8-4-2021

Proximal Policy Optimization for Radiation Source Search

Philippe Erol Proctor
Portland State University

Proximal Policy Optimization for Radiation Source Search

by

Philippe Erol Proctor

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Electrical and Computer Engineering

Thesis Committee:
Christof Teuscher, Chair
John Lipor
James McNames

Portland State University
2021

Abstract

Rapid localization and search for lost nuclear sources in a given area of interest is an important task for the safety of society and the reduction of human harm. Detection, localization and identification are based upon the measured gamma radiation spectrum from a radiation detector. The nonlinear relationship of electromagnetic wave propagation paired with the probabilistic nature of gamma ray emission and background radiation from the environment leads to ambiguity in the estimation of a source's location. In the case of a single mobile detector, there are numerous challenges to overcome such as weak source activity, multiple sources, or the presence of obstructions, i.e. a non-convex environment. Detectors deployed to smaller autonomous systems such as drones or robots have smaller surface area and volume resulting in worse counting statistics per dwell time. Additionally, search algorithms need to be efficient and generalizable to operate across a variety of scenarios.

The motivation of this work is to investigate the sequential decision making capability of *deep reinforcement learning* (DRL) in the nuclear source search context. We focus on a branch of DRL known as stochastic, model-free, on-policy gradients that learns strictly through interaction with an environment to develop a useful policy for a specified goal. A novel neural network architecture (RAD-A2C) based on the *actor critic* (A2C) framework that uses a gated recurrent unit (GRU) for action selection and a *particle filter gated recurrent unit* (PFGRU) for localization is proposed.

Performance is studied in randomized 22×22 m convex and non-convex simulated environments across a range of *signal-to-noise ratio* (SNR)s for a single detector and

single source. The RAD-A2C performance is compared to both an information-driven controller that uses a *bootstrap particle filter* (BPF) and to a *gradient search* (GS) algorithm. We find that the RAD-A2C has comparable performance to the information-driven controller across SNR in a convex environment and at lower computational complexity per action. The RAD-A2C far outperforms the GS algorithm in the non-convex environment with greater than 95% median completion rate.

To all who share their time for the benefit of others.

Acknowledgements

First, I would like to thank my mother Eda Tuncel, and grandmother, Sevim Tuncel for their support in the completion of this thesis. I would also like to thank my father, Michael Proctor, for all of his efforts in discussion of ideas, reviewing drafts, and watching practice presentations. I would also like to thank my accademic advisor Christof Teuscher for all of his help both technically and professionally. His influence and guidance on this research was immeasurable, and I certainly would not have been able to complete this thesis without his guidance and expertise.

I would also like to thank the staff and faculty of PSU's Electrical Engineering Department and Math Department. Their passion for both the material and the students was evident in every course I took. In particular I would like to thank John Lipor, James McNames, Mark Martin, Christof Teuscher, Dacian Daescu, and Ted Willke.

I would specifically like to acknowledge James McNames and John Lipor for setting high expectations of students while also ensuring sufficient guidance. This is the optimal means for personal growth and development within these complex topics that both professors are so well versed in.

I would also like to thank my peers in PSU's ECE program. I feel fortunate to have had the opportunity to learn and work alongside committed and passionate individuals. Specifically I would like to thank Bruno Duregon, Jens Evans, Merlin Carson, Christopher Neighbor, Phillip Kearns, Umair Khan, Hoang Nguyen, Wubin Sheng, Kirk Jungles, Cody Henderson, Alex Higgins, Zachery Stamler,

I would like to thank the faculty and students at University of New Mexico who we

collaborated with throughout the duration of this thesis, Marek Osiński, Adam Hecht, Nathan Withers, Payman Zarkesh-Ha, Mark Wetzel, and Jorge Iván Canales Verdial,

A few researchers from national labs provided valuable feedback in defining the scope of my project. I would like to acknowledge and thank, Reynold Cooper, Tenzing Joshi, and Jason Hite.

Finally, I would like to thank my partner Allie, who supported and motivated me during the many moments of uncertainty on this journey. She listened to my doubts and struggles, proffered sage advice on overcoming stressful challenges, and kept me from becoming a uni-dimensional thesis machine. In short, I am a better person because of her presence in my life.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Problem Definition

1.1.1 Radiation Source Search

The advancement of nuclear technology has brought the positive of energy production and medical diagnosis to society, but also the risks associated with exposure to radiation [1]. Radioactive materials might be used in explosives for dirty bombs which could cause catastrophic damage to the public. Effective detection when to these types of materials are released in the environment is of the utmost importance and measures need to be in place to rapidly locate a source of radiation in an exposure event to limit human harm [2].

