.* [266], Berntsen *et al.* [279] and Cools *et al.* [281]. Exact values are given in Tbl. B.1.

a highly irregular exterior. This creates a challenging benchmark to compare and contrast the methods.

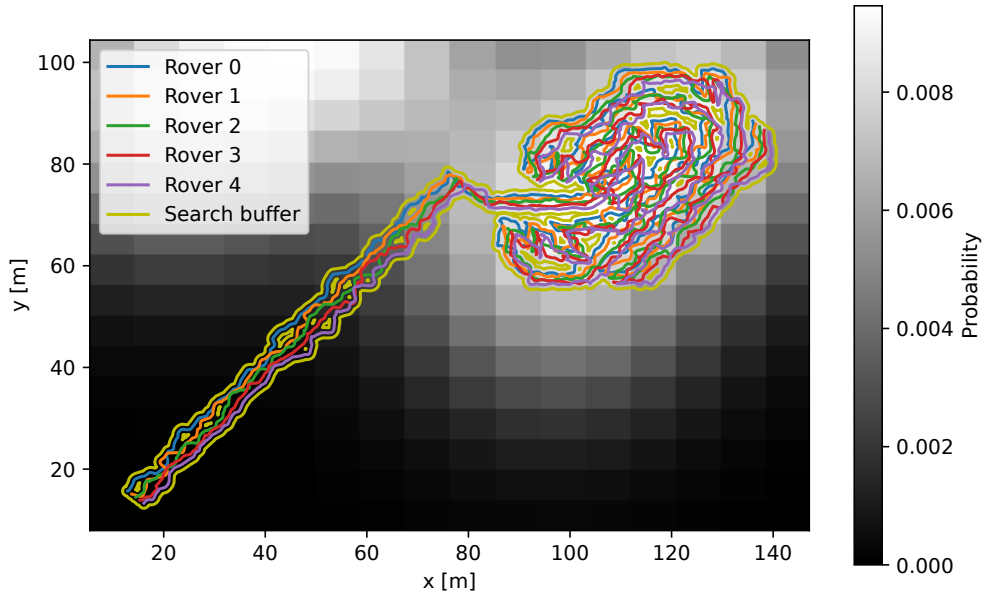


Figure 5.6: Rover trajectories over the area given the mission planned by LHC_GW_CONV with a random PDM $p(\mathbf{x})$. $N = 16$ was selected to emphasise the issue with sampling-based integration methods.

To benchmark the methods, the cubature method is evaluated for each PDM and corresponding paths and the mean of this result (time and integration result) were deemed the baseline. Overall, 105 missions have been simulated using the mission planning from Sect. 5.3.1 and the rover simulation from Sect. 5.3.2. In this case, the

Algorithm 5.1: Modified DCUTRI [266], [279], [281].

Data: Function $f(\mathbf{x})$, the collection of triangles H , and relative and absolute error thresholds ϵ_r, ϵ_a .

Result: Integral estimate $\hat{Q}$ and error estimate $\hat{E}$.

```

1  $m \leftarrow \dim(M)$ ;
2 Calculate  $\hat{Q}_k$  and  $\hat{E}_k$  for  $k \in [1, \dots, m]$ ;
3  $\hat{Q} \leftarrow \sum_{i=1}^m \hat{Q}_i$ ;
4  $\hat{E} \leftarrow \sum_{i=1}^m \hat{E}_i$ ;
5 while  $\hat{E} > \max(\epsilon_a, |\epsilon_r \hat{Q}|)$  do
    /* Vanilla DCUTRI selects a random triangle from  $H$ . */
6     Select the triangle  $h = H_k \in H$  with the largest estimated error  $\hat{E}_k$ ;
7     Subdivide  $h$  into four new sub-triangles ;
8     Calculate  $\hat{Q}_k^{(i)}$  and  $\hat{E}_k^{(i)}$  for  $i \in [1, 2, 3, 4]$ ;
9      $\hat{Q} \leftarrow \hat{Q} + \sum_{i=1}^4 (\hat{Q}_k^{(i)} - \hat{Q}_k)$ ;
10     $\hat{E} \leftarrow \hat{E} + \sum_{i=1}^4 (\hat{E}_k^{(i)} - \hat{E}_k)$ ;
11    Replace  $h$  with the four sub-triangles in  $H$ ;
12     $m \leftarrow m + 3$ ;
13 end
```

sampling method is evaluated from $N = 10$ to $N = 300$, and the relative error and relative time are calculated with

$$e_{rel} = \frac{|c - s|}{c}, \quad (5.17)$$

where c and s are the cubature and sampling values, respectively.

From Eqn. 5.12, it can be seen that the grid that the sampling method uses grows in $\mathcal{O}(NM)$. For the remainder of this study, $M = N$ for simplicity, making this algorithm $\mathcal{O}(N^2)$. This is evident from Fig. 5.7 where it can be seen that after $N = 16.78$, the cubature method is as fast to compute as the sampling method. After $N = 300$, this value is at 25000%. For reinforcement learning, where billions of steps are taken in the modelled environments during training, this can quickly add days to the learning time.

Fig. 5.8 shows the exponential decay of the relative error as N increases. At the crossing point of $N = 16.78$, the relative error is 14.75%. Depending on the application, this level of accuracy might be deemed sufficient. However, for reinforcement learning, as outlined in Sect. 5.2, any noise must be mitigated as it can lead to unstable training and, therefore, diminished results. Even at the high value of $N = 300$, the relative error is still above 0.5%. This size of error is likely to be acceptable, however, the computational cost is too high to justify this method.

A smaller N might lead to faster computation, the physical meaning of N must be considered. The individual rovers have a sensor diameter of 1 m and the area has dimensions 150 m $\times$ 150 m. To ensure that each grid cell is the same size as that of a single rover's sensor, $N = \frac{150}{1^2} = 150$, which would lead to a 800% relative time

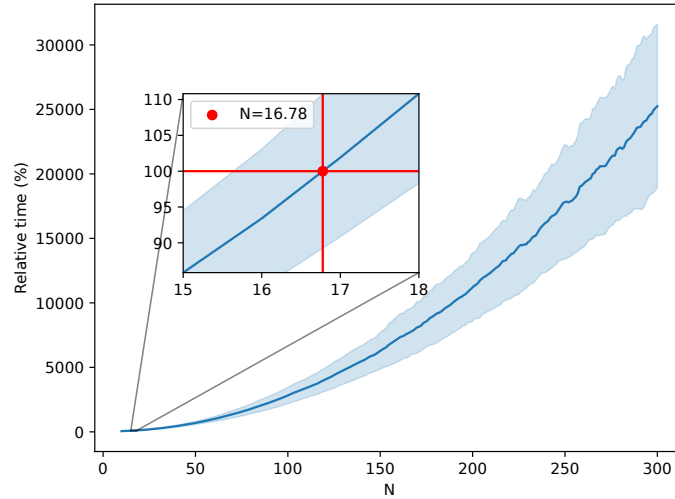


Figure 5.7: The relative time as N increases, showing a trend towards an exponential increase as $N \rightarrow \infty$.

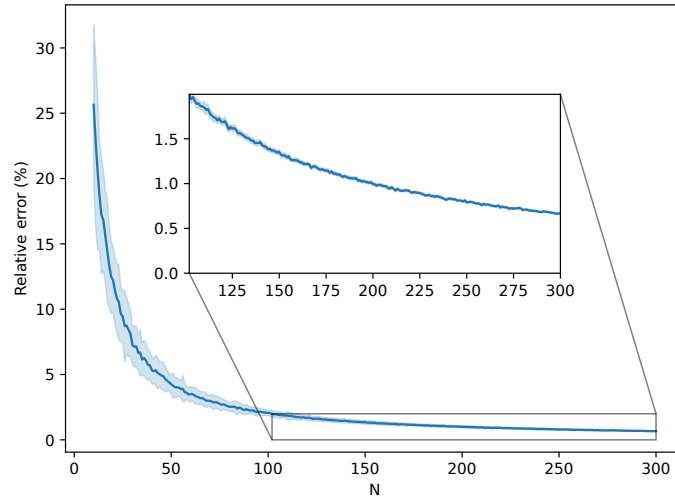


Figure 5.8: The relative error as N increases, showing a trend towards 0% as $N \rightarrow \infty$.

difference between the two methods as seen in Fig. 5.9.

The impact of the varying maximum number of coordinates per trial, denoted as N_{coords} , on the relative error was examined to ensure it was not a dependent variable. The null hypothesis posited that N_{coords} does not affect the relative error. Following the application of an ordinary least squares, a p -value of 0.1404 was computed, surpassing the level of significance (0.05). Thus, the null hypothesis cannot be rejected, indicating that there is no significant evidence to suggest that N_{coords} influences the relative error.

5.5 Chapter Summary And Conclusion

This chapter presents a comprehensive analysis comparing cubature and sampling-based integration of an integrand over a 2D polygon. An example scenario requiring a coordinated group of rovers' search of an area on Mars was used as a test bed. These

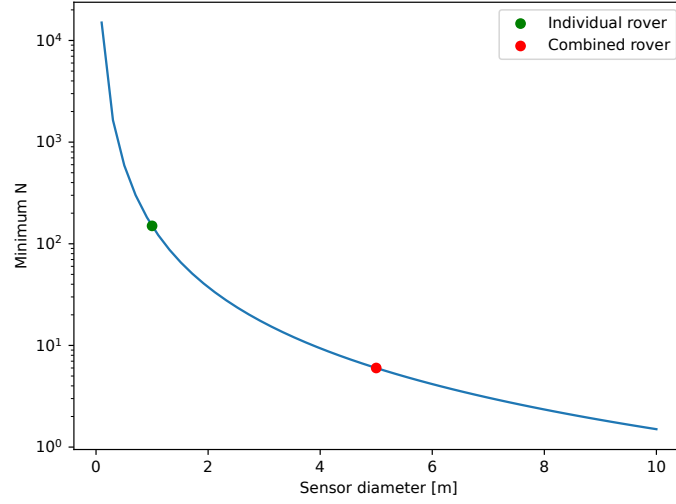


Figure 5.9: Minimum N is required as a function of the sensor diameter.

rovers completed a search mission using the integrand; a random and continuous PDM. The union of the resultant buffered paths was then used as the 2D polygon. Using the trajectories from 105 missions, the aggregated data analysis showed that the cubature approach is substantially more accurate and significantly faster. Both qualities are important in reinforcement learning, where marginal performance gains can quickly snowball into days of saved training time. Furthermore, these results also translate to embedded devices where the computational budget is typically limited. This could be the case for onboard navigation systems on a self-sufficient rover similar to the one used in this chapter.

Overall, the results of this chapter highlight the superiority of the cubature approach in terms of accuracy and computational efficiency, particularly in scenarios requiring coordinated search missions like those encountered in reinforcement learning and embedded systems for autonomous navigation. However, it is important to acknowledge the limitations of this chapter. While the focus was on comparing two prevalent integration methods, numerous other techniques warrant exploration. As highlighted in Sect. 5.2, other potentially faster but less adaptable cubature algorithms exist, which could result in an even greater performance increase. Additionally, the requirement of a continuous integrand for the cubature method may pose challenges in certain scenarios, though potential solutions using approximations have been discussed. Furthermore, it is important to note that the influence of problem characteristics (deterministic or stochastic, for example) on the comparison remains unexplored. By recognising these limitations, future work can build upon these findings to further enhance integration techniques for a wide range of applications.

Chapter 6

Exploring The Efficacy Of Reinforcement Learning For Search Planning

6.1 Section Overview

This chapter introduces a novel algorithm using RL to create efficient search paths for UAVs in the WiSAR environment. This approach leverages a priori data about the search area and the LP in the form of a PDM. This allows the RL policy to learn optimal flight paths that maximise the probability of finding the LP quickly. Experimental results show that this method significantly improves search times compared to traditional coverage planning and search planning algorithms by over 160 %, a difference that can mean life or death in real-world search operations. Additionally, unlike related work from the literature, this approach incorporates a continuous action space enabled by cubature, allowing for more nuanced flight patterns.

This chapter closely follows the work by the thesis author of the peer-reviewed journal paper ‘Deep Reinforcement Learning for Time-Critical Wilderness Search and Rescue Using Drones’ [45].

6.2 Introduction

UAVs offer a cost-effective and agile solution for aerial search. Both PSASU and SMR are placing small fleets of UAVs around Scotland for rapid deployment in a WiSAR scenario. The pilot-observer model is the current approach to flying UAVs for PSASU, where two personnel are required at a minimum per UAV. In this setup, the observer maintains a visual line of sight while the pilot can fly the UAV and inspect the live camera feed. Koester *et al.* [16] identifies that a foot-based searcher has a higher detection rate when not in motion, similar to the behaviour exhibited by pilots [17]. The cognitive load of being in motion while searching is evidently a barrier for pilots,

and offloading one aspect of this may lead to efficiency gains.

To address this challenge and free up precious human resources, search planning algorithms are employed to optimise UAV flight paths [141]. RL’s success in other domains, such as video games and UAV racing [171], [230], suggests its potential for improving WiSAR search planning. The ability of RL to generalise the problem and make decisions that maximise future gains based on extensive training allows it to provide unique solutions to the problem.

WiSAR search planning is an information-rich task and the effective utilisation of prior information — such as the PLS, age, and fitness levels of the LP— is critical. This information can be used to generate a PDM, which describes the probability of detecting the LP at a given location and informs the search stage of the mission. The search problem can be classified as a coverage or exploration problem without this prior information. The former assumes a uniform PDM whilst the latter develops an understanding of the environment in real-time. There is a multitude of algorithms that can generate the PDM [42], [46], [77] and as such, it can be assumed to be a known quantity during search planning.

The core contributions of this chapter to the field are thus:

- the proposal of a novel application of RL to the search planning problem which can outperform benchmark algorithms from the literature [22], [155];
- the use of a continuous PDM, as opposed to an image-based one [22], [155] in order to prevent undesired noisy rewards [259], [260]; and
- a framework for calculating the accumulated probability over the search path through cubature.

Related work is discussed in Sect. 6.3, and the methodology is presented in Sect. 6.4. Results are shown in Sect. 6.5, and a conclusion is drawn in Sect. 6.6.

6.3 Related Work

Lin *et al.* [155] approach the search planning problem by using a gradient descent algorithm in the form of LHC that can advance into any of the surrounding eight cells. However, as discussed in Sect. 2.2.4, LHC alone is not sufficient. Waharte *et al.* [154] found that this class of algorithms do not perform well due to their propensity to get stuck around local maxima. For this reason, Lin *et al.* [155] introduces the notion of global warming to break out of local maxima. This raises the zero probability floor sequentially several times, storing the paths and then reassessing them given the original PDM. Through this and a convolution-based tie-breaking scheme, LHC_GW_CONV is shown to have significantly favourable results. However, only the adjacent areas are considered at every time step.

In order to consider the area as a whole, sampling-based optimisation approaches have been applied to the problem. Morin *et al.* [163] uses ant colony optimisation with a discrete PDM and Ewers *et al.* [17] uses both genetic algorithm and particle swarm optimisation with a pseudo-continuous PDM. However, due to sampling-based optimisation problems, they are prone to long computation times when convergent on a solution.

A core problem with the previously mentioned algorithms is the inability to consider the PDM as a whole when making decisions. Being able to prioritise long-term goals over short-term gains is a key feature of RL.

RL is being used extensively for mission planning, such as by Yuksek *et al.* [172], who used proximal policy optimisation to create a trajectory for two UAVs to avoid no-fly-zones whilst tracking towards the mission objective. This approach has defined start and target locations. However, using no-fly-zones with constant radius is analogous to an inverted PDM. Peake *et al.* [191] uses Recurrent-DDQN for target search in conjunction with A2C for region exploration with a single UAV to find missing people. This method does not use a priori information but explores the area in real-time. This shows that RL is a suitable approach to the search-over-PDM problem that a WiSAR mission requires.

As highlighted in Sect. 2.2.6, search and exploration planning are different problems. Exploration planning has seen many different planning algorithms, such as in work by Zhang *et al.* [140], which uses a partially observable Markov decision process and environment discretisation to handle exploration and search in near real-time. Similarly, Talha *et al.* [20] and Peake *et al.* [191] use RL with environment discretisation to explore the environment whilst searching. Ebrahimi *et al.* [173] also uses RL but localises missing people using radio signal strength indexes without prior knowledge of the searchable domain. These algorithms do not have any a priori knowledge available during the planning stage other than what is discovered in real-time; a different problem.

A core aspect of RL is having a fully observable environment such that the policy can infer why the action resulted in the reward. Thus, being able to represent the PDM effectively is a primary goal. Images can be used as inputs for RL, as done by Mnih *et al.* [230] to play Atari 2600 computer games, but the typically large dimension can be prohibitive. However, finding a different representation is required since PDM generation algorithms most commonly create discrete maps [42], [77]. This was highlighted by Tbl. 2.9 in Sect. 2.2.6 with all ML search planning methods using a discrete PDM. To go from a discrete to continuous PDM, Lin *et al.* [138] uses the Gaussian mixture model to represent the PDM as a sum of bivariate Gaussians. This can be easily used to numerically represent the numerous bivariate Gaussian parameters in an array, which is a suitable format for a RL observation.

There are many RL algorithms to choose from, with PPO [222] and SAC [223] being some of the most prevalent in the literature [172], [282], [283]. Mock *et al.* [282] found

that PPO performed well for low-dimension observation spaces, whilst SAC performed much better for larger ones. The need for a large observation space comes from the fact that the policy would need to have a sense of memory regarding where it had been to encounter *unseen* probability to satisfy the Markov decision process that underpins RL. Mock *et al.* [282] found that a recurrent architecture was comparable to including previous states in the observation (also known as frame stacking). This shows that frame-stacking with SAC is a suitable RL architecture for the current problem.

6.4 Method

This section outlines the RL-based search planning method for this chapter. The environmental model is defined in Sect. 6.4.1.1, followed by the definition of the PDM in Sect. 6.4.1.2. The reward function, and RL setup are then defined in Sect. 6.4.1.3, and Sect. 6.4.2 respectively.

6.4.1 Modelling

6.4.1.1 Environment

The low-level tasks of control [284]–[286], trajectory generation [287], and obstacle avoidance [63], [275], [288] can be assumed to be of a high enough standard to achieve perfect waypoint mission execution performance. Whilst this assumption does not fully hold in practice, since search platforms operate in imperfect environments, the speed of the search platform during a search mission is slow which increases the probability of detection, and this in turn results in sufficient tracking accuracy to ignore this source of error. Nonetheless, the problem of imperfect waypoint mission tracking is partially explored later in Ch. 8.

The UAV within the environment is therefore modelled as the velocity-heading model with a constant step size Δt . The state is represented by the 2D position vector $\mathbf{x} = [x, y]^T \in \mathbb{R}^2$ and its dynamics are described by the non-linear system

$$\begin{aligned} \dot{\mathbf{x}} &= \lambda \begin{bmatrix} \cos(\pi(a_t + 1)) \\ \sin(\pi(a_t + 1)) \end{bmatrix} \\ \mathbf{y} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}, \end{aligned} \quad (6.1)$$

where $a_t \in [-1, 1]$ is the control action at time-step t and the Euler integration scheme, $\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta t \dot{\mathbf{x}}$.

From Eqn. 6.1 it is clear that the state $\mathbf{x}_{t=T}$ is dependent on the states from $t = 0$ to $t = T$. This makes this model suitable for formulating the UAV's motion as a MDP for use within RL.

The reward function and training of the optimal policy are defined in Sect. 6.4.1.3

and Sect. 6.4.2 respectively.

6.4.1.2 Probability Distribution Map

In a real WiSAR scenario, algorithms such as from Hashimoto *et al.* [42], and Šerić *et al.* [77] can be employed to generate the PDM given the search mission profile — PLS, terrain, profile of the LP, and more. This data is not publicly available in any meaningful quantity and is thus unusable for ML in this scenario. Therefore, as common in the search planning literature, the PDM is randomly generated for training and evaluation.

The MMBG from Eqn. 5.5 is used to model the PDM. Parameter values for the random parameter instantiation from Eqn. 5.6 are discussed in Sect. 6.5.1.

If the bounding area were infinite, that is $x_{\min} = y_{\min} = -\infty$ and $x_{\max} = y_{\max} = \infty$, then $\sum p(\mathbf{x}) = 1$. However, as seen from Fig. 6.1, the area enclosed by the rectangular bounds contains less than this. Sect. 6.4.1.3 further discusses handling this such that the available probability is normalised.

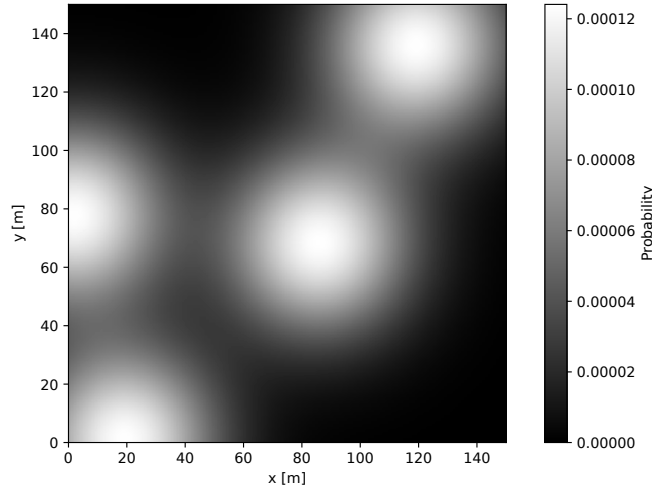


Figure 6.1: An example MMBG PDM.

6.4.1.3 Reward

As the agent moves a constant distance λ m every step, it is assumed that the camera follows this path continuously at a fixed height whilst pointing straight down at all times. Therefore, to represent the *seen area* for a given path at time-step t , the path is buffered by R_{buffer} m to give the polygon H_t . All probability from the PDM enclosed within H_t is then *seen* and denoted by p_t . This value, the seen probability, is calculated through

$$I(C) = \oint_C f(\mathbf{x})dC, \quad (6.2)$$

from Eqn. 5.1. Substituting $C = H_t$ and Eqn. 5.5, defined as

$$p(\mathbf{x}) = \frac{1}{N_{\text{gaussian}}} \sum_{i=0}^{N_{\text{gaussian}}} \frac{\exp\left\{\left[-\frac{1}{2}(\mathbf{x} - \mu_i)^T \sigma_i^{-1}(\mathbf{x} - \mu_i)\right]\right\}}{\sqrt{4\pi^2 \det(\sigma_i)}}, \quad (6.3)$$

gives

$$p_t = \oint_{H_t} p(\mathbf{x}) dH_t. \quad (6.4)$$

Our goal is to maximise the captured probability mass. To gain insight into how the agent's path affects this, the focus is first on the local behaviour of p_t . The area covered by the agent in two steps is analysed, as this provides a foundation for understanding more complex paths. The following lemma demonstrates a crucial property of this two-step area in the simplified case of a uniform PDM.

Lemma 6.1. *For a uniform PDM, $p(\mathbf{x}) = 1$, the area $A(\theta)$ of the region H after two steps, as defined by*

$$A_2(\theta) = \underbrace{4sR_{\text{buffer}}}_{\text{Main Area}} + \underbrace{\pi R_{\text{buffer}}^2}_{\text{Semi-Circle}} + \underbrace{\frac{1}{2}R_{\text{buffer}}^2\theta}_{\text{Rounded corner } g(\theta)} - \underbrace{R_{\text{buffer}}^2 \tan\left(\frac{\theta}{2}\right)}_{\text{Overlap } f(\theta)}, \quad (6.5)$$

$$\forall \theta \in \left[0, 2 \arctan\left(\frac{s}{R_{\text{buffer}}}\right)\right]$$

where R_{buffer} and λ are constants, is maximised when $\theta = 0$ rad.

Proof. Assume, for the sake of contradiction, that there exists a $\theta^* \in \left[0, 2 \arctan\left(\frac{s}{R_{\text{buffer}}}\right)\right]$ such that $A_2(\theta^*) > A_2(0)$. The derivative of A_2 with respect to θ is

$$\frac{dA_2}{d\theta} = \frac{1}{2}R_{\text{buffer}}^2 \left(1 - \sec^2\left(\frac{\theta}{2}\right)\right). \quad (6.6)$$

Since $1 - \sec^2(x) \leq 0 \forall x \in \mathbb{R}$, it follows that $\frac{dA_2}{d\theta} \leq 0 \forall \theta \in \left[0, 2 \arctan\left(\frac{s}{R_{\text{buffer}}}\right)\right]$. Because the derivative is non-positive, $A_2(\theta)$ is a monotonically decreasing function on the given interval. Since $A_2(\theta)$ is monotonically decreasing, for any $\theta^* > 0$ rad, it must be the case that $A_2(\theta^*) \leq A_2(0)$. Our assumption that there exists a θ^* such that $A_2(\theta^*) > A_2(0)$ must be false. Therefore, the maximum value of $A_2(\theta)$ is achieved when $\theta = 0$ rad. $\square$

Further insights can be garnered by applying Green's theorem

$$\oint_C (Ldx + Mdy) = \int \int_D \left(\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y}\right) dA, \quad (6.7)$$

with $\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y} = 1$. This shows that decreasing the boundary $C = H_2$ reduces the area of the region D bounded by C . With a uniform PDM, maximising the geometric area is equivalent to maximising the captured probability mass. From Lem. 6.1, the

buffered polygon formulation maximises the integral when $\theta = 0$ for a uniform PDM. However, if $\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y}$ is not constant, it could be beneficial to increase θ and therefore reduce the area in order to maximise the encapsulated values.

Special consideration must be taken for the case where $\frac{\lambda}{R_{\text{buffer}}} < \frac{\pi}{2}$ as Lem. 6.1 does not hold and must be further explored. This constraint, however, is always met in this chapter.

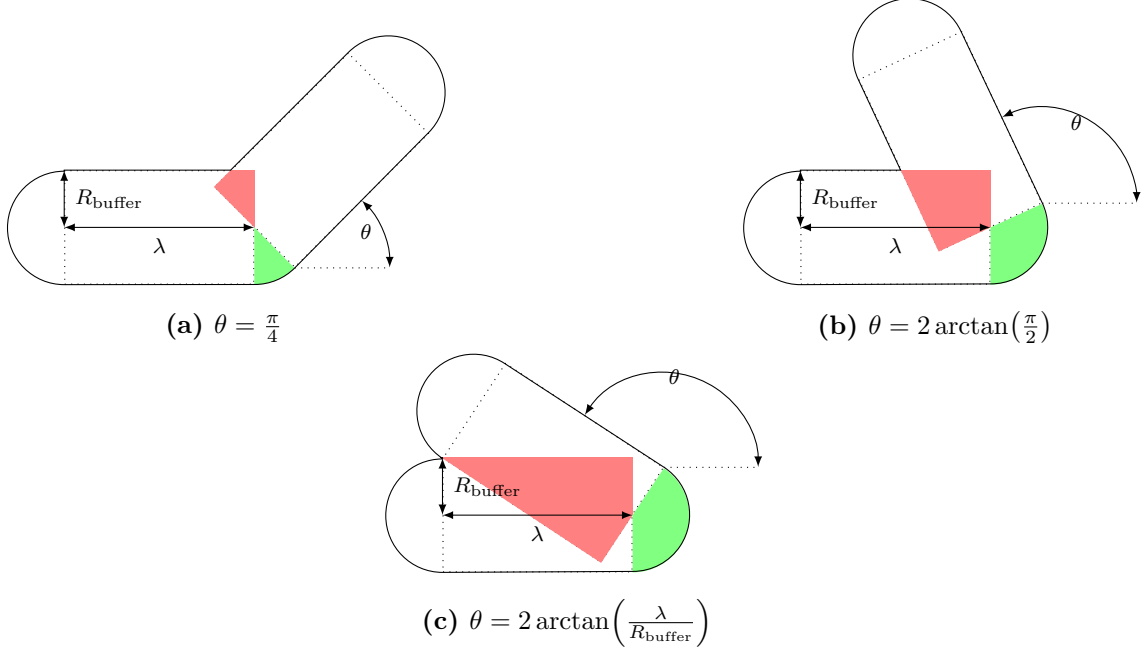


Figure 6.2: Visualising a generalised subsection of search path H_t after two steps (H_2) with different θ . The areas coloured in red and green represent $f(\theta)$ and $g(\theta)$ from Eqn. 6.5 respectively.

The action is correlated to the reward by considering the change in accumulated probability at time t , defined as

$$\Delta p_t = p_t - p_{t-1}. \quad (6.8)$$

To normalise this change in accumulated probability, the scaling constants k is introduced and p_A . Constant k relates the area of a single isolated step to the total search area, $A_{\text{area}} \text{m}^2$. Simplifying Eqn. 6.5 to the single step case using the constants defined in Fig. 6.2, gives the ratio

$$k = \frac{A_{\text{area}}}{R_{\text{buffer}}(\pi R_{\text{buffer}} + 2\lambda)}. \quad (6.9)$$

Constant p_A is the total probability enclosed within the total search area given by substituting $H_t = A$ in Eqn. 6.4. This gives the scaled probability efficiency reward

$$r = \frac{k}{p_A} \Delta p_t. \quad (6.10)$$

The ratio of change in accumulated probability to total probability enclosed within

the search area satisfies the constraint that $0 \leq \frac{\Delta p_t}{p_A} \leq 1$. This ratio is the probability efficiency, $e_{p,t}$, and provides a valuable insight into the performance of a given path.

Further reward shaping is applied to discourage future out-of-bounds actions (w_{oob}) and to penalise visiting areas of very low probability (w_0). The augmented reward r' is given by

$$r' = \begin{cases} -w_{oob}, & \mathbf{x}_t \notin [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \\ w_r r, & \Delta p_t > \epsilon \\ -w_0, & \text{else} \end{cases}. \quad (6.11)$$

6.4.2 Reinforcement Learning

SAC, from Sect. 3.3.3, is used within this chapter due to its superior performance over large observation spaces [282]. The core of the policy network $\pi_{\theta, \text{core}}$ consists of a fully connected network with N_{layers} layers, each with a width of N_{width} . The observation definitions are given in Tbl. 6.1 resulting in $2N_{\text{waypoints}} + 6G + 4$ observation inputs. The policy π_ϕ is constructed to handle multiple input observation spaces and is defined in Fig. 6.3. The inner workings of the policy are defined in Alg. 6.1.

Table 6.1: Definition of the five state observations.

Sub-state	Symbol	Definition
Path	s_{path}	$(\mathbf{x} \mid 0 \ 0^{2 \times N_{\text{waypoints}} - t})^T$
PDM	s_{PDM}	$[\mu_0, \sigma_0, \dots, \mu_G, \sigma_G]^T$
Position	s_{pos}	$\mathbf{x}_t$
Out-of-bounds	s_{oob}	$\mathbf{x}_t \in [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$
Number of steps	s_{steps}	t

Algorithm 6.1: Policy Network π_ϕ Input Process

Input : Observations defined in Tbl. 6.1

Output: Action $\mathbf{a}$

- 1 Extract PDM latent space $z_{\text{PDM}} \leftarrow \pi_{\theta, \text{PDM}}(s_{\text{PDM}})$;
 - 2 Extract path history latent space $z_{\text{path}} \leftarrow \pi_{\theta, \text{path}}(s_{\text{path}})$;
 - 3 Concatenate remaining observations and latent spaces
 $z \leftarrow z_{\text{PDM}} \cup z_{\text{path}} \cup [s_{\text{oob}}, s_{\text{pos}}, s_{\text{steps}}]$;
 - 4 Sample from policy to get action $\mathbf{a} \sim \pi_{\theta, \text{core}}(z)$;
-

6.4.2.1 Policy Network Design

Search planning is an abstract task compared to point-to-point or coverage planning. Therefore, a parameter sweep for the network architecture was performed to ensure sufficient capabilities. A core aspect of this was the large path history observation with $2N_{\text{waypoint}}$ elements since the policy must be able to learn how to avoid self-intersection to avoid penalties.

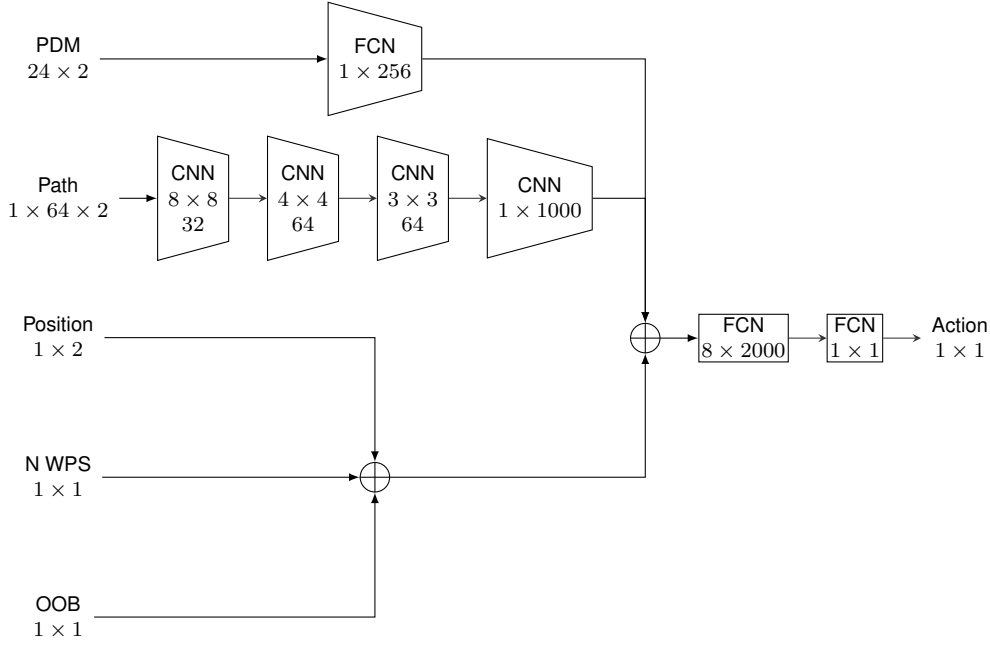


Figure 6.3: Policy network π_ϕ . $\pi_{\theta, \text{path}}$ uses a 2D CNN architecture from Mnih *et al.* [230].

In this sweep, four variables were tuned: number of layers N_{layers} , layer width N_{width} , path output feature dimension, and path feature extractor. N_{layers} and N_{width} correlate to the core policy network, and the other feature extractors are left unchanged and can be seen in Fig. 6.3. The ranges of these sweeps can be seen from Fig. 6.4. The SAC hyperparameters from the main paper were used for all runs.

Three path feature extractors were tested. A standard FCN-based approach (Fig. 6.5a), a 2D CNN from [230] (Fig. 6.5c), and a 1D CNN variant (Fig. 6.5b).

In total, 39 runs were completed over 12 days in the same computational environment as outlined in Sect. 6.5.1. A random sampling strategy was used to ensure broad coverage of the hyperparameter space. Results from this, with a fitted ordinary least squares linear regression where applicable, can be seen in Fig. 6.4.

From the results in Fig. 6.4a and Fig. 6.4b it is evident that a deeper, wider network leads to better results. $N_{\text{layers}} = 6$ has a high mean performance, above $e_{p_D} = 0.15$, but sees an outlier with very low performance and thus an overall higher standard deviation. Fig. 6.4b does taper off in performance after $N_{\text{layers}} = 8$ with no notable improvement of the results.

The overall number of parameters per result was calculated and plotted in Fig. 6.6. This confirms the result that a larger policy network is better. A network size of 8×2000 was selected as seen in Fig. 6.3.

Results from the path output feature dimension in Fig. 6.4d would be expected to show similar results. However, the ordinary least squares has a downward trend implying lower values for being better. Nevertheless, the fitted model did not exhibit statistically significant explanatory power, with a p-value of 0.393. For comparison, the p-values for N_{layers} and N_{width} were 0.00522 and 0.00339, respectively, well below

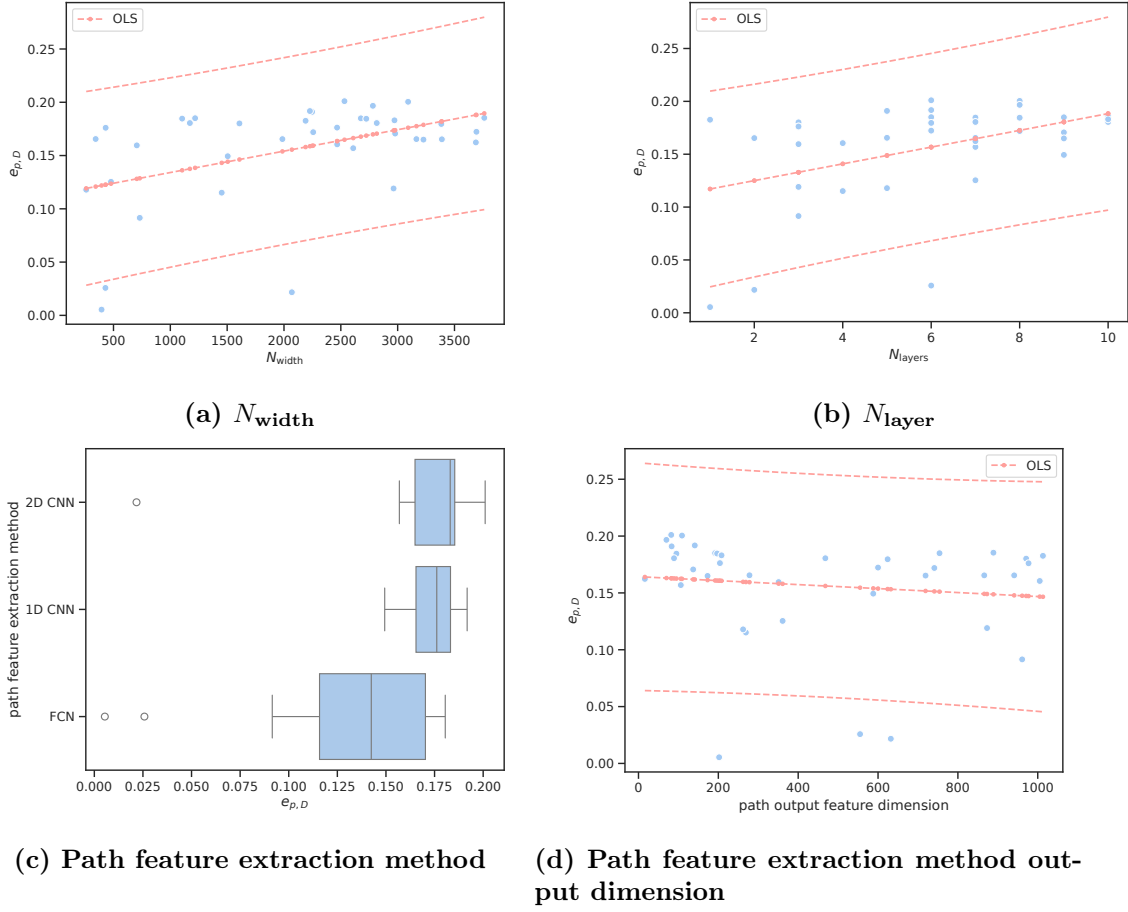


Figure 6.4: Ordinary least squares linear regression fitted to the $e_{p,D}$ results from the hyperparameter sweep. (a) (b) (d): The upper, and lower dashed lines represent the confidence interval, and the middle dashed line represents the fitted linear regression. The dots show the individual data points.

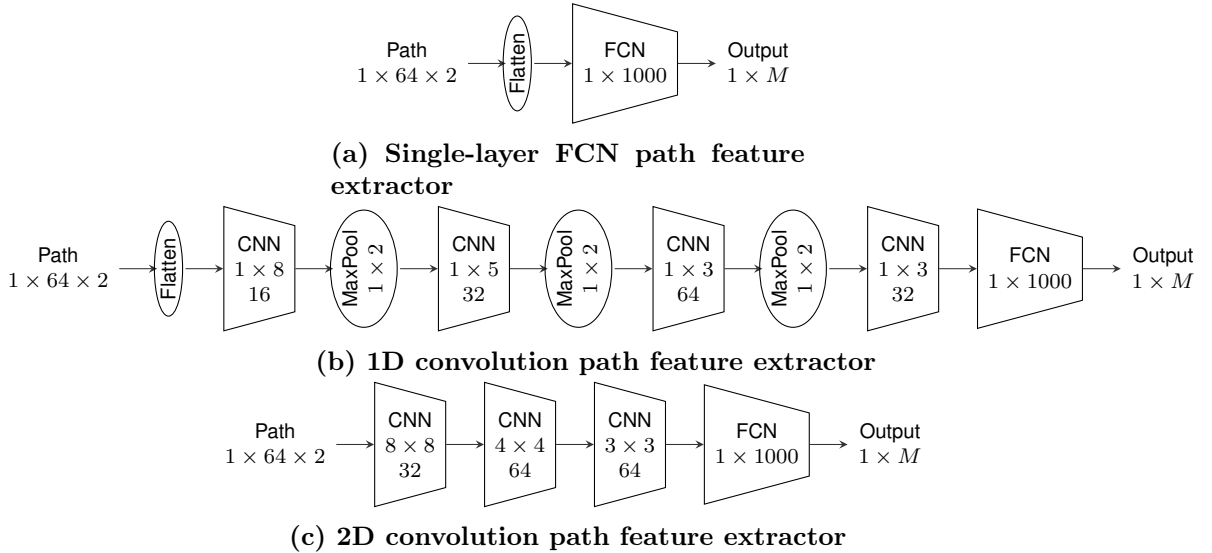


Figure 6.5: Architectures of the three candidate path feature extractors. M is the path output feature dimension.

the standard test p-value of 0.05. Thus, a path output feature dimension of 1000 was selected.

Finally, the result for the path feature extraction method in Fig. 6.4c highlights

that the 2D CNN feature extractor yields the highest median $e_{p,D}$. However, the 1D CNN median result is very similar. A p-value was calculated against each distribution to determine whether the results differed significantly.

Fig. 6.7 shows that distribution of 2D CNN results is statistically significantly different from the rest with p-values below 0.05. Therefore, 2D CNN was used as the path feature extractor as seen in Fig. 6.3.

A limitation of this analysis approach is the need to assume that the results are independent. This is not necessarily true, as changing the feature extractor might change the requirements for N_{width} . Furthermore, the high p-value score for the path feature extraction method implies that more runs could have been undertaken. It also implies that this value is not important to the network. However, a more complex analysis is outside the scope of this chapter.

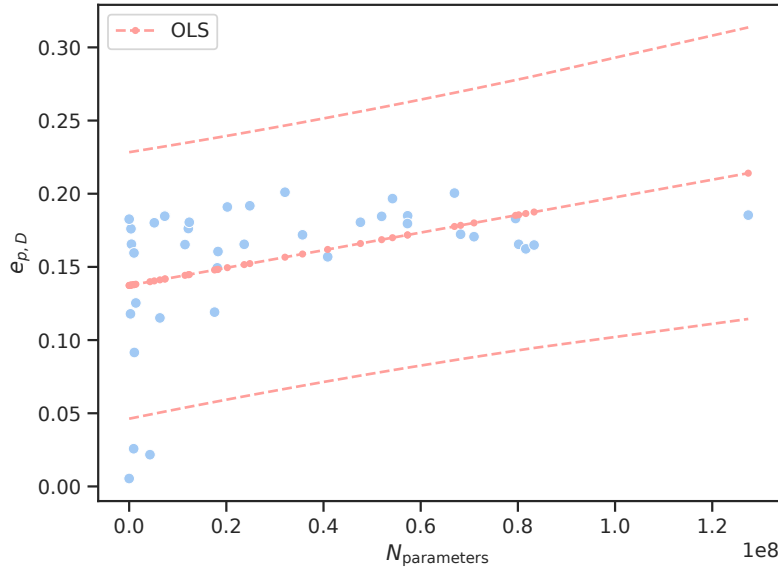


Figure 6.6: $e_{p,D}$ with the number of parameters within the policy network. This value was derived from Fig. 6.4a and Fig. 6.4b. The upper, and lower dashed lines represent the confidence interval, and the middle dashed line represents the fitted linear regression. The dots show the individual data points.