Detection, localization, and identification are based upon the measured gamma radiation spectrum from a radiation detector. Radioactive sources decay at a certain rate called the activity, often measured in disintegrations per second or Becquerels [bq]. Each disintegration has a probability of triggering a gamma photon release via ionization. Localization methods rely upon the intensity [cps] of the gamma photon radiation measured by scintillation detectors composed of materials such as sodium iodide (NaI) [3]. The number of counts per second recorded by a detector is related to the total photons emitted per second through a scaling factor determined by detector characteristics. It is common to approximate each detector measurement as being drawn from a Poisson distribution because the success probability of each count is small and constant [3]. The inverse square relationship, $\frac{1}{r^2}$, is a useful approximation to

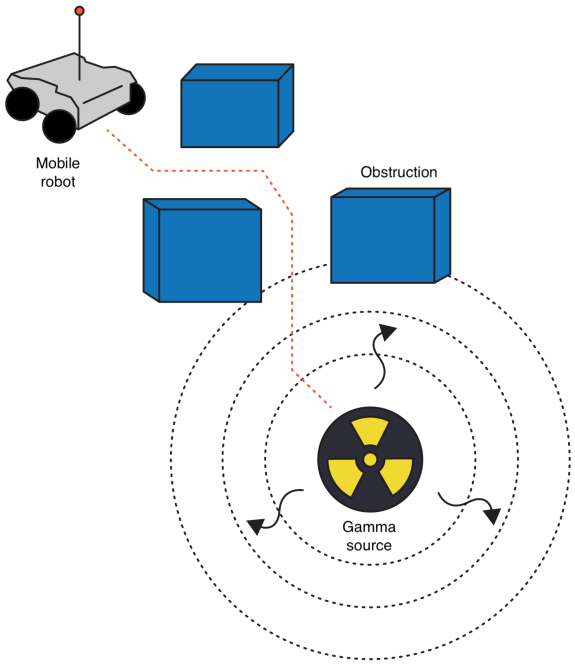


Figure 1.1: Illustration of an autonomous mobile robot operating in a non-convex environment. The unshielded gamma source emits gamma radiation isotropically. Obstructions (blue cubes) attenuate the gamma radiation signal and block the robot's path.

describe the measured intensity of the radiation as a function of the distance between the detector and source. This nonlinear relationship paired with the probabilistic nature of gamma ray emission and background radiation from the environment leads to ambiguity in the estimation of a source's location. Thus, a successful source search requires multiple independent measurements of intensity with varying distances.

In the case of a single mobile detector, there are numerous challenges to overcome. Detectors deployed to smaller autonomous systems such as drones or robots have a smaller surface area and volume resulting in worse counting statistics per dwell time. Common terrestrial materials such as soil and granite contain *naturally occurring radioactive materials* (NORM) that can contribute to a spatially varying background rate [3]. Far distances, shielding with materials such as lead, and the presence of obstructions, i.e. a non-convex environment, can significantly attenuate or block the signal from a radioactive source. Further challenges arise with multiple or weak

sources. Given the high variation in these variables, the development of a generalizable algorithm with minimal priors becomes quite difficult. Additionally, algorithms for localization and search need to be computationally efficient due to energy and time constraints. Figure 1.1 shows an example illustration of a mobile robot performing active nuclear source search in a non-convex environment.

1.1.2 Machine Learning (ML)

ML is broadly concerned with the paradigm of computers learning how to complete tasks through data. *Reinforcement learning* (RL) is a subset of ML focused on developing a control policy that maximizes cumulative reward in an environment. *Deep learning* (DL) is another subset of ML with an emphasis on learning a function of interest using data. A key difference between RL and other subsets of ML is that learning is dependent on the data that is gathered by the policy thereby directly impacting future learning. The intersection of RL and DL has resulted in a powerful framework called *Deep reinforcement learning* (DRL). DRL uses deep neural networks to learn a control policy and approximate state values through trial and error learning in an environment. While training of these networks is computationally intensive, once the weights are learned, inference (the application of a trained ML model) can be performed at lower computational cost. In this thesis, we investigate a branch of DRL known as model-free stochastic on-policy gradients and assess its performance in the task of control in the radiation source search domain.

DRL has far surpassed human expertise in a myriad of other tasks, for example, the board game Go, which has a state space of 10^{174} [4]. Since these algorithms learn strictly through environmental interaction, they can discover and develop heuristics and action trajectories that humans might never have considered in their algorithm design. Radiation source search is a well studied problem, however, data-driven approaches have received less attention, in part because of the high variability mentioned above.

This thesis aims to ascertain if DRL can learn an effective policy that generalizes across a range of scenarios where background rate, source strength and location, and the number of obstructions vary.

1.2 Literature Review

Many solutions have been proposed for nuclear source search and localization across a broad range of scenarios and radiation sensor modalities. These methods are generally limited to the assumptions made about the problem such as the background rate, mobility of the source, shielding presence, and knowledge of obstruction composition. Morelande et al. present a maximum likelihood estimation approach and a Bayesian approach to multi-source localization using multiple fixed detectors in an unobstructed environment [5]. Hite et al. also use a Bayesian approach with Markov chain Monte Carlo to localize a single point source in a cluttered urban environment by modeling the radiation attenuation properties of different materials [6]. Hellfeld et al. focused on a single detector in 3D space moving along a pre-defined path for single and multiple weak sources [7] They utilized an optimization framework with sparsity regularization to estimate the source intensity and coordinates.

There is great interest in autonomous search capabilities for source search to limit human exposure to harmful radiation. Cortez et al. proposed and experimentally tested a robot that used variable velocity uniform search in a single source scenario [8]. Ristic et al. proposed three different formulations of information-driven search with Bayesian estimation all evaluated in simulation. An information-driven search algorithm selects actions that maximize the available information for its estimates of user-specified quantities of interest at each timestep. The first method utilized the Fisher information matrix and a particle filter for a single source and single detector in an open area with constant background [9]. The second and third method both used the Renyi information divergence metric and particle filter to control a

detector/detectors in open/cluttered environments with multiple sources, respectively [10],[11]. In the cluttered environment, the layout was considered to be known before the start of the search. Anderson et al. considered a single mobile detector used for locating multiple sources in a cluttered environment through an optimization based on the Fisher information and travel costs [12]. The obstruction attenuation and nuclear decay models were specified by hand.