6.5 Results

6.5.1 Experimental setup

In order to effectively benchmark the proposed algorithms, two additional baselines are implemented: lawnmower [37] and LHC_GW_CONV [155]. These were chosen due to the former being ubiquitous for coverage planning and the latter being an optimisation-based implementation that struggles to explore the PDM fully. To ensure compatibility in the comparison to the proposed algorithm, the parallel lines for the lawnmower are offset by the step size s and the grid dimensions for LHC_GW_CONV are $(x_{\max} - x_{\min}, y_{\max} - y_{\min})/\lambda$. The maximum number of waypoints N_{waypoint} are converted to a

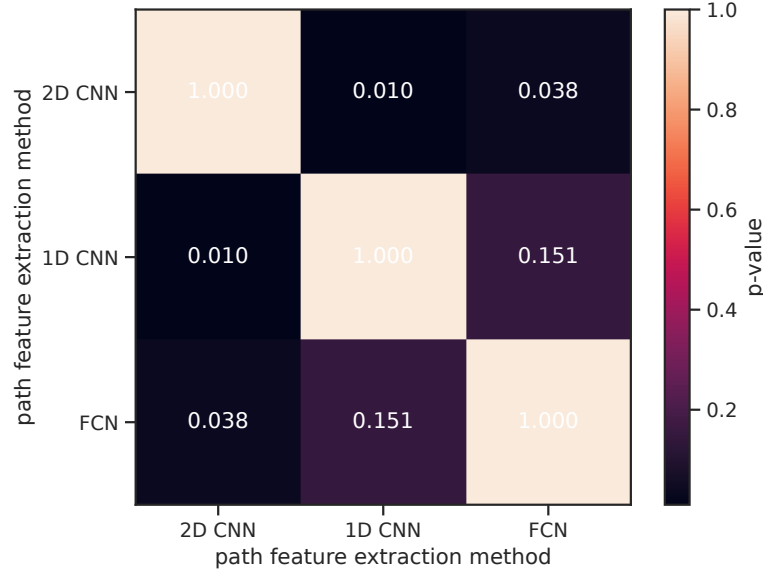


Figure 6.7: p-value matrix from comparing the result distributions categorised by feature extraction method.

maximum distance $D_{\max} = \lambda N_{\text{waypoint}}$ and the generated paths are truncated at this point.

The results for the algorithm implemented in this research, soft actor-critic frame stacking convolution neural network, and titled SAC-FS-CNN from here on in, is the cumulation of three separate training runs with random starting seeds. This aligns with the best practices outlined by Agarwal *et al.* [289] to ensure robust analysis for RL results. Each model was trained for a minimum of 21 days (5×10^8 global steps) with 32 workers on a local Ubuntu 22.04 machine with a AMD Ryzen 9 5950X CPU, a NVIDIA RTX A6000 GPU, and 64GB of RAM.

One evaluation of an algorithm involves generating the random PDM and then creating the resultant search path. This is labelled one run. Each algorithm was evaluated at least 5×10^3 times, and this generated data is based on the following analysis.

Tbl. 6.2 shows the SAC hyperparameters used in this chapter. A low τ and learning rate were selected for a high-level stability. Typically the learning rate for the adaptive moment estimation (Adam) optimiser is 0.001 but this resulted in unstable training. A high number of gradient steps were used to ensure that the policy learns as much from the large buffer, before the target networks shift.

Similarly, Tbl. 6.3 shows the simulation parameters used in this chapter. The x - y range of 0m to 150m provides a controlled arena large enough for meaningful navigation but small enough to prevent sparse-reward issues during early training. The search radius R_{buffer} and step size λ were then selected to ensure that around only 10% of the area was searchable during any single iteration. This ensures that the emphasis is on effective area prioritisation, not complete coverage. The weighting of 1.0 for out-of-bounds w_{woo} compared to 0.5 for progress (w_r) and zero-information (w_0),

Table 6.2: SAC hyperparameters used for this chapter from empirical testing. Other variables were kept at the default values from Raffin *et al.* [290] v2.1.0.

Hyperparameter	Value
Learning rate	1×10^{-6}
Optimiser	Adam
Batch size	1024
Learning starts	8192
Buffer size	5×10^6
Training frequency	10
Gradient steps	50
τ	1×10^{-4}

prioritises survival and staying within the task bounds as the primary objective for the baseline agent. A static number of bivariate Gaussians ($N_{\text{gaussian,min}} = N_{\text{gaussian,max}}$) was required for the fixed-length vector PDM observation defined in Tbl. 6.1

Table 6.3: Simulation parameters used for this chapter.

Parameter	Value	Unit
$N_{\text{gaussian,min}} / \text{max}$	4	
$\sigma_{\text{min}} / \text{max}$	500	
$x_{\text{min}}, y_{\text{min}}$	0	m
$x_{\text{max}}, y_{\text{max}}$	150	m
λ	8	m
R_{buffer}	2.5	m
N_{waypoint}	64	
ϵ	0.1	
w_{oob}	1.0	
w_r	0.5	
w_0	0.5	

6.5.2 Percentage Of Detection

Maximising the probability efficiency (Eqn. 6.10) at all times is critical. This directly correlates to increasing the chances of finding a LP in a shorter time. The *percentage of detection* of SAC-FS-CNN must always outperform the benchmark algorithms. If this is not the case, the search algorithm selection becomes dependent on the endurance and mission. However, if the percentage of detection is always better, one algorithm will be superior regardless of application. To calculate the percentage of detection, the probability efficiency is evaluated at and up to

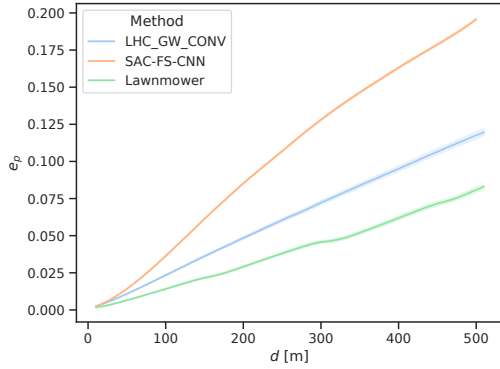
$$d = \frac{(N_{\text{steps}} - i)D}{N_{\text{steps}}} \forall i \in \{N_{\text{steps}}, N_{\text{steps}} - 1, \dots, 1, 0\}, \quad (6.12)$$

with $N_{\text{steps}} = 50$.

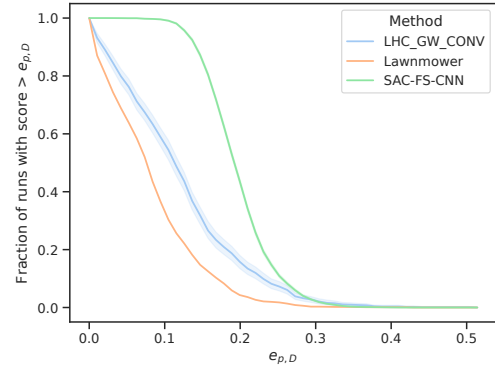
From Fig. 6.8a it is clear that SAC-FS-CNN sufficiently outperforms the benchmark algorithms at all distances. This is further highlighted by the $e_{p,D}$ for SAC-FS-CNN at 238 % of that of lawnmower, and 158 % for LHC_GW_CONV from Tbl. 6.4. This is corroborated by the median $e_{p,D}$ values in Fig. 6.8c. Notably, however, LHC_GW_CONV has a substantial amount of high $e_{p,D}$ outliers.

Likewise, the performance profile from Fig. 6.8b follows the trend. It can be seen that SAC-FS-CNN has close to 100 % of runs with $e_{p,D} > 0.1$ and 50 % at approximately $e_{p,D} > 0.2$. This aligns with results from Fig. 6.8c and Tbl. 6.4.

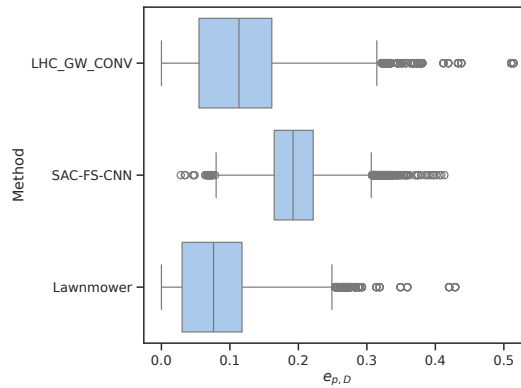
It is of note that LHC_GW_CONV has the most extensive range of values going from 0.0 to 0.5 whilst SAC-FS-CNN only goes from 0.02 to 0.41. This shows that LHC_GW_CONV can perform very well given the right PDM or poorly given the wrong one. On the other hand, the RL approach of SAC-FS-CNN does not suffer from this due to its ability to find a general solution to the problem.



(a) Mean e_p with d showing SAC-FS-CNN outperforming the benchmark algorithms at all distances.



(b) Performance profile of $e_{p,D}$. SAC-FS-CNN has a close to 100 % of runs with $e_{p,D} > 0.1$ compared to the < 50 % for the other algorithms.



(c) Median $e_{p,D}$ for all runs. SAC-FS-CNN has a higher median than the other algorithms. LHC_GW_CONV registers large outliers.

Figure 6.8: Probability efficiency analysis over all runs. The shaded regions in Fig. 6.8a and Fig. 6.8b show the 95 % confidence interval.

Table 6.4: Mean percentage of detection with the standard deviation as error.

Method	p_D	$e_{p,D}$	N
LHC_GW_CONV	0.09 ± 0.06	0.12 ± 0.08	9.3×10^3
Lawnmower	0.06 ± 0.04	0.08 ± 0.06	10×10^3
SAC-FS-CNN	0.15 ± 0.04	0.19 ± 0.04	9.8×10^3

6.5.3 Distance To Find and Percentage Found

Whilst percentage of detection shows the theoretical effectiveness of an algorithm, the intended use case is finding a LP whilst searching within a bounded area. The mission statement is to reduce the time it takes to find the potentially vulnerable person to save lives.

To quantify this requirement, *distance to find* and *percentage found* are introduced. The former gives a clear answer on the capabilities of the various algorithms, whilst the latter should align with the percentage of detection results from Tbl. 6.4 for validation.

Firstly, Gumbel-Softmax [291] is used to sample N_{samples} positions from the PDM to give the set $\chi \in \mathbb{R}^{2 \times N_{\text{samples}}}$ containing all samples. The path is then traversed in incremental steps using Eqn. 6.12 with $N_{\text{step}} = 1 \times 10^4$. At each step, an Euclidean distance check is done from the current position $\mathbf{x}$ to each entry in χ with any points within R_{buffer} marked as seen. The updated set of positions to search for in the next step is then

$$\chi' = \{\chi_i \in \chi : |0\mathbf{x} - \chi_i| > R_{\text{buffer}}\}. \quad (6.13)$$

From Fig. 6.9, it is clear to see that SAC-FS-CNN outperforms the benchmark algorithms with a lower median distance to find as well as a lower inter-quartile range. Tbl. 6.5 shows that the mean distance to find is 15.22% lower than lawnmower, and 4.07% lower than LHC_GW_CONV. This aligns with expectations from the results in Sect. 6.5.2. Likewise, the percentage found values closely match the $e_{p,D}$ values from Tbl. 6.4, showing that this test correlates to the theory.

Whilst SAC-FS-CNN outperforms the benchmarks in the median distance to find, it is of note that the variances of the three algorithms are almost identical as shown by the whiskers in Fig. 6.9. This is due to how the positions are sampled from the random PDM, making it likely for a minimal subset of positions near the start and end of the path. This is evident because the variances of the three algorithms range from 0 to 512; the minimum and maximum simulation distances respectively.

Table 6.5: Mean distance to find with the standard deviation as error.

Method	Percentage found [%]	Mean Distance To Find [m]	N
LHC_GW_CONV	11.86 ± 0.32	249.37 ± 145.70	5×10^6
Lawnmower	7.77 ± 0.27	282.63 ± 146.62	5×10^6
SAC-FS-CNN	19.00 ± 0.39	239.61 ± 138.16	10×10^6

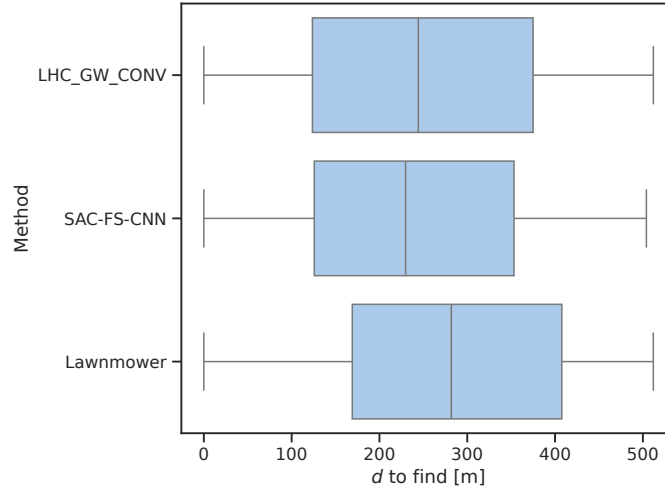


Figure 6.9: Median distance to find. Similarly, the whisker markers across the algorithms show the broad distribution of results from this experiment. SAC-FS-CNN ultimately has the lowest median score.

6.5.4 Area Efficiency

A path with corners has overlapping regions when considering the buffered path, which is evident from Fig. 6.2. The most efficient path in this formulation is thus a straight line such that the area efficiency is $e_{a,D} = 1$. Using Eqn. 6.9, this value is calculated with

$$e_{a,D} = \frac{a_{\text{buffer},D}}{R_{\text{buffer}}(\pi R_{\text{buffer}} + 2Ds)}, \quad (6.14)$$

where $a_{\text{buffer},D}$ is the total area of the buffered path, and D is the number of waypoints in the path.

The aggregated metrics can be seen in Fig. 6.10 which shows a significant difference in area efficiency between the three methods. The lawnmower method consistently achieves the highest area efficiency values, with a median value of 0.97. This suggests that lawnmowers generate paths with minimal overlapping regions within the buffer, resulting in efficient use of the search buffer. However, lawnmower has no variance in area efficiency as it is a coverage planning algorithm and as such always generates the same path. In contrast, the SAC-FS-CNN method demonstrates lower area efficiency, with a median value around 0.90. This indicates that SAC-FS-CNN paths tend to have more overlapping areas within the buffer, leading to suboptimal utilisation. LHC_GW_CONV method exhibits the lowest area efficiency, with a median value of 0.86 due to its inability to make trade-offs now for future gains due to its local hill climbing formulation. However, this is not the case for SAC-FS-CNN as an infinite-horizon discounted reward is at the core of the SAC algorithm, meaning that the current action is taken to maximise the future rewards.

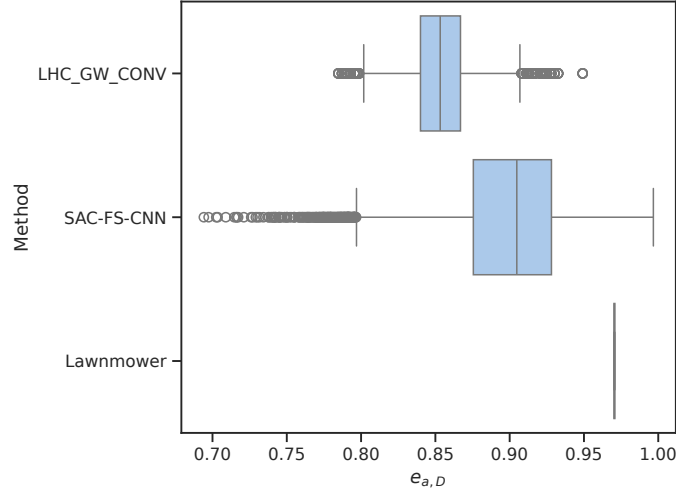


Figure 6.10: Area efficiency of the three algorithms. Lawnmower has many straight segments, which result in a high score of 0.97 and no variance because it does not change. SAC-FS-CNN, on the other hand outperforms the other environment-dependent algorithm LHC_GW_CONV in this metric.

6.6 Conclusion

This chapter proposed SAC-FS-CNN for search planning in WiSAR operations, leveraging a priori information. This was identified as a solution to the challenge of maximising accumulated probability over a given area due to machine learning’s powerful capabilities to identify patterns and generalise complex tasks.

The results indicate that SAC-FS-CNN can outperform benchmark algorithms in the probability efficiency by up to 250 % for lawnmower, and 166 % for LHC_GW_CONV. A similar trend is identified when comparing mean distance to find with RL outperforming the aforementioned algorithms by 15.22 % and 4.07 % respectively. The critical result, however, was that SAC-FS-CNN found 160 % more simulated missing people than LHC_GW_CONV over the same endurance limit. This translates to a substantial advantage in locating missing individuals, potentially saving countless lives during WiSAR operations.

While SAC-FS-CNN exhibits lower area efficiency compared to lawnmower, this trade-off is justified by its superior performance in terms of percentage of detection and distance to find. Future work could focus on improving area efficiency while maintaining its strong performance in these critical metrics.

The integration of RL into WiSAR mission planning holds great potential for the future of search, offering a powerful tool that can significantly increase the success rate of WiSAR efforts.

Chapter 7

Recurrent Auto-Encoders For Enhancing Reinforcement Learning Based Search Planning

7.1 Section Overview

WiSAR operations are often carried out over vast landscapes. The search efforts, however, must be undertaken in minimum time to maximise the victim’s chance of survival. Whilst the advent of cheap multicopters in recent years has changed the way search operations are handled, it has not solved the challenges of the massive areas at hand. The problem, therefore, is not one of complete coverage but one of maximising the information gathered in the limited time available. In this chapter, it is proposed that a combination of a RAE and RL is a more efficient solution to the search problem than previous pure RL or optimisation approaches. The AE training paradigm efficiently maximises the information throughput of the encoder into its latent space representation, which RL is primed to leverage. Without the overhead of independently solving the problem that the RAE is designed for, it is more efficient in learning the control task. Furthermore, three additional architectures are implemented to compare the main proposed architecture comprehensively. Similarly, both soft actor-critic and proximal policy optimisation are used to provide an insight into the performance of both in a highly non-linear and complex application with a large observation space. Results show that the proposed architecture is superior to the benchmarks, with soft actor-critic achieving the best performance. This model further outperformed work from the literature whilst having below a fifth of the total learnable parameters and training in a quarter of the time.

This chapter closely follows the work by the thesis author of the in-review journal paper *Recurrent Auto-Encoders for Enhanced Deep Reinforcement Learning in Wilderness Search and Rescue Planning* which was submitted for publication in 2025 [50].

7.2 Introduction

Ch. 6 has previously shown the strength of RL over analytical and optimal search planning methods. Work by Talha *et al.* [20] and Peake *et al.* [192] applied RL to a similar problem, however these algorithms explore the environment during the search and do not have the complete PDM at the beginning. This is a reasonable scenario to be in at the start of the search, as local area knowledge, maps, and case studies are all available to a pilot before the search begins. Similarly, PDM generation algorithms, such as outlined in Sect. 2.1, are viable solutions to generate the PDM as an input.

The proposed algorithm builds on the previous efforts in Ch. 6, which had severe limitations due to the policy network architecture. The most major limitation was that frame-stacking constrained the maximum number of waypoints in a search path. Another limitation was the long training time required to train the policy due to its large amount of learnable parameters. This, of course, also results in poor runtime performance during deployment, as multiple gigabytes of memory are consumed to hold the model on potentially resource-limited devices.

Mock *et al.* [282] found that using frame-stacking over recurrent architectures leads to comparable performance. However, in this formulation, the observations from early in the simulation become decreasingly unimportant. This is not the case in the search planning algorithm from Ch. 6, and every step has an equal impact on the next step's reward. Hence, the policy must be able to observe the observations back to $t = 0$ s to maintain the MDP.

In this chapter, the aim is to tackle two of the aforementioned problems: find an alternative to frame-stacking and increase general performance (training times, model size, overall efficacy). In Raffin *et al.* [292] a AE is successfully used within the RL loop whilst in [293] the sequence-to-sequence architecture is introduced for text-based applications. This is the basis to approach the first problem and a possible approach to handle the second. It is hypothesised that by splitting out the feature extraction phase as an observation preprocessing step, the powerful AE training setup can be harnessed and thus reduce the training overhead by not having to learn the feature extraction during the RL phase. This will result in two models (AE and RL policy) working in specialised unison and substantially smaller than in Ch. 6 with better performance.

The contributions to the field in this chapter are then:

- a framework to decrease training times and to significantly reduce the number of learnable parameters, whilst enhancing final performance through the proposed AE and RL architecture;
- the empirical evaluation of frame-stacking and recurrent frameworks for large observations; and
- to further the discussion in comparing PPO and SAC in a large observation space domain.

Related work is discussed in Sect. 7.3. The environmental modelling is presented in Sect. 7.4, whilst the RAE and RL architectures are introduced in Sect. 7.5. Implementation details are outlined in Sect. 7.6, followed by results and discussions in Sect. 7.7 and Sect. 7.8. Finally, a conclusion is drawn in Sect. 7.9.

7.3 Related Work

RL has seen significant application across exploration planning domains. Zuluaga *et al.* [21] focuses on urban scenarios and incorporates frontier exploration into the search task where the agent gathers environmental information over time. Niroui *et al.* [189] uses SLAM with RL to explore a cluttered environment, which has many parallels to mining search in real-time. Similarly, Peake *et al.* [192] applies RL to WiSAR and employs dual-policy RL, DDPG and recurrent A2C, which were trained separately to handle the exploration and trajectory planning. Talha *et al.* [20] also uses two RL policies to handle the navigation and exploration separately. These foundational works notably omit consideration of prior probability distributions for target locations. Ch. 6 explored this scenario and showed that RL outperforms search planning methods from the literature, however, with long training times and too many parameters to be practical.

One potential solution to reducing the training overhead is to use hierarchical RL with multiple more specialised models and policies. As previously outlined, Peake *et al.* [192] and Talha *et al.* [20] apply the dual-policy paradigm but still rely on a single policy to handle the top-level exploration planning. This then implies that the search planning from Ch. 6 can be coupled with the lower-level trajectory or navigation planning from the aforementioned work. However, the same issues arise: the mission planning policy is no better than before.

The architecture in Ch. 6 uses frame-stacking to ensure that the MDP is maintained. This is due to the reward at the current time step being dependent on the position of the agent in all previous time steps. If the agent intersects with the historical path, it is penalised by not gaining any new information. Whilst Mock *et al.* [282] found that frame-stacking and recurrent architectures performed similarly, frame-stacking imposes a hard upper limit on the input size. The buffer size can be increased to overcome this issue, but this has its own problems. If the buffer is not filled during training, then inputs associated with data points far in the future cannot be trained, leading to inefficient — or even unstable — training. Therefore, finding another approach to handle the temporal input is imperative.

In Raffin *et al.* [292], the concept of decoupling the feature extraction for RL is explored. It was found that the proposed method was far superior to the standard single-policy approach in RL. Interestingly, it was found that this method was only slightly better than an AE. AEs have two components: an encoder applying a transformation on the input into a latent space, followed by a decoder approximating the reverse of this process Berahmand *et al.* [294]. However, the method from this work requires

frame-stacking again to work effectively. Pleines *et al.* [295] used a LSTM layer [206] within the policy. However, another approach is to couple this with the aforementioned AE to leverage the sequence-to-sequence architecture proposed in work by Park *et al.* [293] to predict the trajectory of a vehicle through a RAE. The sequence-to-sequence architecture was also used in Cho *et al.* [208] to process complex phrase representations for translations; another domain where there are dependencies on the entire variable length dataset for context.

PPO [222], its recurrent variant recurrent proximal policy optimisation (RPPO) [290], [295], and SAC [229] are widely used in control problems such as by Kaufmann *et al.* [171] and Yue *et al.* [296], as well as in other domains such as video gaming [226]. PPO’s stability and simplicity make it ideal for initial policy convergence in deterministic settings, while SAC’s entropy-driven exploration excels in dynamic environments requiring adaptive action distributions [297]. The contrast between PPO’s bounded policy updates (via advantage function clipping) and SAC’s stochasticity (via the maximum entropy formulation) provides a methodological spectrum to evaluate robustness. PPO has become one of the de facto RL algorithms in the literature; however, Mock *et al.* [282] found that it was unable to cope with higher dimensional observation spaces as well as SAC could.

This chapter addresses the training instability of frame-stacking through a RAE architecture that compresses temporal dependencies into latent states. By integrating sequence-to-vector trajectory encoding with decoupled feature extraction, dynamic adaptation to environmental uncertainty is possible while maintaining compatibility with the dimensionality constraints of the policy networks. This approach uniquely resolves the conflict between long-horizon probabilistic reasoning and fixed-size observation spaces.

7.4 Environment Modelling

The parameters used in this section are presented in Tbl. 7.1, and remain the same as the simulation parameters from Ch. 6 (introduced in Tbl. 6.3) to facilitate performance comparisons.

7.4.1 Agent Dynamics

The agent within the environment is modelled as a heading control model with a fixed step size sm . It is assumed that any physical vehicle, such as a drone in the case of WiSAR, executing this mission can accurately track the waypoints via its controller or operator. The agent’s state is represented by the 2D position vector $\mathbf{x} = [x, y]^T \in \mathbb{R}^2$

Table 7.1: Simulation parameters used in this chapter.

Parameter	Value	Units
N_{gaussian}	4	
σ_i	diag (500,500)	
$x_{\min}, y_{\min}$	0	m
$x_{\max}, y_{\max}$	150	m
λ	8	m
R_{buffer}	2.5	m
N_{waypoint}	64	
ϵ	0.1	
w_{oob}	1.0	
w_r	0.5	
w_0	0.5	

and its dynamics are described by the non-linear system

$$\dot{\mathbf{x}} = s \begin{bmatrix} \cos(\pi(a_t + 1)) \\ \sin(\pi(a_t + 1)) \end{bmatrix} \quad (7.1)$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \quad (7.2)$$

where $a_t \in [-1, 1]$ is the agent's action at time-step t and the Euler integration scheme, $\mathbf{x}_{t+1} = \mathbf{x}_t + \delta t \dot{\mathbf{x}}$, is used with $\delta t = 1\text{s}$.

7.4.2 Probability Distribution Map

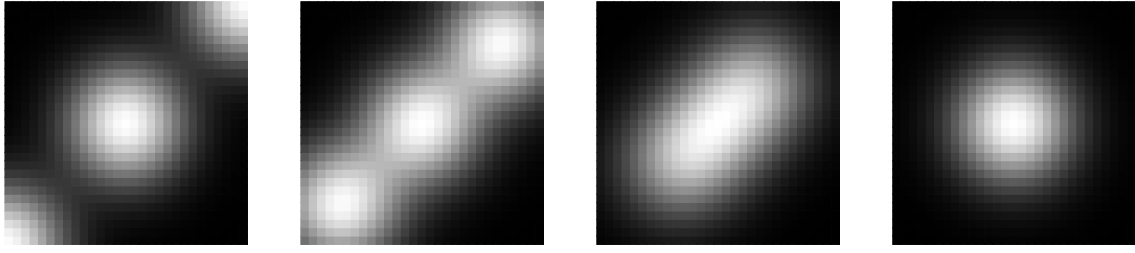
It is assumed that the PDM is known and is modelled as a sum of N_{gaussian} bivariate Gaussians [47]. Yao *et al.* [139] and Yao *et al.* [188] showed that bivariate Gaussians are effective at approximating PDMs. The agent at position $\mathbf{x}$ thus has a probability of being above the true search objective, a missing person for WiSAR, given by

$$p(\mathbf{x}) = \frac{1}{N_{\text{gaussian}}} \sum_{i=0}^{N_{\text{gaussian}}} \frac{\exp\left\{\left[-\frac{1}{2}(\mathbf{x} - \mu_i)^T \sigma_i^{-1}(\mathbf{x} - \mu_i)\right]\right\}}{\sqrt{4\pi^2 \det\{\sigma_i\}}}, \quad (7.3)$$

where $\mu_i \in \mathbb{R}^2$ and $\sigma_i \in \mathbb{R}^{2 \times 2}$ are the mean location and covariance matrix of the i th bivariate Gaussian respectively. The mean is reset at the start of every episode with

$$\mu_i \sim \mathcal{U}([x_{\min}, x_{\max}], [y_{\min}, y_{\max}]), \forall i \in [0, N_{\text{gaussian}}]. \quad (7.4)$$

The covariance is left unchanged to avoid a further tunable simulation parameter. Fig. 7.1 shows how the various PDMs are still highly irregular even with constant covariance due to the randomness introduced by Eqn. 7.4.



(a) The three modes are far apart, with most of the saddle being close to 0. (b) The saddle is aligned with the maxima of the modes. (c) All three maxima have merged into a pill-shaped area of high value. (d) All maxima are aligned leading to a single hotspot.

Figure 7.1: Example PDM $p(\mathbf{x})$ with $N_{\text{gaussian}} = 3$ and constant covariance. Lighter areas have a higher probability, whilst darker ones have a lower probability. During search planning, the agent would avoid lower probability regions whilst targeting high probability ones to maximise total seen probability.

7.4.3 Rewards

The rewards scheme from Sect. 6.4.1.3 is employed in this section. In this formulation, the search efficiency is the objective being maximised by the agent. This value is based on the integration of the PDM, defined as a MMBG (see Eqn. 5.5), over the search area, the accumulated probability at time t , and is given by

$$p_t = \oint_{H_t} p(\mathbf{x}) dH_t. \quad (7.5)$$

The difference of this value, normalised by the maximum probability step, is then defined as

$$r = \frac{p_A A_{\text{area}} (p_t - p_{t-1})}{R_{\text{buffer}} (\pi R_{\text{buffer}} + 2\lambda)}, \quad (7.6)$$

and labelled the scaled probability efficiency reward. More information on the derivation and design of this can be found in Sect. 6.4.1.3.

Further reward shaping is applied to prevent out-of-bounds scenarios and to discourage visiting area of low probability. This is then the augmented reward r' which is given by

$$r' = \begin{cases} -w_{oob}, & \mathbf{x}_t \notin [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \\ w_r r, & \Delta p_t > \epsilon \\ -w_0, & \text{else} \end{cases}. \quad (7.7)$$

7.4.4 Observation Processing

The observation vector at time ts is denoted by s_t . To ensure flexibility when designing the architectures, the available sub-states are given in Tbl. 7.2 with architecture-specific observation space definitions given in Sect. 7.5.

Table 7.2: Definition of the five state observations.

Sub-state	Symbol	Definition
Path	s_{path}	$(\mathbf{x} \parallel 0 \parallel 0^{2 \times N_{\text{waypoints}} - t})^T$
PDM	s_{PDM}	$[\mu_0, \sigma_0, \dots, \mu_G, \sigma_G]^T$
Position	s_{pos}	$\mathbf{x}_t$
Out-of-bounds	s_{oob}	$\mathbf{x}_t \in [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$
Number of steps	s_{steps}	t

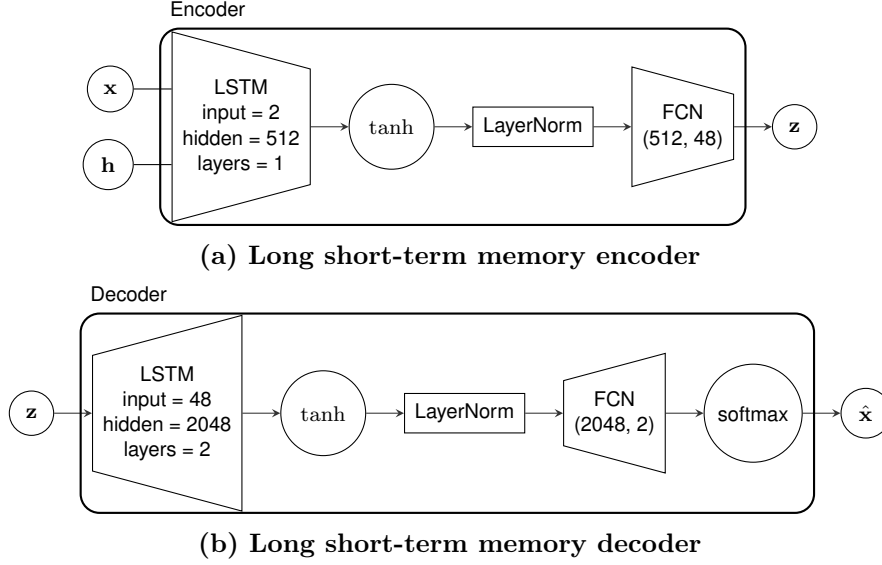


Figure 7.2: RAE architecture using LSTM for encoding and decoding. Using an unbalanced architecture, with the (b) decoder being larger than the (a) encoder, enables higher-quality reconstruction, which results in faster training and better performance. Softsign is applied to the (b) decoder output to ensure that values meet the $s_{\text{path}} \in [-1, 1]$ requirement whilst providing close to linear mappings in this range.

7.5 Model Architectures

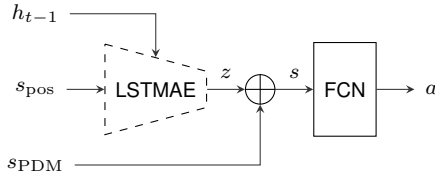
The principal architecture introduced in this chapter is the LSTMAE_SAC. This model leverages a RAE, specifically an LSTM-based AE, as previously defined in Sect. 3.2.1, to handle path history feature extraction. The architecture of the LSTM RAE is given in Fig. 7.2. The long short-term memory auto-encoder (LSTMAE) is coupled with the SAC algorithm, which has demonstrated robust performance in continuous action spaces with large observation spaces. The LSTMAE_SAC architecture is designed to efficiently process temporal information from the agent’s path history, potentially reducing the need for an extremely large network as observed in Ch. 6.

To comprehensively evaluate the efficacy of LSTMAE_SAC, a suite of comparative architectures were developed. These vary in their RL algorithms, path observation augmentation techniques, and inner-model feature extraction methods. The key variants are defined in Tbl. 7.3, and Tbl. 7.3, and were designed to systematically explore the impact of different components, which are the:

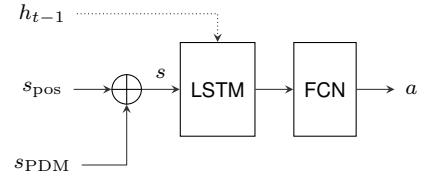
- efficacy of path observation augmentation;
- differences in RL algorithm; and
- impact on inner-model feature extraction in lieu of path observation augmentation.

Table 7.3: Definitions of architectures.

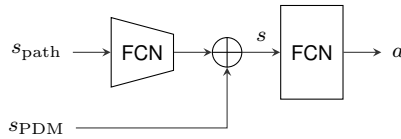
Architecture	Algorithm	Path Observation Augmentation	Inner-model Feature Extraction	Observation Space
LSTMAE_SAC	SAC	LSTMAE	None	$(z_{\text{path}}, s_{\text{PDM}})$
LSTMAE_PPO	PPO	LSTMAE	None	$(z_{\text{path}}, s_{\text{PDM}})$
LSTM_PPO	RPPO	None	None	$(s_{\text{pos}}, s_{\text{PDM}})$
FS_SAC_LSTM	SAC	Frame Stacking	LSTM	$(s_{\text{path}}, s_{\text{PDM}})$
FS_SAC_FCN	SAC	Frame Stacking	FCN	$(s_{\text{path}}, s_{\text{PDM}})$
FS_SAC_CONV2D	SAC	Frame Stacking	CNN	$(s_{\text{path}}, s_{\text{PDM}})$
FS_PPO_LSTM	PPO	Frame Stacking	LSTM	$(s_{\text{path}}, s_{\text{PDM}})$
FS_PPO_FCN	PPO	Frame Stacking	FCN	$(s_{\text{path}}, s_{\text{PDM}})$
FS_PPO_CONV2D	PPO	Frame Stacking	CNN	$(s_{\text{path}}, s_{\text{PDM}})$



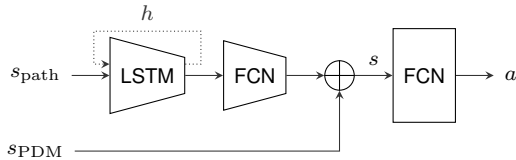
(a) Policy for the LSTMAE variation. Note that the LSTM encoder module is frozen and its parameters are not updated during training.



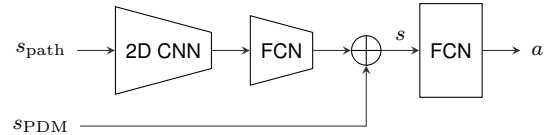
(b) RPPO policy where h_{t-1} is the LSTM hidden state from the previous time-step.



(c) FCN inner-model feature extraction.



(d) LSTM inner-model feature extraction.



(e) 2D CNN inner-model feature extraction.

Figure 7.3: The five proposed policy architectures for use with either PPO, RPPO, or SAC. Only the RPPO uses the architecture from (b).

7.6 Practical Implementation Details

7.6.1 Cubature

The integral is calculated using a cubature integration scheme [266] with constrained Delaunay triangulation [278] to subdivide H_t into triangles for fast computation. The use of pseudo-continuous over discrete integration has been shown to significantly reduce noise in Ch. 5. Whilst noise can be beneficial to promote exploration, this must be controllable and tunable. Reducing noise in the reward function, where cubature is used, is critical to maximising learning efficiency. Else expensive techniques have to be employed [298].

7.6.1.1 Recurrent Encoder

Training During training the test dataset was unbatched and chunked into $N \sim \mathcal{U}(2, k)$ length sections where k is the length of the longest path in the batch. If $N < k$ then the hidden states would be reused for the next section rather than resetting. This significantly improved the speed of convergence. Automatic mixed precision was used to further increase training times. A no-improvement criterion was used where training would terminate if the number of epochs since a loss function decreases breaches a patience threshold.

The hyperparameters used for the RAE are given in Tbl. 7.4 The chosen values represent the configuration that yielded the best trade-off between validation loss stability and computational efficiency during a small hyperparameter sweep. While the Adam optimiser typically uses a learning rate of 0.001, the sweep indicated that a more conservative rate of 0.0001 was necessary for stable training. Furthermore, it was found that gradient clipping and L1 regularisation were crucial in avoiding exploding gradients and in ensuring the latent representation remained interpretable by the decoder.

Table 7.4: RAE parameters used in this chapter.

Parameter	Value
Learning rate	1.000×10^{-4}
Gradient norm clipping	0.500
L1 regularisation coefficient (λ)	1.000×10^{-4}
Patience	40
Batch size	8
# Epochs	5.000×10^3
# Steps	5.000×10^4
Optimiser	Adam

The RAE training dataset of paths was generated using LHC_GW_CONV [155] with the same parameters as this chapter. This resulted in 5×10^4 unique paths for training, which were batch and chunked into $N \sim \mathcal{U}(k_1, \min k_2, k')$ length sections

where $k_1 < k_2$ and $k_1, k_2 \in \mathbb{Z}^+$, and k' is the length of the longest path in the batch. If $N < k'$; then the hidden states would be reused for the next section rather than being reset. This significantly improved the speed of convergence by minimising the backpropagation through time the impact of the vanishing gradients problem [207].

Deployment The recurrent encoder from Sect. 7.5 undergoes separate training from the RL models with its parameters frozen during RL training. This isolation prevents latent space divergence that could destabilise learning during online updates. In our implementation, the encoder resides in the observation preprocessing pipeline rather than the policy network itself. This architectural choice minimises replay buffer memory and compute power requirements during training (critical for SAC’s experience replay mechanism), though deployment permits alternative configurations. The two viable deployment strategies are:

- *hidden state propagation*: stores only the encoder’s hidden states (including cell state), giving a constant runtime performance per step with fixed memory usage (hidden states).
- *full history processing*: maintains complete trajectory histories, with the memory requirements and runtime performance growing linearly with the episode length.

In this work the former approach — hidden state propagation — is used. The full history processing approach would be required for other techniques such as temporal convolution networks [299] or transformers [213].

7.6.2 Further Architecture Details

The RPPO implementation used in this work is from Raffin *et al.* [290] which aligns closely with Pleines *et al.* [295]. The 2D convolution kernel inner-model feature extractor for FS_PPO_CONV2D and FS_SAC_CONV2D is from Mnih *et al.* [230]

7.7 Results

7.7.1 Experimental setup

Each model was trained over 1×10^7 step with 8 workers on a local Ubuntu 22.04 machine with a AMD Ryzen 9 5950X CPU with a NVIDIA RTX A6000 GPU and 64GB of RAM. Runs terminated when invalid training updates were attempted, which only happened during periods of extremely poor performance. At least five runs per architecture were undertaken with randomised starting seeds, which aligns with the best practices outlined in work by Agarwal *et al.* [289] to ensure robust analysis for RL results. The set of run configurations was generated and then randomised during training to avoid any possibility of unforeseen interactions.

The hyperparameters and specific network definitions used for each RL architecture specified in Tbl. 7.5, and were determined via an automated hyperparameter optimisation process. Adam was used for all architectures as it is the de facto standard for RL due to its ability to handle non-stationary objectives and varying gradient scales. A large τ and low learning rate were chosen for the SAC architectures (Tbl. 7.5a) to accelerate the synchronisation of the target networks while maintaining the stability of the parameter updates. This is in contrast with the SAC hyperparameters from Ch. 6 (specifically Tbl. 6.2) which used a very low τ for maximum stability for near-static target networks which indicates that the frame-stacking was causing a high level of observation noise. It was also found that a large batch size with a low learning rate stabilised training substantially. Similarly, in the PPO architectures a narrow clip ranges, and specific GAE λ values were selected to mitigate the risk of catastrophic forgetting and to stabilise the policy gradients.

7.7.2 Evaluation Metrics

This section outlines the metrics used to evaluate the performance of the different architectures. The metrics are:

- **Mean Step Reward:** The average reward received by the agent at each time step during an episode. A higher mean step reward indicates that the agent is making more effective decisions that lead to higher rewards at each step.
- **Mean Rollout Episode Reward:** The average reward accumulated by the agent over a complete episode during the rollout phase. This metric reflects the overall performance of the learned policy in generating high-quality paths.
- **Mean Episode Length:** The average number of time steps taken by the agent before the episode terminates. A longer mean episode length generally indicates that the agent is able to explore the environment more effectively and find longer, more efficient paths.
- **Maximum Probability Efficiency:** The highest achieved probability of reaching the target within a given search area observed during training as defined in Sect. 7.4.3. This metric directly reflects the search efficacy of the architecture, indicating how well it can find the target within the specified search space.
- **Runtime:** The total time or steps taken by the architecture to train. This metric is crucial for practical applications, as it indicates the computational cost of training the architecture.
- **Mean Probability Efficiency:** Similar to the maximum probability efficiency, it is the mean probability efficiency achieved during evaluation. This metric provides a comprehensive assessment of the architecture's search performance across multiple runs.

Table 7.5: Reinforcement learning hyperparameters for the various architectures in this chapter. Default values from [290] v2.1.0 were used if not listed in the table.

(a) Soft actor-critic.			
Hyperparam.	FS_SAC_CONV2D FS_SAC_FCN FS_SAC_LSTM	LSTMAE_SAC	LSTMAE_SAC
Net width	256	256	2048
τ	0.19736	0.19028	0.19028
# envs	8	8	8
Training freq.	1	1	1
Batch size	1024	512	512
Buffer size	5.000×10^6	5.000×10^5	5.000×10^5
Gradient Steps	2	100	100
Learning rate	1.436×10^{-4}	5.917×10^{-6}	5.917×10^{-6}
Learning starts	8092	6547	6547
Target entropy	-1	-1	-1
SDE freq.	5	4	4
Optimiser	Adam	Adam	Adam

(b) Recurrent proximal policy optimisation.

Hyperparam.	LSTM_PPO
# steps	1024
# envs	8
Entropy coef.	2.500×10^{-7}
V_f coef.	2.363×10^{-3}
Batch size	128
Learning rate	8.581×10^{-5}
SDE freq.	2
# epochs	37
Clip range	0.149
GAE λ	0.975
Max. grad.	0.990
Optimiser	Adam

- **Number of Parameters:** The total number of learnable parameters in the neural network architecture. This metric indicates the model’s complexity and computational requirements.

(Continued) Reinforcement learning hyperparameters for the various architectures in this chapter. Default values from [290] v2.1.0 were used if not listed in the table.

(c) Proximal policy optimisation.

Hyperparam.	Architecture		
	FS_PPO_CONV2D FS_PPO_LSTM	FS_PPO_FCN	LSTMAE_PPO
# steps	6126	5126	3104
# envs	8	7	8
Entropy coef.	2.070×10^{-7}	1.070×10^{-7}	0.016
V_f coef.	0.014	3.560×10^{-3}	0.329
Batch size	256	156	64
Learning rate	3.108×10^{-4}	2.108×10^{-4}	4.360×10^{-4}
SDE freq.	3	2	5
# epochs	40	30	58
Clip range	0.148	0.048	0.082
GAE λ	0.901	0.801	0.929
Max. grad.	0.195	0.095	0.614
Optimiser	Adam	Adam	Adam

7.7.3 Architecture

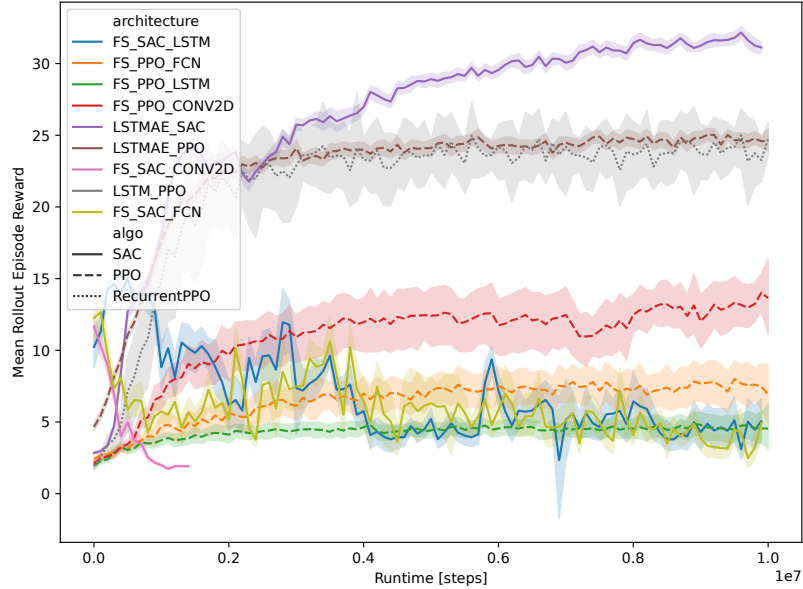
From Fig. 7.4, it is clear that LSTMAE_SAC, LSTMAE_PPO, LSTM_PPO, and FS_PPO_CONV2D had the best rollout reward performance with very stable learning curves. However, the FS_SAC_CONV2D variant was by far the least stable and consistently crashed during training with illegal update steps. This is further corroborated by Tbl. 7.6 with FS_PPO_CONV2D having one of the highest mean runtime steps at 9.82×10^6 and FS_SAC_CONV2D having the lowest at 1.00×10^6 .

With the simulation terminating prematurely if the agent steps out-of-bounds, it is important for the mean episode length to be as close to the maximum simulation length as possible. The worst performance was again FS_SAC_CONV2D with 1.01×10^1 showing a complete lack of generalisation. LSTMAE_SAC had the highest mean episode length at 5.73×10^1 with LSTMAE_PPO and LSTMAE_PPO achieving similar results at 5.11×10^1 and 5.54×10^1 respectively. None of the frame-stacking variants was able to breach the 4.00×10^1 barrier, with FS_SAC_CONV2D coming closest at 3.69×10^1 , which aligns with the performance displayed in Fig. 7.4.

Mean step reward and maximum probability efficiency should strongly correlate from the definition of reward in Eqn. 7.7. However, the latter is the key metric as it directly relates to the search efficacy. Similar to previous metrics, FS_SAC_CONV2D

Table 7.6: Aggregated architecture results over multiple metrics gathered at the end of a training run.

Architecture	Mean Step Reward		Mean Episode Length		Maximum Probability Efficiency		Runtime	
	[$\times 10^{-1}$]		[$\times 10^1$]		[$\times 10^{-1}$]		[s $\times 10^5$]	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
FS_PPO_CONV2D	3.290	0.519	3.690	1.190	1.020	0.188	0.511	0.077
FS_PPO_FCN	2.720	0.937	2.160	1.080	0.659	0.124	0.458	0.154
FS_PPO_LSTM	2.730	1.150	1.640	0.739	0.437	0.089	0.948	0.249
FS_SAC_CONV2D	2.260	1.030	1.010	0.163	0.363	0.295	0.098	0.046
FS_SAC_FCN	2.070	0.977	2.210	1.920	0.790	0.250	0.799	0.527
FS_SAC_LSTM	1.360	1.140	3.200	2.510	0.843	0.167	2.800	1.250
LSTMAE_PPO	5.200	0.610	5.110	0.425	1.760	0.067	1.560	0.499
LSTMAE_SAC	5.370	0.431	5.730	0.609	1.920	0.116	2.550	0.551
LSTM_PPO	4.130	0.512	5.540	0.740	1.640	0.195	3.550	1.280

**Figure 7.4: Mean rollout episode reward over global step for all architectures highlighting the training stability provided by the LSTMAE (LSTMAE_SAC in purple, LSTMAE_PPO in brown). Likewise, LSTM_PPO (grey) also benefits from the recurrent architecture. The PPO-based algorithms (solid line) tend to perform better on the bottom end of the performance scale. However, SAC (solid line) performs best overall in LSTMAE_SAC.**

displayed the poorest performance with a score of 3.63×10^{-2} . FS_PPO_CONV2D, again, was the best of the frame-stacking variants with a score of 1.02×10^{-1} . LSTMAE_PPO and LSTM_PPO also performed inline with previous results with 1.76×10^{-1} and 1.64×10^{-1} respectively. Whilst LSTMAE_PPO has a higher mean maximum probability efficiency, it also has a lower standard deviation of 6.71×10^{-3} compared to 1.95×10^{-2} across more runs. This could indicate the additional stability offered by having a static LSTMAE compared to the internalised LSTM module of LSTM_PPO

from Fig. 7.3b. Furthermore, this could hint at the vanishing gradient problem that recurrent networks suffer from.

Ultimately, LSTMAE_SAC showed the best performance across mean step reward (1.70×10^{-2} higher than LSTMAE_PPO), mean episode length (1.90×10^{-2} higher than LSTM_PPO), and maximum probability efficiency with a value of 1.92×10^{-1} (1.60×10^{-2} higher than LSTMAE_PPO). Contrary to the runtime results from Tbl. 7.6, Fig. 7.4 shows a training curve that had not reached its maxima whilst starkly outperforming the competing variants.

7.7.4 Ablation Study: Reinforcement Algorithm

SAC and PPO have been extensively compared in the literature, however, it is typically only the environment that is being changed. In this comparison the result from the various architectures are grouped by RL algorithm with a consistent environment and hyperparameters. This will give an indication of the robustness and sensitivity to network changes.

Fig. 7.5 displays the inter-quartile ranges of the maximum probability efficiency data presented in Tbl. 7.7. This shows that neither PPO nor SAC have a clear advantage for the frame-stacking variant. However, the LSTMAE results clearly show that SAC outperforms PPO here. The aggregated mean for PPO is 1.43×10^{-1} compared to 1.37×10^{-1} for SAC, showing that PPO perhaps has a slight edge. However, applying a p-test gives a p-value of 0.61, implying that the results are inconclusive.

Isolating the top frame-stacking (FS_PPO_CONV2D and FS_SAC_LSTM) architectures results in a different outcome. SAC has a higher aggregated mean maximum probability efficiency of 1.54×10^{-1} whilst PPO only achieves 1.43×10^{-1} . A p-test, however, also reveals that the results are inconclusive, with a p-value of 0.48.

Table 7.7: Aggregated algorithm (PPO, RPPO, SAC) results over multiple metrics gathered at the end of a training run.

Algorithm	Mean Step Reward		Mean Episode Length		Maximum Probability Efficiency		Runtime	
	[$\times 10^{-1}$]		[$\times 10^1$]		[$\times 10^{-1}$]		[s $\times 10^5$]	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
PPO	3.530	1.340	3.170	1.650	0.982	0.540	0.908	0.538
SAC	3.140	1.900	3.420	2.360	1.120	0.657	1.670	1.290
RPPO	4.130	0.512	5.540	0.740	1.640	0.195	3.550	1.280

7.7.5 Ablation Study: Path Feature Extraction

The main contribution of this chapter is the use of the LSTMAE to handle the feature extraction. Therefore, evaluating its efficacy is imperative compared to the much

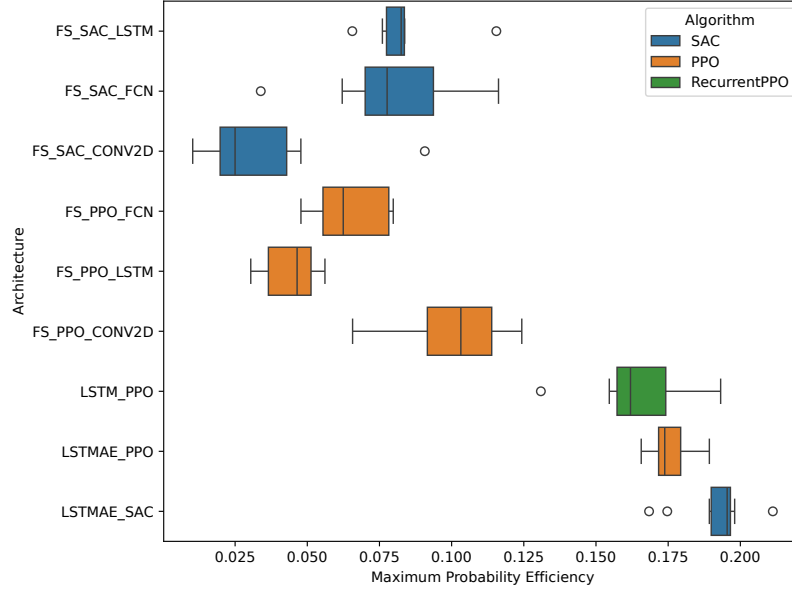


Figure 7.5: Maximum probability efficiency during training for all architecture variants with colour coded by RL algorithm.

simpler frame-stacking. Tbl. 7.8 shows the aggregated performance of the various feature extractors, which closely align to those from Sect. 7.7.3. LSTM_{AE} _{*} is the most performant path feature extraction method across all key performance metrics. All frame-stacking variants have low mean episode lengths and low mean step reward. Most importantly, the mean maximum probability efficiency for the best frame-stacking variant, FS_{*}_CONV2D, is less than half as performant as LSTM_{AE} _{*}.

As expected from Sect. 7.7.4, LSTM_{AE} _{*} also outperforms LSTM _{*} in all metrics from Tbl. 7.8. This further highlights the efficiency of the RAE to capture the relevant information before RL training to allow the RL policy to focus on control.

Table 7.8: Aggregated path feature extraction results over multiple metrics gathered at the end of a training run.

Path Feature Extraction	Mean Step Reward		Mean Episode Length		Maximum Probability Efficiency	
	$[\times 10^{-1}]$		$[\times 10^1]$		$[\times 10^{-1}]$	
	Mean	Std.	Mean	Std.	Mean	Std.
FS _* _CONV2D	2.880	0.896	2.620	1.630	0.760	0.405
FS _* _FCN	2.420	0.984	2.180	1.480	0.720	0.199
FS _* _LSTM	2.250	1.300	2.190	1.710	0.580	0.232
LSTM _{AE} _*	5.540	0.652	5.620	0.650	2.010	0.304
LSTM _*	4.130	0.512	5.540	0.740	1.640	0.195

7.7.6 Benchmark

Ch. 6 has shown that FS_SAC_FCN with a large enough network can outperform optimisation-based algorithms from the literature in. A key limitation of this approach was the large number of parameters and long training times required to achieve these results. The results from Sect. 7.7.3 were achieved using a core policy of dimensions 2×256 whereas Ch. 6 used 8×2000 . A 2×2000 version of LSTMAE_SAC was also trained for a fairer comparison.

Tbl. 7.9 shows the mean probability efficiency from 2000 generated paths. It is evident that the smaller variants of both FS_SAC_FCN and LSTMAE_SAC performed poorer than their larger counterparts. However, the smaller LSTMAE_SAC is only worse than the large FS_SAC_FCN by a margin of 2.40×10^{-2} whilst having 0.27% the number of learnable parameters. Large LSTMAE_SAC, which has 14.0% of the amount of learnable parameters of large FS_SAC_FCN, clearly outperforms all others variants. A performance difference of 4.00×10^2 might hint at an insignificant result. However, the sample sizes were above 2000, resulting in a p-value of 0.0151 above the threshold of 0.05, showing a meaningful difference in distributions. Large LSTMAE_SAC therefore outperforms large FS_SAC_FCN whilst being only 14% of the size and only needing to be trained for 23% of the time (90 days for Ch. 6) and 21 days for large LSTMAE_SAC before it reached the no-improvement termination criterion.

Table 7.9: Mean probability efficiency from over 2000 generated paths by the respective architectures.

Architecture	Probability Efficiency [$\times 10^{-1}$]		Core Policy Shape	# Parameters
	Mean	Std.		
LSTMAE_SAC	2.000	0.570	2×2000	2.170×10^7
LSTMAE_SAC	1.720	0.617	2×256	4.250×10^5
FS_SAC_FCN from Ch. 6	1.960	0.471	8×2000	1.550×10^8
FS_SAC_FCN	0.090	0.191	2×256	4.360×10^5
LHC_GW_CONV [155]	1.210	0.802	n/a	n/a

7.8 Discussion

These results demonstrate the effectiveness of the proposed LSTMAE_SAC architecture for search planning. The LSTMAE effectively captures temporal dependencies within the path, improving performance and stability compared to simpler frame-stacking methods. The RAE training harness maximises the information throughput in the latent representation, which is akin to lossy compression. This prevents the

onus from being on the RL algorithm to perform the same function. Furthermore, the recurrent network allows maximum throughput regardless of path length — from 2 steps to N — since every neuron within the policy is trainable regardless of state. The training stability seen in Fig. 7.4 underlines this with both LSTMAE_PPO and LSTMAE_SAC having stable learning curves.

While further investigation is needed to determine the optimal RL algorithm overall architectures of the algorithm, the results show that SAC has the edge over PPO when used with LSTM AE. This aligns with results from Mock *et al.* [282], which suggested that SAC is better for larger dimensional observation spaces.

Isolating the path feature extraction from the RL algorithm in Sect. 7.7.4 suggested that the frame-stacking with LSTM was the poorest performer with LSTMAE being the best. This further gives credence to the theory that the RAE is superior to standalone RL. On the other hand, LSTM_PPO is close in performance to LSTMAE_PPO, which suggests that simple frame-stacking is the main issue.

The significant performance improvement of the larger LSTMAE_SAC over the larger FS_SAC_FCN in Sect. 7.7.6 highlights the true efficiency and potential of the proposed approach. Similarly, the minor performance difference between large FS_SAC_FCN and small LSTMAE_SAC shows the power of the highly specialised network architecture proposed in this work. This suggests that the LSTMAE_SAC can generally achieve high performance with fewer parameters and shorter training times, making it a promising approach for real-world applications where computational budgets can be limited.

7.9 Conclusion

A novel approach to enhancing WiSAR missions through the integration of an LSTM-based RAE with RL was presented in this chapter. The challenges associated with traditional search planning methods, particularly in large and complex environments, necessitate solutions to optimise efficiency and effectiveness in locating missing persons.

The proposed framework addresses key limitations of existing methodologies by decoupling feature extraction from the policy training phase, thereby significantly reducing the number of learnable parameters and improving training speeds. By employing a RAE architecture, it has been demonstrated that it is possible to maintain high performance while ensuring the model is lightweight enough for deployment on resource-constrained devices. Furthermore, a reduction in learnable parameters makes this model more accessible as fewer performant computers and graphics cards can be used for training. The empirical results indicate that this approach enhances the probability of detection and thus accelerates the overall search process, which is critical in time-sensitive rescue scenarios.

Furthermore, this chapter contributes to the growing body of literature on RL applications in search planning by providing a comprehensive evaluation of various

architectures, including comparisons between PPO and SAC algorithms in large observation domains. SAC outperformed PPO for the proposed architecture, whereas PPO and SAC were similar for the rest. However, FS_PPO_CONV2D was the best non-recurrent architecture whilst FS_SAC_CONV2D was the worst, showing that this result is application specific. These findings underscore that RL is not a golden bullet and the importance of careful model engineering when tackling difficult problems.

Chapter 8

Dynamically Aware Reinforcement Learning Search Planning

8.1 Section Overview

This chapter increases the level of fidelity in the RL search planning of Ch. 6 and Ch. 7 by introducing a dynamically aware approach, addressing limitations of simplified PDMs and modelling the terrain. Earlier applications of the AE paradigm, specifically using an LSTM-based RAE, improved training time, model size, and search performance; however, these approaches lacked real-world considerations like accurate PDM representation and dynamic constraints.

To overcome these challenges, this chapter presents a comprehensive framework, including a 6-degree-of-freedom quadcopter model for waypoint tracking and a 2D plane-based terrain model for analysing search algorithms. A key contribution is the development of a CNN-based AE for encoding image-based PDM and terrain into a compact latent representation. This enables a dynamically and terrain-aware, continuous action-space RL search path planner, with two proposed architectures: one extending the successful AE framework and another primarily CNN-based to incorporate system dynamics into the observation space while maintaining policy freedom.

Both models outperformed prior work from Ch. 6 and Ch. 7, as well as three benchmark algorithms. Notably, the non-AE architecture performed the best, contrary to prior results.

In this chapter related search planning algorithms and RL-based path planning methods are discussed in Sect. 8.3. The environmental model is introduced in Sect. 8.4, with Sect. 8.5 exploring the use of the RAE architecture to encode path histories. Sect. 8.6 describes the RL policy architecture including the observation, reward function, and action definitions. Finally, results are presented and discussed in Sect. 8.7, with a conclusion being drawn in Sect. 8.8.

8.2 Introduction

Ch. 7 showed that applying the AE paradigm to RL search planning provides many benefits. The training time, model size, and search performance were all improved in comparison to the frame-stacking approach from Ch. 6 through the use of an LSTM-based RAE. This is likely due to the RAE training harness being able to maximise the information throughput during the latent space projection whilst SAC (the RL algorithm used) could focus on interpreting the observation. These results show that careful model architecture design and engineering are key to maximising performance.

In previous chapters, the core challenge was in maintaining the MDP whilst encoding the path history in a manner efficient for both training and computation. This was achieved for the problem definition, however, in order to undertake this comparison, the PDM observation was simplified to the vector of normalised parameters of the continuous PDM. In order to represent a real PDM tools like the Gaussian mixture model [139] must be used to estimate a numerical representation. This process can be skipped since a real PDM is likely to be in an image format, which CNNs [200], [201] are designed to handle. Work by Mnih *et al.* [230] showed the capabilities of leveraging image-based observation spaces through the CNN architecture in RL. To then build on the prior success of the RAE to encode the path history, a convolution auto-encoder (CAE) should be likewise explored to encode the PDM.

Like the simplified PDM representation, Ch. 6 and Ch. 7 assumed a purely time-limited system with constant speed that is traversing the waypoint path with perfect accuracy. This is not the case in the real world where a UAV is a complex non-linear system that requires many complex subsystems to control [115], [148], [151], [283], [300]–[304] and whose endurance is dependent on more than just time [173], [305]–[307]. Therefore, in order to fully leverage RL, the underlying dynamic system should be part of the training environment, and the system’s performance should directly depend on the flight path.

Similarly, Ch. 6 and Ch. 7 assumed a flat landscape. This is not the case for WiSAR, where the terrain can be undulating and unforgiving. Koester [24] reports the elevation difference from the PLS to the find location for the LP datasets, which highlights the importance of the altitude for search planning. Changes in altitude also impact the endurance performance of a quadcopter due to the increased control action required to ascend and the energy losses associated with normal operations. Therefore, it is essential to model the terrain within the learning and evaluation environment, as search will occur in three-dimensional areas, and endurance is dependent on paths that efficiently traverse the landscape.

The contributions to the literature of this chapter are then the:

- development of a six-degree-of-freedom quadcopter model with a waypoint-based path-tracking subsystem for the analysis and development of search path algorithms;

- implementation of a continuous terrain model to model the challenging search landscape;
- development of a CNN-based AE to facilitate the encoding of an image-based PDM and terrain, in order to produce a lower dimensional latent representation in place of approximation methods; and
- dynamically and terrain aware, continuous action-space RL search path planner with two proposed architectures: AE (RAE and CAE, akin to that of Ch. 7), and CNN.

8.3 Related Work

One of the allures of RL is the ability to learn emergent behaviour based on interactions with a black-box model of the environment [308], [309]. It is therefore important to maximise the degrees of freedom in the environment that the agent has to increase the chances of the RL policy.

Both Manjanna *et al.* [43] and Manjanna *et al.* [197] incorporate the notion of dynamic awareness into the training. This gives the policy a more granular insight into the platform for which it is generating the search path. However, while this work increases the observability of the environment, it restricts the action space through discretisation. This limits RL in discovering novel policies that could improve performance.

Increasing the degrees of freedom can come in the form of replacing the navigation stack with a single RL policy. However, while RL can perform low-level high-frequency control — as shown by Kaufmann *et al.* [171], Degraeve *et al.* [310], and Carlucho *et al.* [311] — Shahna *et al.* [312] showed the power of using RL for the high-level decision making. For this reason, Mehta *et al.* [313] introduces a waypoint-based RL policy for robot arm control that encodes dynamics. Through discretising the problem as a sequence of multi-armed bandits, a continuous action space is maintained whilst enabling RL to make high-level decisions. For search planning specifically, this can take the form of using RL to control a dynamic system through high-level waypoints whilst still learning the underlying platform dynamics.

A key concern when increasing the degrees of freedom for RL is maintaining the MDP without compromising the system architecture. Previous chapters (specifically Ch. 6 and Ch. 7) and Raffin *et al.* [292] have demonstrated the efficacy of decoupling the feature extraction from the learning. However, as the observation and action space grows, the RL policies will become more unstable during training [282]. It is, therefore, imperative to explore advanced networks [314] to facilitate more efficient and stable learning.

The RAE [293] has been successfully applied to encode the path history in Ch. 7 using the LSTM [206] architecture. However, the GRU [208], and RNN [205] have also been successfully applied in various contexts. Likewise, MHSA, introduced by Vaswani

et al. [213], has directly contributed to the rise in the prominence of large language model in everyday life and could prove fruitful in the context of RAEs.

Nonetheless, the allure of increasing the complexity of a successful model should be critically considered. Given the image-like shape of a PDM, it is important to explore a CNN [200], [201] or CSA [217] based approach [230], [315]. Valencia *et al.* [316] successfully applied a purely image-based RL for physical robotic hand manipulation. Similarly, Deng *et al.* [317] implemented a CNN-based RL policy with hierarchical trajectory generation.

This chapter, therefore, explores search planning using a RL policy that is aware of the dynamic properties of the underlying system. Two architectures are developed to solve this problem: one built on the success of previous chapters using the AE architecture, and the other using a primarily CNN-based approach. The observation space will provide insights into the system's dynamics, while the action space will remain continuous, adding additional degrees of freedom to the policy's control space.

8.4 Environment Modelling

This section introduces and develops the dynamic environmental model used to train the RL policy. Sect. 8.4.5 describes how the individual components are combined into a cohesive model.

8.4.1 Six Degree-Of-Freedom UAV Model

The non-linear dynamics of the UAV are defined in the left-handed reference frame of Fig. 8.1. The system is modelled as a point-mass with four forces $F_i \forall i \in \{1, 2, 3, 4\}$ and moments $M_i \forall i \in \{1, 2, 3, 4\}$ acting on it [300]. Drag is modelled using through the square of the linear velocity. This non-linear system of equations is defined as

$$\begin{aligned} m\ddot{\mathbf{x}} &= \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - R_E^B \begin{bmatrix} 0 \\ 0 \\ k_T \sum_{i=1}^4 P_i^2 \end{bmatrix} - R_E^B (\mathbf{C}_d R_B^E \dot{\mathbf{x}})^2, \\ \mathbf{I}\ddot{\boldsymbol{\phi}} &= \begin{bmatrix} lk_T(P_4^2 - P_2^2) \\ lk_T(P_3^2 - P_1^2) \\ k_Q(P_1^2 - P_2^2 + P_3^2 - P_4^2) \end{bmatrix}, \end{aligned} \quad (8.1)$$

where k_T and k_Q are thrust and torque constants respectively [318], and $P_i \forall i \in \{1, 2, 3, 4\}$ are the propeller rotational speeds.

The model is linearised at $T = mg$, with the small angle assumptions ($\sin(x) \approx x$

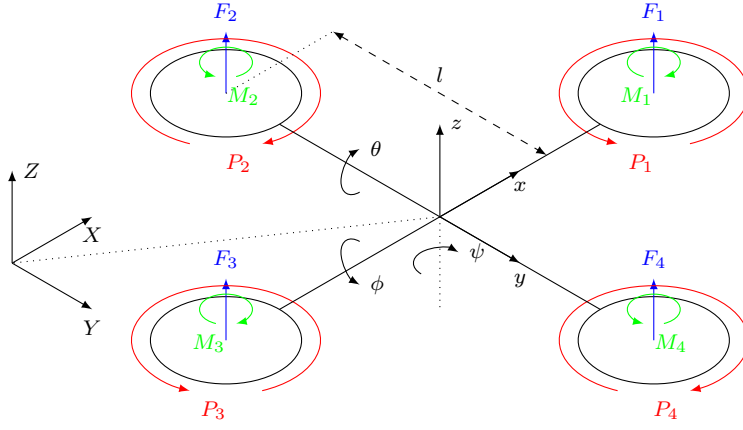


Figure 8.1: Configuration of the UAV.

and $\cos(x) \approx 1$), and an assumption that drag is negligible, to give

$$\begin{aligned}
 \ddot{x} &= -g\theta, \\
 \ddot{y} &= g\phi, \\
 \ddot{z} &= g - T/m, \\
 \dot{p} &= \tau_\phi/I_z, \\
 \dot{q} &= \tau_\theta/I_y, \\
 \dot{r} &= \tau_\psi/I_z.
 \end{aligned} \tag{8.2}$$

The torques and thrust are mapped to the propeller control commands using the linear transformation

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \\ T \end{bmatrix} = \begin{bmatrix} 0 & -lk_T & 0 & lk_T \\ -lk_T & 0 & lk_T & 0 \\ k_Q & -k_Q & k_Q & -k_Q \\ k_T & k_T & k_T & k_T \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}. \tag{8.3}$$

8.4.2 Control

A tracking linear quadratic regulator (LQR) controller [319]–[322] is applied for the low-level UAV state control. The linearised model from Eqn. 8.2 can be described as an infinite-horizon linear time invariant system

$$\begin{aligned}
 \dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u}, \\
 \mathbf{x} &= [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r] \in \mathbb{R}^{12}, \\
 \mathbf{u} &= [\tau_\phi, \tau_\theta, \tau_\psi, T] \in \mathbb{R}^4.
 \end{aligned} \tag{8.4}$$

The optimal feedback control law is then found by minimising the cost function

$$J = \int_0^\infty (x^T Q x + u^T R u) dt. \tag{8.5}$$

The feedback control law that minimises this value is

$$u = -Kx, \quad (8.6)$$

where K is given by

$$K = R^{-1}(B^T P), \quad (8.7)$$

and P is the solution to the continuous-time algebraic Riccati equation

$$A^T P + PA - PBR^{-1}(B^T P) + Q = 0. \quad (8.8)$$

The control Q , and error R matrices are tuned using the GA (values, and further details are provided in Appendix C) to minimise the weighted error of an altitude, horizontal velocities, and yaw step-input. This reflects the setpoints generated by the trajectory planning procedure outlined in Sect. 8.4.3.

8.4.3 Trajectory Planning

A quintic polynomial scheme is used to handle the trajectory planning in the form

$$x(t) = c_5 t^5 + c_4 t^4 + c_3 t^3 + c_2 t^2 + c_1 t + c_0. \quad (8.9)$$

Given the boundary conditions $x(0)$, $x(T)$, $\dot{x}(0)$, $\dot{x}(T)$, $\ddot{x}(0)$, and $\ddot{x}(T)$, Eqn. 8.9 can be put into matrix form

$$\underbrace{\begin{bmatrix} x(0) \\ x(T) \\ \dot{x}(0) \\ \dot{x}(T) \\ \ddot{x}(0) \\ \ddot{x}(T) \end{bmatrix}}_x = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{bmatrix}}_M \underbrace{\begin{bmatrix} c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix}}_c, \quad (8.10)$$

which is a system of linear equations. The coefficient matrix c can be solved for using

$$\mathbf{c} = M^{-1}\mathbf{x}. \quad (8.11)$$

A *look-ahead* scheme is used to define the target velocity at $t = T$ with both $\ddot{x}(0) = \ddot{x}(T) = 0$, and the current position and velocity are used for $x(0)$ and $\dot{x}(0)$ respectively. Look-ahead is defined using the current position A , target position $x(T)$ (denoted as B) and the next target waypoint C , as seen in Fig. 8.2. The target velocity

is calculated using

$$\cos(\theta) = \frac{(-\mathbf{r}_{AB}) \cdot \mathbf{r}_{BC}}{|\mathbf{r}_{AB}| |\mathbf{r}_{BC}|} \quad (8.12)$$

$$\mathbf{v}_B^- = \mathbf{v}_B^+ = V \cdot \frac{\mathbf{r}_{AC}}{|\mathbf{r}_{AC}|} \cdot \left(\frac{1 - \cos(\theta)}{2} \right)^n \quad (8.13)$$

where n is a tunable parameter, V is the base velocity magnitude, and $\mathbf{r}_{xy}$ is the vector from a waypoint x to another y (defined in Fig. 8.2). n defines how smoothly the waypoint path is traversed with the impact of varying n being seen in Fig. 8.3.

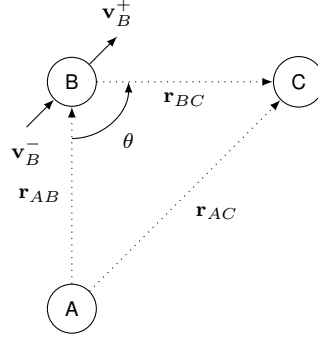


Figure 8.2: Look-ahead velocity relative definitions.

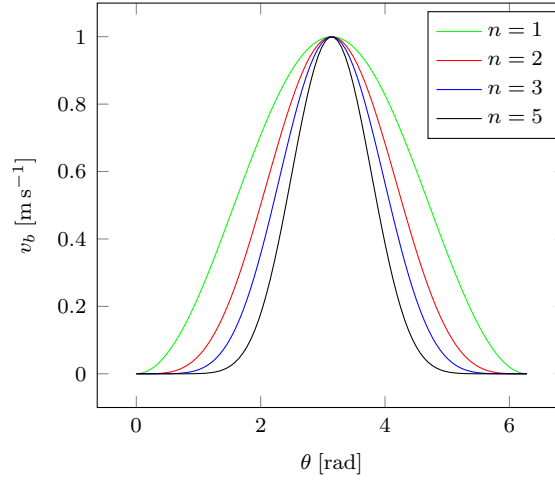


Figure 8.3: v_b over θ with different values for n .

The look-ahead transforms the typical velocity degree-of-freedom to a positional one. The waypoint controller outputs the next target waypoint C and not the current target B , which removes the need for superfluous target velocity control points whilst smoothing the trajectory. In order to implement this, the first step considers $C = B$ before switching to the aforementioned scheme. This approach of using a quintic polynomial scheme with look-ahead guarantees continuity for position, velocity, and acceleration but does not respect dynamic constraints nor does it adapt to distances.

A constant T in Eqn. 8.10 would lead to slow traversal of the path over short distances and physically impossible accelerations for large steps. The former is only a problem from an efficiency point-of-view, however the latter could result the system

to become unstable. An optimal formulation is therefore applied to meet a maximum acceleration constraint whilst minimising time. This is defined as

$$T^* = \min_{T>0} T \text{ s.t. } \max_{q \in \{x,y,z\}} \max_{t \in [0,T]} |\ddot{q}(t)| \leq a_{\max}, \quad (8.14)$$

where $\max_{t \in [0,T]} |\ddot{q}(t)|$ can be found by finding the stationary points of the third derivative through

$$\frac{d^3 x(t)}{dt^3} = 60c_5 t^2 + 24c_4 t + 6c_3 = 0. \quad (8.15)$$

Eqn. 8.15 can be solved for t using the quadratic formula such that

$$t^* = \frac{-24c_4 \pm \sqrt{576c_4^2 - 1440c_5c_3}}{120c_5}, \quad (8.16)$$

$$\max_{t \in [0,T]} |\ddot{q}(t)| = |\max(\{\ddot{q}(\text{clip}(t_i, 0, T)) | t_i \in \{0, t_0^*, t_1^*, T\}\})|.$$

A couple of considerations must be made to implement this efficiently.

Lemma 8.1. *Eqn. 8.11 is not solvable at $T = 0$.*

Proof. Assume, for the sake of contradiction, that Eqn. 8.11 is solvable for $T = 0$. Since matrix M is square it can be inverted by

$$M^{-1} = \frac{1}{\det(M)} \text{adj}(M). \quad (8.17)$$

However, when $T = 0$ the determinant of M is 0. Our initial assumption that there exists as solution to Eqn. 8.11 when $T = 0$ is therefore incorrect. This means that M is not invertible and there exists no solution for $T = 0$. $\square$

Lemma 8.2. *Eqn. 8.14 is solvable with $T^* \in (0, \infty)$ and $a_{\max} \in (0, \infty)$.*

Proof. Consider the two extreme cases:

- for sufficiently small T , $\ddot{q}(T) \gg a_{\max}$; and
- for sufficiently large T , $\ddot{q}(T) \ll a_{\max}$.

Then, by the Intermediate Value Theorem, there exists a minimal $T^* > 0$ such that $\ddot{q}(T^*) = a_{\max}$. $\square$

Thus, from Lem. 8.1, Eqn. 8.14 and Eqn. 8.16 can be modified to be in the form

$$f(T)|_q = \begin{cases} \max_{q \in \{x,y,z\}} \max_{t \in [0,T]} |\ddot{q}(t)| - a_{\max}, & t > 0 \\ \infty, & \text{else} \end{cases}, \quad (8.18)$$

which is suitable for solving for T^* via the bisection method.

The bisection method [323] is a root-finding technique that can be applied to any continuous function for which the values at the start end of the range have opposite

signs. Whilst it is very simple and robust method, this comes at the cost of being slow due to the bisection approach. The procedure of the bisection method is given in Alg. 8.1.

Algorithm 8.1: Bisection method.

Data: Function $f(x)$, left value a , right value b , tolerance ϵ , number of iterations N . The condition $f(a)f(b) < 0$ states that the sign of $f(a)$ and $f(b)$ cannot be the same.

Result: c such that $f(x = c) \leq |0 + \epsilon|$

```

1  $i \leftarrow 0$ ;
2 while  $i < N$  do
3    $c \leftarrow (a + b)/2$ ;
4   if  $(b - a)/2 \leq \epsilon$  or  $f(c) = 0$  then
5     stop;
6   end
7    $i \leftarrow i + 1$ ;
8   if  $f(c)f(a) < 0$  then
9      $b \leftarrow c$ ;
10  else
11     $a \leftarrow c$ ;
12  end
13 end

```

However, the vanilla bisection method from Alg. 8.1 requires the solution to be in a predefined range resulting in wasted computation if the range is too large or invalid results if the solution is outwith the specified range. The right expansion modification from Alg. 8.2 is therefore applied with an initial best guess range of $a = 0$ and $b > a$ which guarantees a solution in combination with Lem. 8.2.

Algorithm 8.2: Bisection method with right expansion. This can be adapted to left expansion by decreasing a instead of increasing b on line 4.

Data: Function $f(x)$, left value a , right value b , tolerance ϵ , number of iterations N .

Result: c such that $f(x = c) \leq |0 + \epsilon|$

```

1  $i \leftarrow 0$ ;
2  $k \leftarrow b$ ;
3 while  $f(a)f(b) < 0$  and  $i < N$  do
4    $b \leftarrow b + ik$ ;
5    $i \leftarrow i + 1$ ;
6 end
7  $c \leftarrow \text{bisection}(f, a, b, \epsilon, N - i)$ ;

```

Eqn. 8.18 is therefore solvable for T^* using the bisection method with right expansion. This gives a search space of $T^* \in [0, \infty)$ meaning that the maximum acceleration constraints are always met whilst minimising time.

8.4.4 Terrain Modelling

The terrain is modelled as a tilted plane. The altitude, $A \in \mathbb{R}$, at a specified position, $\mathbf{p} = [p_x, p_y, p_z]^T$, is given by

$$A(\mathbf{p}, m, \psi, c) = m \begin{bmatrix} \cos(\psi) & \sin(\psi) \end{bmatrix} \cdot \begin{bmatrix} p_x \\ p_y \end{bmatrix} + c, \quad (8.19)$$

where $m \in \mathbb{R}$ is the slope gradient, $\psi \in \mathbb{R}$ is the slope direction, and $c \in \mathbb{R}$. At the start of a simulation, the terrain parameters of Eqn. 8.19 are randomly sampled:

- slope gradient $m \sim \mathcal{U}(0, m_{\max})$;
- slope direction $\psi \sim \mathcal{U}(0, 2\pi)$; and
- elevation offset constant, $c \sim \mathcal{U}(c_{\min}, c_{\max})$.

The range of $A_{\min}$ and $A_{\max}$ will vary between simulations, due to the random sampling of the parameters. It is therefore imperative to normalise these values to be useful for ML applications. The normalised terrain, $\tilde{A}(\mathbf{p})$, is thus given by the optimisation

$$\begin{aligned} \tilde{A}(\mathbf{p}) &= \frac{2A(\mathbf{p}, m, \psi, c) - A_{\min} - A_{\max}}{A_{\max} - A_{\min}}, \\ A_{\min/\max} &= \min/\max_{\mathbf{p} \in \mathbb{R}^{3 \times 1}, m, c, \psi \in \mathbb{R}} A(\mathbf{p}, m, \psi, c), \\ &\text{s.t.} \\ y_{\min} &\leq p_y \leq y_{\max}, \\ x_{\min} &\leq p_x \leq x_{\max}, \\ 0 &\leq m \leq m_{\max}, \\ c_{\min} &\leq c \leq c_{\max}, \\ 0 &\leq \psi \leq 2\pi, \end{aligned} \quad (8.20)$$

such that $\tilde{A}(\mathbf{p}) \in [-1, 1]$ if the parameter ranges for m , c , and ψ , and the bounds for p_x and p_y remain constant. This ensures that the normalisation bounds remain constant with randomised terrain parameters such that the relative magnitudes are observable as seen in Fig. 8.4.

Distance checks are carried out to detect terrain collisions in the form

$$p_z \leq A(\mathbf{p}, m, \psi, c). \quad (8.21)$$

8.4.5 Component Integration

This section introduced four components that, when combined, describe a model of the environment consisting of a single UAV with a waypoint-based hierarchical control architecture defined in Fig. 8.5.

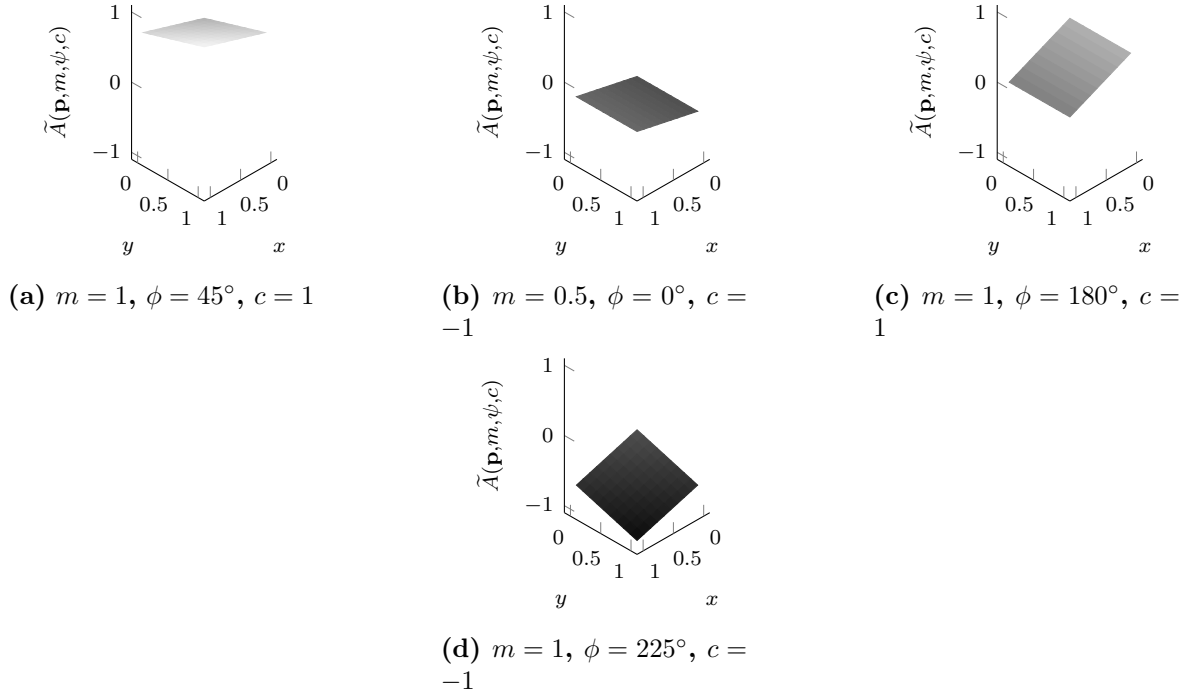


Figure 8.4: Demonstration of the normalised terrain $\tilde{A}(\mathbf{p})$ randomisation over a $x - y$ unit square.

The environment takes a policy action, a as an input, which is denormalised and translated into a waypoint command based on the policy action definition introduced in Sect. 8.6. The trajectory planner from Sect. 8.4.3 then converts this into a state setpoint for every time-step t . This state setpoint is then converted into a torque and thrust command by the controller described in Sect. 8.4.2, where the altitude setpoint is given by the terrain model from Sect. 8.4.4. Finally, the torque and thrust commands are provided to the six degree-of-freedom UAV model from Sect. 8.4.1.