RL and DRL have also been applied to the control of single robots. Landgren used a multi-armed bandit approach to control nuclear source search in an indoor environment [13]. This was implemented on a Turtlebot3 and used to find multiple radioactive sources in a lab through radiation field sampling. Liu et al. used double Q-learning to control a single detector search for a single radioactive source with a varying sized wall in simulation [14]. The model performed well when the test search environment matched its training set but did not generalize when new geometries were introduced and had to be retrained. This approach is the most similar to the one used in this thesis.

In contrast to the majority of the methods mentioned above, our algorithm does not directly rely on any hard-coded modeling assumptions for decision making. This gives greater flexibility to our approach and allows the opportunity for generalization to a greater variety of situations. For example, our approach was only trained on up to five obstructions in an environment at any one time but can easily operate when greater than five obstructions are present. Additionally, it would be relatively simple to retrain the agent to account for a moving source or novel obstruction types and layouts, among other things. This comes with the caveat that there is a heavy reliance upon the assumptions made in modeling an environment that are likely to fail in capturing the intricacies of reality (reality gap). This is an area of intense interest in the DRL research space [15].

1.3 Research Goals and Hypothesis

This thesis focuses on the search for a single radiation source by a single detector in a simulated 2D environment with background radiation, variable source intensity and location, variable detector starting position, and obstructions. Our first goal is to assess design decisions of our proposed NN architecture, the RAD-A2C, such as the localization module and certain hyperparameters relative to performance. Our second goal is to evaluate the RAD-A2C in a convex environment through comparison against a modified information-driven search algorithm previously proposed in the nuclear source search literature and a gradient search algorithm. Our third goal is to examine the effect of obstructions on the RAD-A2C performance in a non-convex environment with comparison to a gradient search algorithm. We hypothesize that the RAD-A2C model will match the performance of an information-driven search algorithm across a range of signal-to-noise ratios in the convex environment and outperform gradient search in the non-convex environment.

1.4 Approach

1.4.1 Network Architecture

The partial observability of the radiation search task necessitates the incorporation of observations across time to ensure consistent completion of the task. *Recurrent neural network* (RNN)s are a type of artificial neural network that incorporates state information across time through its hidden state [16]. *Long short-term memory* (LSTM) networks are a subset of RNNs that have finer control over state storage and propagation and have greater stability during training [17]. The *gated recurrent unit* (GRU) is a modified version of the LSTM that requires fewer parameters and faster training at the loss of propagating information across longer time sequences [18]. Our scenario of interest has a maximum sequence length of 120 making it amenable to

using the GRU over the LSTM.

In addition to the use of RNNs, the agent architecture was modularized into source localization and action selection. The localization module was trained exclusively to predict the source location and then pass this prediction to the action selection module. The *particle filter gated recurrent unit* (PFGRU) embeds a *bootstrap particle filter* (BPF) into a recurrent neural network to approximate a posterior state distribution [19]. Action selection was structured in the *actor-critic* (A2C) architecture and used the current observation, source location and hidden state in each selection [20].

1.4.2 Training Framework

Proximal policy optimization (PPO) is a recently proposed model-free policy gradient method that has achieved consistent performance across a variety of RL domains [21]. Model-free methods require that the agent learns a policy from its experiences throughout training, whereas model-based methods focus on using a learned or given model to plan action selection. Model-free methods are worse in terms of sample efficiency than model-based or Q-learning because learning takes place in an episodic fashion, i.e. the policy is updated on a per-episode basis. An episode is a sequence of states, actions, and rewards concluded with a terminal state. The benefit is that it allows the agent to directly optimize policy parameters through the maximization of the reward signal. The decision to use model-free policy gradients was motivated by the stability and ease of hyperparameter tuning during training.

1.4.3 Simulation Environment

One of the fundamental components of learning in RL is the environment. In this thesis, the agent is trained in a 22×22 m simulated environment consisting of a randomly sampled source activity, background rate, source location, and detector starting location. At each timestep, radiation measurements are sampled from a

Poisson distribution with rate proportional to the inverse square distance. Additionally, obstructions are randomly placed into the environment that obscures the radiation signal. A key advantage of this environment randomization is that it induces efficient exploration of policy space by presenting the agent with a diverse set of experiences. In addition, each policy update is the result of 400 episodes each with different environment initializations. This stabilizes the gradient and encourages the agent to hone in on the important features of the observations that result in an increased reward.

1.5 Contributions

The main contributions of this thesis are:

- The RAD-A2C, a novel NN architecture for radiation source search that utilizes a GRU and the PFGRU to effectively solve radiation source search in convex and non-convex environments. More details in Section 3.2.3.
- Development of a model-free proximal policy optimization algorithm to train the RAD-A2C found in Section 3.2.3
- A graph-based radiation simulation environment that can be customized with and without obstructions found in Section 4.1.
- Comparison of the RAD-A2C to a modification of a well-cited information-maximization nuclear search algorithm and a standard gradient search algorithm, found in Chapter 5 and Section 4.5.