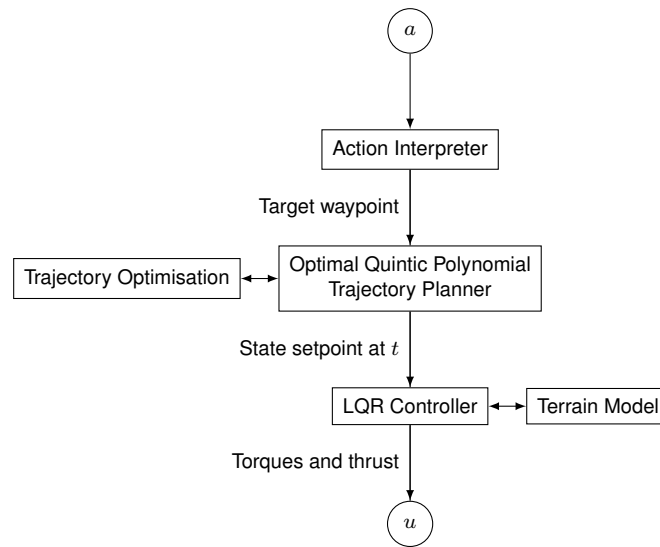


Figure 8.5: Hierarchical control architecture flowchart with a and u being the policy action from the RL policy, and dynamic command respectively. The state feedback and time dependencies are not shown in this diagram.

8.5 Auto-Encoders

Ch. 7 explored the use of the RAE, introduced in Sect. 3.2, using the LSTM architecture to perform feature extraction for the RL policy. This poses the question of whether alternative recurrent architectures could be more suited for this problem and if the AE formulation could be used for the remaining observed states. Sect. 8.5.1 will explore the former and Sect. 8.5.2 will develop a CNN-based AE for the PDM and terrain feature extraction.

8.5.1 Path History Recurrent Auto-Encoder

Six RAE architectures are implemented to evaluate the efficacy of the underlying base architecture (LSTM, GRU, or RNN) as well as the addition of MHSA (see Sect. 3.1). The architectures using MHSA have the suffix *-Attn* appended due to the *attention* capabilities that MHSA provides. Tbl. 8.1 introduces the six permutations evaluated, and Fig. 8.6 outlines the generalised architecture followed. Whilst each architecture is different internally, the training harness follows the RAE methodology outlined in Sect. 3.2.1 (which was successfully applied in Ch. 7)

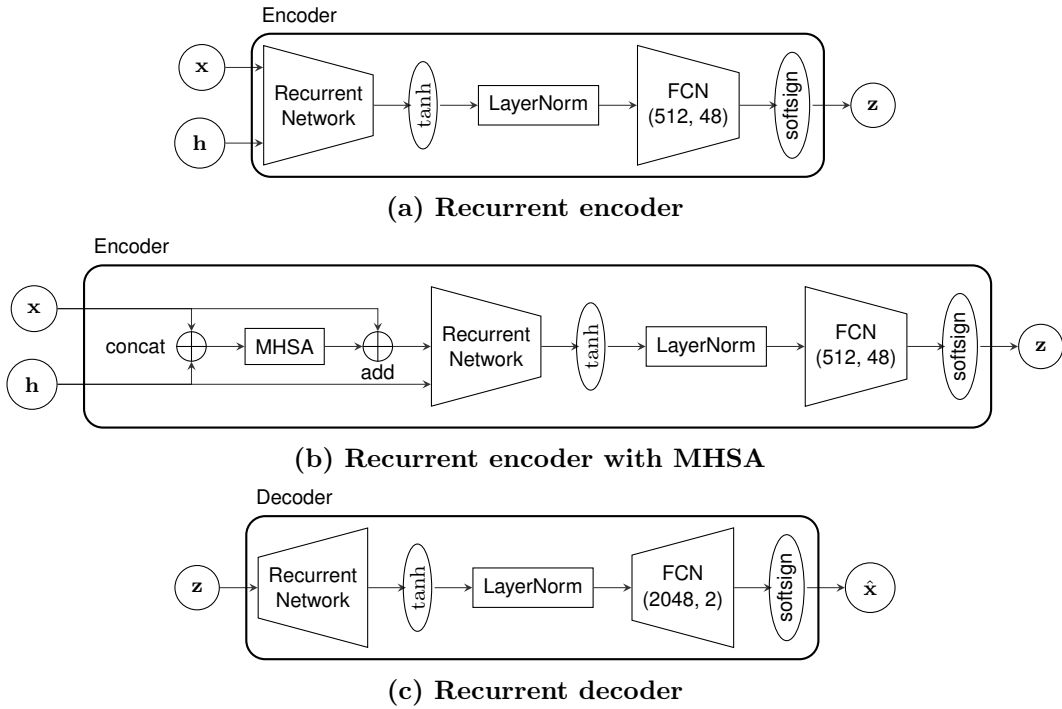


Figure 8.6: Generic RAE architectures for (a)(b) encoding and (c) decoding either MHSA with a recurrent network or a standalone recurrent network for encoding and recurrent network for decoding. Like Ch. 8, the (c) decoder is larger than the (a)(b) encoder, which results in faster training and better performance by increasing the reconstruction potential. Softsign is applied to the latent space encoding and reconstruction to ensure that values meet the $s_{\text{path}} \in [-1, 1]$ requirement whilst providing close to linear mappings in this range.

The training dataset of paths was generated using LHC_GW_CONV [155] with the same parameters as this chapter. This resulted in 5×10^4 unique paths for training,

which were batch and chunked into $N \sim \mathcal{U}(k_1, \text{mink}_2, k')$ length sections where $k_1 < k_2$ and $k_1, k_2 \in \mathbb{Z}^+$, and k' is the length of the longest path in the batch. If $N < k'$; then the hidden states would be reused for the next section rather than being reset. This significantly improved the speed of convergence by minimising the backpropagation through time and thus reducing the impact of the vanishing gradients problem [207]. Automatic mixed precision was employed to reduce training times further. A no-improvement criterion was used where training would terminate if the number of epochs since a loss function decreases breaches a patience threshold.

Four synthetic benchmarks, defined in Tbl. 8.4, were used for the evaluation of the six different architectures. Six models were trained for each architecture, resulting in a total of 64 models. Each training run was initialised with a randomly generated seed to ensure unique training behaviour between otherwise identical setups [324]. The number of parameters per model is given in Tbl. 8.3, which highlights that LSTM is the most complex, whilst RNN is the simplest as in Ch. 7, an unbalanced architecture was used to maximise the learning potential in the encoder whilst minimising the model size and inference times during deployment. The hyperparameters of the RAE are presented in Tbl. 8.2.

Table 8.1: Definition of the six RAE architectures used within this chapter. The suffix *-Attn* denotes architectures using MHSA due to the *attention* capabilities.

Architecture	Base	Addition
LSTM	LSTM	
LSTM-Attn	LSTM	MHSA
GRU	GRU	
GRU-Attn	GRU	MHSA
RNN	RNN	
RNN-Attn	RNN	MHSA

8.5.1.1 Results

Tbl. 8.5 shows the results obtained from models trained using Adam. GRU-Attn performs the best in terms of the mean score in the line and random benchmark, whilst LSTM and LSTM-Attn perform best in the circle and periodic benchmarks respectively. However, GRU-Attn has a lower mean score than LSTM-Attn, for the line and random benchmarks, by only 9.30 % and 18.83 % respectively. GRU-Attn, whilst having low mean scores, has the highest standard deviation across all the four benchmarks which implies an unstable model. LSTM-Attn, on the other hand, has the lowest standard deviation across the benchmarks. This shows that, whilst not dominant, the LSTM-Attn architecture, trained with Adam, performs better than results might initially suggest.

This is not the case when trained with stochastic gradient descent (SGD), as seen in Tbl. 8.6. LSTM-Attn performs best in all benchmarks, except line, where LSTM is the best by a thin margin of 8.29 %. The SGD trained GRU-Attn performs particularly

Table 8.2: RAE hyperparameters for the path history encoding in this chapter.

Hyperparam.	RNN, GRU,LSTM	RNN-Attn, GRU-Attn,LSTM-Attn
Learning Rate	0.0001	0.0001
Momentum	0.9	0.9
k	128	128
Batch size	1	1
Encoder width	512	256
Encoder layers	2	2
Decoder width	2048	2048
Decoder layers	2	2
N_{steps}	10000	10000
N_{epoch}	250	250
Latent dimension	64	64
Clip grad. norm.	0.5	0.5
Patience	25	25
MHSA Heads	n/a	8
MHSA Input Dim.	n/a	256
MHSA Encoding Dim.	n/a	246

Table 8.3: Number of parameters in the encoder and decoder.

Architecture	# Decoder Parameters	# Encoder Parameters
GRU	38 200 000	2 400 000
GRU-Attn	38 200 000	1 200 000
LSTM	50 900 000	3 190 000
LSTM-Attn	50 900 000	1 470 000
RNN	12 700 000	823 000
RNN-Attn	12 700 000	676 000

Table 8.4: Evaluation path generation method definitions. Each path is defined as $\mathbf{P} = [\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_N]$, and is normalised such that $\mathbf{P} = \tilde{\mathbf{P}} / \max(\|\tilde{\mathbf{P}}\|)$, where $\tilde{\mathbf{P}}$ is the unnormalised path, and each element of the path $p_i \in [-1, 1]$. Additionally, $\epsilon_{i,j} \sim \mathcal{U}(0, 1)$, $i \in [0, 1, N - 1]$, $x_i = \frac{i}{N-1}$, $N = 256$ and $\eta = 0.1$.

Name	$[p_x, p_y]^T$
Circle	$\begin{bmatrix} \cos(2\pi x_i) \\ \sin(2\pi x_i) \end{bmatrix} + \eta \begin{bmatrix} \epsilon_{i,0} \\ \epsilon_{i,1} \end{bmatrix}$
Line	$\begin{bmatrix} x_i \\ x_i \end{bmatrix} + \eta \begin{bmatrix} \epsilon_{i,0} \\ \epsilon_{i,1} \end{bmatrix}$
Periodic	$\begin{bmatrix} x_i \\ \sin(2\pi x_i) \end{bmatrix} + \eta \begin{bmatrix} \epsilon_{i,0} \\ \epsilon_{i,1} \end{bmatrix}$
Random	$\begin{bmatrix} \epsilon_{i,0} \\ \epsilon_{i,1} \end{bmatrix}$

poorly, and when compared to RNN with equal or worse loss scores but over twice the amount of parameters.

Table 8.5: Aggregated L2 norm reconstruction loss for the models trained using the Adam optimiser. 64 samples are used per analysis and architecture.

Architecture	Circle		Line		Periodic		Random	
	$[\times 10^{-1}]$		$[\times 10^{-1}]$		$[\times 10^{-1}]$		$[\times 10^{-1}]$	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
GRU	4.239	0.260	3.219	0.297	3.443	0.343	3.291	0.551
GRU-Attn	4.641	0.422	1.376	0.469	2.817	0.251	1.407	0.437
LSTM	4.187	0.038	2.849	0.044	3.489	0.041	2.889	0.130
LSTM-Attn	4.266	0.035	1.504	0.036	2.801	0.029	1.672	0.089
RNN	4.207	0.036	3.163	0.055	3.862	0.049	3.186	0.126
RNN-Attn	4.267	0.094	3.370	0.397	3.573	0.186	3.454	0.389

The LSTM-Attn architecture has the best performance over almost all benchmarks when trained with SGD and is competitive when trained with Adam. The SGD trained LSTM-Attn configuration is therefore used for the remainder of this chapter. However, it is worth noting that this comes at a computational cost, as LSTM-Attn has over 400 % as many parameters as RNN.

Table 8.6: Aggregated L2 norm reconstruction losses for the models trained using the SGD. 64 samples are used per analysis and architecture.

Architecture	Circle		Line		Periodic		Random	
	$[\times 10^{-1}]$		$[\times 10^{-1}]$		$[\times 10^{-1}]$		$[\times 10^{-1}]$	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
GRU	6.498	0.104	1.133	0.093	2.746	0.045	0.903	0.057
GRU-Attn	5.342	0.040	1.326	0.019	2.753	0.029	0.860	0.052
LSTM	5.926	0.213	0.784	0.016	2.566	0.035	0.868	0.048
LSTM-Attn	4.378	0.109	0.849	0.066	2.525	0.048	0.845	0.033
RNN	6.319	0.042	1.083	0.008	2.706	0.035	0.907	0.077
RNN-Attn	6.358	0.371	1.419	0.327	2.871	0.132	0.958	0.097

8.5.2 Probability Distribution Map And Terrain Convolution Neural Network Auto-Encoder

Two CNN-based AE architectures are defined to project the rasterised PDM and terrain to a smaller latent representation. Fig. 8.7a shows the encoder of the vanilla CAE which uses the Gaussian error linear unit (GELU) activation function along with maximum pooling. Fig. 8.7b uses the same architecture but also employs the CSA from Sect. 3.1.4.2. Both architectures use the same CNN-based decoder from Fig. 8.7c.

Training a CAE is simpler than a RAE as this architecture does not consider time series data. The reconstruction of the input $\mathbf{x}$, which is an image, from $\mathbf{z}$ to obtain $\hat{\mathbf{x}}$ is then handled using transpose convolution neural network (TCNN) networks. Similarly to the RAE, the pixel-to-pixel reconstruction loss is calculated using the L2 norm from Eqn. 3.17.

The dataset used for training and evaluation was continuously sampled from the definitions of the PDM and terrain. The two networks per datatype were trained six times, resulting in a total of 24 models. These were analysed on 2×10^4 datapoints per architecture per application (PDM or terrain). SGD was solely used in this analysis following the results in Sect. 8.5.1.

Tbl. 8.8 shows the number of parameters per model. The total number of parameters was kept low due to the definition of the CNN, with the terrain CAE having the smallest latent dimension. This was due to the simple definition of the terrain in Eqn. 8.19 being dependent on only four parameters. The CAE hyperparameters are presented in Tbl. 8.7.

Table 8.7: CAE hyperparameters for the PDM and terrain encoding in this chapter.

Hyperparam.	CNN	CNN-Attn
Learning Rate	0.001	0.001
Momentum	0.9	0.9
Batch size	8	8
N_{steps}	100000	100000
N_{epoch}	5000	5000
Latent dimension (PDM)	64	64
Latent dimension (terrain)	16	64
Patience	50	50

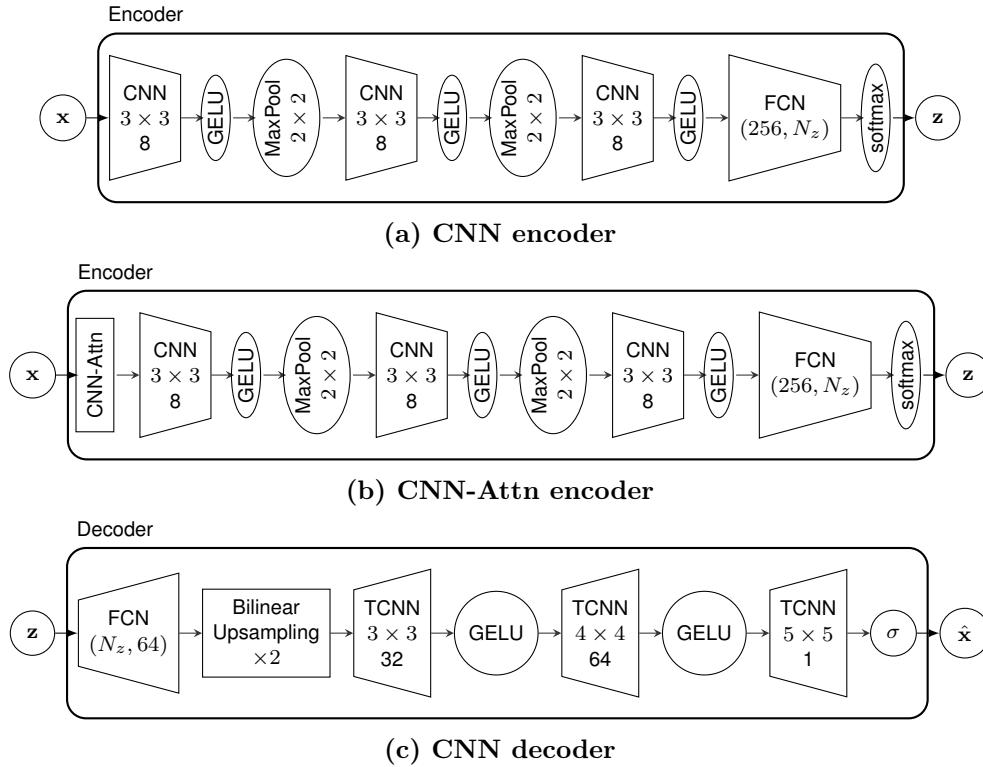


Figure 8.7: The architecture of the (a) CNN encoder, (b) CNN-Attn encoder, and the shared (c) CNN decoder. N_z is the latent space dimension size such that $z \in \mathbb{R}^{N_z}$.

Table 8.8: Number of parameters in the CAE encoder and decoder.

	Architecture	# Encoder Parameters	# Decoder Parameters	Latent Dimension
PDM	CNN	17 185	38 913	64
	CNN-Attn	17 194	38 913	64
Terrain	CNN	4849	35 841	16
	CNN-Attn	4858	35 841	16

8.5.2.1 Results

Table 8.9: Aggregated L2 norm reconstruction losses for the CAE architectures for both applications. $N = 20\,000$ per architecture and application.

Architecture	PDM [$\times 10^{-2}$]		Terrain [$\times 10^{-3}$]	
	Mean	Std.	Mean	Std.
CNN	1.571	0.445	0.245	0.395
CNN-Attn	0.825	0.313	0.210	0.295

Tbl. 8.9 shows the aggregated L2 norm reconstruction losses for the CNN and CNN-Attn based CAE architectures. CNN-Attn has a score 52.5% of that of CNN for the PDM application, which highlights the high impact of the CSA. This result is especially noteworthy as the CNN-Attn architecture only requires 9 more parameters than the CNN architecture for PDMs. The overall result for the terrain application is the same, albeit not as extreme, where CNN-Attn has a score 16.7% lower than CNN with 9 more parameters.

The CNN-Attn architecture is, therefore, the one used for both PDM and terrain encoding in the rest of this chapter.

8.6 Search Planning

The six degree-of-freedom dynamic model from Sect. 8.4 introduces new challenges and opportunities for the application of RL, which have not been encountered in either Ch. 6 nor Ch. 7. By learning the physical properties of the model, the policy can traverse the landscape faster and more efficiently than a naive waypoint planner. This, in turn, could result in better paths leading to better chances of finding a LP.

Two fundamental architectures are thus introduced in this chapter: one is based on the success of Ch. 7 and uses a RAE for path history feature extraction, and another which moves away from the RAE design and learns a CNN policy online. The overall data flow is presented in Fig. 8.8 and both use the CAE from Sect. 8.5.2 to carry out the PDM and terrain feature extraction. Similarly, both use the same action-space and reward function whilst differing in the observation space definition. The latter

is outlined for the respective architectures in Sect. 8.6.1 and Sect. 8.6.2, whilst the action-space and reward are defined in Sect. 8.6.3 and Sect. 8.6.4 respectively. Finally, the search path constraints are presented in Sect. 8.6.5.

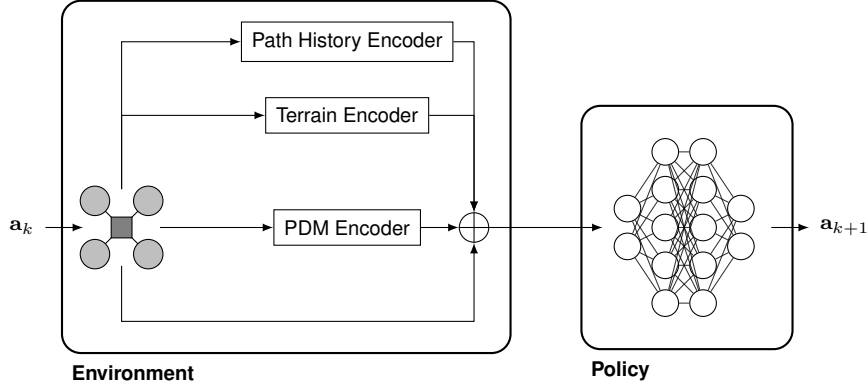


Figure 8.8: RAE and CAE based architecture layout diagram. The terrain, PDM, and path history encoders are pre-trained and deployed within the simulation to reduce memory usage during training.

8.6.1 Architecture A: Recurrent Auto-Encoder

Ch. 7 showed that the RAE was a good solution to processing the varying sized path history observation. Sect. 8.5.1 then explored other recurrent architectures and determined that the LSTM recurrent unit combined with MHSA into the LSTM-Attn RAE architecture was the most performant across multiple benchmarks. This, combined with the CNN-Attn AE from Sect. 8.5.2 to encode the PDM and terrain, then provides an AE-based policy similar to Ch. 7 and is thus titled hybrid auto encoder soft actor-critic (HAE_SAC).

The observation for this architecture is defined as the vector

$$\mathbf{s}_{\text{RAE},k} = \begin{bmatrix} f_{\theta_{\text{path}}}(s_{\text{path},k}) \\ f_{\theta_{\text{PDM}}}(s_{\text{PDM}}) \\ f_{\theta_{\text{terrain}}}(s_{\text{terrain}}) \\ \frac{2(\sum_{i=0}^k U_i) - (\sum U)_{\text{limit}}}{(\sum U)_{\text{limit}}} \\ \frac{2(\sum_{i=0}^k E_i) - (\sum E)_{\text{limit}}}{(\sum E)_{\text{limit}}} \\ \frac{2t_k - T_{\text{limit}}}{T_{\text{limit}}} \\ \frac{\mathbf{v}_k}{1 + |\mathbf{v}_k|} \\ \mathbf{u}_{k-1} \end{bmatrix}, \quad (8.22)$$

where T_{limit} , $(\sum U)_{\text{limit}}$, and $(\sum E)_{\text{limit}}$ are tunable environmental parameters that control the length of the simulation and are defined in Sect. 8.7. $\mathbf{v}_k$ is the UAV velocity at step k , and $\mathbf{u}_{k-1}$ is the previous action defined in Sect. 8.6.3.

Whilst the value of k increases monotonically, the simulated time steps between each successive waypoints are not constant. This is due to the trajectory planning

scheme outlined in Sect. 8.4.3 which optimises time between waypoints. The path history could therefore grow by a large number of points with every step which would reduce the efficacy of the path history encoder. Three time-wise evenly spaced points and the final position are therefore sampled from the set of new points in the path history to decrease the total number of encodings required.

8.6.2 Architecture B: Convolution Neural Network

In the previous iterations of the RL-based search planning the limiting factor has been the path history. Ch. 6 applied frame-stacking but Ch. 7 found this to be ineffective primarily due to unstable training most likely caused by the zero-value padding which lead to slow training as well as the hard limit on the number of waypoints. The RAE was therefore successfully applied to the problem which lead to reduced network sizes, decreased training times, and better search planning performance. However, as briefly alluded to in Sect. 8.6.1, recurrent networks are not a golden-bullet and also have soft upper limits on the number of successive data points before performance is impacted. A method independent of time which is still able to fully encode the entire path history is therefore required.

This architecture, titled convolution neural network soft actor-critic (CNN_SAC), therefore uses the notion of the search area mask to do this. This mask is the path history buffered by the search radius r and then rasterised into an image with grid-cells intersecting the polygon being marked as 1 and 0 otherwise as defined by

$$\begin{aligned}
 \mathcal{G} &= \{(i, j) \mid i \in [0, 1, \dots, N-1] \wedge j \in [0, 1, \dots, M-1]\}, \\
 M(\mathbf{x})_H &= \begin{cases} 1, & \mathbf{x} \in H \\ 0, & \mathbf{x} \notin H \end{cases}, \\
 m_x &= \frac{x_{\max} - x_{\min}}{N}, \\
 m_y &= \frac{y_{\max} - y_{\min}}{M}, \\
 \mathbf{T} &= \begin{bmatrix} m_x & 0 & x_{\min} + \frac{m_x}{2} \\ 0 & m_y & y_{\min} + \frac{m_y}{2} \\ 0 & 0 & 1 \end{bmatrix}, \\
 \mathbf{s}_{\text{search}, \cdot} &= M \left(\mathbf{T} \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \right) \quad \forall (i, j) \in \mathcal{G},
 \end{aligned} \tag{8.23}$$

where $\mathbf{T}$ is the affine transformation matrix from pixel- to world-space, and H is the search area polygon. However, as shown in Fig. 8.9a, this leads to a loss of precision due to aliasing. This problem is the same as faced, and overcome using continuous methods, in Ch. 5. In this scenario, however, the use of a continuous representation is not feasible as this would require the use of a recurrent architecture, which is already

achieved by the RAE architecture.

A second search area mask centred around the previous $\mathbf{C}$ waypoint, named the *egocentric* search area mask and seen in Fig. 8.9b, is therefore used. This mask provides a high-quality insight into the immediately surrounding path history and through the use of both the global and egocentric masks the entire path history is observable. This modifies the affine transformation matrix $\mathbf{T}$ from Eqn. 8.23 to become

$$\mathbf{T} = \begin{bmatrix} \frac{2s}{N} & 0 & C_{x,k-1} + \frac{s(1-N)}{N} \\ 0 & \frac{2s}{M} & C_{y,k-1} + \frac{s(1-M)}{M} \\ 0 & 0 & 1 \end{bmatrix}. \quad (8.24)$$

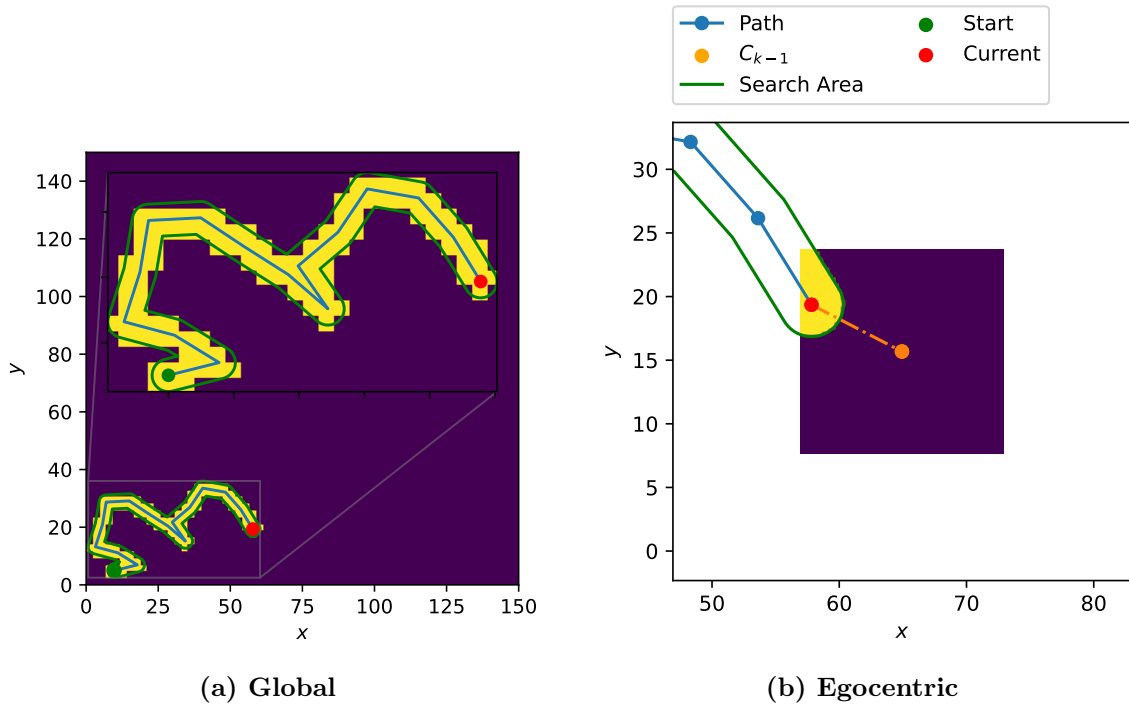


Figure 8.9: Example rasterised path histories masks for the (a) global and (b) egocentric cases. The global mask provides a coarse top-level view of the visited regions whereas the egocentric mask allows higher quality insights into the local information space without suffering from severe aliasing.

The observation for this architecture is then defined as the vector

$$\mathbf{s}_{\text{CNN},k} = \begin{bmatrix} g_{\theta_{\text{search,ego}}}(s_{\text{search,ego},k}) \\ g_{\theta_{\text{search,global}}}(s_{\text{search,global},k}) \\ f_{\theta_{\text{PDM}}}(s_{\text{PDM}}) \\ f_{\theta_{\text{terrain}}}(s_{\text{terrain}}) \\ \frac{2(\sum_{i=0}^k U_i) - (\sum U)_{\text{limit}}}{(\sum U)_{\text{limit}}} \\ \frac{2(\sum_{i=0}^k E_i) - (\sum E)_{\text{limit}}}{(\sum E)_{\text{limit}}} \\ \frac{2t_k - T_{\text{limit}}}{T_{\text{limit}}} \\ \tilde{\mathbf{x}}_k \\ \frac{\mathbf{v}_k}{1+|\mathbf{v}_k|} \\ \mathbf{u}_{k-1} \end{bmatrix}, \quad (8.25)$$

where $\mathbf{s}_{\text{search,ego}} \in \mathbb{R}^{(1 \times 48 \times 48)}$ and $\mathbf{s}_{\text{search,global}} \in \mathbb{R}^{(1 \times 48 \times 48)}$ are the egocentric and global search area masks respectively. These are transformed by the transformation denoted by g to highlight that this is a component of the RL policy as seen in Fig. 8.10. Transformations labelled by f_{θ} are the encoder of the respective AE with parameters θ . $\tilde{\mathbf{x}}_k$ is the bounds-based normalised position at step k .

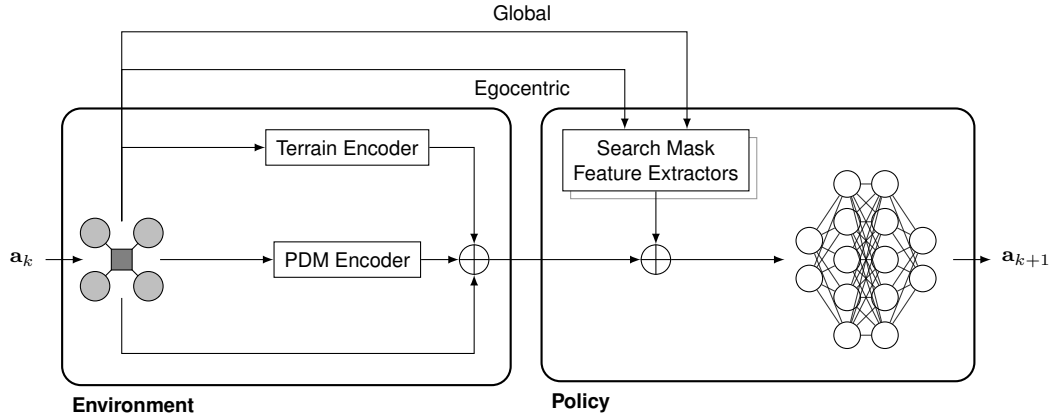


Figure 8.10: CNN policy architecture layout diagram. The terrain and PDM encoders are pre-trained and deployed within the simulation to reduce memory usage during training. The search area feature extractors follow the architecture outlined by Mnih *et al.* [230].

8.6.3 Action-space

The action-space is defined as $\mathbf{a} = [a_x, a_y]$ where $a_x, a_y \in [-1, 1]$. This is converted to the next target position $\mathbf{C}$ and current target position $\mathbf{B}$, as defined in Sect. 8.4.3, through

$$\begin{aligned} \mathbf{u}_k &= s \cdot \min \left(\frac{\mathbf{a}_k}{\|\mathbf{a}_k\|}, 1 \right), \\ \mathbf{B}_k &= \mathbf{x}_k + \mathbf{u}_{k-1}, \\ \mathbf{C}_k &= \mathbf{B}_k + \mathbf{u}_k, \end{aligned} \quad (8.26)$$

where $\min$ is the element-wise minimum, and $\mathbf{x}_k$ is the current position. The $\min(\cdot, 1)$ term limits the possible magnitude of $\mathbf{u}_k$ such that $\|\mathbf{u}_k\| \leq 1$ as seen in Fig. 8.11.

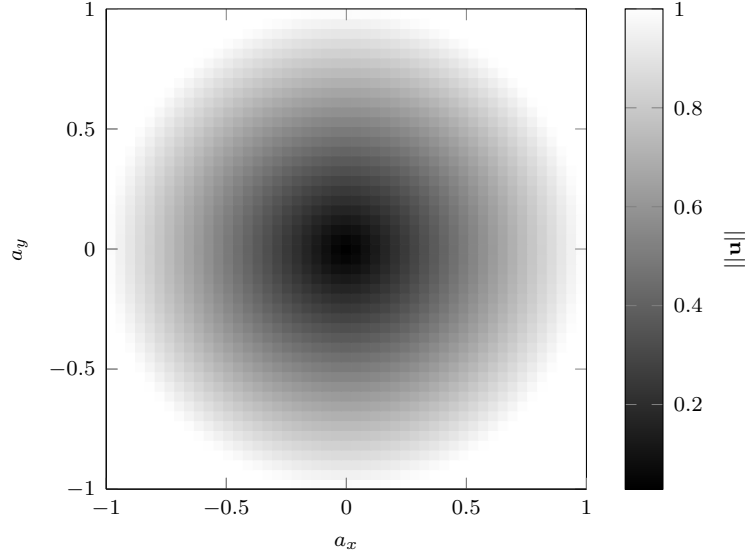


Figure 8.11: The magnitude of the action space defined in Eqn. 8.26 over the a_x and a_y axis. This figure demonstrates that this approach avoids the $\sqrt{2}$ magnitude in the corners.

8.6.4 Reward Function

The rewards scheme from Sect. 6.4.1.3 and Sect. 7.4.3 is employed in this section with a minor modification. In this chapter, the Douglas-Ramer-Peucker algorithm [325], a line simplification algorithm, is used to reduce the search path complexity before calculating the search area. Line simplification is required because of the underlying dynamic system, which produces many intermediary steps between each waypoint, which improves performance during the cubature procedure. The performance gain is achieved by the reduction of the number of triangles created during the constrained Delaunay triangulation (see Ch. 5) by removing unnecessary points. In this formulation, the search efficiency is the objective being maximised by the agent. This value is based on the integration of the PDM, defined as a MMBG (see Eqn. 5.5), over the search area, the accumulated probability at time t , and is given by

$$p_t = \oint_{H_t} p(\mathbf{x}) dH_t. \quad (8.27)$$

The difference of this value, normalised by the maximum probability step, is given by

$$r = \frac{p_A A_{area}(p_t - p_{t-1})}{R_{buffer}(\pi R_{buffer} + 2\lambda)}, \quad (8.28)$$

and labelled the scaled probability efficiency reward. More information on the derivation and design of this can be found in Sect. 6.4.1.3.

Further reward shaping is applied to prevent out-of-bounds scenarios and to discourage visiting area of low probability. This is then the augmented reward r' which is given by

$$r' = \begin{cases} -w_{oob}, & \mathbf{x}_t \notin [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \\ w_r r, & \Delta p_t > \epsilon \\ -w_0, & \text{else} \end{cases}, \quad (8.29)$$

with the reward function parameters from Tbl. 8.10.

Table 8.10: Reward function parameters.

Parameter	Value
w_{oob}	1
w_r	25
ϵ	1×10^{-10}
w_0	0.05

8.6.5 Search Path Constraints

The simulation environment has four termination criteria, which impose constraints on the generated search path. Like Ch. 6 and Ch. 7, the environment terminates when the current position, or any position along the resultant simulated path, is out-of-bounds. However, unlike the aforementioned sections, the number of waypoints limit replaced with a time limit T_{limit} such that. Furthermore, a control limit, $(\sum U)_{\text{limit}}$, is introduced to promote efficient control of the system, which assumes a correlation between power draw the square of the propeller speed [305]. These constraints are defined as

$$\begin{aligned} \mathbf{x}_k &\in [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}], \\ \sum_{i=0}^k T_{i-1:i} &\leq T_{\text{limit}}, \\ \sum_{i=0}^k \sum_{j=1}^4 \delta t P_{j,i-1:i}^2 &\leq (\sum U)_{\text{limit}}, \end{aligned} \quad (8.30)$$

where $\mathbf{P}$ refers to the propeller rotational velocities from Eqn. 8.3, and k is the total number of simulation steps at a constant time step of δt .

8.7 Results and Discussion

8.7.1 Parameter Configurations

Neither the dynamics of the UAV nor the terrain were modelled in Ch. 6 and Ch. 7. It is thus important to consider this when comparing analytical results to those of this chapter.

During preliminary testing it was found that the non-dynamically aware algorithms were inefficient and resulted in substantially shorter paths than the benchmark or dynamically aware algorithms. Both a *short* and *long* parameter configurations are therefore used throughout this section.

Similarly, the RL methods from the aforementioned chapters do not consider a sloped terrain, nor MMBGs with varying number of modes or different covariance matrices. This further gives rise to the *flat* and *sloped* parameter configurations.

This then results in the four permutations given in Tbl. 8.11 with the parameter definitions found in Tbl. 8.12. *short-flat* is biased towards the work from prior chapters, where as *long-sloped* favours the algorithms presented in this chapter.

Finally, the remaining simulation parameters are given in Tbl. 8.13 and the RL parameters are given in Tbl. 8.14

Table 8.11: The permutation matrix of the parameter groups defined in Tbl. 8.12.

	Short	Long
Flat	Short-Flat	Long-Flat
Sloped	Short-Sloped	Long-Sloped

Table 8.12: Parameters used to define the search path generation constraints and randomisation.

Configuration	Parameter	Value	Unit
Short	T_{limit}	256	s
	$(\sum U)_{\text{limit}}$	20000	
Long	T_{limit}	512	s
	$(\sum U)_{\text{limit}}$	40000	
Flat	$N_{\text{gaussian,min / max}}$	4	
	$\sigma_{\text{min / max}}$	500	
	ρ_{max}	0	
	$m_{\text{min / max}}$	0	m m ⁻¹
	$c_{\text{min / max}}$	0	m
Sloped	$N_{\text{gaussian,min}}$	1	
	$N_{\text{gaussian,max}}$	5	
	σ_{min}	100	
	σ_{max}	1000	
	ρ_{max}	0.5	
	m_{min}	0	m m ⁻¹
	m_{max}	1	m m ⁻¹
	$c_{\text{min / max}}$	0	m

8.7.2 Benchmark Algorithms

As in Ch. 6 and Ch. 7, the LHC_GW_CONV [155] (introduced in Sect. 5.3.1) and lawnmower [37] algorithms will be used to provide comparative benchmarks. These

Table 8.13: Simulation parameters used in this chapter.

Parameter	Value	Unit
λ	8	m
R_{buffer}	2.5	m
V	2	m s^{-1}
a_{max}	1	m s^{-2}
$x_{\text{min}}, y_{\text{max}}$	0	m
$x_{\text{min}}, y_{\text{max}}$	150	m
ϵ	1×10^{-10}	
w_{oob}	1.0	
w_r	25	
w_0	0.05	

Table 8.14: Reinforcement learning hyperparameters for CNN_SAC and HAE_SAC in this chapter. Default values from [290] v2.1.0 were used if not listed in the table.

Hyperparam.	CNN_SAC and HAE_SAC
Net width	2048
Net depth	2
τ	0.05
# envs	32
Training freq.	32
Batch size	512
Buffer size	2 000 000
Gradient Steps	64
N_{epochs}	5×10^7
Learning rate (start)	0.0003
Learning rate (end)	0.00015
Learning rate (inc. end)	4×10^7
Learning starts	16384
Target entropy	-2
Optimiser	Adam

are expected to perform poorly due to a lack of dynamic and terrain awareness.

Since RL, with a continuous action space, has shown a clear advantage over the aforementioned benchmark methods, an additional algorithm has been implemented. Maximum positioning with the travelling salesman problem (Max_TSP) is combination of the PDM points-of-interest identification from [138] and the travelling salesman problem-based mode predictive control pre-seeding for prioritised coverage planning from Schweppe *et al.* [326]. The core of this algorithm works on a two step principle: identification of areas of interest followed by solving of the path through a travelling salesman problem formulation.

The identification of the areas of interest over the PDM in Max_TSP is an iterative process, which uses a high definition discretised PDM to select the point with the maximum probability. An area surrounding this point, equivalent to the search radius

r , is then masked out and the process repeats.

Secondly, a fully connected graph is constructed to find the search path from the set of points. Each point then becomes a node in the graph and every edge $(\mathbf{u}, \mathbf{v})$ is assigned a score $c = \|\mathbf{v} - \mathbf{u}\|_2$. The travelling salesman problem path originates at the start location, which is randomly sampled from within the search area, and is solved for such that it terminates at the start location.

The simulated annealing [158], [327] travelling salesman problem solver from *networkx* v.3.4.2 [328] with default parameters was used. The PDM was discretised into a 64×64 image, and 128 points of interest were identified.

8.7.3 Dynamics and Static Analysis

The benchmark algorithms were designed with the core assumption that a UAV navigating the search path is able to perfectly follow it. This, however, is a naive assumption since it would require the UAV to move so slowly as to be useless, particularly in the event of a corner. Two analyses are therefore performed for the benchmark algorithms; dynamic and static.

A dynamic analysis is one where the entire resultant path of the UAV is assessed after a simulation execution of the search path. Importantly, this includes the offset from the ideal path due to errors in the tracking.

The static analysis is then the counterpart to the dynamic analysis without the consideration of the dynamics. However, due to the introduction of the control action limit, the path is truncated at the final time as the dynamic analysis. This is carried out by taking the mean speed of the UAV along the dynamic path and truncating the original waypoint path at the resultant distance of the final time and mean speed.

8.7.4 Short-Flat Results

Tbl. 8.15 shows that $T_{\text{limit}} = 256\text{ s}$, from Tbl. 8.12, is never reached by any of the algorithms. However, the control action limit, $(\sum U)_{\text{limit}} = 20000$ is the termination condition for all algorithms. In terms of both the time and control action limits, both CNN_SAC and HAE_SAC are most performant.

Time and control action alone are not indicative of the performance since a UAV can hover and maximise these metrics. A better insight is provided by Fig. 8.12 which shows the area of the search area. It follows that a search with a larger area is better, on average, at maximising the chances of finding a LP than one with a smaller area. In this regard, both CNN_SAC and HAE_SAC are substantially better than the benchmark algorithms with values just below $a_D = 1500$. Interestingly, the benchmark algorithms are grouped around $a_D = 750$. This shows that the dynamic-awareness is critical to simply maximise area covered. This is further highlighted by the search path area efficiency results from Fig. 8.13, which shows that the static analysis of the benchmark algorithms is close to 1.0, whilst their dynamic analysis has a value around 0.6.

Table 8.15: $T_{\text{limit}} = 256 \text{ s}$, $U_{\text{limit}} = 20\,000$, $E_{\text{limit}} = 1 \times 10^5 \text{ J}$.

Algorithm	Environment	$e_{a,D}$ [$\times 10^{-1}$]		d [$\times 10^2$]		$e_{p,D}$ [$m \times 10^{-1}$]		$\sum^u$ [$\times 10^4$]		t [$s \times 10^2$]	
		Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
CNN_SAC	Dynamic	8.732	0.558	3.323	0.334	1.592	0.370	1.946	0.057	1.413	0.055
HAE_SAC	Dynamic	8.273	0.710	3.356	0.346	1.504	0.373	1.945	0.047	1.413	0.049
Max_TSP	Static	9.129	0.752	1.220	0.238	0.614	0.194	1.927	0.059	1.399	0.043
	Dynamic	5.566	0.825	2.682	0.221	0.850	0.269	1.927	0.059	1.399	0.043
LSTMAE_SAC	Static	9.558	0.722	1.140	0.222	0.522	0.236	1.874	0.218	1.360	0.158
	Dynamic	5.631	0.767	2.803	0.358	0.731	0.319	1.874	0.218	1.360	0.158
FS_SAC_CONV2D	Static	9.871	0.273	1.179	0.174	0.454	0.218	1.912	0.094	1.388	0.068
	Dynamic	5.789	0.722	2.798	0.193	0.619	0.285	1.912	0.094	1.388	0.068
Lawnmower	Static	9.253	0.402	1.291	0.184	0.355	0.257	1.929	0.064	1.400	0.046
	Dynamic	6.177	0.510	2.609	0.188	0.473	0.335	1.929	0.064	1.400	0.046
LHC_GW_CONV	Static	9.999	0.000	1.354	0.000	0.207	0.181	1.937	0.000	1.408	0.000
	Dynamic	5.599	0.000	2.772	0.000	0.229	0.201	1.937	0.000	1.408	0.000

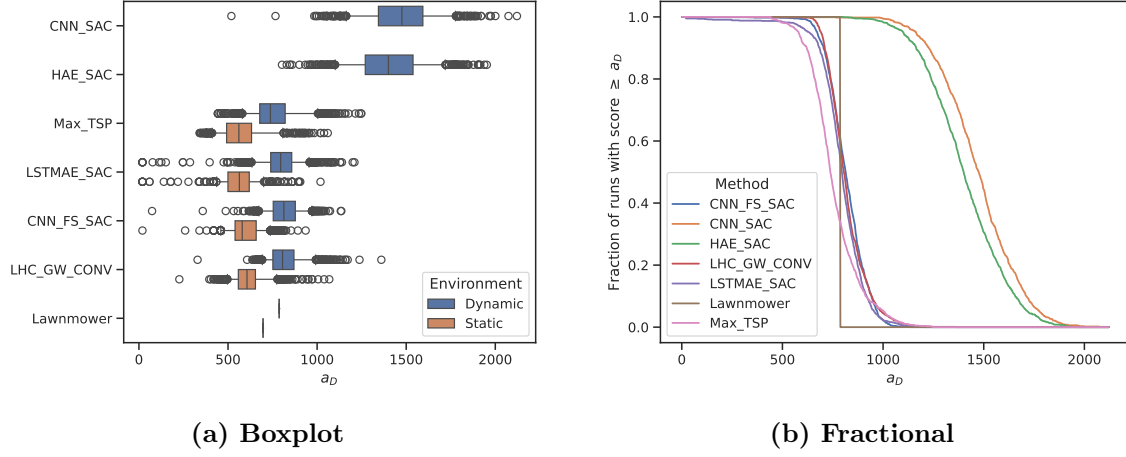


Figure 8.12: Final search path area for each algorithm using the *short-flat* parameter setup. The static environment definition is not compatible with the architectures of CNN_SAC, and HAE_SAC. Therefore, these data points are unavailable in (a).

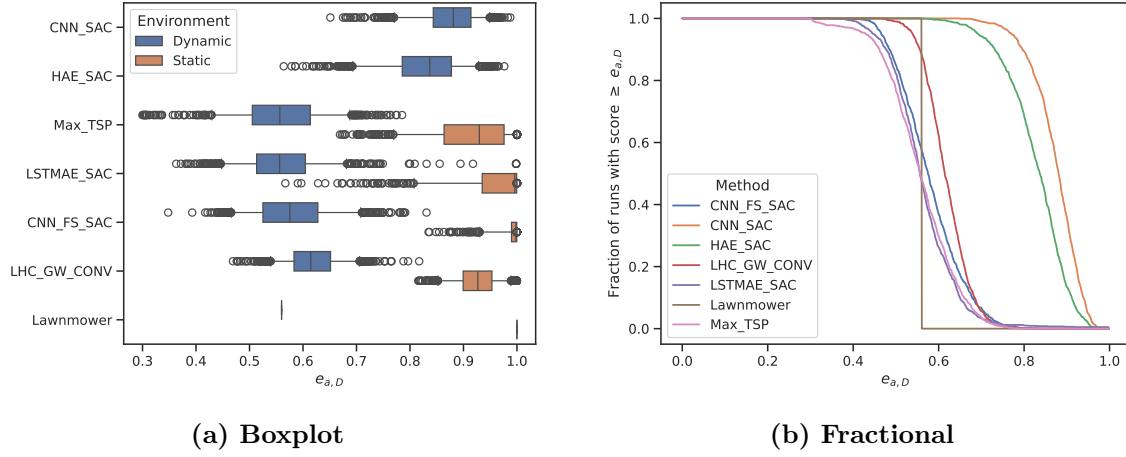


Figure 8.13: Final search path area efficiency for each algorithm using the *short-flat* parameter setup. The static environment definition is not compatible with the architectures of CNN_SAC, and HAE_SAC. Therefore, these data points are unavailable in (a).

The core metric, however, is the final search path probability efficiency shown in Fig. 8.14. CNN_SAC and HAE_SAC have metric values of 1.592×10^{-1} and 1.504×10^{-1} respectively, which is almost twice that of the next best algorithm; Max_TSP with 0.850×10^{-1} . It is noteworthy that the static analysis performs worse than the dynamic analysis for all benchmark algorithms. This is explained by the low path length of the static analysis in Tbl. 8.12.

Looking at the search path accumulated probability per unit length from Fig. 8.15, however, shows that the static analysis of Max_TSP outperforms all other methods. CNN_SAC and HAE_SAC perform the best within the dynamic analysis with lawn-mower performing particularly poorly.

Table 8.16: $T_{\text{limit}} = 512 \text{ s}$, $U_{\text{limit}} = 40\,000$, $E_{\text{limit}} = 1 \times 10^5 \text{ J}$.

Algorithm	Environment	$e_{a,D}$ [$\times 10^{-1}$]		d [$\times 10^2$]		$e_{p,D}$ [$m \times 10^{-1}$]		$\sum u$ [$\times 10^4$]		t [$s \times 10^2$]	
		Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
CNN_SAC	Dynamic	7.459	0.901	6.799	0.703	3.785	1.216	3.907	0.308	2.837	0.234
HAE_SAC	Dynamic	6.670	0.863	6.996	0.538	3.328	1.215	3.943	0.116	2.861	0.098
Max_TSP	Static	7.890	0.631	4.196	0.682	2.859	1.101	3.553	0.375	2.581	0.272
	Dynamic	6.935	0.630	5.386	0.718	3.254	1.235	3.553	0.375	2.581	0.272
Lawnmower	Static	9.848	0.021	7.185	1.663	1.456	1.602	3.954	0.038	2.865	0.034
	Dynamic	9.216	0.538	7.761	1.495	1.477	1.624	3.954	0.038	2.865	0.034
LHC_GW_CONV	Static	8.734	0.266	4.741	0.551	1.273	1.135	3.889	0.309	2.825	0.226
	Dynamic	7.682	0.317	6.004	0.595	1.400	1.231	3.889	0.309	2.825	0.226

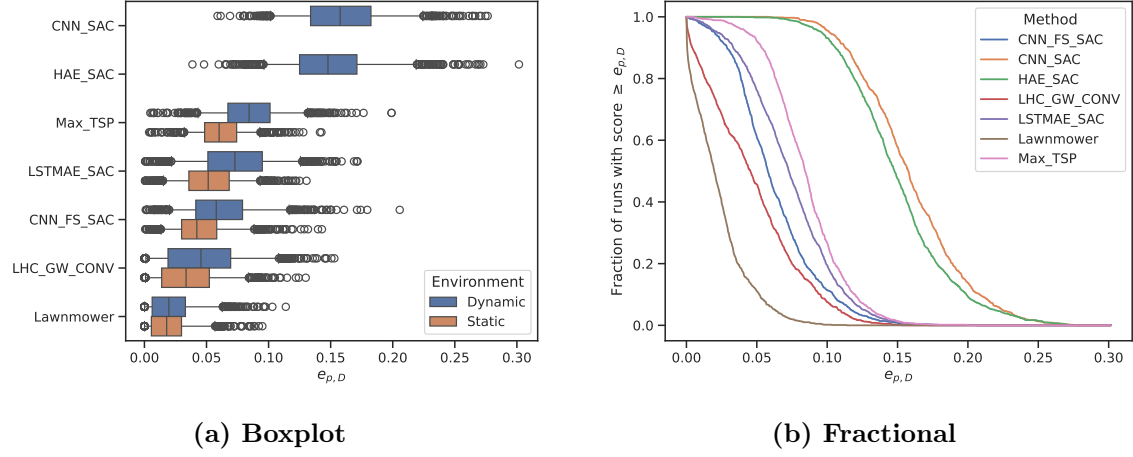


Figure 8.14: Final search path probability efficiency for each algorithm using the *short-flat* parameter setup. The static environment definition is not compatible with the architectures of CNN_SAC, and HAE_SAC. Therefore, these data points are unavailable in (a).

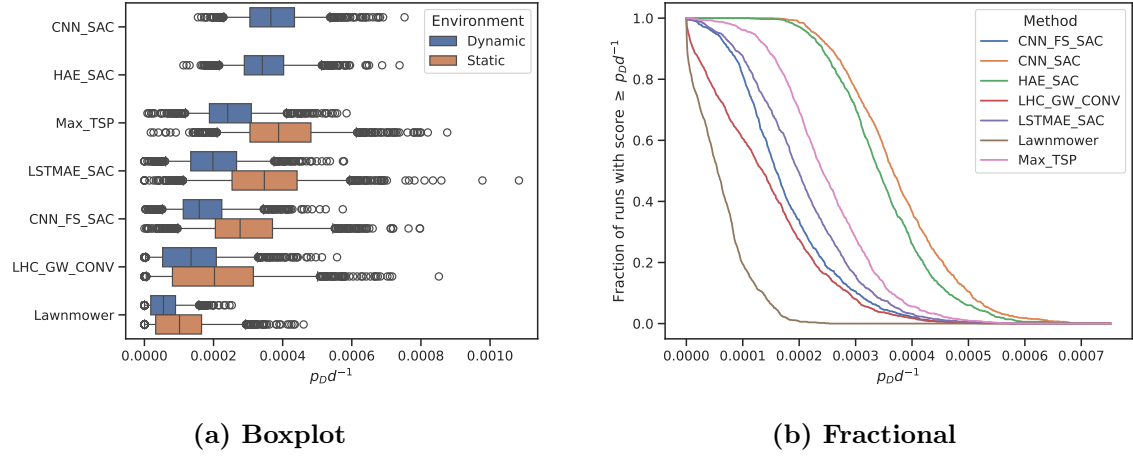


Figure 8.15: Final search path accumulated probability per unit length for each algorithm using the *short-flat* parameter setup. The static environment definition is not compatible with the architectures of CNN_SAC, and HAE_SAC. Therefore, these data points are unavailable in (a).

8.7.5 Long-Sloped Results

The time limit for the *long-sloped* parameter setup, from Tbl. 8.12, is never reached whilst the control limit is. This is similar to the *short-flat* results, however in this configuration the lawnmower algorithm is the most performant by a small margin. As previously discussed, endurance is not sufficient to search but the search area is what is important. In this regard lawnmower is also the most performant, as shown in Fig. 8.16, and Tbl. 8.16. Fig. 8.16b in particular highlights the exceptionally high search path area performance of the lawnmower algorithm. This could be due to the long and straight paths, which are simple for the UAV to traverse. The search path area efficiency, shown in Fig. 8.17, confirms this. A high area efficiency has few overlaps and corners, which the lawnmower avoids. HAE_SAC performs worst in this metric with CNN_SAC being in the middle of the pack.

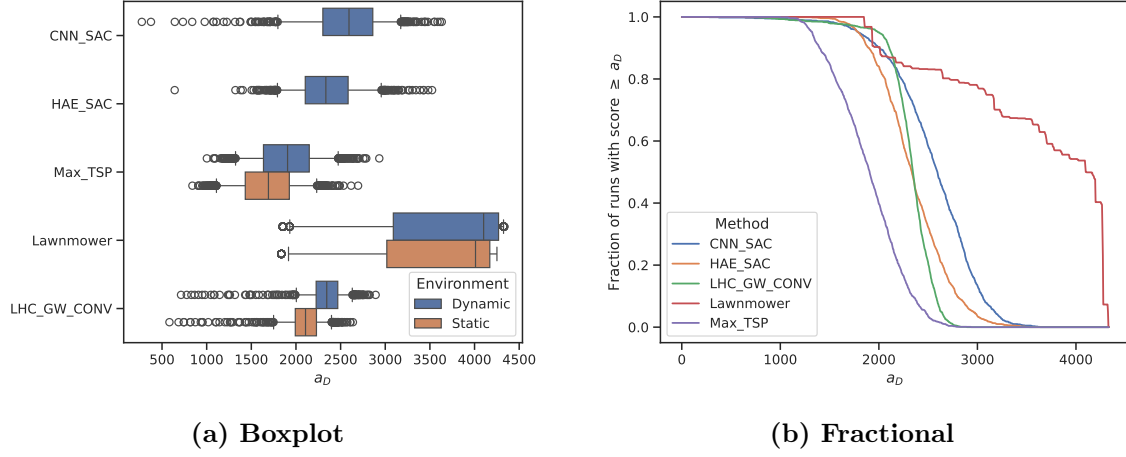


Figure 8.16: Final search path area for each algorithm using the *long-sloped* parameter setup. The static environment definition is not compatible with the architectures of CNN_SAC, and HAE_SAC. Therefore, these data points are unavailable in (a).

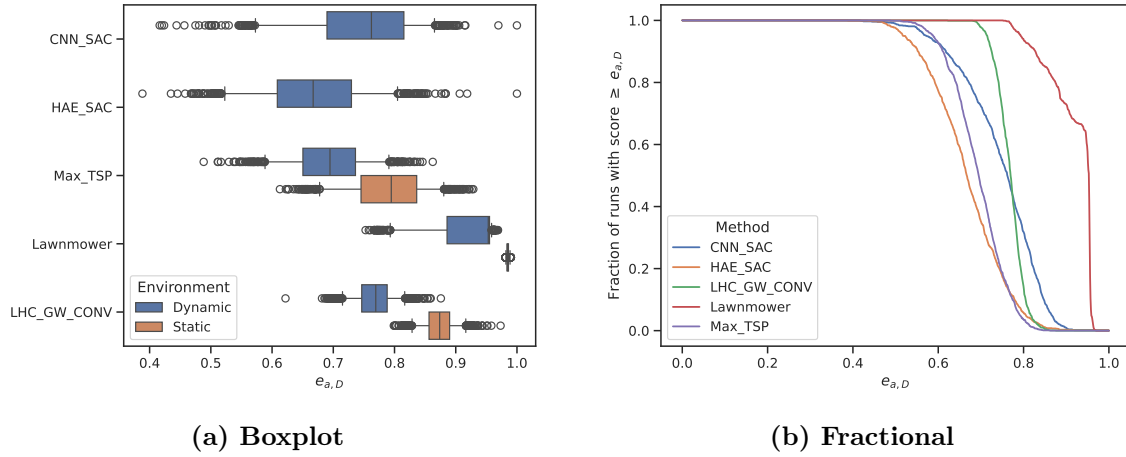


Figure 8.17: Final search path area efficiency for each algorithm using the *long-sloped* parameter setup. The static environment definition is not compatible with the architectures of CNN_SAC, and HAE_SAC. Therefore, these data points are unavailable in (a).

The search area, endurance, and area efficiency are irrelevant, however, if the high probability areas are not effectively traversed. Fig. 8.18 shows that even with poor search path characteristics CNN_SAC and HAE_SAC have the highest probability efficiency with values of 3.785×10^{-1} and 3.328×10^{-1} respectively. However, HAE_SAC only outperformed Max_TSP by 0.074×10^{-1} . Noteworthy is the very low probability efficiency scores for both LHC_GW_CONV and lawnmower at just 1.400×10^{-1} and 1.477×10^{-1} respectively.

Looking at the accumulated probability per unit length in Fig. 8.19 shows that Max_TSP utilises the search area most efficiently. However, Max_TSP has the lowest search path area (from Fig. 8.16) implying that the locations searched by it are highly effective but the path itself is inefficient for a UAV. CNN_SAC has the second most efficient area allocation, with HAE_SAC being third. This, in conjunction with the probability efficiency results from Fig. 8.18 and the area results from Fig. 8.16, suggest

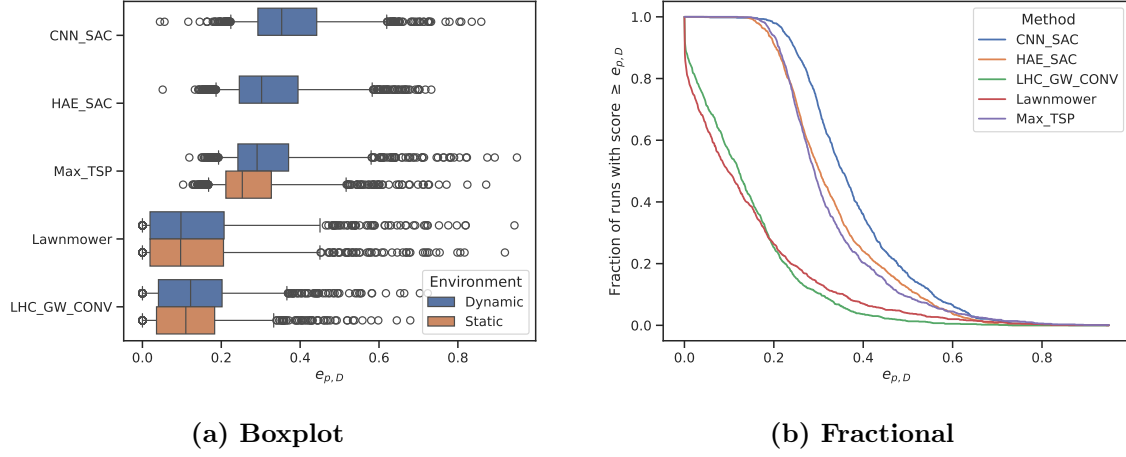


Figure 8.18: Final search path probability efficiency for each algorithm using the *long-sloped* parameter setup. The static environment definition is not compatible with the architectures of CNN_SAC, and HAE_SAC. Therefore, these data points are unavailable in (a).

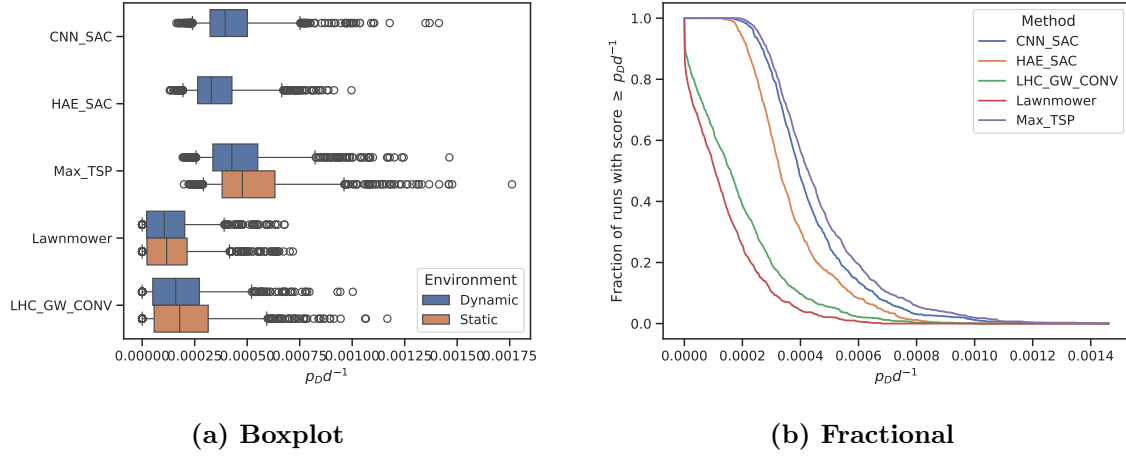


Figure 8.19: Final search path accumulated probability per unit length for each algorithm using the *long-sloped* parameter setup. The static environment definition is not compatible with the architectures of CNN_SAC, and HAE_SAC. Therefore, these data points are unavailable in (a).

that the algorithms from this chapter leverage their dynamic awareness to maximise endurance first and search location second. However, given the ultimate goal of maximising the probability of finding the LP, this is a viable trade-off as probability efficiency is maximised.

8.8 Section Summary and Conclusion

In this chapter two RL architectures, HAE_SAC and CNN_SAC, were developed and trained using SAC. The training environment modelled a six degree of freedom quadcopter with a navigation stack consisting of a low-level LQR controller and an optimal polynomial trajectory generation scheme with look-ahead functionality to reduce the action-space complexity. Furthermore, this environment modelled terrain elevation to reduce to the simulation-to-reality gap as WiSAR typically occurs in mountainous

terrain. Likewise, to reduce the simulation-to-reality gap, the PDM was used in an image-based representation which expands the flexibility and increases the practicality of these architectures.

Following the performance gained from applying the LSTM-based RAE to the path history encoding problem in Ch. 7, an in-depth analysis of six competing RAE architectures was carried out in Sect. 8.5.1. This explored the use of vanilla recurrent architectures, as well as the addition of MHSA to potentially enhance the encoding potential. The opportunity was also taken to explore training through SGD and Adam, two of the most popular optimisation methods for ML, to compare performance. It was found that the LSTM-Attn architecture trained using SGD on data generated by LHC_GW_CONV outperformed the competing architectures in three out of the four synthetic benchmarks by a significant margin. LSTM-Attn was only beaten by the LSTM architecture in the *line* benchmark whilst still outperforming the remaining models. A similar study was done on the CNN-based AE for encoding the PDM and terrain. In this investigation, as with the path history encoding, the attention-based CNN-Attn outperformed the vanilla CNN architecture in both applications.

HAE_SAC thus used the findings of Sect. 8.5 to develop a hybrid AE architecture that employed a LSTM-Attn RAE for the path history encoding, and a CNN-Attn AE for the PDM and terrain encoding. The alternative CNN-based architecture, CNN_SAC, was developed alongside HAE_SAC to explore a simpler formulation. A final third (non-RL) algorithm, Max_TSP, was also developed since the benchmark algorithms, lawnmower and LHC_GW_CONV, from Ch. 6 and Ch. 7 were insufficient to challenge the RL-based algorithms.

The search planning results from the analysis of two simulation configuration showed the exceptional performance of both HAE_SAC and CNN_SAC compared to the benchmark algorithms. Max_TSP, however, also performed well compared to the other benchmark algorithm. It is therefore clear that RL is a powerful technique for search planning that is capable of learning the underlying dynamics of the platform. Ultimately, the architecturally simpler, CNN_SAC outperformed HAE_SAC. This would imply that the additional AEs for the feature extraction stage result in a more challenging observation-space than the purely image-based one used for CNN_SAC.

Chapter 9

Conclusions, Limitations, and Future Work

9.1 Summary and Conclusions

This work explored the application of ML within the search phase of WiSAR. Specifically, the PDM generation and search planning were targeted as this directly applies to the application of autonomy for UAVs. PDM generation alone is a valuable tool to inform the search leader which areas to prioritise. A search planning algorithm must be used to achieve this area prioritisation on an autonomous platform like a UAV. Thus, this work has developed a holistic framework that can take the LP parameters (PLS, profile, time, and more) and generate the aforementioned search path.

The use of ML for the core PDM generation algorithm in Ch. 4 was considered, but a lack of data availability made this approach infeasible. Therefore, a different profile-driven approach was developed that combined GIS maps with psychological datasets. A limited PLS dataset was resampled using a GP kernel to drive the design of the experiment. A PDM was generated for each sampled PLS location, and the resultant aggregated land cover find location categorisation dataset was compared to the input dataset from the literature using the Kullback-Leibler symmetric divergence metric. Ultimately, it was found that the proposed method matched the input dataset. Adding the GP-driven analysis and a higher fidelity Head2Paths behaviour further decreased the divergence score. A comparative comparison could not be carried out due to a lack of data. However, the algorithm's efficacy was proven to be better than a random distribution by order of magnitude; the random distribution and the proposed method had a score of 159.16 and 61.56, respectively.

Ch. 5 explored the use of continuous and discrete integration in the context of calculating the efficacy of a search path. This case study was undertaken to validate using the continuous integration method that was implemented because the search planning literature exclusively uses discrete integration. It was found that continuous integration techniques were orders of magnitude faster and more accurate than the discrete methods. For the discrete methods to compare in speed, the discretised PDM

must be less than 17×17 in size. However, for the relative error of the discrete approach to be within 0.5 % of the continuous method, the discretised PDM must be larger than 300×300 at which the relative calculation time exceeds 10 000 %. It is, therefore, clear that the continuous integration approach should be taken wherever possible to decrease both computational times and integration errors.

The case study on the integration techniques directly supported using a continuous action space for RL as it enabled fast and virtually noise-less reward calculations. With a continuous action space comes a continuous state, which results in a complex search path that greatly benefits from the continuous integration method. Ch. 6 was the first iteration of the search planning algorithm and used frame-stacking to represent the path history. This was a requirement of RL as entire state observability must be maintained, and revisiting areas already seen would result in inefficient searches. This resulted in long training times and a large number of parameters, but a search path that outperformed the benchmark algorithms with a final probability efficiency of 0.19. This result showed that RL with a continuous action space and the proposed reward function was a viable method.

Ch. 7 challenged the use of frame-stacking and proposed a RAE-based approach. It was hypothesised that a dedicated, pre-trained and static feature extractor would allow RL to better focus the training on the control aspect of waypoint planning. This was evaluated over many architectures, including frame-stacking and various RL algorithms, but with a substantially smaller policy network size than used in Ch. 6. It was found that the RAE architecture marginally outperformed the frame-stacking one with a probability efficiency of 0.20. However, the standout result was a reduction in the number of learnable parameters to 14 % of the original architecture as well as a reduction in training time to 23 %. This primarily showed that feature engineering for the search planning problem is paramount in improving RL training characteristics.

Finally, in Ch. 8, a non-linear dynamical model of a quadcopter was implemented alongside a hierarchical optimal quintic trajectory planning and LQR control scheme. The simulated environment was further enhanced through the addition of a terrain model with collision detection. Furthermore, the endurance restriction was improved from being time-based to an approach rooted in the simulation system. This was done to reward search paths that were efficient for the underlying dynamic platform, which had not been done previously. Two models, HAE_SAC and CNN_SAC, were developed and trained in this environment. An additional third algorithm, based on the travelling salesman problem, was also implemented in order to increase competitiveness. It was found that the purely CNN-based CNN_SAC outperformed all other algorithms across the benchmarks. HAE_SAC, which used the LSTM-Attn and CNN architectures for the RAE and CAE, respectively, was also highly performant, albeit with a more complex architecture. Overall, however, the two dynamically aware methods were dominant, which shows the potential for RL to be highly customised to the deployment algorithm.

9.2 Future Work and Limitations

Sections Sect. 9.2.1, and Sect. 9.2.2 will discuss the future work and limitations of the PDM generation algorithm introduced in Ch. 4, and the RL-based search planning algorithms introduced in Ch. 6, Ch. 7, and Ch. 8 respectively. Finally, Sect. 9.2.3 discusses how the aforementioned methods can be combined into a comprehensive framework for search planning.

9.2.1 Probability Distribution Map Generation

Data availability is typically one of the core limitations for training ML models. This was no different in this work for the development and validation of the PDM generation algorithm developed in Ch. 4. Whilst the algorithm is driven through statistical data in the form of the land cover find location categorisation and mobility datasets, the lower level decision-making had to be tuned manually. This is a common approach used by the likes of Frakes *et al.* [39] and Drexel *et al.* [40], however the weights of the parameters are subjective without real quantitative origins. This can be replaced with statistical analysis if more LP data were available. Similarly, the use of ML models to replace the whole behaviour-based model, or subsets of the model, would also be a worthwhile endeavour. This could replace the weighted vector approach from Sect. 4.4.1.1 or the entire Head2Water behaviour from Sect. 4.4.1.1. Finally, as briefly discussed in Sect. 4.5, a large enough dataset from SMR or PSASU to validate the model does not exist. Out of approximately 300 PLS data points, less than 10 had an associated find location. Sava *et al.* [61] develops a framework called *MapScore* to evaluating a PDM using real-world data. However, MapScore is localised to specific locations in the USA and, at the time of writing, is not accessible. This is the reason for the comparison of the aggregated dataset of the land cover find location categorisation of the generated PDM and the input dataset, which may lead to unquantifiable noisy results based on how the land cover was defined.

It is therefore evident that LP data collection must be improved in order to increase the efficacy of PDM generation for the use by SMR or PSASU. The path, along with velocity data, taken by the LP from the moment they are lost till they are found is the most compelling to future research. Given a large enough dataset, a ML model can be trained to replicate the LP traversing the terrain using imitation learning [184]. How can this dataset be collected?

One approach would be to monitor the exact location of every person partaking in a mountainous activity. This poses many ethical and practical hurdles, such as data privacy laws. Another approach could be to introduce a voluntary scheme for mountaineers to track their movement using GPs, through apps like Strava [329], and to aggregate this dataset. This approach would, however, be skewed towards the tech-savvy physically active, which poses the question of how data on the youth or elderly categories would be collected. A combination of the aforementioned methods

with commitments from benefiting organisations like SMR and PSASU to assist and contribute to the data collection efforts is likely the most effective and realistic.

Data collection aside, there are a multitude of research avenues to explore. The use of clustering methods should be explored to increase the number of behavioural profiles. Current categories primarily focus on age, and activity, but people are multifaceted, which would likely yield new insights. Large language models, in the form of *AI agents* *, could then, for example, be used to extract key identifiers about the LP from the information at hand in real-time. This belief of the profile could then evolve over time as more data is collected, which would morph the PDM (time-dependent PDMs) and reduce uncertainty.

The use of large language models can also be extended to the PDM generation phase. As was discussed in Sect. 4.8, additional environmental variables should be included in the PDM stage, but creating a model that actions these new data layers is difficult. To solve this, a large language model could be used as a surrogate of an LP trying to decide whether to take shelter due to strong winds, or to keep following a river.

9.2.2 Search Planning

The simulation-to-reality gap within RL is the difference between the training and deployment environments. RL skirts the problem caused by a lack of training data by simply modelling the deployment domain, but simulations cannot be identical to the real world. However, the presence of *unknown-unknowns*, from the awareness-understanding matrix [330], can lead to substantial loss of performance or even catastrophic failure when deploying the trained RL policy.

Whilst the search planning algorithms presented in Ch. 6, Ch. 7, and Ch. 8 are likely to be robust to catastrophic failure due to the action-space definition using offset-based waypoints, the potential loss of performance during deployment cannot be ignored. For Ch. 6 and Ch. 7, this might be due to the search platform not meeting the assumption that it can perfectly track the waypoints. Manjanna *et al.* [43] encountered this problem during field-testing the generated waypoint-based paths. Ch. 8 embodied the dynamics of the search platform, a UAV, during training, which should minimise this issue. However, this leads to further questions surrounding the simulation-to-reality problem. How accurate is the model of the UAV? Can the trajectory planning and LQR controller be implemented in the real system? Is the terrain model adequate to represent the real DEM?

Quantifying the impact of the simulation-to-reality gap is thus of utmost importance. If this is within a safe threshold, deploying in real-world WiSAR scenarios will be straightforward. However, a more likely scenario is that the simulation-to-reality gap is substantial, requiring the training environment and procedure to be evaluated.

*AI agents are applied large language models that can reason, pursue goals, leverage tooling, and complete tasks.

Validation and verification of the UAV model to align it with those used by SMR and PSASU would likely be the first port of call. Adding elements like wind, air humidity, and pressure to the simulation would further improve its realism. Similarly, the current terrain and PDM models are independently sampled from random distributions. However, it is evident from Ch. 4 that they are tightly coupled, which should be further explored.

Alternative solutions to bridge the simulation-to-reality gap also exist and should be explored [331]. Dynamics randomisation [332], sim-to-real-to-sim [333], [334], action uncertainty [335], and imitation Learning [184] are a subset of the plethora of tools and could apply to the search planning problem.

The offset-based action space used by all search planning algorithms in some form should be further explored. The policy is currently restricted from going out-of-bounds through the termination of the environment, alongside a negative penalty. This trains the policy to avoid leaving the search area but does not guarantee it. Through algorithms like Maskable-PPO [336], action masking could provide this guarantee. Similarly, implementing dynamic clipping of the action would make it impossible for the policy to command an action, leading to the system going out of bounds. Mehta *et al.* [313] proposes a solution based on constrained optimisation to guarantee obstacle avoidance constraints similarly. The out-of-bounds problem is a fruitful endeavour for future work and can further lead to non-rectangular search area definitions, which would be more useful in real-world WiSAR scenarios.

The methods described in this work have focused on planning; putting the search platform in the right locations. To do this, it was assumed that a perfect sensor was used, which always detects the LP if they are within the sensor’s footprint. Future methods should consider the imperfections of cameras, the use of infra-red sensors, electromagnetic noise coming from the mobile devices of an LP [304], and other variables.

Multi-agent, multi-platform algorithms should be explored in order to close the simulation-to-reality gap. It is unrealistic to assume that a single platform is used sequentially in WiSAR given the vast size of the search areas; the reality is more likely to be that multiple different search teams are simultaneously using different UAVs based on personal preferences. One promising avenue to solve this problem is multi-agent RL [309]. This would open the doors to searching an area with any number of drones, with different dynamical properties, that a policy can leverage for optimal performance. As with PDMs, large language models could also be used here to orchestrate the platforms, and to react to real-time developments in the search.

Looking beyond WiSAR, search planning has applications in any field where the problem statement is *finding something as fast as possible given a prior distribution*. One of these fields is wildfire detection [337], where high-risk areas can be identified a priori through historical data, and satellite imagery. Search planning, not coverage planning, is required here because of the large areas at hand, and time urgency. A

single area can be searched over multiple days by a single fixed-wing UAV, but fires only take minutes to start, and hours to spread. Furthermore, the PDM of wildfire locations might also change on an hourly basis making it imperative to search the high probability areas in a timely manner.

9.2.3 End-to-End System Integration

Building on the PDM generation (Ch. 4), and RL search planning (Ch. 6, Ch. 7, and Ch. 8) modules that have been developed, the next phase should focus on combining these components into a cohesive end-to-end search pipeline. Each element has been validated independently — demonstrating reliable PDM generation from behavioural and GIS data, followed by search planning under simulated conditions. Reflecting on these successes, the future work should focus on facilitating data exchange among modules.

Another data layer that should be considered are markers such as a LP leaving behind a rucksack, or footprints. These inform and update the current estimation of the location of the LP alongside the PLS initial reporting. This data should then be fed back into the PDM generation, and search planning models, to update the current mission. Large language tools can be used to reason about the significance of a marker, with a human in the loop providing feedback on how to proceed.

Finally, while simulation provided valuable insights, live field trials will be indispensable for stress-testing the end-to-end workflow. Lessons learned from controlled exercises — ranging from operator interface usability to communication failure modes — will guide refinements to both software and operational protocols. In particular, the human-autonomy collaboration strategies should be carefully evaluated, leveraging retrospective analysis of decision logs to improve trust and transparency. These iterative evaluations will not only validate the integrated system’s robustness but also inform guidelines for deployment in real-world search-and-rescue missions as well as build trust in such an automated system.

References

- [1] D. Heywood and L. Tufnell, ‘Snow avalanche hazards in the Glen Nevis and Glen Coe areas of Scotland,’ *Disasters*, vol. 9, no. 1, pp. 51–57, 1985, ISSN: 1467-7717. DOI: 10.1111/j.1467-7717.1985.tb00910.x.
- [2] S. J. Page, T. A. Bentley and L. Walker, ‘Scoping the nature and extent of adventure tourism operations in Scotland: How safe are they?’ *Tourism Management*, vol. 26, no. 3, pp. 381–397, Jun. 2005, ISSN: 0261-5177. DOI: 10.1016/j.tourman.2003.11.018.
- [3] B. Sharp, University of Strathclyde Faculty of Education Scottish School of Sports Studies, Leverhulme Trust, Mountaineering Council of Scotland and Scottish Mountain Rescue, *Strategies for Improving Mountain Safety*. [Glasgow]: University of Strathclyde. Faculty of Education. Scottish School of Sports Studies, 2001, ch. v, 213 pages : illustrations ; 30 cm.
- [4] R. G. W. Ward, ‘Avalanche Hazard in the Cairngorm Mountains, Scotland,’ *Journal of Glaciology*, vol. 26, no. 94, pp. 31–41, Jan. 1980, ISSN: 0022-1430, 1727-5652. DOI: 10.3189/S0022143000201020.
- [5] J. Allen and C. Bonington, *Cairngorm John: A Life in Mountain Rescue*. La Vergne: Sandstone Press Ltd, 2019, ISBN: 978-1-912240-70-8.
- [6] C. Williams, ‘The worst mountain disaster in British history,’ *BBC News*, Nov. 2021.
- [7] J. Steinemann and A. Morgan, ‘Statistics Report,’ Scottish Mountain Rescue, Scotland, Tech. Rep. 2024, Apr. 2025.
- [8] S. Carrell, ‘Flying to the rescue: Scottish mountain teams are turning to drones,’ *The Guardian*, Jan. 2022, ISSN: 0261-3077.
- [9] L. Skeleton and C. Smith, ‘Remotely Piloted Aircraft System (RPAS) Evaluation Report,’ Scottish Police Authority, SPA Policing Performance Committee, Tech. Rep. 1, Nov. 2020.
- [10] K. Solaiman, T. Sun, A. Nesen, B. Bhargava and M. Stonebraker, ‘Applying Machine Learning and Data Fusion to the “Missing Person” Problem,’ *Computer*, vol. 55, no. 6, pp. 40–55, Jun. 2022, ISSN: 1558-0814. DOI: 10.1109/MC.2022.3145507.

- [11] P. Rudol and P. Doherty, ‘Human body detection and geolocalization for UAV search and rescue missions using color and thermal imagery,’ *IEEE Aerospace Conference Proceedings*, 2008, ISSN: 1095323X. DOI: 10.1109/AERO.2008.4526559.
- [12] A. P. Mäyrä, A. Agapiou, L. Hildebrand, K. M. Ojala, K. Mikedi and M. Statheropoulou, ‘Optical sensors for urban search and rescue operations,’ *Electro-Optical and Infrared Systems: Technology and Applications VIII*, vol. 8185, no. September 2011, 81850F, 2011, ISSN: 0277786X. DOI: 10.1117/12.898146.
- [13] T. Niedzielski, Mirosława Jurecka, M. Jurecka *et al.*, ‘A real-time field experiment on search and rescue operations assisted by unmanned aerial vehicles,’ *Journal of Field Robotics*, vol. 35, no. 6, pp. 906–920, Mar. 2018. DOI: 10.1002/rob.21784.
- [14] A. Symington, S. Waharte, S. Julier and N. Trigoni, ‘Probabilistic target detection by camera-equipped UAVs,’ in *2010 IEEE International Conference on Robotics and Automation*, Anchorage, AK: IEEE, May 2010, pp. 4076–4081, ISBN: 978-1-4244-5038-1. DOI: 10.1109/ROBOT.2010.5509355.
- [15] P. Doherty and P. Rudol, ‘A UAV Search and Rescue Scenario with Human Body Detection and Geolocalization,’ in *AI 2007: Advances in Artificial Intelligence*, M. A. Orgun and J. Thornton, Eds., vol. 4830, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–13, ISBN: 978-3-540-76926-2. DOI: 10.1007/978-3-540-76928-6_1.
- [16] R. Koester, D. C. Cooper, J. R. Frost and R. Q. Robe, ‘Sweep Width Estimation for Ground Search and Rescue,’ *undefined*, 2004.
- [17] J.-H. Ewers, D. Anderson and D. Thomson, ‘Optimal path planning using psychological profiling in drone-assisted missing person search,’ *Advanced Control for Applications*, e167, Sep. 2023, ISSN: 2578-0727, 2578-0727. DOI: 10.1002/adc2.167.
- [18] J. Wu, L. Cheng, S. Chu and Y. Song, ‘An autonomous coverage path planning algorithm for maritime search and rescue of persons-in-water based on deep reinforcement learning,’ *Ocean Engineering*, vol. 291, p. 116 403, Jan. 2024, ISSN: 0029-8018. DOI: 10.1016/j.oceaneng.2023.116403.
- [19] B. Ai, M. Jia, H. Xu *et al.*, ‘Coverage path planning for maritime search and rescue using reinforcement learning,’ *Ocean Engineering*, vol. 241, p. 110 098, Dec. 2021, ISSN: 0029-8018. DOI: 10.1016/j.oceaneng.2021.110098.
- [20] M. Talha, A. Hussein and M. Hossny, ‘Autonomous UAV Navigation in Wilderness Search-and-Rescue Operations Using Deep Reinforcement Learning,’ in *AI 2022: Advances in Artificial Intelligence*, H. Aziz, D. Corrêa and T. French, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International

- Publishing, 2022, pp. 733–746, ISBN: 978-3-031-22695-3. DOI: 10.1007/978-3-031-22695-3_51.
- [21] J. G. C. Zuluaga, J. P. Leidig, C. Trefftz and G. Wolffe, ‘Deep Reinforcement Learning for Autonomous Search and Rescue,’ in *NAECON 2018 - IEEE National Aerospace and Electronics Conference*, Jul. 2018, pp. 521–524. DOI: 10.1109/NAECON.2018.8556642.
 - [22] A. Subramanian, R. Alimo, T. Bewley and P. Gill, ‘A Probabilistic Path Planning Framework for Optimizing Feasible Trajectories of Autonomous Search Vehicles Leveraging the Projected-Search Reduced Hessian Method,’ in *AIAA Scitech 2020 Forum*, AIAA, Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2020, ISBN: 978-1-62410-595-1. DOI: 10.2514/6.2020-0987.
 - [23] A. Macwan, G. Nejat and B. Benhabib, ‘Target-Motion Prediction for Robotic Search and Rescue in Wilderness Environments,’ *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 5, pp. 1287–1298, Oct. 2011, ISSN: 1941-0492. DOI: 10.1109/TSMCB.2011.2132716.
 - [24] R. J. Koester, *Lost Person Behavior: A Search and Rescue Guide on Where to Look for Land, Air, and Water*. Charlottesville, VA: dbS Productions, 2008, ISBN: 978-1-879471-39-9.
 - [25] C. D. Heth and E. H. Cornell, ‘Characteristics of Travel by Persons Lost in Albertan Wilderness Areas,’ *Journal of Environmental Psychology*, vol. 18, no. 3, pp. 223–235, Sep. 1998, ISSN: 0272-4944. DOI: 10.1006/jevp.1998.0093.
 - [26] D. Reilly, C. Wren, S. Giles, L. Cunningham and P. Hargreaves, ‘CASPER: Computer Assisted Search Prioritisation and Environmental Response Application,’ in *2013 Sixth International Conference on Developments in eSystems Engineering*, IEEE, Dec. 2013, pp. 225–230. DOI: 10.1109/DeSE.2013.48.
 - [27] P. J. Doherty, Q. Guo, J. Doke and D. Ferguson, ‘An analysis of probability of area techniques for missing persons in Yosemite National Park,’ *Applied Geography*, vol. 47, pp. 99–110, Feb. 2014, ISSN: 0143-6228. DOI: 10.1016/j.apgeog.2013.11.001.
 - [28] E. Sava, C. Twardy, R. Koester and M. Sonwalkar, ‘Evaluating Lost Person Behavior Models,’ *Transactions in GIS*, vol. 20, no. 1, pp. 38–53, 2016, ISSN: 1467-9671. DOI: 10.1111/tgis.12143.
 - [29] J. Doke, ‘Analysis of Search Incidents and Lost Person Behavior in Yosemite National Park,’ Thesis, University of Kansas, Department of Geography, Dec. 2012. HDL: 1808/10846.
 - [30] D. Perkins, P. Roberts, G. Feeney and P. Mrt, ‘The U.K. Missing Person Behaviour Study,’ *CSR*, vol. 28, pp. 1–66, 2011, ISSN: 2046-9527.
 - [31] Grampian Police, ‘Missing persons: Understanding, planning, responding,’ Grampian Police, 4 Burnside St, Rothes, Aberlour AB38 7AA, Tech. Rep. 1, 2007.