1.6 Document Overview

This chapter briefly defined nuclear source search and its associated challenges, reviewed localization/search techniques previously put forward in the literature, laid out the

general approach taken to solve this problem, and lists the contributions of this document. The next chapter covers the basics of deep learning and focuses on RNNs and the PFGRU. Chapter 3 covers the general RL problem, policy gradients, DRL, and PPO. In Chapter 4, we detail the implementation of our simulation environment, the RAD-A2C architecture and training details, the comparison algorithms, and evaluation metrics. In Chapter 5, we show the results of our DRL algorithm in both convex and non-convex simulation environments, compare performance against the other search algorithms, and discuss the implications of our findings with respect to our hypothesis. In Chapter 6, we summarize this thesis and highlight some potential avenues for future work.

Chapter 2

Deep Neural Networks

This chapter opens with the basics of supervised learning and moves into *deep learning* (DL). Section 2.2-2.4 focus on two types of network architectures essential to our radiation source search approach. Section 2.5 briefly overviews classical state space tracking through filtering and section 2.5.2 covers the data-driven approach to state space tracking with the particle filter neural network used in our model.

2.1 Supervised Learning

The objective of *machine learning* (ML) is for a computer to learn a mapping from some input space to some output space using data. For example, suppose we have a collection of images and the associated label of whether a certain object is present in the image or not. We are interested in devising a mapping from the pixel inputs to a decision of object presence. Humans have developed many rule-based approaches through logic and reasoning to solve this and many other types of problems in the field of computer vision [22]. However, this methodology of manual model development quickly runs aground when the domain or task becomes too complex, and thus, we turn to the ML paradigm which provides a principled and scalable alternative to solving this and many other types of problems. Thus, the field of computer vision has quickly shifted to DL to achieve state of the art results [23].

First, we define the supervised ML context. Let $\mathcal{X}$ be the set of inputs, $\mathcal{Y}$ be the label set, and the data distribution be $\mathcal{D}$. The desired learner's output will be

a mapping or hypothesis $h : \mathcal{X} \to \mathcal{Y}$ that is learned from a set of training data $\mathcal{S} = \{(x_i, y_i)\}_{i=0}^{n} \sim \mathcal{D}$. Next, we define a measure of success that determines how well the hypothesis fits the data as

$$\mathbb{E}_{(x,y)\sim\mathcal{S}}[\mathcal{L}(h(x), y)], \tag{2.1}$$

where $\mathcal{L}$ is some loss function such as the mean square error or classification loss [24]. The choice of loss function is essential to the desired outcome of the learned parameters. The aim is to find a hypothesis from some set of hypothesis $\mathcal{H}$ that minimizes 2.1. Importantly, all of the operations of the learning process must be differentiable such that gradient descent can be performed to improve the parameters.

2.2 Feedforward Neural Network

As specified in the introduction, DL is a subset of ML that is concerned with directly approximating target functions through a combination of weights, biases, and non-linearities. *Feedforward neural networks* (FNN) are made up of a layer of "neurons" connected in sequence across hidden layers as shown in Figure 2.1. Each hidden layer contains h_l hidden neurons that describes the "width" of the network and there are L hidden layers that describe the "depth." The neurons each have a differentiable activation function represented by the piecewise linear function inside the neuron. Hornik et al. proved the universal function approximation capability for a multilayer FNN with arbitrary activation functions [25].

While theoretically true, in practice, it can be difficult to train the larger neural networks that are required for more complex functions as the data requirements quickly increase. The FFN can be expressed in the following form where x_0 is the input to

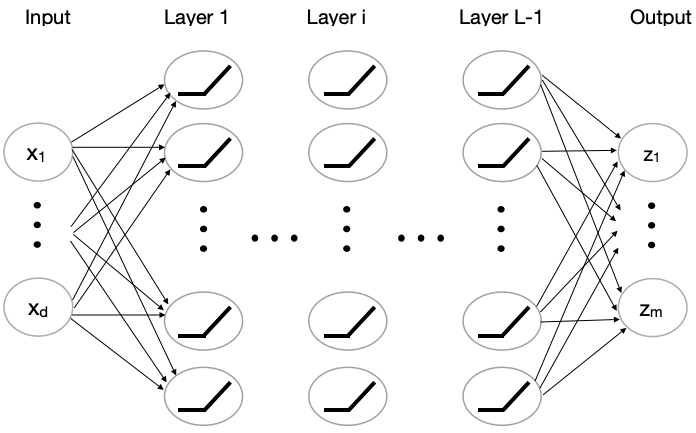


Figure 2.1: FNN Architecture. Each hidden layer contains h_l hidden neurons and there are L hidden layers. The neurons each have an activation function represented by the piecewise linear function inside the neuron.

the initial layer and σ is an arbitrary activation function:

$$y_{l+1} = W_{l+1}^T x_l + b_{l+1} \text{ for } l = 0 \dots L-1, \tag{2.2}$$

$$x_{l+1} = \sigma(y_{l+1}), \tag{2.3}$$

where W_{l+1} varies in dimension depending on the previous layer input and the number of neurons in layer $l+1$. We denote the final output layer to be $h(x) = z_L$, labeled as "Output" in Figure 2.1. The weight matrix and bias term of each layer are updated through optimization using stochastic gradient descent [26]. First, define $W = \{W_1, \dots, W_L\}$, then the gradient is calculated through back-propagation of the error signal on a loss function using the chain rule, and the weights are updated as

$$W_{k+1} = W_k - \eta \nabla \mathbb{E}_{(x,y)\sim\mathcal{B}}[\mathcal{L}(h(x), y)], \tag{2.4}$$

where k is the iterate number, η is the learning rate, $\mathcal{L}(\cdot)$ is some loss function, and $\mathcal{B}$ is some batch of training examples of size c [27].