- [32] S. M. LaValle *et al.*, ‘Rapidly-exploring random trees: A new tool for path planning,’ *The annual research report*, 1998.
- [33] B. Paden, M. Čáp, S. Z. Yong, D. Yershov and E. Frazzoli, ‘A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles,’ *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, Mar. 2016, ISSN: 2379-8904. DOI: 10.1109/TIV.2016.2578706.
- [34] H. Shim, T. Koo, F. Hoffmann and S. Sastry, ‘A comprehensive study of control design for an autonomous helicopter,’ in *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, vol. 4, Dec. 1998, 3653–3658 vol.4. DOI: 10.1109/CDC.1998.761749.
- [35] J. Liu, J. Zhao, Z. Liu *et al.*, ‘Development and Application of the Coverage Path Planning Based on a Biomimetic Robotic Fish,’ *Journal of Field Robotics*, rob.22531, Feb. 2025, ISSN: 1556-4959, 1556-4967. DOI: 10.1002/rob.22531.
- [36] J. Jin and L. Tang, ‘Coverage path planning on three-dimensional terrain for arable farming,’ *Journal of Field Robotics*, vol. 28, no. 3, pp. 424–440, 2011, ISSN: 1556-4967. DOI: 10.1002/rob.20388.
- [37] E. Galceran and M. Carreras, ‘A survey on coverage path planning for robotics,’ *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013, ISSN: 09218890. DOI: 10.1016/j.robot.2013.09.004.
- [38] P. Cheng, J. Keller and V. Kumar, ‘Time-optimal UAV trajectory planning for 3D urban structure coverage,’ in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 2750–2757. DOI: 10.1109/IRoS.2008.4650988.
- [39] B. Frakes, Thomas Flowe and Kirk R. Sherrill, ‘National Park Service Travel Time Cost Surface Model (TTCSM),’ National Park Service, Fort Collins, CO 80525, Tech. Rep. 2015/933, Mar. 2015, pp. 1–28.
- [40] S. Drexel, S. Zimmermann-Janschitz and R. J. Koester, ‘Network analysis for search areas in WiSAR operations,’ *International Journal of Emergency Services*, vol. 7, no. 3, pp. 192–202, Feb. 2018, ISSN: 2047-0894. DOI: 10.1108/IJES-02-2017-0005.
- [41] L. Heintzman, A. Hashimoto, N. Abaid and R. K. Williams, ‘Anticipatory Planning and Dynamic Lost Person Models for Human-Robot Search and Rescue,’ in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 8252–8258. DOI: 10.1109/ICRA48506.2021.9562070.
- [42] A. Hashimoto, L. Heintzman, R. Koester and N. Abaid, ‘An agent-based model reveals lost person behavior based on data from wilderness search and rescue,’ *Scientific Reports*, vol. 12, no. 1, p. 5873, Dec. 2022, ISSN: 2045-2322. DOI: 10.1038/s41598-022-09502-4.

- [43] S. Manjanna, H. Van Hoof and G. Dudek, ‘Policy Search on Aggregated State Space for Active Sampling,’ in *Proceedings of the 2018 International Symposium on Experimental Robotics*, J. Xiao, T. Kröger and O. Khatib, Eds., vol. 11, Cham: Springer International Publishing, 2020, pp. 211–221, ISBN: 978-3-030-33950-0. DOI: 10.1007/978-3-030-33950-0_19.
- [44] J.-H. Ewers, D. Anderson and D. Thomson, ‘Predictive Probability Density Mapping for Search and Rescue Using an Agent-Based Approach With Sparse Data,’ *IEEE Access*, vol. 13, pp. 60 802–60 813, 2025, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2025.3557693.
- [45] J.-H. Ewers, D. Anderson and D. Thomson, ‘Deep reinforcement learning for time-critical wilderness search and rescue using drones,’ *Frontiers in Robotics and AI*, vol. 11, p. 1 527 095, Feb. 2025, ISSN: 2296-9144. DOI: 10.3389/frobt.2024.1527095.
- [46] J.-H. Ewers, D. Anderson and D. Thomson, ‘GIS Data Driven Probability Map Generation for Search and Rescue Using Agents,’ in *IFAC World Congress 2023*, 2023, pp. 1466–1471. DOI: 10.1016/j.ifacol.2023.10.1834.
- [47] J.-H. Ewers, S. Swinton, D. Anderson, E. McGookin and D. Thomson, ‘Enhancing Reinforcement Learning in Sensor Fusion: A Comparative Analysis of Cubature and Sampling-based Integration Methods for Rover Search Planning,’ in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Abu Dhabi, United Arab Emirates: IEEE, Oct. 2024, pp. 7825–7830, ISBN: 979-8-3503-7770-5. DOI: 10.1109/IROS58592.2024.10801978.
- [48] J.-H. Ewers, J. Gibbs and D. Anderson, ‘Stone Soup Multi-Target Tracking Feature Extraction for Autonomous Search and Track in Deep Reinforcement Learning Environment,’ in *2025 28th International Conference on Information Fusion (FUSION)*, Rio de Janeiro, Brazil: IEEE, Jul. 2025, pp. 1–8, ISBN: 978-1-0370-5623-9. DOI: 10.23919/FUSION65864.2025.11123969.
- [49] J.-H. Ewers, D. Cormack, J. Gibbs and D. Anderson, *Multi-Target Radar Search and Track Using Sequence-Capable Deep Reinforcement Learning*, Feb. 2025. DOI: 10.48550/arXiv.2502.13584. arXiv: 2502.13584 [cs].
- [50] J.-H. Ewers, D. Anderson and D. Thomson, *Recurrent Auto-Encoders for Enhanced Deep Reinforcement Learning in Wilderness Search and Rescue Planning*, 2025. DOI: 10.48550/ARXIV.2502.19356.
- [51] S. Swinton, J.-H. Ewers, E. McGookin, D. Anderson and D. Thomson, ‘Autonomous mission planning for planetary surface exploration using a team of micro rovers,’ *Frontiers in Robotics and AI*, vol. 12, 2025, ISSN: 2296-9144. DOI: 10.3389/frobt.2025.1565173.
- [52] R. J. Koester, *International Search & Rescue Incident Database (ISRID)*, 2000.

- [53] M. Jurecka and T. Niedzielski, ‘A procedure for delineating a search region in the UAV-based SAR activities,’ *Geomatics, Natural Hazards and Risk*, vol. 8, no. 1, pp. 53–72, Jan. 2017, ISSN: 1947-5705. DOI: 10.1080/19475705.2016.1238853.
- [54] D. Lambert, *The Field Guide to Geology*, Updated ed. New York: Facts on File, 1998, ISBN: 978-0-8160-3823-7.
- [55] USDA-NRCS, *Watershed Boundary Datasets*, Department of Agriculture-Natural Resources Conservation Service, US Geological Survey, and US Environmental Protection Agency, 2022.
- [56] O. Conrad, B. Bechtel, M. Bock *et al.*, ‘System for Automated Geoscientific Analyses (SAGA) v. 2.1.4,’ *Geoscientific Model Development*, vol. 8, no. 7, pp. 1991–2007, Jul. 2015, ISSN: 1991-9603. DOI: 10.5194/gmd-8-1991-2015.
- [57] R. Fisher, S. Hobgen, N. Riwu Kaho and I. Mandaya, *A Practical Manual for Natural Resource Management, Disaster Risk and Development Planning Using Free Geospatial Data and Software*. Jun. 2017.
- [58] QGIS Development Team, *QGIS Geographic Information System*. QGIS Association, 2022.
- [59] R. J. Mattson, ‘Establishing Search Priorities,’ in *Search Theory and Applications*, ser. NATO Conference Series, K. B. Haley and L. D. Stone, Eds., Boston, MA: Springer US, 1980, pp. 93–97, ISBN: 978-1-4615-9188-7. DOI: 10.1007/978-1-4615-9188-7_7.
- [60] J. Surowiecki, *The Wisdom of Crowds: Why the Many Are Smarter than the Few*, Repr. London: Abacus, 2014, ISBN: 978-0-349-11605-1.
- [61] E. Sava, C. Twardy, R. Koester and M. Sonwalkar, *MapScore*, Apr. 2012. DOI: 10.5281/zenodo.5579128.
- [62] M. Ciolli, L. Mengon, A. Vitti and P. Zatelli, ‘A GIS-based decision support system for the management of SAR operations in mountain areas,’ *Geomatics workbooks*, vol. 6, pp. 1–17, 2006.
- [63] D. Levine, B. Luders and J. How, ‘Information-Rich Path Planning with General Constraints Using Rapidly-Exploring Random Trees,’ in *AIAA Infotech@Aerospace 2010*, Atlanta, Georgia: American Institute of Aeronautics and Astronautics, Apr. 2010, ISBN: 978-1-60086-963-1. DOI: 10.2514/6.2010-3360.
- [64] S. Karaman and E. Frazzoli, ‘Sampling-based algorithms for optimal motion planning,’ *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 2011, ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364911406761.
- [65] D. H. Douglas, ‘Least-cost Path in GIS Using an Accumulated Cost Surface and Slope-lines,’ *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 31, no. 3, pp. 37–51, Oct. 1994, ISSN: 0317-7173, 1911-9925. DOI: 10.3138/D327-0323-2JUT-016M.

- [66] J. Kinsella-Shaw, B. Shaw and M. Turvey, 'Perceiving 'Walk-on-able' Slopes,' *Ecological Psychology*, vol. 4, no. 4, pp. 223–239, Dec. 1992, ISSN: 1040-7413. DOI: 10.1207/s15326969eco0404_2.
- [67] C. G. Homer, Joyce A. Fry and Christopher A. Barnes, 'The National Land Cover Database,' U.S. Geological Survey, Reston, VA, USGS Numbered Series 2012-3020, 2012.
- [68] W. Tobler, 'Three presentations on geographical analysis and modeling: Non-isotropic geographic modeling; speculations on the geometry of geography; and global spatial analysis,' National Center For Geographics Information and Analysis, University of California, Tech. Rep. 93-1, Feb. 1993, pp. 1–26.
- [69] W. Naismith, 'Excursions: Cruach Ardran, Stobinian, and Ben More,' *Scottish Mountaineering club journal*, vol. 2, no. 3, p. 136, 1892.
- [70] E. Langmuir, *Mountaincraft and Leadership*. Mountain Training Boards of England and Scotland, 2013, ISBN: 978-0-9568869-0-3.
- [71] P. Scarf, 'A mathematical excursion in the isochronic hills,' *Mathematics Today*, pp. 163–167, Oct. 2011.
- [72] M. J. Campbell, P. E. Dennison, B. W. Butler and W. G. Page, 'Using crowd-sourced fitness tracker data to model the relationship between slope and travel rates,' *Applied Geography*, vol. 106, pp. 93–107, May 2019, ISSN: 01436228. DOI: 10.1016/j.apgeog.2019.03.008.
- [73] A. Kay, 'Pace and critical gradient for hill runners: An analysis of race records,' Jan. 2012, ISSN: 1559-0410. DOI: 10.1515/1559-0410.1456'] .
- [74] M. Saleh, Y. Esa and A. Mohamed, 'Applications of Complex Network Analysis in Electric Power Systems,' *Energies*, vol. 11, no. 6, p. 1381, May 2018, ISSN: 1996-1073. DOI: 10.3390/en11061381.
- [75] I. Habibi, E. S. Emamian and A. Abdi, 'Quantitative analysis of intracellular communication and signaling errors in signaling networks,' *BMC Systems Biology*, vol. 8, no. 1, p. 89, Dec. 2014, ISSN: 1752-0509. DOI: 10.1186/s12918-014-0089-z.
- [76] D. Sarkar, P. Konwar, A. De and S. Goswami, 'A graph theory application for fast and efficient search of optimal radialized distribution network topology,' *Journal of King Saud University - Engineering Sciences*, vol. 32, no. 4, pp. 255–264, May 2020, ISSN: 10183639. DOI: 10.1016/j.jksues.2019.02.003.
- [77] L. Šerić, T. Pinjušić, K. Topić and T. Blažević, 'Lost Person Search Area Prediction Based on Regression and Transfer Learning Models,' *ISPRS International Journal of Geo-Information*, vol. 10, no. 2, p. 80, Feb. 2021, ISSN: 2220-9964. DOI: 10.3390/ijgi10020080.

- [78] E. S. Olivas, Ed., *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. Hershey, PA: Information Science Reference, 2010, ISBN: 978-1-60566-767-6.
- [79] N. Keijsers, N. Stolwijk, G. Renzenbrink and J. Duysens, ‘Prediction of walking speed using single stance force or pressure measurements in healthy subjects,’ *Gait & Posture*, vol. 43, pp. 93–95, Jan. 2016, ISSN: 09666362. DOI: 10.1016/j.gaitpost.2015.09.027.
- [80] A. Feng and A. S. Gordon, ‘Latent Terrain Representations for Trajectory Prediction,’ in *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Computing with Multifaceted Movement Data*, ser. MOVE’19, New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 1–4, ISBN: 978-1-4503-6951-0. DOI: 10.1145/3356392.3365218.
- [81] L. Wang, X. Geng, X. Ma, F. Liu and Q. Yang, ‘Cross-City Transfer Learning for Deep Spatio-Temporal Prediction,’ *arXiv:1802.00386 [cs]*, May 2018. arXiv: 1802.00386 [cs].
- [82] European Union, Copernicus Land Monitoring Service, *Corine Land Cover Change (CHA) 2012 - 2018*, Jun. 2019.
- [83] European Union, Copernicus Land Monitoring Service, *European Digital Elevation Model (EU-DEM)*, Apr. 2016.
- [84] Y. Li, M. Chen, Z. Dou, X. Zheng, Y. Cheng and A. Mebarki, ‘A review of cellular automata models for crowd evacuation,’ *Physica A: Statistical Mechanics and its Applications*, vol. 526, p. 120752, Jan. 2019, ISSN: 0378-4371. DOI: 10.1016/j.physa.2019.03.117.
- [85] M. Wojtyra, ‘Multibody Simulation Model of Human Walking,’ *Mechanics Based Design of Structures and Machines*, vol. 31, no. 3, pp. 357–379, Jan. 2003, ISSN: 1539-7734, 1539-7742. DOI: 10.1081/SME-120022855.
- [86] R. W. Bohannon, ‘Comfortable and maximum walking speed of adults aged 20—79 years: Reference values and determinants,’ *Age and Ageing*, vol. 26, no. 1, pp. 15–19, 1997, ISSN: 0002-0729, 1468-2834. DOI: 10.1093/ageing/26.1.15.
- [87] A. Macwan, J. Vilela, G. Nejat and B. Benhabib, ‘A Multirobot Path-Planning Strategy for Autonomous Wilderness Search and Rescue,’ *IEEE Transactions on Cybernetics*, vol. 45, no. 9, pp. 1784–1797, Sep. 2015, ISSN: 2168-2275. DOI: 10.1109/TCYB.2014.2360368.
- [88] A. Hashimoto and N. Abaid, ‘An Agent-Based Model of Lost Person Dynamics for Enabling Wilderness Search and Rescue,’ in *ASME 2019 Dynamic Systems and Control Conference*, American Society of Mechanical Engineers Digital Collection, Nov. 2019, pp. 1–8. DOI: 10.1115/DSCC2019-9222.

- [89] J. Canny, ‘A Computational Approach to Edge Detection,’ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986, ISSN: 0162-8828. DOI: 10.1109/TPAMI.1986.4767851.
- [90] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge: Cambridge University Press, 2003, ISBN: 9780521642989.
- [91] B. G. Cangan, L. Heintzman, A. Hashimoto, N. Abaid and R. K. Williams, ‘Anticipatory Human-Robot Path Planning for Search and Rescue,’ *arXiv:2009.03976 [cs]*, Sep. 2020. arXiv: 2009.03976 [cs].
- [92] B. G. Cangan, ‘Risk-Aware Human-In-The-Loop Multi-Robot Path Planning for Lost Person Search and Rescue,’ Thesis, Virginia Tech, Jul. 2019.
- [93] S. S. Skiena, *The Algorithm Design Manual*. London: Springer London, 2008, ISBN: 978-1-84800-070-4. DOI: 10.1007/978-1-84800-070-4.
- [94] S. Kullback and R. A. Leibler, ‘On Information and Sufficiency,’ *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, Mar. 1951, ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177729694.
- [95] N. L. Johnson, S. Kotz and N. Balakrishnan, *Continuous Univariate Distributions* (Wiley Series in Probability and Mathematical Statistics), 2nd ed. New York: Wiley, 1994, ISBN: 978-0-471-58494-0.
- [96] L. Lin and M. A. Goodrich, ‘A Bayesian approach to modeling lost person behaviors based on terrain features in Wilderness Search and Rescue,’ *Computational and Mathematical Organization Theory*, vol. 16, no. 3, pp. 300–323, Jul. 2010, ISSN: 1381-298X, 1572-9346. DOI: 10.1007/s10588-010-9066-2.
- [97] C. P. D. Birch, S. P. Oom and J. A. Beecham, ‘Rectangular and hexagonal grids used for observation, experiment and simulation in ecology,’ *Ecological Modelling*, vol. 206, no. 3, pp. 347–359, Aug. 2007, ISSN: 0304-3800. DOI: 10.1016/j.ecolmodel.2007.03.041.
- [98] F. Bourgault, T. Furukawa and H. F. Durrant-Whyte, ‘Optimal Search for a Lost Target in a Bayesian World,’ in *Field and Service Robotics*, S. Yuta, H. Asama, E. Prassler, T. Tsubouchi and S. Thrun, Eds., vol. 24, Berlin/Heidelberg: Springer-Verlag, 2006, pp. 209–222, ISBN: 978-3-540-32801-8.
- [99] J. B. Tenenbaum, *Bayesian Modeling of Human Concept Learning* (Advances in Neural Information Processing Systems 11). Cambridge: MIT press, 1999, ISBN: 978-0-262-11245-1.
- [100] A. Sanborn and T. Griffiths, ‘Markov Chain Monte Carlo with People,’ in *Advances in Neural Information Processing Systems*, vol. 20, Cambridge: MIT Press, Jan. 2007, pp. 1265–1272.
- [101] V. E. Johnson, ‘A Bayesian χ^2 test for goodness-of-fit,’ *The Annals of Statistics*, vol. 32, no. 6, Dec. 2004, ISSN: 0090-5364. DOI: 10.1214/009053604000000616.

- [102] D. K. Rossmo, *Geographic Profiling*. Boca Raton, Fla.: CRC Press, 2000, ISBN: 978-0-8493-8129-4.
- [103] M. Quigley, M. Goodrich, S. Griffiths, A. Eldredge and R. Beard, ‘Target Acquisition, Localization, and Surveillance Using a Fixed-Wing Mini-UAV and Gimbaled Camera,’ in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain: IEEE, 2005, pp. 2600–2605, ISBN: 978-0-7803-8914-4. DOI: 10.1109/ROBOT.2005.1570505.
- [104] R. J. Koester, K. B. Chiacchia, C. R. Twardy, D. C. Cooper, J. R. Frost and R. Q. Robe, ‘Use of the Visual Range of Detection to Estimate Effective Sweep Width for Land Search and Rescue Based On 10 Detection Experiments in North America,’ *Wilderness & Environmental Medicine*, vol. 25, no. 2, pp. 132–142, Jun. 2014, ISSN: 1080-6032, 1545-1534. DOI: 10.1016/j.wem.2013.09.016.
- [105] R. J. Koester, ‘Enhancements to Statistical Probability of Area Models based upon updated ISRID data collection for Autistic Spectrum Disorders and Typically Developing Children,’ *The Journal of Search and Rescue*, vol. 4, no. 1, Apr. 2020, ISSN: 22305734, 22305734. DOI: 10.61618/BTZC1005.
- [106] G. S. Kashyap, D. Mahajan, O. C. Phukan, A. Kumar, A. E. I. Brownlee and J. Gao, *From Simulations to Reality: Enhancing Multi-Robot Exploration for Urban Search and Rescue*, Nov. 2023. DOI: 10.48550/arXiv.2311.16958. arXiv: 2311.16958 [cs].
- [107] A. Stentz, ‘Optimal and efficient path planning for partially-known environments,’ in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, May 1994, 3310–3317 vol.4. DOI: 10.1109/ROBOT.1994.351061.
- [108] L. Tang, H. Wang, P. Li and Y. Wang, *Real-time Trajectory Generation for Quadrotors using B-spline based Non-uniform Kinodynamic Search*, Nov. 2019. arXiv: 1904.12348 [cs].
- [109] X. Zhou, P. Wu, H. Zhang, W. Guo and Y. Liu, ‘Learn to Navigate: Cooperative Path Planning for Unmanned Surface Vehicles Using Deep Reinforcement Learning,’ *IEEE Access*, vol. 7, pp. 165 262–165 278, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2953326.
- [110] P. B. Sujit and R. Beard, ‘Multiple UAV path planning using anytime algorithms,’ in *2009 American Control Conference*, Jun. 2009, pp. 2978–2983. DOI: 10.1109/ACC.2009.5160222.
- [111] L. Kästner, T. Buiyan, X. Zhao, Z. Shen, C. Marx and J. Lambrecht, *Connecting Deep-Reinforcement-Learning-based Obstacle Avoidance with Conventional Global Planners using Waypoint Generators*, Apr. 2021. DOI: 10.48550/arXiv.2104.03663. arXiv: 2104.03663 [cs].

- [112] L. Kästner, X. Zhao, Z. Shen and J. Lambrecht, *Obstacle-aware Waypoint Generation for Long-range Guidance of Deep-Reinforcement-Learning-based Navigation Approaches*, Sep. 2021. DOI: 10.48550/arXiv.2109.11639. arXiv: 2109.11639 [cs].
- [113] B. Morrell, R. Thakker, G. Merewether *et al.*, ‘Comparison of Trajectory Optimization Algorithms for High-Speed Quadrotor Flight Near Obstacles,’ *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4399–4406, 2018, ISSN: 23773766. DOI: 10.1109/LRA.2018.2868866.
- [114] A. Sonmez, E. Kocyigit and E. Kugu, ‘Optimal path planning for UAVs using Genetic Algorithm,’ in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, Jun. 2015, pp. 50–55. DOI: 10.1109/ICUAS.2015.7152274.
- [115] B. Geiger, J. Horn, A. DeLullo, A. Niessner and L. Long, ‘Optimal Path Planning of UAVs Using Direct Collocation with Nonlinear Programming,’ in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, ser. Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics, Aug. 2006. DOI: 10.2514/6.2006-6199.
- [116] P. E. Hart, N. J. Nilsson and B. Raphael, ‘A Formal Basis for the Heuristic Determination of Minimum Cost Paths,’ *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, Jul. 1968, ISSN: 2168-2887. DOI: 10.1109/TSSC.1968.300136.
- [117] I. Nikolos, K. Valavanis, N. Tsourveloudis and A. Kostaras, ‘Evolutionary algorithm based offline/online path planner for UAV navigation,’ *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 33, no. 6, pp. 898–912, Dec. 2003, ISSN: 1941-0492. DOI: 10.1109/TSMCB.2002.804370.
- [118] J. Yu and S. M. LaValle, ‘Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics,’ *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, Oct. 2016, ISSN: 1941-0468. DOI: 10.1109/TR0.2016.2593448.
- [119] Yujin and G. Xiaoxue, ‘Optimal Route Planning of Parking Lot Based on Dijkstra Algorithm,’ in *2017 International Conference on Robots & Intelligent System (ICRIS)*, Oct. 2017, pp. 221–224. DOI: 10.1109/ICRIS.2017.62.
- [120] K. Wei, Y. Gao, W. Zhang and S. Lin, ‘A Modified Dijkstra’s Algorithm for Solving the Problem of Finding the Maximum Load Path,’ in *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*, Mar. 2019, pp. 10–13. DOI: 10.1109/INF0CT.2019.8711024.
- [121] E. W. Dijkstra, ‘A note on two problems in connexion with graphs,’ *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959, ISSN: 0029-599X, 0945-3245. DOI: 10.1007/BF01386390.

- [122] A. Alyasin, E. I. Abbas and S. D. Hasan, ‘An Efficient Optimal Path Finding for Mobile Robot Based on Dijkstra Method,’ in *2019 4th Scientific International Conference Najaf (SICN)*, Apr. 2019, pp. 11–14. DOI: 10.1109/SICN47020.2019.9019345.
- [123] A. Patel, M. A. V. Saucedo, C. Kanellakis and G. Nikolakopoulos, ‘STAGE: Scalable and Traversability-Aware Graph based Exploration Planner for Dynamically Varying Environments,’ in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 5949–5955. DOI: 10.1109/ICRA57147.2024.10610939.
- [124] N. Aksoy, F. Cakil and I. G. Tekdemir, ‘Performance Analysis of Deterministic and Probabilistic Path Planning Algorithms in Complex Environments,’ in *2024 Innovations in Intelligent Systems and Applications Conference (ASYU)*, Oct. 2024, pp. 1–6. DOI: 10.1109/ASYU62119.2024.10757028.
- [125] A. Stentz, ‘The focussed D* algorithm for real-time replanning,’ in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI’95, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Aug. 1995, pp. 1652–1659, ISBN: 978-1-55860-363-9.
- [126] S. Koenig and M. Likhachev, ‘Fast replanning for navigation in unknown terrain,’ *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, Jun. 2005, ISSN: 1941-0468. DOI: 10.1109/TR0.2004.838026.
- [127] K. Daniel, A. Nash, S. Koenig and A. Felner, ‘Theta*: Any-angle path planning on grids,’ *J. Artif. Int. Res.*, vol. 39, no. 1, pp. 533–579, Sep. 2010, ISSN: 1076-9757.
- [128] C. Cai, J. Chen, Q. Yan, F. Liu and R. Zhou, ‘A prior information-based coverage path planner for underwater search and rescue using autonomous underwater vehicle (AUV) with side-scan sonar,’ *IET Radar, Sonar & Navigation*, vol. 16, no. 7, pp. 1225–1239, 2022, ISSN: 1751-8792. DOI: 10.1049/rsn2.12256.
- [129] Y. Du, ‘Multi-UAV Search and Rescue with Enhanced A* Algorithm Path Planning in 3D Environment,’ *International Journal of Aerospace Engineering*, vol. 2023, Y. Wang, Ed., pp. 1–18, Feb. 2023, ISSN: 1687-5974, 1687-5966. DOI: 10.1155/2023/8614117.
- [130] G. Sotirchos and Z. Ajanovic, *Search-based versus Sampling-based Robot Motion Planning: A Comparative Study*, Jun. 2024. DOI: 10.48550/arXiv.2406.09623. arXiv: 2406.09623 [cs].
- [131] K. Bergman, O. Ljungqvist and D. Axehill, ‘Improved Path Planning by Tightly Combining Lattice-Based Path Planning and Optimal Control,’ *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 1, pp. 57–66, Mar. 2021, ISSN: 2379-8904. DOI: 10.1109/TIV.2020.2991951.

- [132] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Pr, 1984, ISBN: 978-0-691-07951-6.
- [133] J. Kuffner and S. LaValle, ‘RRT-connect: An efficient approach to single-query path planning,’ in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, Apr. 2000, 995–1001 vol.2. DOI: 10.1109/ROBOT.2000.844730.
- [134] M. Xanthidis, J. M. Esposito, I. Rekleitis and J. M. O’Kane, ‘Motion Planning by Sampling in Subspaces of Progressively Increasing Dimension,’ *Journal of Intelligent & Robotic Systems*, vol. 100, no. 3-4, pp. 777–789, Dec. 2020, ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-020-01217-w.
- [135] M. Brunner, B. Brüggemann and D. Schulz, ‘Hierarchical rough terrain motion planning using an optimal sampling-based method,’ in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 5539–5544. DOI: 10.1109/ICRA.2013.6631372.
- [136] L. Palmieri, S. Koenig and K. O. Arras, ‘RRT-based nonholonomic motion planning using any-angle path biasing,’ in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2775–2781. DOI: 10.1109/ICRA.2016.7487439.
- [137] B. Lindqvist, A.-A. Agha-Mohammadi and G. Nikolakopoulos, ‘Exploration-RRT: A multi-objective Path Planning and Exploration Framework for Unknown and Unstructured Environments,’ in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 3429–3435. DOI: 10.1109/IROS51168.2021.9636243.
- [138] L. Lin and M. A. Goodrich, ‘Hierarchical Heuristic Search Using a Gaussian Mixture Model for UAV Coverage Planning,’ *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2532–2544, Dec. 2014, ISSN: 2168-2267, 2168-2275. DOI: 10.1109/TCYB.2014.2309898.
- [139] P. Yao, H. Wang and H. Ji, ‘Gaussian mixture model and receding horizon control for multiple UAV search in complex environment,’ *Nonlinear Dynamics*, vol. 88, no. 2, pp. 903–919, Apr. 2017, ISSN: 1573-269X. DOI: 10.1007/s11071-016-3284-1.
- [140] Y. Zhang, B. Luo, A. Mukhopadhyay *et al.*, ‘Shrinking POMCP: A Framework for Real-Time UAV Search and Rescue,’ in *2024 International Conference on Assured Autonomy (ICAA)*, Oct. 2024, pp. 48–57. DOI: 10.1109/ICAA64256.2024.00016.

- [141] M. A. Goodrich, B. S. Morse, D. Gerhardt *et al.*, ‘Supporting wilderness search and rescue using a camera-equipped mini UAV: UAV-Enabled WiSAR,’ *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 89–110, Jan. 2008, ISSN: 15564959. DOI: 10.1002/rob.20226.
- [142] V. Lumelsky, S. Mukhopadhyay and K. Sun, ‘Dynamic path planning in sensor-based terrain acquisition,’ *IEEE Transactions on Robotics and Automation*, vol. 6, no. 4, pp. 462–472, Aug. 1990, ISSN: 2374-958X. DOI: 10.1109/70.59357.
- [143] E. M. Arkin, S. P. Fekete and J. S. Mitchell, ‘Approximation algorithms for lawn mowing and milling,’ *Computational Geometry*, vol. 17, no. 1-2, pp. 25–50, Oct. 2000, ISSN: 09257721. DOI: 10.1016/S0925-7721(00)00015-8.
- [144] L. E. Dubins, ‘On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents,’ *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957, ISSN: 0002-9327. DOI: 10.2307/2372560. JSTOR: 2372560.
- [145] H. Choset and P. Pignon, ‘Coverage Path Planning: The Boustrophedon Cellular Decomposition,’ in *Field and Service Robotics*, A. Zelinsky, Ed., London: Springer London, 1998, pp. 203–209, ISBN: 978-1-4471-1273-0. DOI: 10.1007/978-1-4471-1273-0_32.
- [146] R. Bähnemann, N. Lawrance, J. J. Chung, M. Pantic, R. Siegwart and J. Nieto, ‘Revisiting Boustrophedon Coverage Path Planning as a Generalized Traveling Salesman Problem,’ in *Field and Service Robotics*, G. Ishigami and K. Yoshida, Eds., vol. 16, Singapore: Springer Singapore, 2021, pp. 277–290, ISBN: 978-981-15-9460-1. DOI: 10.1007/978-981-15-9460-1_20.
- [147] K. Helsgaun, ‘Solving the equality generalized traveling salesman problem using the Lin–Kernighan–Helsgaun Algorithm,’ *Mathematical Programming Computation*, vol. 7, no. 3, pp. 269–287, Sep. 2015, ISSN: 1867-2949, 1867-2957. DOI: 10.1007/s12532-015-0080-8.
- [148] D.-N. Bui, T. H. Khuat, M. D. Phung, T.-H. Tran and D. L. Tran, ‘Model Predictive Control for Optimal Motion Planning of Unmanned Aerial Vehicles,’ in *2024 International Conference on Control, Robotics and Informatics (ICCRI)*, Jul. 2024, pp. 1–6. DOI: 10.1109/ICCRI64298.2024.00025.
- [149] A. Joos, C. Seiferth, L. Schmitt and W. Fichter, ‘Parameters for Nonlinear Model Predictive Control in Unmanned Aerial Vehicle Path-Planning Applications,’ *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 484–492, Feb. 2017, ISSN: 0731-5090. DOI: 10.2514/1.G000311.
- [150] M. Ramezani, H. Habibi, J. L. Sanchez-Lopez and H. Voos, ‘UAV Path Planning Employing MPC-Reinforcement Learning Method Considering Collision Avoidance,’ in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, Jun. 2023, pp. 507–514. DOI: 10.1109/ICUAS57906.2023.10156232.

- [151] D. L. Vinokursky, O. S. Mezentseva, P. V. Samoylov, K. Y. Ganshin and O. A. Baklanova, ‘Model predictive control for path planning of UAV group,’ *IOP Conference Series: Materials Science and Engineering*, vol. 1155, no. 1, p. 012 092, Jun. 2021, ISSN: 1757-899X. DOI: 10.1088/1757-899X/1155/1/012092.
- [152] H. Wei and Y. Shi, ‘Mpc-based motion planning and control enables smarter and safer autonomous marine vehicles: Perspectives and a tutorial survey,’ *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 1, pp. 8–24, Jan. 2023, ISSN: 2329-9274. DOI: 10.1109/JAS.2022.106016.
- [153] Z. Ge, J. Jiang and M. Coombes, *Multi-UAV Search and Rescue in Wilderness Using Smart Agent-Based Probability Models*, Nov. 2024. DOI: 10.48550/arXiv.2411.10148. arXiv: 2411.10148 [cs].
- [154] S. Waharte and N. Trigoni, ‘Supporting Search and Rescue Operations with UAVs,’ in *2010 International Conference on Emerging Security Technologies*, University of Oxford, Sep. 2010, pp. 142–147. DOI: 10.1109/EST.2010.31.
- [155] L. Lin and M. A. Goodrich, ‘UAV intelligent path planning for wilderness search and rescue,’ *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, vol. 0, no. 1, pp. 709–714, 2009. DOI: 10.1109/IROS.2009.5354455.
- [156] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, ‘Optimization by Simulated Annealing,’ *Science*, vol. 220, no. 4598, pp. 671–680, May 1983. DOI: 10.1126/science.220.4598.671.
- [157] V. Granville, M. Krivanek and J.-P. Rasson, ‘Simulated annealing: A proof of convergence,’ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 652–656, Jun. 1994, ISSN: 1939-3539. DOI: 10.1109/34.295910.
- [158] K. J. Worrall, ‘Guidance and search algorithms for mobile robots: Application and analysis within the context of urban search and rescue,’ Ph.D. dissertation, University of Glasgow, 2008.
- [159] E. W. McGookin, ‘Optimisation of sliding mode controllers for marine applications: A study of methods and implementation issues,’ Ph.D. dissertation, University of Glasgow, 1997.
- [160] M. Mitchell, *An Introduction to Genetic Algorithms*. The MIT Press, 1998, ISBN: 978-0-262-28001-3. DOI: 10.7551/mitpress/3927.001.0001.
- [161] D. Whitley, ‘A genetic algorithm tutorial,’ *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, Jun. 1994, ISSN: 09603174. DOI: 10.1007/BF00175354.
- [162] L. Hong, Y. Wang, Y. Du, X. Chen and Y. Zheng, ‘UAV search-and-rescue planning using an adaptive memetic algorithm,’ *Frontiers of Information Technology & Electronic Engineering*, vol. 22, no. 11, pp. 1477–1491, Dec. 2011, ISSN: 2095-9230. DOI: 10.1631/FITEE.2000632.

- [163] M. Morin, I. Abi-Zeid and C.-G. Quimper, ‘Ant colony optimization for path planning in search and rescue operations,’ *European Journal of Operational Research*, vol. 305, no. 1, pp. 53–63, Feb. 2023, ISSN: 0377-2217. DOI: 10.1016/j.ejor.2022.06.019.
- [164] Y. Wang, M.-X. Zhang and Y.-J. Zheng, ‘A Hyper-Heuristic Method for UAV Search Planning,’ in *Advances in Swarm Intelligence*, Y. Tan, H. Takagi, Y. Shi and B. Niu, Eds., vol. 10386, Cham: Springer International Publishing, 2017, pp. 454–464, ISBN: 978-3-319-61833-3.
- [165] H. Bayerlein, M. Theile, M. Caccamo and D. Gesbert, ‘Multi-UAV Path Planning for Wireless Data Harvesting With Deep Reinforcement Learning,’ *IEEE Open Journal of the Communications Society*, vol. 2, pp. 1171–1187, 2021, ISSN: 2644-125X. DOI: 10.1109/OJCOMS.2021.3081996.
- [166] Z. Cui and Y. Wang, ‘UAV Path Planning Based on Multi-Layer Reinforcement Learning Technique,’ *IEEE Access*, vol. 9, pp. 59 486–59 497, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3073704.
- [167] B. Li and Y. Wu, ‘Path Planning for UAV Ground Target Tracking via Deep Reinforcement Learning,’ *IEEE Access*, vol. 8, pp. 29 064–29 074, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2971780.
- [168] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding and F. Shu, ‘Path Planning for UAV-Mounted Mobile Edge Computing With Deep Reinforcement Learning,’ *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5723–5728, May 2020, ISSN: 1939-9359. DOI: 10.1109/TVT.2020.2982508.
- [169] M. Theile, H. Bayerlein, R. Nai, D. Gesbert and M. Caccamo, ‘UAV Coverage Path Planning under Varying Power Constraints using Deep Reinforcement Learning,’ in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 1444–1449. DOI: 10.1109/IROS45743.2020.9340934.
- [170] C. Yan and X. Xiang, ‘A Path Planning Algorithm for UAV Based on Improved Q-Learning,’ in *2018 2nd International Conference on Robotics and Automation Sciences (ICRAS)*, Jun. 2018, pp. 1–5. DOI: 10.1109/ICRAS.2018.8443226.
- [171] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun and D. Scaramuzza, ‘Champion-level drone racing using deep reinforcement learning,’ *Nature*, vol. 620, no. 7976, pp. 982–987, Aug. 2023, ISSN: 1476-4687. DOI: 10.1038/s41586-023-06419-4.
- [172] B. Yuksek, M. Umut Demirezen, G. Inalhan and A. Tsourdos, ‘Cooperative Planning for an Unmanned Combat Aerial Vehicle Fleet Using Reinforcement Learning,’ *Journal of Aerospace Information Systems*, vol. 18, no. 10, pp. 739–750, 2021, ISSN: 1940-3151. DOI: 10.2514/1.I010961.

- [173] D. Ebrahimi, S. Sharafeddine, P.-H. Ho and C. Assi, ‘Autonomous UAV Trajectory for Localizing Ground Objects: A Reinforcement Learning Approach,’ *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1312–1324, Apr. 2021, ISSN: 1558-0660. DOI: 10.1109/TMC.2020.2966989.
- [174] M. Shurrah, R. Mizouni, S. Singh and H. Otkrok, ‘Reinforcement learning framework for UAV-based target localization applications,’ *Internet of Things*, vol. 23, p. 100 867, Oct. 2023, ISSN: 2542-6605. DOI: 10.1016/j.iot.2023.100867.
- [175] Z. Zong, D. Li, X. Dong *et al.*, ‘Risk-Aware Enabled Path Planning for Drones Flight in Unknown Environment,’ *Journal of Intelligent & Robotic Systems*, vol. 111, no. 2, p. 47, Apr. 2025, ISSN: 1573-0409. DOI: 10.1007/s10846-025-02243-2.
- [176] M. Mohammadi and M. W. Shafer, ‘UAV Path Planning for Precision Multi-Target Localization,’ *IEEE Access*, vol. 13, pp. 63 715–63 728, 2025, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2025.3558700.
- [177] P. Chen, J. Pei, W. Lu and M. Li, ‘A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance,’ *Neurocomputing*, vol. 497, pp. 64–75, Aug. 2022, ISSN: 09252312. DOI: 10.1016/j.neucom.2022.05.006.
- [178] L. Chen, Z. Jiang, L. Cheng, A. C. Knoll and M. Zhou, ‘Deep Reinforcement Learning Based Trajectory Planning Under Uncertain Constraints,’ *Frontiers in Neurorobotics*, vol. 16, May 2022, ISSN: 1662-5218. DOI: 10.3389/fnbot.2022.883562.
- [179] J. Raajan, P. V. Srihari, J. P. Satya, B. Bhikkaji and R. Pasumarthi, ‘Real Time Path Planning of Robot using Deep Reinforcement Learning,’ *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 602–15 607, 2020, ISSN: 24058963. DOI: 10.1016/j.ifacol.2020.12.2494.
- [180] J. Li, S. Song and X. Shi, ‘Deep reinforcement learning based trajectory real-time planning for hypersonic gliding vehicles,’ *Proceedings of the Institution of Mechanical Engineers, Part G*, vol. 238, no. 16, pp. 1665–1682, Dec. 2024, ISSN: 0954-4100. DOI: 10.1177/09544100241278023.
- [181] S. Josef and A. Degani, ‘Deep Reinforcement Learning for Safe Local Planning of a Ground Vehicle in Unknown Rough Terrain,’ *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6748–6755, Oct. 2020, ISSN: 2377-3766. DOI: 10.1109/LRA.2020.3011912.
- [182] M. Wei, S. Wang, J. Zheng and D. Chen, ‘UGV Navigation Optimization Aided by Reinforcement Learning-Based Path Tracking,’ *IEEE Access*, vol. 6, pp. 57 814–57 825, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2872751.

- [183] C. Sammut, ‘Behavioral Cloning,’ in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., Boston, MA: Springer US, 2011, pp. 93–97, ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_69.
- [184] J. Ho and S. Ermon, *Generative Adversarial Imitation Learning*, Jun. 2016. arXiv: 1606.03476 [cs].
- [185] J. Song, H. Ren, D. Sadigh and S. Ermon, ‘Multi-agent generative adversarial imitation learning,’ in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS’18, Red Hook, NY, USA: Curran Associates Inc., Dec. 2018, pp. 7472–7483.
- [186] S. Ross, G. J. Gordon and J. A. Bagnell, *A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning*, Mar. 2011. DOI: 10.48550/arXiv.1011.0686. arXiv: 1011.0686 [cs].
- [187] T. Laperrière-Robillard, M. Morin and I. Abi-Zeid, ‘Supervised learning for maritime search operations: An artificial intelligence approach to search efficiency evaluation,’ *Expert Systems with Applications*, vol. 206, p. 117857, Nov. 2022, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2022.117857.
- [188] P. Yao, Q. Zhu and R. Zhao, ‘Gaussian Mixture Model and Self-Organizing Map Neural-Network-Based Coverage for Target Search in Curve-Shape Area,’ *IEEE Transactions on Cybernetics*, vol. 52, no. 5, pp. 3971–3983, May 2022, ISSN: 2168-2275. DOI: 10.1109/TCYB.2020.3019255.
- [189] F. Niroui, K. Zhang, Z. Kashino and G. Nejat, ‘Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments,’ *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 610–617, Apr. 2019, ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2019.2891991.
- [190] A. Patel, C. Kanellakis and G. Nikolakopoulos, ‘Traversable Frontiers Based Autonomous Exploration Strategy for Deploying MAVs in Subterranean Environments,’ in *2022 Eighth Indian Control Conference (ICC)*, Dec. 2022, pp. 260–265. DOI: 10.1109/ICC56513.2022.10093322.
- [191] A. Peake, J. McCalmon, Y. Zhang, B. Raiford and S. Alqahtani, ‘Wilderness Search and Rescue Missions using Deep Reinforcement Learning,’ in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Nov. 2020, pp. 102–107. DOI: 10.1109/SSRR50563.2020.9292613.
- [192] A. Peake, J. McCalmon, Y. Zhang, D. Myers, S. Alqahtani and P. Pauca, ‘Deep Reinforcement Learning for Adaptive Exploration of Unknown Environments,’ in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, Jun. 2021, pp. 265–274. DOI: 10.1109/ICUAS51884.2021.9476756.