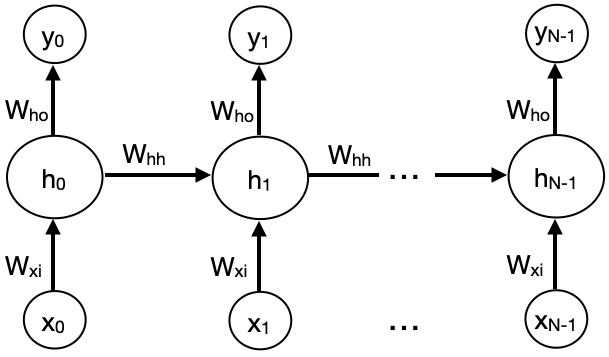


Figure 2.2: RNN Architecture. The learned weight matrices W_{ho}, W_{xi}, W_{hh} are the same across all sequence steps so the only changes are the input, output and hidden state. The h_n represents the hidden state which is passed between sequence steps and is combined with the input to carry information across time. Each input x_n is input to the network sequentially.

2.3 Recurrent Neural Networks (RNN)

A major drawback of the FFN is that it does not factor in information across time explicitly. This can be addressed through a clever setup of the input vector such as a sliding window of inputs or an input map. RNNs are a type of network architecture that handles inputs with a temporal or spatial dimension such as sequence data explicitly through the use of feedback through a hidden state [16].

Figure 2.2 shows the basic idea of an RNN with sequence inputs $x_0, \ldots, x_{N-1}$ where each $x_i \in \mathbb{R}^d$. The learned weight matrices W_{ho}, W_{xi}, W_{hh} are the same across all sequence steps so the only changes are the input, output and hidden state. The h_n represents the hidden state which is passed between sequence steps and is combined with the input to carry information across time:

$$h_{n+1} = \sigma(W_{xi}^T x_{n+1} + W_{hh}^T h_n + b_h), \tag{2.5}$$

$$y_{n+1} = \sigma(W_{ho}^T h_{n+1} + b_q). \tag{2.6}$$

The weight matrices and biases are updated via the loss between the desired output and the prediction y_n with *back-propagation through time* (BPTT) [28]. BPTT is necessary because there is a dependence on the previous sequence of inputs and hidden

states in determining the current output. However, in the current implementation of the RNN, BPTT often results in vanishing or exploding gradients when dealing with long-range temporal dependencies [29].

2.4 Gated Recurrent Unit (GRU)

The GRU is part of the *long-short term memory* (LSTM) family which are types of RNNs. These networks use gates to address the vanishing and exploding gradients encountered when using BPTT and increase the network's ability to establish dependencies across long temporal gaps. Hochreiter et al. proposed the first LSTM architecture with great success in learning temporal relationships with time intervals greater than 1,000 steps [17]. The GRU is a variation on the LSTM proposed by Cho et al. that uses similar principles to the LSTM but reduces the number of gates and thereby reduces the number of parameters [30]. The following set of equations describe the GRU operations,

$$z_{n+1} = \sigma(W_{xr}^T x_{n+1} + W_{hr}^T h_n + b_h), \tag{2.7}$$

$$r_{n+1} = \sigma(W_{xz}^T x_{n+1} + W_{hz}^T h_n + b_h), \tag{2.8}$$

$$\tilde{h}_{n+1} = \tanh(W_{xh}^T x_{n+1} + W_{hh}^T (r_{n+1} \odot h_n) + b_h), \tag{2.9}$$

$$h_{n+1} = (1 - z_{n+1}) \odot h_n + z_{n+1} \tilde{h}_{n+1}, \tag{2.10}$$

where $\sigma(\cdot)$ is the sigmoid activation function and $\tanh(\cdot)$ is the hyperbolic tangent activation function. Clearly, the GRU has more parameters than the standard RNN but the huge gain is in training stability and the increase in range for sequence relationships.

Figure 2.3 shows the design of a single GRU cell taken from Olah [31]. Each box represents a weight matrix and activation function and the circles represent mathematical operations. The conjoining lines represent the concatenation of the

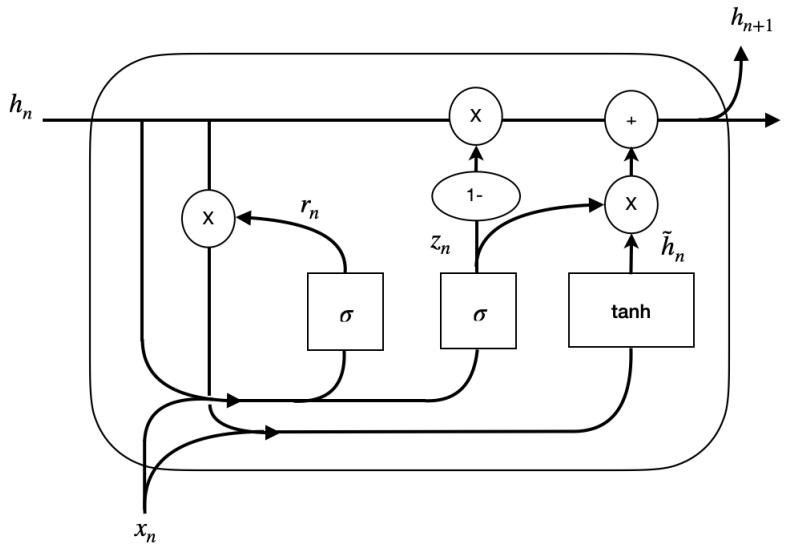


Figure 2.3: GRU Architecture. Each box represents a weight matrix and activation function and the circles represent mathematical operations. The conjoining lines represent concatenation of the quantity and diverging lines represent the copying. The crux of the reset (r_n) and update (z_n) gates are to modify the candidate hidden state ($\tilde{h}_n$) which then becomes the output hidden state (h_n) [31].

quantity and diverging lines represent the copying. The crux of the reset (r_n) and update (z_n) gates are to modify the candidate hidden state ($\tilde{h}_n$), which then becomes the output hidden state (h_n). The reset gate determines how much of the previous hidden state to factor into the new hidden state and the update gate determines the convex combination of the previous hidden state and the candidate hidden state. This cell is a drop-in replacement for the hidden state h_n found in Figure 2.2.

Modifying the hidden state with gates alleviates the unstable gradient issues found in the RNN architecture due to greater control of the flow of information over time and to the additive nature of the hidden state. For example, if an input feature at the beginning of a sequence is important to some future output, this can be preserved or forgotten through the respective gates (update, reset) that eliminate the need to back-propagate the error gradient across many timesteps. The addition of the previous and current hidden state also guarantees this is the only component of the cell for which the gradient has to be calculated across time eliminating a major factor in gradient instability [32].

2.5 State Space Tracking Methods

Here, we give an overview of the classic *bootstrap particle filter* (BPF) and the data-driven approach called the *particle filter gated recurrent unit* (PFGRU) as both techniques are used in the experimental methods. State space tracking is a methodology for estimation of a dynamical system's internal state across time. This is accomplished using sequential Monte Carlo methods to predict and filter the process's dynamic posterior distribution given observations, a process model, and a measurement model. The process model specifies how the state of the system changes through time and the measurement model specifies the relationship between the state and observations. The process and measurement models are typically denoted as,

$$\begin{aligned} x_{n+1} &= f_{n+1}(x_n) + v_{n+1}, \\ y_{n+1} &= h_{n+1}(x_{n+1}) + w_{n+1}, \end{aligned} \tag{2.11}$$

where v_n, w_n, are white noise processes and f_n, h_n, are the process and measurement models, respectively. These models can be nonlinear and time-varying.

2.5.1 Particle Filter

The BPF is an effective heuristic to track the internal state of a dynamical process. It has been proven that an optimal estimate of the state can be recovered from the posterior state distribution, however, it is often computationally intractable to track when the state dimension is high [33]. Thus, methods such as the BPF attempt to approximate the posterior state through a set of samples, $\{x_n^i, w_n^i\}_{i=1}^{N_p}$, often referred to as particles and weights, respectively. This leads to the approximation,

$$P(x_{n+1}|y_{0:n+1}) \approx \sum_{i=1}^{N_p} w_{n+1}^i \delta(x_{n+1} - x_{n+1}^i), \tag{2.12}$$

where $P(x_{n+1}|y_{0:n+1})$ is the marginal posterior, w^i_{n+1} is the i^{th} particle weight, x^i_{n+1} is the i^{th} particle state, $\delta(\cdot)$ is the Dirac Delta function, and N_p is the number of particles. At each timestep, the particles are propagated through the process model and a measurement prediction is generated with the measurement model. The particle weights are calculated recursively as,

$$w^i_{n+1} \propto \frac{p(y_{n+1}|x^i_{n+1})p(x^i_{n+1}|x^i_n)}{q(x^i_{n+1}|x^i_n, y_{n+1})} w^i_n, \tag{2.13}$$

where $p(y_{n+1}|x^i_{n+1})$ is the measurement likelihood, $p(x^i_{n+1}|x^i_n)$ is the transition density, and $q(x^i_{n+1}|x^i_n, y_{n+1})$ is an importance density [33]. Particles are drawn from a user-specified importance density q_x. In our implementation, the importance density is set equal to the prior distribution to reduce the weight update step to the measurement likelihood and the previous weight:

$$w^i_{n+1} \propto p(y_{n+1}|x^i_{n+1})w^i_n. \tag{2.14}$$

If a particle has a low probability for a given measurement, this effectively removes the particle's contribution to the estimated posterior which can adversely affect state estimation over the trajectory and is known as the degeneracy problem. Particle degeneracy can be tracked by the following metric to characterize the number of effective particles at a given time step,

$$N_{e,n} = \frac{1}{\sum_{i=1}^{N_p}(w^i_n)^2}. \tag{2.15}$$

Particle degeneracy can be alleviated by resampling the particles and reinitializing the weights when the number of effective particles becomes too low. In Chapter 4, the specific resampling scheme used in the comparison algorithm will be discussed.

2.5.2 Particle Filter Gated Recurrent Unit (PFGRU)

The PFGRU is an embedding of the BPF into a GRU architecture proposed by Ma et al [19]. As in the BPF, there are a set of particles and weights used for filtering and prediction of the posterior state distribution. In the case of the PFGRU, the particles are represented by the set of hidden or latent state vectors, $\{h_n^i\}_{i=1}^{N_{gp}}$. The latent states are propagated and the weights updated at each timestep by a learned transition and measurement function denoted as,

$$\begin{aligned} h_{n+1}^i &= f_{tr}(h_n^i, \zeta_{n+1}^i) \\ y_{n+1}^i &= f_{out}(h_{n+1}^i), \end{aligned} \tag{2.16}$$

where $\zeta_n^i \sim p(\zeta_{n+1}^i | h_{n+1}^i)$ is a learned noise term akin to the process noise in the BPF. The weight update also relies on a learned likelihood function,

$$w_{n+1}^i = \eta f_{obs}(y_{n+1}, h_{n+1}^i) w_n^i, \tag{2.17}$$

where η is a normalization factor.