- [193] N. Benalaya, I. Amdouni, C. Adjih, A. Laouiti and L. A. Saidane, ‘Deep Reinforcement Learning Approach for UAV Search Path Planning In Discrete Time and Space,’ in *2024 International Wireless Communications and Mobile Computing (IWCMC)*, May 2024, pp. 1437–1442. DOI: 10.1109/IWCMC61514.2024.10592510.
- [194] N. Benalaya, C. Adjih, I. Amdouni, A. Laouiti and L. Saidane, ‘UAV Search Path Planning For Livestock Monitoring,’ in *2022 IEEE 11th IFIP International Conference on Performance Evaluation and Modeling in Wireless and Wired Networks (PEMWN)*, Nov. 2022, pp. 1–6. DOI: 10.23919/PEMWN56085.2022.9963839.
- [195] S. Manjanna, H. van Hoof and G. Dudek, ‘Reinforcement Learning with Non-uniform State Representations for Adaptive Search,’ in *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Aug. 2018, pp. 1–7. DOI: 10.1109/SSRR.2018.8468649.
- [196] E. Tola, V. Lepetit and P. Fua, ‘DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo,’ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 815–830, May 2010, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2009.77.
- [197] S. Manjanna, M. A. Hsieh and G. Dudek, ‘Scalable multirobot planning for informed spatial sampling,’ *Autonomous Robots*, vol. 46, no. 7, pp. 817–829, Oct. 2022, ISSN: 0929-5593, 1573-7527. DOI: 10.1007/s10514-022-10048-7.
- [198] A. C. Kapoutsis, S. A. Chatzichristofis, S. A. Chatzichristofis, E. B. Kosmatopoulos and E. B. Kosmatopoulos, ‘DARP: Divide Areas Algorithm for Optimal Multi-Robot Coverage Path Planning,’ *Journal of Intelligent and Robotic Systems*, vol. 86, no. 3, pp. 663–680, Jun. 2017. DOI: 10.1007/s10846-016-0461-x.
- [199] W. S. McCulloch and W. Pitts, ‘A logical calculus of the ideas immanent in nervous activity,’ *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943, ISSN: 0007-4985, 1522-9602. DOI: 10.1007/BF02478259.
- [200] K. Fukushima, ‘Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,’ *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, Apr. 1980, ISSN: 0340-1200, 1432-0770. DOI: 10.1007/BF00344251.
- [201] Y. LeCun, Y. Bengio and G. Hinton, ‘Deep learning,’ *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, ISSN: 1476-4687. DOI: 10.1038/nature14539.
- [202] R. Kiros, Y. Zhu, R. Salakhutdinov *et al.*, ‘Skip-Thought Vectors,’ *arXiv:1506.06726 [cs]*, Jun. 2015. arXiv: 1506.06726 [cs].

- [203] T. Yao, Y. Pan, Y. Li and T. Mei, ‘Incorporating Copying Mechanism in Image Captioning for Learning Novel Objects,’ in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 5263–5271. DOI: 10.1109/CVPR.2017.559.
- [204] J. Wang, J. Wang, W. Fang and H. Niu, ‘Financial Time Series Prediction Using Elman Recurrent Random Neural Networks,’ *Computational Intelligence and Neuroscience*, vol. 2016, no. 1, p. 4742515, 2016, ISSN: 1687-5273. DOI: 10.1155/2016/4742515.
- [205] J. L. Elman, ‘Finding structure in time,’ *Cognitive Science*, vol. 14, no. 2, pp. 179–211, Apr. 1990, ISSN: 0364-0213. DOI: 10.1016/0364-0213(90)90002-E.
- [206] S. Hochreiter and J. Schmidhuber, ‘Long Short-Term Memory,’ *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.
- [207] S. Hochreiter, ‘The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions,’ *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 06, no. 02, pp. 107–116, Apr. 1998, ISSN: 0218-4885, 1793-6411. DOI: 10.1142/S0218488598000094.
- [208] K. Cho, B. van Merriënboer, C. Gulcehre *et al.*, *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*, Sep. 2014. DOI: 10.48550/arXiv.1406.1078. arXiv: 1406.1078 [cs, stat].
- [209] R. Cahuantzi, X. Chen and S. Güttel, ‘A Comparison of LSTM and GRU Networks for Learning Symbolic Sequences,’ in *Intelligent Computing*, K. Arai, Ed., vol. 739, Cham: Springer Nature Switzerland, 2023, pp. 771–785, ISBN: 978-3-031-37963-5. DOI: 10.1007/978-3-031-37963-5_53.
- [210] D. Bahdanau, K. Cho and Y. Bengio, *Neural Machine Translation by Jointly Learning to Align and Translate*, Sep. 2014. DOI: 10.48550/arXiv.1409.0473. arXiv: 1409.0473 [cs, stat].
- [211] M.-T. Luong, H. Pham and C. D. Manning, *Effective Approaches to Attention-based Neural Machine Translation*, Sep. 2015. DOI: 10.48550/arXiv.1508.04025. arXiv: 1508.04025 [cs].
- [212] D. Britz, A. Goldie, M.-T. Luong and Q. Le, *Massive Exploration of Neural Machine Translation Architectures*, Mar. 2017. DOI: 10.48550/arXiv.1703.03906. arXiv: 1703.03906 [cs].
- [213] A. Vaswani, N. Shazeer, N. Parmar *et al.*, *Attention Is All You Need*, Jun. 2017. arXiv: 1706.03762 [cs].
- [214] J. Cheng, L. Dong and M. Lapata, *Long Short-Term Memory-Networks for Machine Reading*, Sep. 2016. DOI: 10.48550/arXiv.1601.06733. arXiv: 1601.06733 [cs].

- [215] A. P. Parikh, O. Täckström, D. Das and J. Uszkoreit, *A Decomposable Attention Model for Natural Language Inference*, Sep. 2016. DOI: 10.48550/arXiv.1606.01933. arXiv: 1606.01933 [cs].
- [216] T. Xu, P. Zhang, Q. Huang *et al.*, *AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks*, Nov. 2017. DOI: 10.48550/arXiv.1711.10485. arXiv: 1711.10485 [cs].
- [217] X. Wang, R. Girshick, A. Gupta and K. He, *Non-local Neural Networks*, Apr. 2018. DOI: 10.48550/arXiv.1711.07971. arXiv: 1711.07971 [cs].
- [218] H. Zhang, I. Goodfellow, D. Metaxas and A. Odena, *Self-Attention Generative Adversarial Networks*, Jun. 2019. DOI: 10.48550/arXiv.1805.08318. arXiv: 1805.08318 [stat].
- [219] N. Parmar, A. Vaswani, J. Uszkoreit *et al.*, *Image Transformer*, Jun. 2018. DOI: 10.48550/arXiv.1802.05751. arXiv: 1802.05751 [cs].
- [220] Y. Zhou, D. Arpit, I. Nwogu and V. Govindaraju, *Is Joint Training Better for Deep Auto-Encoders?* Jun. 2015. DOI: 10.48550/arXiv.1405.1380. arXiv: 1405.1380 [stat].
- [221] T. Wong and Z. Luo, ‘Recurrent Auto-Encoder Model for Large-Scale Industrial Sensor Signal Analysis,’ in *Engineering Applications of Neural Networks*, E. Pimenidis and C. Jayne, Eds., vol. 893, Cham: Springer International Publishing, 2018, pp. 203–216, ISBN: 978-3-319-98204-5. DOI: 10.1007/978-3-319-98204-5_17.
- [222] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, *Proximal Policy Optimization Algorithms*, Aug. 2017. DOI: 10.48550/arXiv.1707.06347. arXiv: 1707.06347 [cs].
- [223] T. Haarnoja, A. Zhou, P. Abbeel and S. Levine, *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*, Aug. 2018. DOI: 10.48550/arXiv.1801.01290. arXiv: 1801.01290 [cs, stat].
- [224] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (Wiley Series in Probability and Statistics), 1st ed. Wiley, Apr. 1994, ISBN: 978-0-470-31688-7. DOI: 10.1002/9780470316887.
- [225] J. Schulman, ‘Optimizing expectations: From deep reinforcement learning to stochastic computation graphs,’ Ph.D. dissertation, EECS Department, University of California, Berkeley, Dec. 2016.
- [226] OpenAI, ‘Dota 2 with Large Scale Deep Reinforcement Learning,’ *arXiv:1912.06680 [cs]*, Dec. 2019. DOI: 10.48550/arXiv.1912.06680. arXiv: 1912.06680 [cs].

- [227] S. Kakade and J. Langford, ‘Approximately Optimal Approximate Reinforcement Learning,’ in *Proceedings of the Nineteenth International Conference on Machine Learning*, ser. ICML ’02, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jul. 2002, pp. 267–274, ISBN: 978-1-55860-873-3.
- [228] J. Schulman, P. Moritz, S. Levine, M. Jordan and P. Abbeel, *High-Dimensional Continuous Control Using Generalized Advantage Estimation*, Oct. 2018. DOI: 10.48550/arXiv.1506.02438. arXiv: 1506.02438 [cs].
- [229] T. Haarnoja, A. Zhou, K. Hartikainen *et al.*, *Soft Actor-Critic Algorithms and Applications*, Jan. 2019. arXiv: 1812.05905 [cs, stat].
- [230] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, ‘Human-level control through deep reinforcement learning,’ *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, ISSN: 1476-4687. DOI: 10.1038/nature14236.
- [231] H. van Hasselt, A. Guez and D. Silver, *Deep Reinforcement Learning with Double Q-learning*, Dec. 2015. DOI: 10.48550/arXiv.1509.06461. arXiv: 1509.06461 [cs].
- [232] S. Fujimoto, H. van Hoof and D. Meger, *Addressing Function Approximation Error in Actor-Critic Methods*, Oct. 2018. DOI: 10.48550/arXiv.1802.09477. arXiv: 1802.09477 [cs].
- [233] 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, Jun. 2020, ISSN: 15579603. DOI: 10.1109/TAES.2019.2949384.
- [234] D. Reilly, M. Taylor, P. Fergus, C. Chalmers and S. Thompson, ‘Misper-Bayes: A Bayesian Network Model for Missing Person Investigations,’ *IEEE Access*, vol. 9, pp. 49 990–50 000, Mar. 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3069081.
- [235] Geographic Information Science and Analysis Team, *Lidar Composite Digital Terrain Model Scotland (Phase2) 1m resolution*, DEM, nr83ne, nr84ne, nr84se, nr93nw, nr94nw, nr94sw, nr95sw, nr93ne, nr94ne, nr94se, nr95se, ns03nw, ns04nw, ns04sw, nr82ne, nr83se, nr92n, nr92sw, nr93sw, nr91ne, nr92ne, nr92se, nr93se, ns02ne, ns02nw, ns02se, ns02sw, ns03se, ns03sw, Jul. 2014.
- [236] R. Morton, C. Marston, A. O’Neil and C. Rowland, *Land Cover Map 2020 (land parcels, GB)*, 2021. DOI: 10.5285/0E99D57E-1757-451F-AC9D-92FD1256F02A.
- [237] D. Morris and R. Flavin, ‘A digital terrain model for hydrology,’ in *Proc 4th International Symposium on Spatial Data Handling*, vol. 1, Zurich, Jul. 1990, pp. 250–262.
- [238] Ordnance Survey, *OS MasterMap Highways Network®*, Mar. 2021.

- [239] H. Haverkort, L. Toma and Y. Zhuang, ‘Computing visibility on terrains in external memory,’ *ACM Journal of Experimental Algorithmics*, vol. 13, Feb. 2009, ISSN: 1084-6654, 1084-6654. DOI: 10.1145/1412228.1412233.
- [240] K. Gast, R. Kram and R. Riemer, ‘Preferred walking speed on rough terrain; is it all about energetics?’ *Journal of Experimental Biology*, vol. 222, no. 9, jeb.185447, Jan. 2019, ISSN: 1477-9145, 0022-0949. DOI: 10.1242/jeb.185447.
- [241] P. Getreuer, ‘Linear Methods for Image Interpolation,’ *Image Processing On Line*, vol. 1, pp. 238–259, Sep. 2011, ISSN: 2105-1232. DOI: 10.5201/ipol.2011.g_lmii.
- [242] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (Adaptive Computation and Machine Learning). Cambridge, Mass: MIT Press, 2006, ISBN: 978-0-262-18253-9.
- [243] J. Park and J. Choi, ‘Gaussian Process Online Learning With a Sparse Data Stream,’ *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5977–5984, Oct. 2020, ISSN: 2377-3766. DOI: 10.1109/LRA.2020.3010752.
- [244] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger and A. G. Wilson, *GPpy-Torch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration*, Jun. 2021. DOI: 10.48550/arXiv.1809.11165. arXiv: 1809.11165 [cs, stat].
- [245] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions: With Formulas, Graphs and Mathematical Tables [Conference under the Auspices of the National Science Foundation and the Massachusetts Institute of Technology]* (Dover Books on Advanced Mathematics), Unabridged, unaltered and corr. republ. of the 1964 ed. New York: Dover publ, 1972, ISBN: 978-0-486-61272-0.
- [246] D. Cornford, I. Nabney and C. Williams, ‘Modelling Frontal Discontinuities in Wind Fields,’ *Journal of Nonparametric Statistics*, vol. 14, Apr. 2002. DOI: 10.1080/10485250211392.
- [247] A. G. Wilson and H. Nickisch, *Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)*, Mar. 2015. DOI: 10.48550/arXiv.1503.01057. arXiv: 1503.01057 [cs, stat].
- [248] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, Jan. 2017. DOI: 10.48550/arXiv.1412.6980. arXiv: 1412.6980 [cs].
- [249] M. Mortensen and H. P. Langtangen, ‘High performance Python for direct numerical simulations of turbulent flows,’ *Computer Physics Communications*, vol. 203, pp. 53–65, Jun. 2016, ISSN: 00104655. DOI: 10.1016/j.cpc.2016.02.005. arXiv: 1602.03638 [cs].
- [250] C. Coleman, S. Lyon, L. Maliar and S. Maliar, ‘Matlab, Python, Julia: What to Choose in Economics?’ *Computational Economics*, vol. 58, no. 4, pp. 1263–1288, Dec. 2021, ISSN: 0927-7099, 1572-9974. DOI: 10.1007/s10614-020-09983-3.

- [251] L. Oden, ‘Lessons learned from comparing C-CUDA and Python-Numba for GPU-Computing,’ in *2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Mar. 2020, pp. 216–223. DOI: 10.1109/PDP50117.2020.00041.
- [252] O. E. Wattne and F. Volden, ‘Waylosing and wayfinding in the outdoors: A typology of wayfinding approaches to problem-solving when temporarily lost,’ *The Journal of Navigation*, vol. 76, no. 6, pp. 603–626, Nov. 2023, ISSN: 0373-4633, 1469-7785. DOI: 10.1017/S0373463324000067.
- [253] P. A. Dudchenko, ‘Human navigation,’ in *Why People Get Lost: The Psychology and Neuroscience of Spatial Cognition*, P. Dudchenko, Ed., Oxford University Press, Jul. 2010, ISBN: 978-0-19-921086-2. DOI: 10.1093/acprof:oso/9780199210862.003.0004.
- [254] I. H. Sarker, ‘Machine Learning: Algorithms, Real-World Applications and Research Directions,’ *SN Computer Science*, vol. 2, no. 3, p. 160, May 2021, ISSN: 2662-995X, 2661-8907. DOI: 10.1007/s42979-021-00592-x.
- [255] M. I. Jordan and T. M. Mitchell, ‘Machine learning: Trends, perspectives, and prospects,’ *Science*, vol. 349, no. 6245, pp. 255–260, Jul. 2015, ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aaa8415.
- [256] C. Stokel-Walker and R. Van Noorden, ‘What ChatGPT and generative AI mean for science,’ *Nature*, vol. 614, no. 7947, pp. 214–216, Feb. 2023, ISSN: 0028-0836, 1476-4687. DOI: 10.1038/d41586-023-00340-6.
- [257] D. F. Nettleton, A. Orriols-Puig and A. Fornells, ‘A study of the effect of different types of noise on the precision of supervised learning techniques,’ *Artificial Intelligence Review*, vol. 33, no. 4, pp. 275–306, Apr. 2010, ISSN: 1573-7462. DOI: 10.1007/s10462-010-9156-z.
- [258] J. R. Quinlan, ‘Induction of decision trees,’ *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar. 1986, ISSN: 1573-0565. DOI: 10.1007/BF00116251.
- [259] Y. Guo and W. Wang, ‘A robust adaptive linear regression method for severe noise,’ *Knowledge and Information Systems*, vol. 65, no. 11, pp. 4613–4653, Nov. 2023, ISSN: 0219-3116. DOI: 10.1007/s10115-023-01924-4.
- [260] R. Fox, A. Pakman and N. Tishby, *Taming the Noise in Reinforcement Learning via Soft Updates*, Mar. 2017. DOI: 10.48550/arXiv.1512.08562. arXiv: 1512.08562 [cs, math].
- [261] K. Sun, Y. Zhao, S. Jui and L. Kong, ‘Exploring the Training Robustness of Distributional Reinforcement Learning Against Noisy State Observations,’ in *Machine Learning and Knowledge Discovery in Databases: Research Track*, D. Koutra, C. Plant, M. Gomez Rodriguez, E. Baralis and F. Bonchi, Eds., ser. Lecture Notes in Computer Science, Cham: Springer Nature Switzerland,

- 2023, pp. 36–51, ISBN: 978-3-031-43424-2. DOI: 10.1007/978-3-031-43424-2_3.
- [262] T. P. Lillicrap, J. J. Hunt, A. Pritzel *et al.*, *Continuous control with deep reinforcement learning*, Jul. 2019. DOI: 10.48550/arXiv.1509.02971. arXiv: 1509.02971 [cs, stat].
- [263] J. Hollenstein, S. Auddy, M. Saveriano, E. Renaudo and J. Piater, *Action Noise in Off-Policy Deep Reinforcement Learning: Impact on Exploration and Performance*, Jun. 2023. DOI: 10.48550/arXiv.2206.03787. arXiv: 2206.03787 [cs].
- [264] T. Everitt, V. Krakovna, L. Orseau, M. Hutter and S. Legg, *Reinforcement Learning with a Corrupted Reward Channel*, Aug. 2017. DOI: 10.48550/arXiv.1705.08417. arXiv: 1705.08417 [cs, stat].
- [265] J. Lyness and R. Cools, ‘A Survey of Numerical Cubature over Triangles,’ *Proceedings of Symposia in Applied Mathematics American Mathematical Society Providence, RI*, vol. 48, Apr. 1994, ISSN: 9780821802915. DOI: 10.1090/psapm/048/1314845.
- [266] R. Cools, L. Pluym and D. Laurie, ‘Algorithm 764: Cubpack++: A C++ package for automatic two-dimensional cubature,’ *ACM Transactions on Mathematical Software*, vol. 23, no. 1, pp. 1–15, Mar. 1997, ISSN: 0098-3500. DOI: 10.1145/244768.244770.
- [267] I. Robinson and M. Hill, ‘Algorithm 816: R2d2lri: An algorithm for automatic two-dimensional cubature,’ *ACM Trans. Math. Softw.*, vol. 28, no. 1, pp. 75–100, Mar. 2002, ISSN: 0098-3500. DOI: 10.1145/513001.513006.
- [268] M. Hill and I. Robinson, ‘A Comparison of Strategies for the Automatic Computation of Two-Dimensional Integrals over Infinite Domains,’ *Numerical Algorithms*, vol. 34, no. 2-4, pp. 325–338, Dec. 2003, ISSN: 1017-1398. DOI: 10.1023/B:NUMA.0000005348.15222.6a.
- [269] K. Farley, K. Williford and K. et al. Stack, ‘Mars 2020 mission overview,’ *Space Science Reviews*, vol. 216, no. 142, 2020. DOI: 10.1007/s11214-020-00762-y.
- [270] T. Fong, B. D. Allen and S. Azimi, ‘Autonomous systems & robotics for lunar surface infrastructure,’ in *Aiaa Ascend*, 2021.
- [271] T. Vaquero, M. Troesch and S. Chien, ‘An approach for autonomous multi-rover collaboration for mars cave exploration: Preliminary results,’ in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space (iSAIRAS 2018)*, Madrid, Spain, Jun. 2018.

- [272] T. Furukawa, F. Bourgault, B. Lavis and H. Durrant-Whyte, ‘Recursive Bayesian search-and-tracking using coordinated uavs for lost targets,’ in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, Orlando, FL, USA: IEEE, 2006, pp. 2521–2526, ISBN: 978-0-7803-9505-3. DOI: 10.1109/ROBOT.2006.1642081.
- [273] J. W. Bergstrom, W. Delamere and A. McEwen, ‘MRO high resolution imaging science experiment (HiRISE): Instrument test, calibration and operating constraints,’ in *55th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law*, 2004, Q–3.
- [274] M. Baxter, ‘Modelling and visualisation of the mars terrain for rover simulations,’ M.S. thesis, University of Glasgow, 2022.
- [275] S. Swinton and E. McGookin, ‘A Novel, RRT* Based Approach to the Coordination of Multiple Planetary Rovers,’ in *2022 UKACC 13th International Conference on Control (CONTROL)*, Apr. 2022, pp. 88–93. DOI: 10.1109/Control55989.2022.9781363.
- [276] I. E. Sutherland, R. F. Sproull and R. A. Schumacker, ‘A Characterization of Ten Hidden-Surface Algorithms,’ *ACM Computing Surveys*, vol. 6, no. 1, pp. 1–55, Mar. 1974, ISSN: 0360-0300. DOI: 10.1145/356625.356626.
- [277] E. Clementini, P. Di Felice and P. Van Oosterom, ‘A small set of formal topological relationships suitable for end-user interaction,’ in *Advances in Spatial Databases*, G. Goos, J. Hartmanis, D. Abel and B. Chin Ooi, Eds., vol. 692, Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 277–295, ISBN: 978-3-540-47765-5. DOI: 10.1007/3-540-56869-7_16.
- [278] L. P. Chew, ‘Constrained delaunay triangulations,’ *Algorithmica*, vol. 4, no. 1-4, pp. 97–108, 1987, ISSN: 0178-4617. DOI: 10.1007/BF01553881.
- [279] J. Berntsen and T. O. Espelid, ‘Algorithm 706: DCUTRI: An algorithm for adaptive cubature over a collection of triangles,’ *ACM Transactions on Mathematical Software*, vol. 18, no. 3, pp. 329–342, Sep. 1992, ISSN: 0098-3500. DOI: 10.1145/131766.131772.
- [280] J. N. Lyness, ‘Symmetric integration rules for hypercubes. III. Construction of integration rules using null rules,’ *Mathematics of Computation*, vol. 19, no. 92, pp. 625–637, 1965, ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-1965-0201069-7.
- [281] R. Cools and A. Haegemans, ‘Algorithm 824: CUBPACK: A package for automatic cubature; framework description,’ *ACM Transactions on Mathematical Software*, vol. 29, no. 3, pp. 287–296, Sep. 2003, ISSN: 0098-3500. DOI: 10.1145/838250.838253.

- [282] J. W. Mock and S. S. Muknahallipatna, ‘A Comparison of PPO, TD3 and SAC Reinforcement Algorithms for Quadraped Walking Gait Generation,’ *Journal of Intelligent Learning Systems and Applications*, vol. 15, no. 1, pp. 36–56, Feb. 2023. DOI: 10.4236/jilsa.2023.151003.
- [283] J. Xu, T. Du, M. Foshey *et al.*, ‘Learning to fly: Computational controller design for hybrid UAVs with reinforcement learning,’ *ACM Transactions on Graphics*, vol. 38, no. 4, pp. 1–12, Aug. 2019, ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3306346.3322940.
- [284] R. Tedrake, *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. Course Notes for MIT 6.832, 2023.
- [285] E. Fresk and G. Nikolakopoulos, ‘Full quaternion based attitude control for a quadrotor,’ in *2013 European Control Conference (ECC)*, Zurich: IEEE, Jul. 2013, pp. 3864–3869, ISBN: 978-3-033-03962-9. DOI: 10.23919/ECC.2013.6669617.
- [286] P. Wang, M. Zhu and S. Shen, *Environment Transformer and Policy Optimization for Model-Based Offline Reinforcement Learning*, Oct. 2023. DOI: 10.48550/arXiv.2303.03811. arXiv: 2303.03811.
- [287] H. Yu, P. Wang, J. Wang *et al.*, ‘Catch Planner: Catching High-Speed Targets in the Flight,’ *IEEE/ASME Transactions on Mechatronics*, vol. 28, no. 4, pp. 2387–2398, Aug. 2023, ISSN: 1941-014X. DOI: 10.1109/TMECH.2023.3286102.
- [288] C. Richter, A. Bry and N. Roy, ‘Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments,’ in *Robotics Research*, M. Inaba and P. Corke, Eds., vol. 114, Cham: Springer International Publishing, 2016, pp. 649–666, ISBN: 978-3-319-28872-7. DOI: 10.1007/978-3-319-28872-7_37.
- [289] R. Agarwal, M. Schwarzer, P. S. Castro, A. Courville and M. G. Bellemare, *Deep Reinforcement Learning at the Edge of the Statistical Precipice*, Jan. 2022. arXiv: 2108.13264 [cs, stat].
- [290] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus and N. Dormann, ‘Stable-Baselines3: Reliable Reinforcement Learning Implementations,’ *Journal of Machine Learning Research*, no. 22, 2021.
- [291] J. Li, T. Chen, R. Shi, Y. Lou, Y.-L. Li and C. Lu, *Localization with Sampling-Argmax*, Oct. 2021. DOI: 10.48550/arXiv.2110.08825. arXiv: 2110.08825 [cs].
- [292] A. Raffin, A. Hill, R. Traoré, T. Lesort, N. Díaz-Rodríguez and D. Filliat, *Decoupling feature extraction from policy learning: Assessing benefits of state representation learning in goal based robotics*, <https://arxiv.org/abs/1901.08651v3>, Jan. 2019.

- [293] S. H. Park, B. Kim, C. M. Kang, C. C. Chung and J. W. Choi, ‘Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture,’ in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2018, pp. 1672–1678. DOI: 10.1109/IVS.2018.8500658.
- [294] K. Berahmand, F. Daneshfar, E. S. Salehi, Y. Li and Y. Xu, ‘Autoencoders and their applications in machine learning: A survey,’ *Artificial Intelligence Review*, vol. 57, no. 2, p. 28, Feb. 2024, ISSN: 1573-7462. DOI: 10.1007/s10462-023-10662-6.
- [295] M. Pleines, M. Pallasch, F. Zimmer and M. Preuss, *Generalization, Mayhems and Limits in Recurrent Proximal Policy Optimization*, May 2022. DOI: 10.48550/arXiv.2205.11104. arXiv: 2205.11104 [cs].
- [296] L. Yue, R. Yang, Y. Zhang, L. Yu and Z. Wang, ‘Deep Reinforcement Learning for UAV Intelligent Mission Planning,’ *Complexity*, vol. 2022, e3551508, Mar. 2022, ISSN: 1076-2787. DOI: 10.1155/2022/3551508.
- [297] J. Shianifar, M. Schukat and K. Mason, ‘Optimizing Deep Reinforcement Learning for Adaptive Robotic Arm Control,’ in *Highlights in Practical Applications of Agents, Multi-Agent Systems, and Digital Twins: The PAAMS Collection*, A. González-Briones, V. Julian Inglada, A. El Bolock *et al.*, Eds., vol. 2149, Cham: Springer Nature Switzerland, 2025, pp. 293–304, ISBN: 978-3-031-73058-0. DOI: 10.1007/978-3-031-73058-0_24.
- [298] J. Wang, Y. Liu and B. Li, *Reinforcement Learning with Perturbed Rewards*, Feb. 2020. DOI: 10.48550/arXiv.1810.01032. arXiv: 1810.01032 [cs].
- [299] C. Lea, R. Vidal, A. Reiter and G. D. Hager, *Temporal Convolutional Networks: A Unified Approach to Action Segmentation*, Aug. 2016. DOI: 10.48550/arXiv.1608.08242. arXiv: 1608.08242 [cs].
- [300] H. A. F. Almurib, P. T. Nathan and T. N. Kumar, ‘Control and path planning of quadrotor aerial vehicles for search and rescue,’ in *SICE Annual Conference 2011*, Sep. 2011, pp. 700–705.
- [301] E. Cetinsoy, ‘Design and simulation of a holonomic quadrotor UAV with sub-rotor control surfaces,’ Dec. 2012, pp. 1164–1169, ISBN: 978-1-4673-2125-9. DOI: 10.1109/ROBIO.2012.6491127.
- [302] H. C. Fernando, A. T. De Silva, M. D. De Zoysa, K. A. Dilshan and S. R. Munasinghe, ‘Modelling, simulation and implementation of a quadrotor UAV,’ in *2013 IEEE 8th International Conference on Industrial and Information Systems, ICIIS 2013 - Conference Proceedings*, IEEE, Dec. 2013, pp. 207–212, ISBN: 978-1-4799-0910-0. DOI: 10.1109/ICIInfS.2013.6731982.

- [303] Y. Li, H. Eslamiat, N. Wang, Z. Zhao, A. K. Sanyal and Q. Qiu, ‘Autonomous waypoints planning and trajectory generation for multi-rotor UAVs,’ in *Proceedings of the Workshop on Design Automation for CPS and IoT - DESTION '19*, Montreal, Quebec, Canada: ACM Press, Apr. 2019, pp. 31–40, ISBN: 978-1-4503-6699-1. DOI: 10.1145/3313151.3313163.
- [304] N. Mimmo, P. Bernard and L. Marconi, ‘Avalanche Victim Search via Robust Observers,’ *IEEE Transactions on Control Systems Technology*, vol. 29, no. 4, pp. 1450–1461, Jul. 2021, ISSN: 1558-0865. DOI: 10.1109/TCST.2020.3016665.
- [305] E. Rodríguez-Novillo and A. Sanchez-Carmona, ‘Battery consumption estimation methodology for electric unmanned aerial systems,’ *The Aeronautical Journal*, vol. 126, no. 1303, pp. 1527–1545, Sep. 2022, ISSN: 0001-9240, 2059-6464. DOI: 10.1017/aer.2022.7.
- [306] Y. Zeng and R. Zhang, ‘Energy-Efficient UAV Communication With Trajectory Optimization,’ *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, Jun. 2017, ISSN: 1558-2248. DOI: 10.1109/TWC.2017.2688328.
- [307] H. Sallouha, M. M. Azari and S. Pollin, ‘Energy-Constrained UAV Trajectory Design for Ground Node Localization,’ in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018, pp. 1–7. DOI: 10.1109/GLOCOM.2018.8647530.
- [308] M. Botvinick, J. X. Wang, W. Dabney, K. J. Miller and Z. Kurth-Nelson, ‘Deep Reinforcement Learning and Its Neuroscientific Implications,’ *Neuron*, vol. 107, no. 4, pp. 603–616, Aug. 2020, ISSN: 0896-6273. DOI: 10.1016/j.neuron.2020.06.014.
- [309] P. Hernandez-Leal, B. Kartal and M. E. Taylor, ‘A survey and critique of multiagent deep reinforcement learning,’ *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, Nov. 2019, ISSN: 1573-7454. DOI: 10.1007/s10458-019-09421-1.
- [310] J. Degraeve, F. Felici, J. Buchli *et al.*, ‘Magnetic control of tokamak plasmas through deep reinforcement learning,’ *Nature*, vol. 602, no. 7897, pp. 414–419, Feb. 2022, ISSN: 1476-4687. DOI: 10.1038/s41586-021-04301-9.
- [311] I. Carlucho, M. De Paula, S. Wang, Y. Petillot and G. G. Acosta, ‘Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning,’ *Robotics and Autonomous Systems*, vol. 107, pp. 71–86, Sep. 2018, ISSN: 0921-8890. DOI: 10.1016/j.robot.2018.05.016.
- [312] M. H. Shahna, S. Adel Alizadeh Kolagar and J. Mattila, ‘Integrating DeepRL with Robust Low-Level Control in Robotic Manipulators for Non-Repetitive Reaching Tasks,’ in *2024 IEEE International Conference on Mechatronics and Automation (ICMA)*, Aug. 2024, pp. 329–336. DOI: 10.1109/ICMA61710.2024.10632966.

- [313] S. A. Mehta, S. Habibian and D. P. Losey, ‘Waypoint-Based Reinforcement Learning for Robot Manipulation Tasks,’ in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2024, pp. 541–548. DOI: 10.1109/IROS58592.2024.10802681.
- [314] B. Wang, Z. Liu, Q. Li and A. Prorok, ‘Mobile Robot Path Planning in Dynamic Environments Through Globally Guided Reinforcement Learning,’ *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6932–6939, Oct. 2020, ISSN: 2377-3766. DOI: 10.1109/LRA.2020.3026638.
- [315] G. Chen, L. Pan, Y. Chen *et al.*, ‘Robot Navigation with Map-Based Deep Reinforcement Learning,’ in *2020 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, Oct. 2020, pp. 1–6. DOI: 10.1109/ICNSC48988.2020.9238090.
- [316] D. Valencia, H. Williams, Y. Xing, T. Gee, M. Liarokapis and B. A. MacDonald, ‘Image-Based Deep Reinforcement Learning with Intrinsically Motivated Stimuli: On the Execution of Complex Robotic Tasks,’ in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2024, pp. 587–594. DOI: 10.1109/IROS58592.2024.10801857.
- [317] Q. Deng, Y. Zhao, R. Li *et al.*, ‘A Pseudo-Hierarchical Planning Framework with Dynamic-Aware Reinforcement Learning for Autonomous Driving,’ in *2024 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2024, pp. 2345–2352. DOI: 10.1109/IV55156.2024.10588785.
- [318] S. Bouabdallah, P. Murrieri and R. Siegwart, ‘Design and control of an indoor micro quadrotor,’ in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004*, vol. 5, Apr. 2004, 4393–4398 Vol.5. DOI: 10.1109/ROBOT.2004.1302409.
- [319] G. C. Goodwin, S. F. Graebe, M. E. Salgado *et al.*, *Control System Design*. Prentice Hall Upper Saddle River, 2001, vol. 240.
- [320] L. M. Argentim, W. C. Rezende, P. E. Santos and R. A. Aguiar, ‘PID, LQR and LQR-PID on a quadcopter platform,’ in *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*, Dhaka, Bangladesh: IEEE, May 2013, pp. 1–6, ISBN: 978-1-4799-0399-3. DOI: 10.1109/ICIEV.2013.6572698.
- [321] A. Ur Rehman, M. U. Khan, M. S. Shah, Q. Maqsood, M. F. Ullah and S. Uba, ‘Aircraft Pitch Control based on Genetic Algorithm Tuning with PID and LQR Controller,’ in *2021 6th International Multi-Topic ICT Conference (IMTIC)*, Nov. 2021, pp. 1–6. DOI: 10.1109/IMTIC53841.2021.9719736.
- [322] Vishal and J. Ohri, ‘GA tuned LQR and PID controller for aircraft pitch control,’ in *2014 IEEE 6th India International Conference on Power Electronics (IICPE)*, Dec. 2014, pp. 1–6. DOI: 10.1109/IICPE.2014.7115839.

- [323] S. Rajasekar, *Numerical Methods: Classical and Advanced Topics*, First edition. Abingdon, Oxon: CRC Press, 2024, ISBN: 978-1-032-64993-1.
- [324] A. Patterson, S. Neumann, M. White and A. White, ‘Empirical Design in Reinforcement Learning,’ *Journal of Machine Learning Research*, vol. 25, no. 318, pp. 1–63, 2024, ISSN: 1533-7928.
- [325] W. Shi and C. and Cheung, ‘Performance Evaluation of Line Simplification Algorithms for Vector Generalization,’ *The Cartographic Journal*, vol. 43, no. 1, pp. 27–44, Mar. 2006, ISSN: 0008-7041. DOI: 10.1179/000870406X93490.
- [326] K. Schweppe, L. Moshagen and G. Schildbach, *On the Application of Model Predictive Control to a Weighted Coverage Path Planning Problem*, Nov. 2024. DOI: 10.48550/arXiv.2411.08634. arXiv: 2411.08634 [eess].
- [327] X. Geng, Z. Chen, W. Yang, D. Shi and K. Zhao, ‘Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search,’ *Applied Soft Computing*, vol. 11, no. 4, pp. 3680–3689, Jun. 2011, ISSN: 1568-4946. DOI: 10.1016/j.asoc.2011.01.039.
- [328] A. A. Hagberg, D. A. Schult and P. J. Swart, ‘Exploring network structure, dynamics, and function using NetworkX,’ in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11–15.
- [329] K. Lee and I. N. and Sener, ‘Strava Metro data for bicycle monitoring: A literature review,’ *Transport Reviews*, vol. 41, no. 1, pp. 27–47, Jan. 2021, ISSN: 0144-1647. DOI: 10.1080/01441647.2020.1798558.
- [330] R. Saravanan, *The Climate Demon: Past, Present, and Future of Climate Prediction*. Cambridge: Cambridge University Press, 2021, ISBN: 978-1-316-51076-6. DOI: 10.1017/9781009039604.
- [331] W. Zhao, J. P. Queralta and T. Westerlund, ‘Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey,’ in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, Dec. 2020, pp. 737–744. DOI: 10.1109/SSCI47803.2020.9308468.
- [332] X. B. Peng, M. Andrychowicz, W. Zaremba and P. Abbeel, ‘Sim-to-Real Transfer of Robotic Control with Dynamics Randomization,’ in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 3803–3810. DOI: 10.1109/ICRA.2018.8460528.
- [333] M. Torne, A. Simeonov, Z. Li *et al.*, *Reconciling Reality through Simulation: A Real-to-Sim-to-Real Approach for Robust Manipulation*, Nov. 2024. DOI: 10.48550/arXiv.2403.03949. arXiv: 2403.03949 [cs].

- [334] E. Valassakis, Z. Ding and E. Johns, ‘Crossing the Gap: A Deep Dive into Zero-Shot Sim-to-Real Transfer for Dynamics,’ in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 5372–5379. DOI: 10.1109/IROS45743.2020.9341617.
- [335] F. L. D. Silva, P. Hernandez-Leal, B. Kartal and M. E. Taylor, ‘Uncertainty-Aware Action Advising for Deep Reinforcement Learning Agents,’ *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 5792–5799, Apr. 2020, ISSN: 2374-3468. DOI: 10.1609/aaai.v34i04.6036.
- [336] C.-Y. Tang, C.-H. Liu, W.-K. Chen and S. D. You, ‘Implementing action mask in proximal policy optimization (PPO) algorithm,’ *ICT Express*, vol. 6, no. 3, pp. 200–203, Sep. 2020, ISSN: 2405-9595. DOI: 10.1016/j.icte.2020.05.003.
- [337] K. D. Julian, Mykel J. Kochenderfer and M. J. Kochenderfer, ‘Distributed Wild-fire Surveillance with Autonomous Aircraft using Deep Reinforcement Learning,’ *arXiv: Robotics*, Oct. 2018.
- [338] S. Linnainmaa, ‘Taylor expansion of the accumulated rounding error,’ *BIT*, vol. 16, no. 2, pp. 146–160, Jun. 1976, ISSN: 0006-3835, 1572-9125. DOI: 10.1007/BF01931367.
- [339] P. Werbos, ‘Beyond regression: New tools for prediction and analysis in the behavioral sciences,’ Ph.D. dissertation, Harvard University, Committee on Applied Mathematics, Harvard University, Cambridge, MA, USA, 1974.
- [340] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman and G. Hinton, ‘Back-propagation and the brain,’ *Nature Reviews Neuroscience*, vol. 21, no. 6, pp. 335–346, Jun. 2020, ISSN: 1471-0048. DOI: 10.1038/s41583-020-0277-3.
- [341] J. G. Fernández, S. Keemink and M. van Gerven, ‘Gradient-free training of recurrent neural networks using random perturbations,’ *Frontiers in Neuroscience*, vol. 18, Jul. 2024, ISSN: 1662-453X. DOI: 10.3389/fnins.2024.1439155.
- [342] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. Wiley, 1949, ISBN: 978-0-471-36727-7.
- [343] S. Löwel and W. Singer, ‘Selection of Intrinsic Horizontal Connections in the Visual Cortex by Correlated Neuronal Activity,’ *Science*, vol. 255, no. 5041, pp. 209–212, Jan. 1992, ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1372754.
- [344] M. Gupta, A. Ambikapathi and S. Ramasamy, ‘HebbNet: A Simplified Hebbian Learning Framework to do Biologically Plausible Learning,’ in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2021, pp. 3115–3119. DOI: 10.1109/ICASSP39728.2021.9414241.

- [345] D. Krotov and J. J. Hopfield, ‘Unsupervised learning by competing hidden units,’ *Proceedings of the National Academy of Sciences*, vol. 116, no. 16, pp. 7723–7731, Apr. 2019. DOI: 10.1073/pnas.1820458116.
- [346] G. Lagani, F. Falchi, C. Gennaro and G. Amato, ‘Comparing the performance of Hebbian against backpropagation learning using convolutional neural networks,’ *Neural Comput. Appl.*, vol. 34, no. 8, pp. 6503–6519, Apr. 2022, ISSN: 0941-0643. DOI: 10.1007/s00521-021-06701-4.
- [347] A. G. Baydin, B. A. Pearlmutter, A. A. Radul and J. M. Siskind, ‘Automatic differentiation in machine learning: A survey,’ *Journal of Machine Learning Research*, vol. 18, no. 153, pp. 1–43, 2018.
- [348] S. Pouyanfar, S. Sadiq, Y. Yan *et al.*, ‘A Survey on Deep Learning: Algorithms, Techniques, and Applications,’ *ACM Comput. Surv.*, vol. 51, no. 5, 92:1–92:36, Sep. 2018, ISSN: 0360-0300. DOI: 10.1145/3234150.
- [349] A. Zohrevand and Z. Imani, ‘An Empirical Study of the Performance of Different Optimizers in the Deep Neural Networks,’ in *2022 International Conference on Machine Vision and Image Processing (MVIP)*, Feb. 2022, pp. 1–5. DOI: 10.1109/MVIP53647.2022.9738743.
- [350] M. D. Zeiler, *ADADELTA: An Adaptive Learning Rate Method*, <https://arxiv.org/abs/1212.5701>, Dec. 2012.
- [351] J. Kiefer and J. Wolfowitz, ‘Stochastic Estimation of the Maximum of a Regression Function,’ *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, Sep. 1952, ISSN: 0003-4851. DOI: 10.1214/aoms/1177729392.
- [352] D. E. Rumelhart, G. E. Hinton and R. J. Williams, ‘Learning representations by back-propagating errors,’ *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, ISSN: 1476-4687. DOI: 10.1038/323533a0.
- [353] S. Ruder, *An overview of gradient descent optimization algorithms*, Jun. 2017. DOI: 10.48550/arXiv.1609.04747. arXiv: 1609.04747 [cs].
- [354] M. Andrychowicz, F. Wolski, A. Ray *et al.*, *Hindsight Experience Replay*, Feb. 2018. DOI: 10.48550/arXiv.1707.01495. arXiv: 1707.01495 [cs].
- [355] K. Adem and S. Kiliçarslan, ‘Performance Analysis of Optimization Algorithms on Stacked Autoencoder,’ in *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Oct. 2019, pp. 1–4. DOI: 10.1109/ISMSIT.2019.8932880.
- [356] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison and G. E. Dahl, *On Empirical Comparisons of Optimizers for Deep Learning*, Jun. 2020. DOI: 10.48550/arXiv.1910.05446. arXiv: 1910.05446 [cs].