Particle degeneracy is a similar issue to the BPF but the PFGRU utilizes a soft resampling scheme to maintain model differentiability. This is achieved by sampling particle indices from a multinomial distribution with probabilities determined by a convex combination of a uniform distribution and the particle weight distribution. The new weights are then determined by,

$$w_{n+1}^{'} = \frac{w_{n+1}^{a_n^j}}{\alpha w_{n+1}^{a_{n+1}^j} + (1-\alpha)(1/N_p)}, \tag{2.18}$$

where α is the mixture coefficient parameter. The loss function consists of two components to capture the important facets of state space tracking. The first component is the mean squared loss between the mean particle and the predicted quantity. The sec-

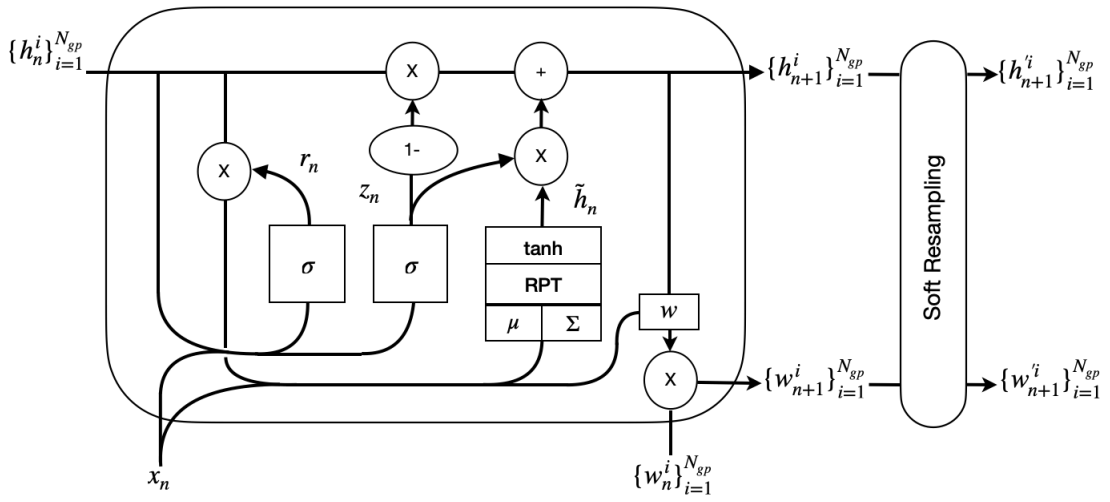


Figure 2.4: PFGRU Architecture. The hidden state h_n^i and weights w_n^i are elements of a set of size N_{gp}. Each box represents a weight matrix and activation function and the circles represent mathematical operations. The conjoining lines represent concatenation of the quantity and diverging lines represent the copying. The crux of the reset (r_n) and update (z_n) gates are to modify the candidate hidden state ($\tilde{h}_n$) which then becomes the output hidden state (h_n). The hidden state and weights are resampled using a soft-resampling scheme at each timestep to preserve differentiability. Recreated from [19].

ond component is the *evidence lower bound* (ELBO) loss that measures the difference in distribution of the particle distribution relative to the observation likelihood, for more details see [19]. The total loss is expressed as,

$$\mathcal{L}(\theta) = \mathcal{L}_{MSE} + \beta * \mathcal{L}_{ELBO}, \tag{2.19}$$

where β is a weighting parameter determined by the user.

2.6 Machine Learning Summary

This chapter introduced the core ideas of ML and the state space tracking framework for dynamical parameter estimation that are both utilized in our radiation source search implementation. Supervised learning is an ML paradigm that facilitates the approximation of any arbitrary function provided the appropriate data, function class, and loss are used. DL is a class of functions that have proven universal function approximation capability, FFN being one of the more basic instantiations. However,

the FFN is poor for learning patterns across temporal/spatial sequence data and so we turned to the RNN to incorporate information across timesteps with its hidden state. The PFGRU is a variation of the RNN that emulates the BPF by using weighted hidden state vectors instead of particles and learns the process and measurement models from data. The next chapter will introduce the main themes of classical reinforcement learning, how DL is used to improve reinforcement learning, and the deep reinforcement learning training framework used in this thesis.

Chapter 3

Reinforcement Learning and Deep Reinforcement Learning

This chapter introduces the *reinforcement learning* (RL) problem and then develops the framework to effectively solve problems of this nature. Sections 3.1.1-3.1.6 layout the essential concepts of RL that form the basis of our algorithmic approach to the radiation source search problem. Sections 3.2-3.2.2 highlights how these RL concepts can be augmented with NNs to transform the learning problem into a parameter optimization problem. Section 3.2.3 covers the *actor critic* (A2C) policy architecture and Section 3.3 details the optimization procedure used to train our controller.