Appendix A

Derivations

A.1 Derivative Of The Sigmoid Function

Let the sigmoid function $\sigma(x)$ be

$$\sigma(x) = \frac{1}{1 + \exp(-x)} = \frac{\exp(x)}{1 + \exp(x)} \quad (\text{A.1})$$

To find the derivative with respect to x , the quotient rule is applied

$$\frac{d}{dx} \frac{f(x)}{g(x)} = \frac{\frac{d}{dx} f(x) g(x) - f(x) \frac{d}{dx} g(x)}{g(x)^2} \quad (\text{A.2})$$

with $f(x) = \exp(x)$ and $g(x) = 1 + \exp(x)$. The derivatives of $f(x)$ and $g(x)$ are then

$$\frac{d}{dx} f(x) = \frac{d}{dx} g(x) = \exp(x)$$

Substituting $f(x)$, $g(x)$, $\frac{d}{dx} f(x)$, and $\frac{d}{dx} g(x)$ into Eqn. A.2 gives

$$\frac{d}{dx} \frac{f(x)}{g(x)} = \frac{\exp(x)(1 + \exp(x)) - \exp(x) \exp(x)}{(1 + \exp(x))^2}$$

This can be rearranged into

$$\frac{d}{dx} \sigma(x) = \frac{\exp(x)}{1 + \exp(x)} \left(\frac{1 + \exp(x)}{1 + \exp(x)} - \frac{\exp(x)}{1 + \exp(x)} \right)$$

and Eqn. A.1 substituted in to finally give

$$\frac{d}{dx} \sigma(x) = \sigma(x)(1 - \sigma(x)) \quad (\text{A.3})$$

A.2 Derivative Of The Tanh Function

Let the hyperbolic tangent $\tanh(x)$ be

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = \frac{\sinh(x)}{\cosh(x)} \quad (\text{A.4})$$

where the hyperbolic cosine and sine are

$$\cosh(x) = \frac{\exp(x) + \exp(-x)}{2} \quad (\text{A.5})$$

and

$$\sinh(x) = \frac{\exp(x) - \exp(-x)}{2} \quad (\text{A.6})$$

respectively. The derivatives of Eqn. A.5 and Eqn. A.6 are

$$\frac{\partial}{\partial x} \cosh(x) = \frac{\exp(x) - \exp(-x)}{2} = \sinh(x) \quad (\text{A.7})$$

$$\frac{\partial}{\partial x} \sinh(x) = \frac{\exp(x) + \exp(-x)}{2} = \cosh(x) \quad (\text{A.8})$$

Furthermore, the derivative of $1/\cosh(x)$ using Eqn. A.7 is found using the chain rule

$$\frac{d}{dx}(f(g(x))) = \frac{df(g(x))}{dg(x)} \frac{dg(x)}{dx}$$

such that

$$\frac{\partial}{\partial x} \frac{1}{\cosh(x)} = \frac{\partial x}{\partial(\cosh(x))}^{-1} = -\frac{\sinh(x)}{\cosh^2(x)} \quad (\text{A.9})$$

To find the derivative of Eqn. A.4 the product rule is applied

$$\frac{d}{dx}(f(x)g(x)) = \frac{d}{dx}f(x)g(x) + f(x)\frac{d}{dx}g(x)$$

with $f(x) = \sinh(x)$ and $g(x) = \cosh(x)$ to get

$$\frac{d}{dx} \tanh(x) = \sinh(x) \frac{d}{dx} \frac{1}{\cosh(x)} + \frac{1}{\cosh(x)} \frac{d}{dx} \sinh(x) \quad (\text{A.10})$$

Substituting in Eqn. A.9 and Eqn. A.8 gives

$$\frac{d}{dx} \tanh(x) = -\frac{\sinh^2(x)}{\cosh^2(x)} + \frac{\cosh(x)}{\cosh(x)}$$

Finally, Eqn. A.2 can be simplified by substituting Eqn. A.4 to get

$$\frac{d}{dx} \tanh(x) = 1 - \tanh^2(x) \quad (\text{A.11})$$

Appendix B

Values for 2D Cubature 2T Degree 13

Table B.1: Triangle cubature rule of degree 13 with 37 points from Berntsen *et al.* [279], Cools *et al.* [266] and Cools *et al.* [281].

x	y	w
0.333 333 333	0.333 333 333	0.025 869 883
0.024 862 169	0.024 862 169	0.004 003 900
0.024 862 169	0.950 275 663	0.004 003 900
0.950 275 663	0.024 862 169	0.004 003 900
0.414 192 543	0.414 192 543	0.023 434 449
0.414 192 543	0.171 614 915	0.023 434 449
0.171 614 915	0.414 192 543	0.023 434 449
0.230 293 878	0.230 293 878	0.023 295 470
0.230 293 878	0.539 412 244	0.023 295 470
0.539 412 244	0.230 293 878	0.023 295 470
0.113 919 982	0.113 919 982	0.015 508 472
0.113 919 982	0.772 160 037	0.015 508 472
0.772 160 037	0.113 919 982	0.015 508 472
0.495 457 300	0.495 457 300	0.005 395 806
0.495 457 300	0.009 085 400	0.005 395 806
0.009 085 400	0.495 457 300	0.005 395 806
0.468 861 355	0.468 861 355	0.016 097 767
0.468 861 355	0.062 277 290	0.016 097 767
0.062 277 290	0.468 861 355	0.016 097 767
0.022 076 290	0.851 306 504	0.007 722 917
0.022 076 290	0.126 617 206	0.007 722 917
0.851 306 504	0.022 076 290	0.007 722 917
0.851 306 504	0.126 617 206	0.007 722 917
0.126 617 206	0.022 076 290	0.007 722 917
0.126 617 206	0.851 306 504	0.007 722 917
0.018 620 523	0.689 441 971	0.008 911 495
0.018 620 523	0.291 937 506	0.008 911 495
0.689 441 971	0.018 620 523	0.008 911 495
0.689 441 971	0.291 937 506	0.008 911 495
0.291 937 506	0.018 620 523	0.008 911 495
0.291 937 506	0.689 441 971	0.008 911 495
0.096 506 481	0.635 867 859	0.018 519 342
0.096 506 481	0.267 625 659	0.018 519 342
0.635 867 859	0.096 506 481	0.018 519 342
0.635 867 859	0.267 625 659	0.018 519 342
0.267 625 659	0.096 506 481	0.018 519 342
0.267 625 659	0.635 867 859	0.018 519 342

Appendix C

Genetic Algorithm Tuning of the Linear-Quadratic Regulator Matrices

To tune the weights of the Q and R matrices for Eqn. 8.5, the GA [159]–[162] method is used. GA is a sampling-based method that is efficient at optimising unconstrained problems with well defined crossover and mutation schemes. In this scenario the problem is defined as

$$q^*, r^* = \arg \min_{q \in \mathbb{R}^{12}, r \in \mathbb{R}^4} J(q, r), \quad (\text{C.1})$$

where states q and r being optimised are reshaped to give $Q = \text{diag}(q)$ and $R = \text{diag}(r)$. The objective function is defined as

$$J(q, r) = \sum_{j=0}^{12} w_j \|g(\text{diag}(q), \text{diag}(q), x_0, x_T)_{:,j} - x_{T,j}\|_2, \quad (\text{C.2})$$

$$\mathbf{w} = \begin{bmatrix} 0 & 0 & 25 & 1 & 1 & 1 & 25 & 25 & 0 & 1 & 1 & 1 \end{bmatrix}, \quad (\text{C.3})$$

$$\mathbf{x}_T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0.3 & 2.5 & 2.5 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (\text{C.4})$$

$$\mathbf{x}_0 = \begin{bmatrix} 1 & 1 & 0.1 & 0 & 0 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (\text{C.5})$$

where $g(Q, R, \mathbf{x}_0, \mathbf{x}_T)$ is simulation of the system defined in Ch. 8 with the LQR matrices Q and R . Within this simulation the system starts from initial conditions $\mathbf{x}_0$ and receives a step input of target state $\mathbf{x}_T$. The state indices are those from Eqn. 8.4. The notation $g(\cdot)_{:,j}$ denotes all values of the state indexed by j . The optimal

LQR values were then found to be

$$Q^* = \text{diag}(q^*) = \text{diag} \left(\begin{bmatrix} 1.374 \times 10^{-4} \\ 1.492 \times 10^{-4} \\ 1.469 \times 10^{-4} \\ 7.158 \times 10^{-5} \\ 1.285 \times 10^{-4} \\ 1.457 \times 10^{-6} \\ 3.287 \times 10^{-5} \\ 4.038 \times 10^{-5} \\ 1.634 \times 10^{-4} \\ 6.638 \times 10^{-5} \\ 1.077 \times 10^{-4} \\ 6.371 \times 10^{-5} \end{bmatrix}^T \right), \quad (\text{C.6})$$

and

$$R^* = \text{diag}(r^*) = \text{diag} \left(\begin{bmatrix} 1.336 \times 10^{-1} \\ 2.499 \times 10^{-1} \\ 3.541 \times 10^1 \\ 4.855 \times 10^{-5} \end{bmatrix}^T \right). \quad (\text{C.7})$$

The system velocity-heading control input, shown in Fig. C.1, is drastically improved in terms of response time, when compared to a manually tuned and random weighted matrix weights. The setpoint target is defined in Eqn. C.4, with a constant velocity target of 2.5 ms in the horizontal directions, a yaw target of 0.3 rad, and a slight altitude change from 0.1 m to 0 m. Importantly, the remaining states remain at stable, and at zero.

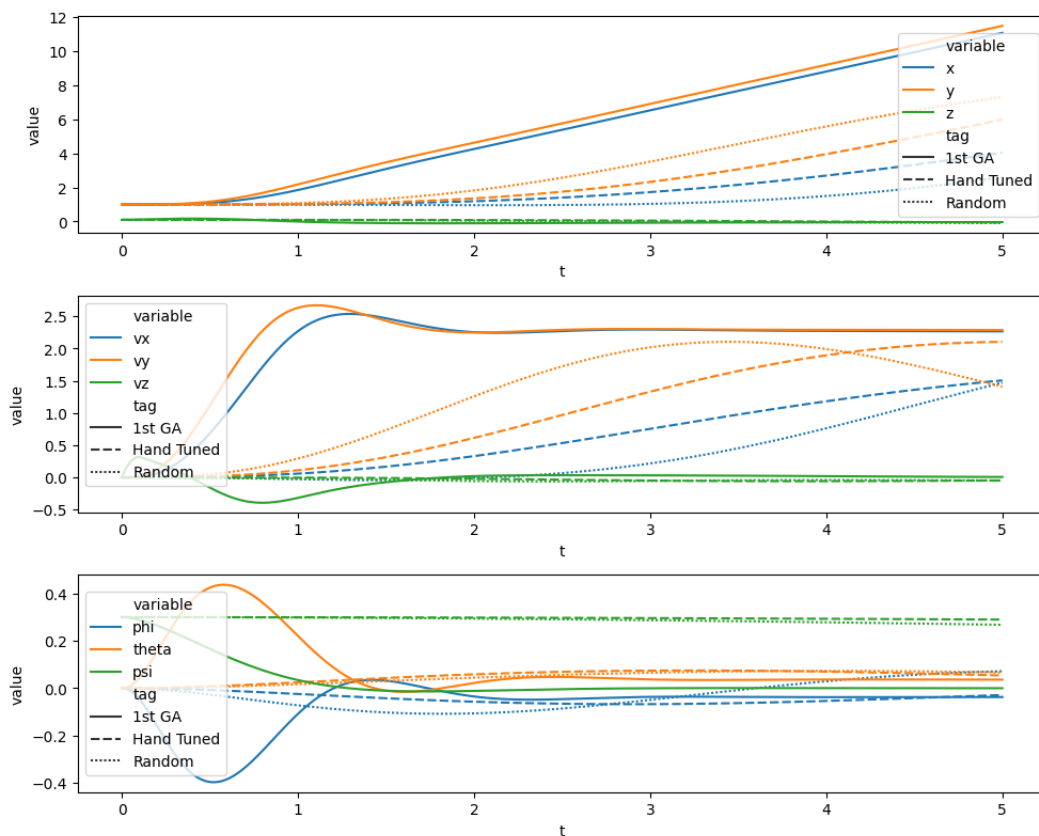


Figure C.1: System velocity-altitude-heading control input response plotted for the three main state groupings — position, velocity, and euler angles — for the random (dotted), hand-tuned (dashed), and GA-tuned (solid) cases.

Appendix D

Additional Results From Ch. 8

D.1 Short, flat

Additional figures for *short-flat* the configuration in Sect. 8.7 are given in this section.

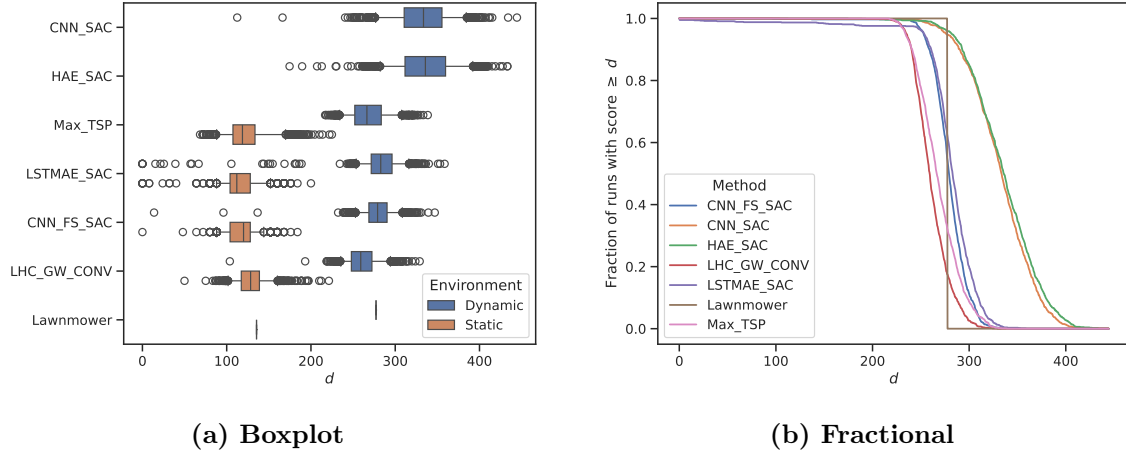


Figure D.1: Final search path length for each algorithm using the *short-flat* parameter setup.

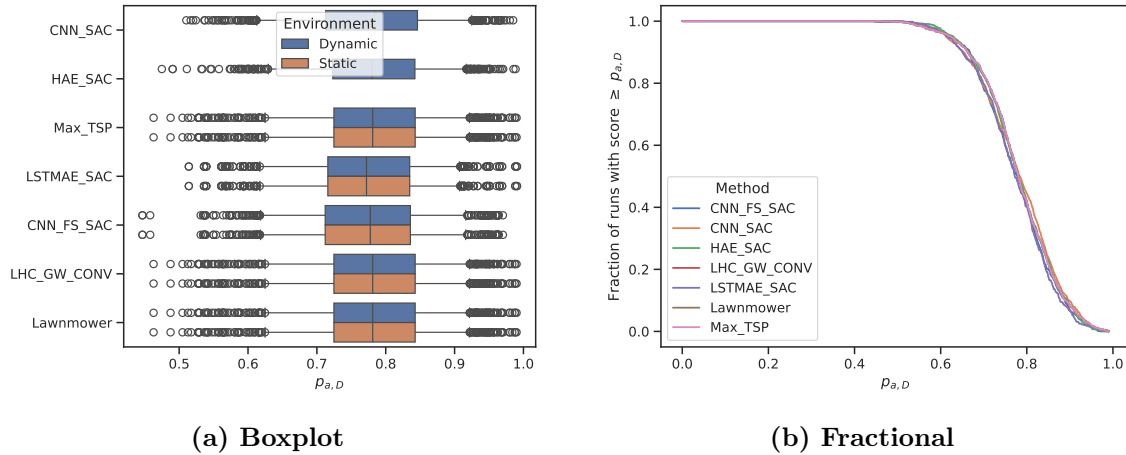


Figure D.2: Final area probability for each algorithm using the *short-flat* parameter setup.

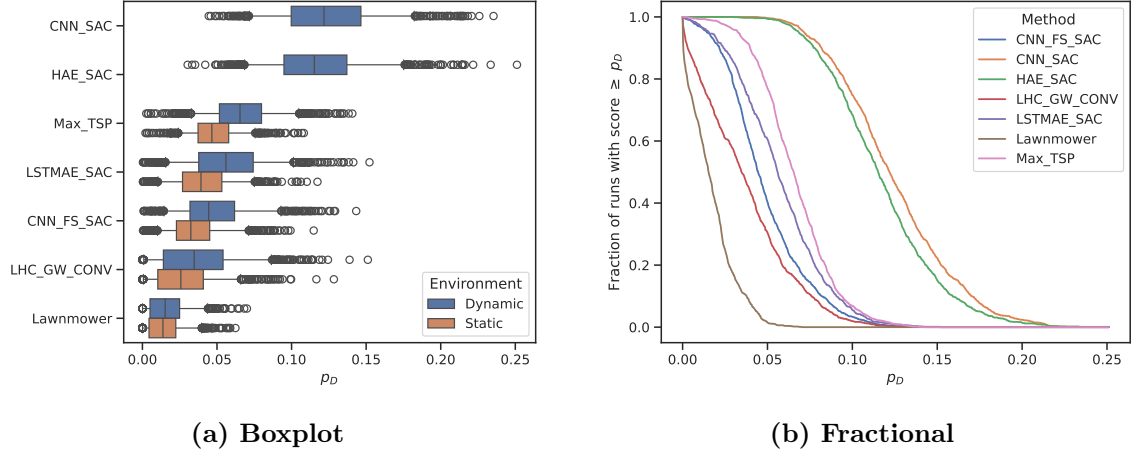


Figure D.3: Final search path accumulated probability per unit length for each algorithm using the *short-flat* parameter setup.

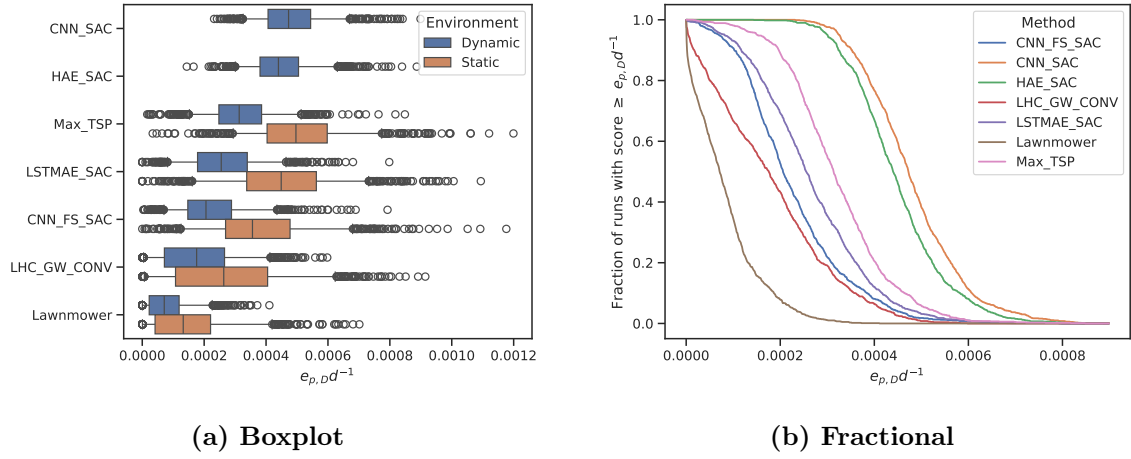


Figure D.4: Final search path probability efficiency per unit length for each algorithm using the *short-flat* parameter setup.

D.2 Long, sloped

Additional figures for *long-sloped* the configuration in Sect. 8.7 are given in this section.

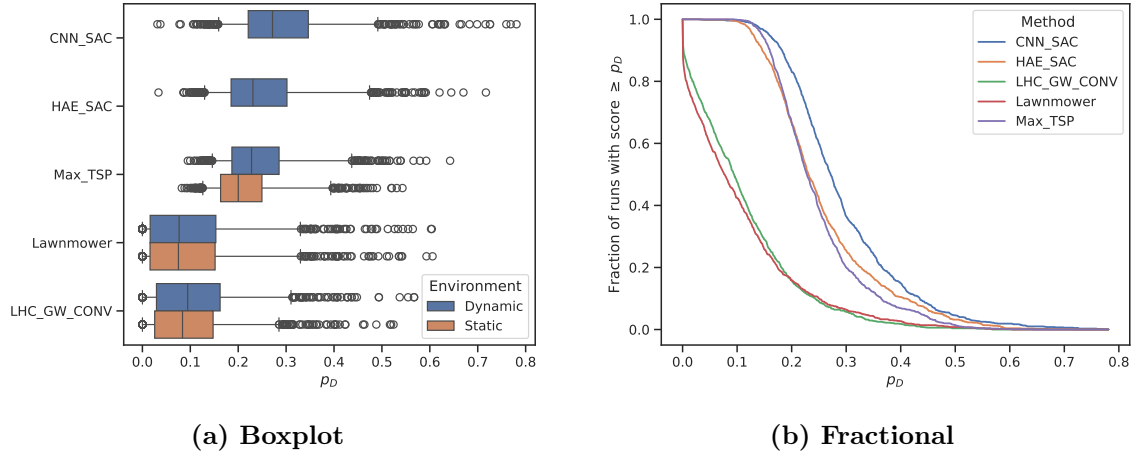


Figure D.5: Final search path accumulated probability per unit length for each algorithm using the *long-sloped* parameter setup.

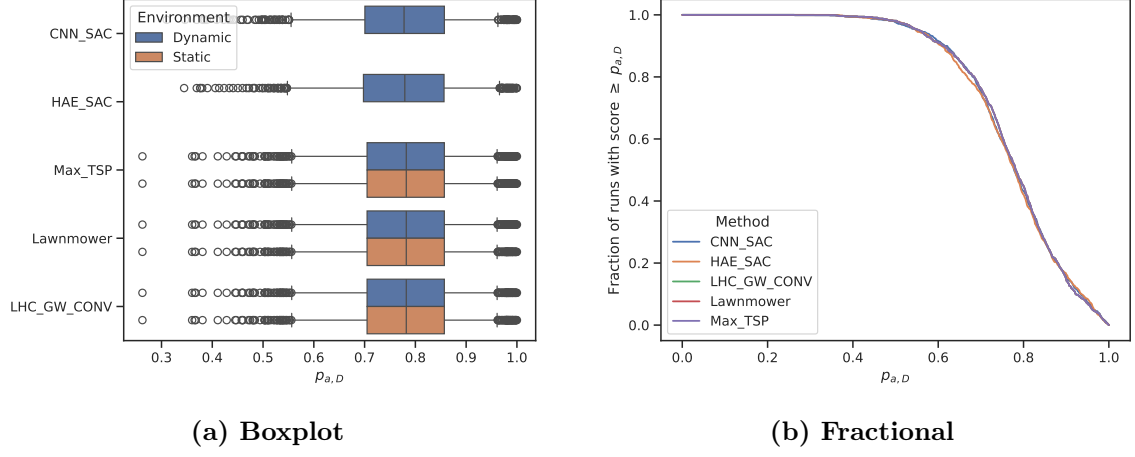


Figure D.6: Final area probability for each algorithm using the *long-sloped* parameter setup.

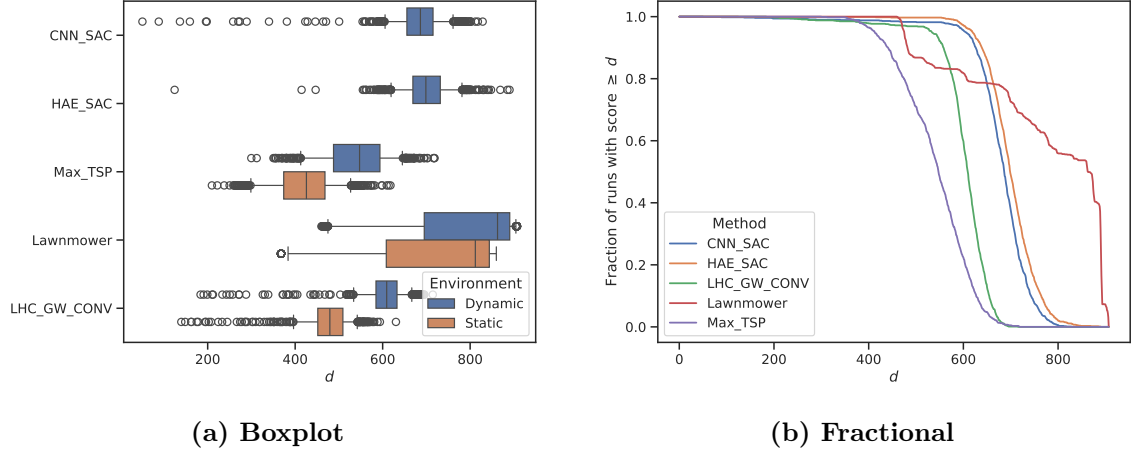


Figure D.7: Final search path length for each algorithm using the *long-sloped* parameter setup.

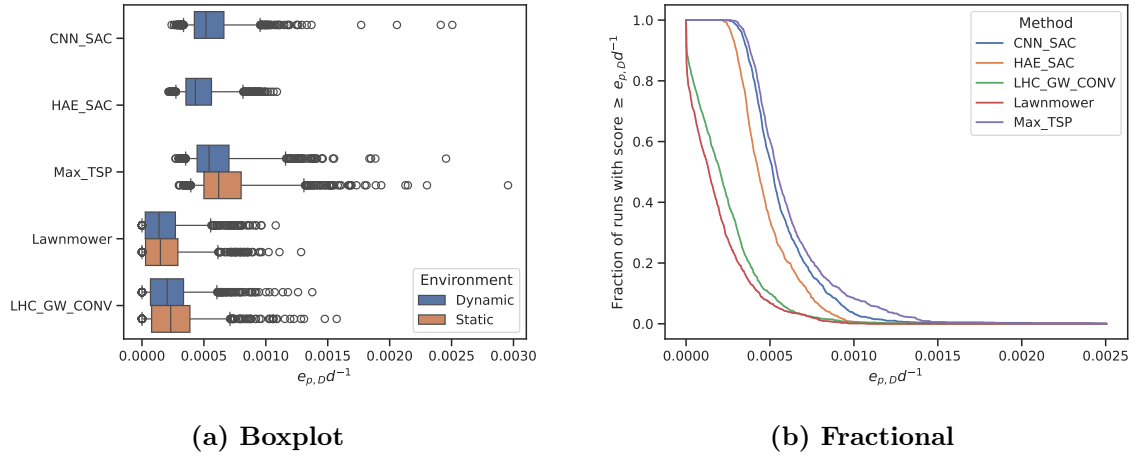


Figure D.8: Final search path probability efficiency per unit length for each algorithm using the *long-sloped* parameter setup.

Appendix E

Further Machine Learning Technical Background

E.1 Backpropagation

Learning the optimal policy is the essence of ML as without it there is no learning. At the core of ML lies the backpropagation of error [338], [339]; simply known as backpropagation. Whilst other methods exist to update the policy parameters like perturbation or Hebbian learning [340], the power of backpropagation comes from the applications of auto-differentiation libraries and dynamic programming.

Backpropagation, by itself, is not very useful for ML. The true power comes from the combination of backpropagation and optimisation algorithms. This group of methods are concerned with updating the model in the best way to minimise the loss function as quickly, in terms of epochs, as possible.

Alternatives to backpropagation are discussed in Sect. E.1.1 and Sect. E.1.2. The backpropagation method is described in Sect. E.1.3 and two popular optimisation methods for ML are outlined in Sect. E.2

E.1.1 Perturbation Learning

Perturbation learning is a correlative method that estimates the impact of a parameter by applying noise to it and re-evaluating the network. This is measured as a global scalar reinforcement signal which signals if a perturbation is accepted or rejected [340]. This approach is very sample inefficient, growing in $\mathcal{O}(n^2)$ complexity making it unfeasible for general deep learning where parameter counts can easily grow into the millions. Whilst unsuitable for most applications, perturbation learning has shown promising results in niche uses like training recurrent networks [341]. This application is promising because recurrent networks are renowned to suffer from the vanishing gradient problem during the typical backpropagation-through-time leading to slow convergence. However, even in this application backpropagation is more performant.

E.1.2 Hebbian Learning

Hebbian learning works on the Hebb’s postulate [342], which is summarised as “neurons that fire together, wire together” by Löwel *et al.* [343]. Applying this to a neural network requires strengthening synaptic connections between two neurons if the pre- and post-synaptic neurons are fired. This however does not use any notion of error or feedback and this learning rule is thus unsupervised. The Hebbian update is given by

$$\Delta w = \eta xz, \quad (\text{E.1})$$

where x and z are the pre- and post-synaptic neuron values respectively, and η is the learning rate. This formulation uses the perceptron model of a neuron (introduced in Sect. 3.1.1) and leads to unstable training. A modified Hebbian update is used in HebbNet [344] and BIO [345] which perform comparably to backpropagation for small networks. Whilst Hebbian learning is applicable to certain scenarios, Lagani *et al.* [346] found that it is unsuitable for layered networks. The limitation stems from Hebbian learning’s inability to coordinate multilayer credit assignment — updates in early layers remain decoupled from final output errors without additional feedback mechanisms.

E.1.3 The Backpropagation Method

Backpropagation is a direct application of the generalised chain rule [340];

$$\frac{\partial z}{\partial u} = \sum_{i=0}^k \frac{\partial v_i}{\partial u} \frac{\partial z}{\partial v_i}. \quad (\text{E.2})$$

This is a powerful formulation, which enables calculation of the gradient of a parameter u with respect to a signal z from the downstream parameters v . This gradient-based approach achieves a $\mathcal{O}(n)$ computational complexity, which matches the forward pass making this approach suitable for large networks from a computational perspective.

Whilst the derivative can be manually evaluated or the Hessian calculated using expensive methods, automatic differentiation has been the major breakthrough [347] that has enabled backpropagation to become the de facto method in deep learning [348]. This leverages dynamic programming during the backward-pass through the network to recursively propagate the gradients through the network; hence *backpropagation*.

A simple example is presented in Sect. E.1.5 to highlight the power of backpropagation. This example highlights that:

- backpropagation only requires a single scalar output from the loss function;
- all elements of the network must be differentiable to propagate the gradient;
- the results of all values from the forward pass must be stored;

- the derivative of a parameter early in the network requires the derivative of all subsequent related parameters; and
- the importance of activation function selection.

E.1.4 The Importance Of Activation Functions

Activation functions are a critical aspect of deep learning architectures. They typically apply the transformation $f : (-\infty, \infty) \mapsto [-1, 1]$ which prevents large parameters from making the forward- and backward-pass unstable. Automatic differentiation works on the principle of isolating the gradient calculation into sub-modules and it is therefore imperative for the activation function to be carefully selected. Whilst one function might be computationally cheap to calculate, its derivative might not be. Three of the most popular activation functions are the:

- $\text{sigmoid}(x) = \frac{\exp(x)}{1+\exp(x)} \rightarrow \frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$;
- $\text{ReLU}(x) = \begin{cases} x, & x > 0 \\ 0, & \text{else} \end{cases} \rightarrow \frac{d\text{ReLU}(x)}{dx} = \begin{cases} 1, & x > 0 \\ 0, & \text{else} \end{cases}$; and
- $\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)} \rightarrow \frac{d\tanh(x)}{dx} = 1 - \tanh^2(x)$.

The derivations of the derivatives of the sigmoid and tanh functions can be found in Sect. A.1 and Sect. A.2 respectively whilst the ReLU derivative is self-explanatory. It is clear that the derivatives of these functions are cheap to calculate due to the usage of dynamic programming, which stores the forward-pass values.

E.1.5 Backpropagation Example

Applying backpropagation to a simple supervised learning example sheds light on how this method is used and how powerful it is.

The dataset has input x and corresponding label y . The network consists of two neurons with sigmoid activation functions such that

$$\begin{aligned} y_1 &= \sigma(W_0x + b_0), \\ y_2 &= \sigma(W_1y_1 + b_1), \\ \hat{y} &= y_2, \end{aligned} \tag{E.3}$$

with the mean-square error loss function

$$\mathcal{L}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^n (y_i - \hat{y}_i)^2, \tag{E.4}$$

which, when $n = 1$, simplifies to

$$\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2. \tag{E.5}$$

The required derivatives are then $\frac{\partial \mathcal{L}}{\partial W_0}$, $\frac{\partial \mathcal{L}}{\partial W_1}$, $\frac{\partial \mathcal{L}}{\partial b_0}$, and $\frac{\partial \mathcal{L}}{\partial b_1}$ in order to update the parameters using a gradient-based method.

The derivative of the loss with respect to y_2 is then

$$\frac{\partial \mathcal{L}}{\partial y_2} = -2(y - \hat{y}). \quad (\text{E.6})$$

It follows that the derivative of the loss with respect to W_1 , using Eqn. A.3, is given by

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial W_1} &= \frac{\partial \mathcal{L}}{\partial y_2} \frac{\partial y_2}{\partial W_1} \\ &= -2(y - \hat{y})y_1\sigma(W_1y_1 + b_1)(1 - \sigma(W_1y_1 + b_1)) \\ &= -2(y - \hat{y})y_1y_2(1 - y_2). \end{aligned} \quad (\text{E.7})$$

However, Eqn. E.6 is already known because of the application of dynamic programming. Therefore, Eqn. E.7 is simplified to

$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial \mathcal{L}}{\partial y_2} y_1y_2(1 - y_2). \quad (\text{E.8})$$

It then follows that

$$\frac{\partial \mathcal{L}}{\partial b_1} = \frac{\partial \mathcal{L}}{\partial y_2} y_1y_2(1 - y_2), \quad (\text{E.9})$$

$$\frac{\partial \mathcal{L}}{\partial y_1} = \frac{\partial \mathcal{L}}{\partial y_2} W_1y_2(1 - y_2), \quad (\text{E.10})$$

$$\frac{\partial \mathcal{L}}{\partial W_0} = \frac{\partial \mathcal{L}}{\partial y_1} xy_1(1 - y_1), \quad (\text{E.11})$$

$$\frac{\partial \mathcal{L}}{\partial b_0} = \frac{\partial \mathcal{L}}{\partial y_1} y_1(1 - y_1). \quad (\text{E.12})$$

E.2 Optimisation Algorithms

Optimisation algorithms for ML are intended to minimise the loss function in as few epochs as possible without converging on a local minima. Many exist such as Ad-aGrad [349] or AdaDelta [350], but SGD [351] and Adam [248] remain some of the most prevalent within the literature. This is due to their relative simplicity and robustness meaning that with minimal parameter tuning a good solution can be found.

E.2.1 Stochastic Gradient Descent

Consider the following problem that is trying to minimise the loss function

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=0}^N \mathcal{L}(x_i, y_i, \theta), \quad (\text{E.13})$$

where $\mathcal{L}(x_i, y_i, \theta)$ corresponds to the i th observation in the dataset with values x_i and label y_i . Vanilla gradient descent minimises Eqn. E.13 through the iterative process

$$\theta \leftarrow \theta - \eta \frac{1}{N} \sum_{i=0}^N \nabla_{\theta} \mathcal{L}(x_i, y_i, \theta). \quad (\text{E.14})$$

SGD [351] approximates the gradient at a single sample and updates the parameters through

$$\theta \leftarrow \theta - \eta \nabla_{\theta} \mathcal{L}(x_i, y_i, \theta). \quad (\text{E.15})$$

SGD updates the parameters θ as it traverses the dataset, and can be run multiple times. If it is run more than once, the dataset can be shuffled.

Algorithm E.1: Stochastic gradient descent.

Data: Parameters θ , values $\mathbf{x}$, labels $\mathbf{y}$, learning rate η

Result: Updated parameters θ_t

```

1  $\theta_0 \leftarrow \theta$  ;
2  $t \leftarrow 0$  ;
3 while  $\theta_t$  not converged do
4    $t \leftarrow t + 1$  ;
5    $i \leftarrow \text{mod}(t, |\mathbf{x}|)$  ;
6   if  $i = 0$  then
7     | Shuffle  $\mathbf{x}, \mathbf{y}$  whilst ensuring they remain aligned;
8   end
9    $\theta_t \leftarrow \theta_{t-1} - \eta \cdot \nabla_{\theta} \mathcal{L}(x_i, y_i, \theta_{t-1})/M$  ;
10 end
```

Algorithm E.2: Minibatch stochastic gradient descent.

Data: Parameters θ , values $\mathbf{x}$, labels $\mathbf{y}$, learning rate η , and minibatch size M

Result: Updated parameters θ_t

```

1  $\theta_0 \leftarrow \theta$  ;
2  $t \leftarrow 0$  ;
3 while  $\theta_t$  not converged do
4    $t \leftarrow t + 1$  ;
5   Shuffle  $\mathbf{x}, \mathbf{y}$  whilst ensuring they remain aligned;
6    $\hat{\nabla}_{\theta} \leftarrow 0$ ;
7   for  $i \in [1, \dots, M]$  do
8     |  $\hat{\nabla}_{\theta} \leftarrow \hat{\nabla}_{\theta} + \nabla_{\theta} \mathcal{L}(x_i, y_i, \theta_{t-1})$ ;
9   end
10   $\theta_t \leftarrow \theta_{t-1} - \eta \cdot \hat{\nabla}_{\theta}/M$  ;
11 end
```

Rumelhart *et al.* [352] introduced the momentum method, stochastic gradient descent with momentum (SGD-M), which changes the update to

$$\begin{aligned} \Delta\theta &\leftarrow \alpha \cdot \Delta\theta - \eta \cdot \nabla_{\theta} \mathcal{L}(x_i, y_i, \theta_{t-1}), \\ \theta &\leftarrow \theta + \Delta\theta, \end{aligned} \quad (\text{E.16})$$

where $\alpha \in [0, 1)$ controls the impact of the momentum. The algorithm that implements minibatch SGD-M can be seen in Alg. E.3. It is noteworthy that with $\alpha = 0$ and $M = 1$, this simplifies into online SGD as seen in Alg. E.1. Similarly, with $\alpha = 0$ and $M > 1$, this simplifies into minibatch SGD.

Algorithm E.3: Minibatch stochastic gradient descent with momentum.

Data: Parameters θ , values $\mathbf{x}$, labels $\mathbf{y}$, learning rate η , momentum α , and minibatch size M

Result: Updated parameters θ_t

```

1  $\theta_0 \leftarrow \theta$  ;
2  $t \leftarrow 0$  ;
3  $\Delta\theta_t \leftarrow 0$  ;
4 while  $\theta_t$  not converged do
5    $t \leftarrow t + 1$  ;
6   Shuffle  $\mathbf{x}, \mathbf{y}$  whilst ensuring they remain aligned;
7    $\hat{\nabla}_\theta \leftarrow 0$ ;
8   for  $i \in [1, \dots, M]$  do
9      $\hat{\nabla}_\theta \leftarrow \hat{\nabla}_\theta + \nabla_\theta \mathcal{L}(x_i, y_i, \theta_{t-1})$ ;
10  end
11   $\Delta\theta_t \leftarrow \alpha \cdot \Delta\theta_{t-1} - \eta \cdot \hat{\nabla}_\theta / M$  ;
12   $\theta_t \leftarrow \theta_{t-1} - \Delta\theta_t$  ;
13 end
```

E.2.2 Adam

Adam [248] is a stochastic optimiser that is widely used within machine learning. Like SGD-M in Alg. E.3, Adam compute the adaptive learning rate for each parameter. However, Adam computes the exponentially decaying average of past squared gradients v_t and an exponentially decaying of past gradients m_t . The latter of the two is more akin to the momentum from SGD-M.

The momentum calculation of Adam is defined as follows:

$$\begin{aligned} m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla_\theta \mathcal{L}(x_i, y_i, \theta_{t-1}), \\ v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot (\nabla_\theta \mathcal{L}(x_i, y_i, \theta_{t-1}))^2. \end{aligned} \tag{E.17}$$

Kingma *et al.* [248] noticed a bias towards zero during the initial steps and counteracted this with the moment estimates

$$\begin{aligned} \hat{m}_t &= m_t / (1 - \beta_1^t), \\ \hat{v}_t &= v_t / (1 - \beta_2^t). \end{aligned} \tag{E.18}$$

The parameters are then finally updated using

$$\theta_t = \theta_{t-1} - \eta \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon), \tag{E.19}$$

where default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$ are suggested. This can be seen in algorithmic form in Alg. E.4.

Algorithm E.4: Adaptive momentum estimation.

Data: Parameters θ , values $\mathbf{x}$, labels $\mathbf{y}$, learning rate η , exponential decay rates for moment estimates $\beta_1, \beta_2 \in [0, 1)$. ϵ is a small number, such as 1×10^{-8} , to avoid division by zero. Kingma *et al.* [248] suggest the default values of $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. All operations are element-wise.

Result: Updated parameters θ_t

```

1  $\theta_0 \leftarrow \theta$  ;
2  $m_0 \leftarrow 0$  ;
3  $v_0 \leftarrow 0$  ;
4  $t \leftarrow 0$  ;
5 while  $\theta'$  not converged do
6    $t \leftarrow t + 1$  ;
7    $i \leftarrow \text{mod}(t, |\mathbf{x}|)$  ;
8    $\hat{\nabla}_\theta \leftarrow \nabla_\theta \mathcal{L}(x_j, y_j, \theta_{t-1})$  ;
9    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \hat{\nabla}_\theta$  ;
10   $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_r) \cdot (\hat{\nabla}_\theta \odot \hat{\nabla}_\theta)$  ;
11   $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$  ;
12   $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$  ;
13   $\theta_t \leftarrow \theta_{t-1} - \eta \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$  ;
14 end
```

E.2.3 Comparing Stochastic Gradient Descent and Adam

Zohrevand *et al.* [349] shows that Adam is the best optimiser for CNN applications in two handwriting recognition benchmarks. Ruder [353] suggest that the adaptive learning rate for each parameter make it robust to sub-optimal hyperparameters, which is particularly important where tuning resources are limited. Similarly, the individual learning rate make Adam effective for changing reward landscapes encountered in problems such as RL as shown by Andrychowicz *et al.* [354].

This, however, changes in the context of AEs with Adem *et al.* [355] showing that SGD outperforms Adam when training a stacked AE. SGD further outperforms ADADELTA [350], which is the optimiser used in the seminal work by Cho *et al.* [208], which introduced the GRU. Choi *et al.* [356] has similar findings with SGD and SGD-M performing better than Adam.

It is therefore evident that an optimal optimiser for all ML applications does not exist. One must carefully analyse the application and conduct empirical studies for the problem at hand to find the best optimiser. Whilst there are many options, SGD, SGD-M, and Adam are some of the most effective across a spread of domains.