3.1 Reinforcement Learning

RL is the branch of machine learning focused on some agent learning a useful/desired behavior through direct experience within an environment. For example, an agent or policy in our problem context is the controller that selects actions for the detector movement. These experiences consist of past states and rewards received by an agent taking actions in the environment. The agent begins with only a specified goal and an ability to sense the environment and must learn an effective policy that maps states to actions to achieve the goal [34]. This trial-and-error learning allows the agent to explore and optimize its behavior through the reward signal to maximize the expected cumulative reward achievable in a given episode. An episode is a sequence of states, actions, and rewards concluded with a terminal state. Some well-known challenges of RL are determining how to attribute delayed reward to actions and balancing

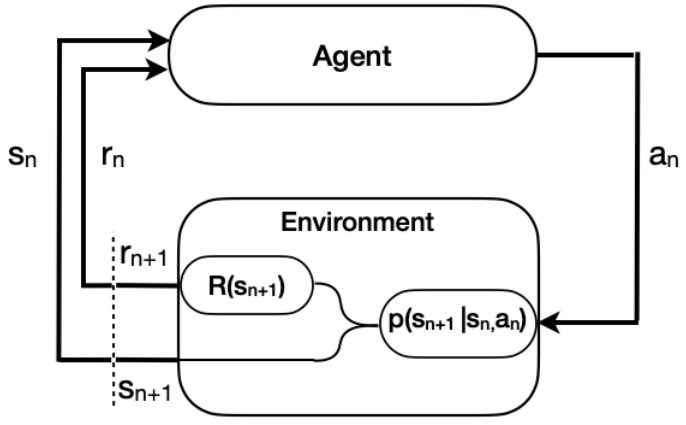


Figure 3.1: RL is classically formulated as an MDP. At every timestep $(n+1)$, the agent receives the full state s_{n+1} and the reward r_{n+1} associated with having taken action a_n from state s_n.

exploration of the environment with the exploitation of behaviors with ostensible reward [34].

3.1.1 Markov Decision Process

The common framework to discuss RL often starts with the *Markov decision process* (MDP). MDPs obey the Markov property and assume that environment is fully observable, or that there is no uncertainty in the observations the agent receives [34]. The finite MDP is defined by the tuple $\langle S, A, \mathcal{T}, R \rangle$ at each time step, n, by:

- the state, $s_n \in S$, in the finite set of states;
- the action, $a_n \in A$, in the finite set of actions;
- the reward function, $R(s_n, a_n, s_{n+1})$;
- the transition probability distribution, $\mathcal{T}(s_{n+1}|s_n, a_n) = p(s_{n+1}|s_n, a_n)$.

The transition probability distribution and reward function are considered fixed and unknown in the RL setting. A trajectory or episode is a sequence of observations and actions up to a timestep n, defined as $\tau_n = (s_0, a_0, s_1, a_1, ..., s_{n-1}, a_{n-1}, s_n)$. The general environment-agent interaction is illustrated in Figure 3.1.

3.1.2 Return and Policy

The source search is considered to be an episodic task which dictates finite trajectories and that the environment and agent are reset to an initial state distributed according to initial state distribution, $\rho_0(s)$, after episode completion. The cumulative episode return is defined as,

$$R(\tau) := \sum_{n=0}^{N-1} \gamma^k r_n, \tag{3.1}$$

where N is the length of the trajectory and $\gamma \in [0, 1)$ is the discount factor. The reward-to-go is defined similarly to 3.1 but accounts for the time index and so provides clearer attribution of reward,

$$\hat{R}_n = \sum_{n'=n}^{N-1} \gamma^k R(s_{n'}, a_{n'}, s_{n'+1}). \tag{3.2}$$

The aim of RL is maximize the expectation of $R(\tau)$ through a policy learned by interaction with the environment. A policy is a mapping from states to actions in either a deterministic $\pi(s_n)$ or stochastic $\pi(a_n|s_n)$ manner. Since the policy and environment transitions are stochastic, a trajectory conditioned on observations can be written as

$$p(\tau|\pi) = \rho_0(s) \prod_{n=0}^{N-1} p(s_{n+1}|s_n, a_n)\pi(a_n|s_n). \tag{3.3}$$

The expected return can then be expressed as

$$J(\pi) = \int_\tau p(\tau|\pi)R(\tau)d\tau = \mathbb{E}_\pi[R(\tau)], \tag{3.4}$$

and the optimal policy is the one that satisfies the formulation

$$\pi^* = \text{argmax}_\pi J(\pi). \tag{3.5}$$

A policy can be either be implemented directly through parameterizations or through value functions that use a look-up table.

3.1.3 Q, Value, and Advantage Functions

The agent learns from the environment reward signal by updating its value functions. The value function estimates cumulative reward attainable from a given state (or state-action pairs) that gives the agent a notion of the quality of its state [34]. This is also a means of judging the quality of a policy, as the value is defined as the expected cumulative reward across the trajectory when starting from state s and acting according to policy π thereafter or more succinctly,

$$V^{\pi}(s) = \mathbb{E}_{\tau\sim\pi}[R(\tau)|s_0 = s]. \tag{3.6}$$

The Q (or state-action value) function is defined similarly to the state value function but fixes both the initial action and state,

$$Q^{\pi}(s,a) = \mathbb{E}_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a]. \tag{3.7}$$

The Q function is fundamental to the RL paradigm known as Q-learning that seeks to learn the Q function for a given environment and then select useful actions according to the Q values [34]. In smaller state spaces, the value functions can be implemented in a tabular format such that the output for a given state just requires a table search. We can relate V^{π} and Q^{π} through the following:

$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi}[Q^{\pi}(s,a)]. \tag{3.8}$$

The advantage function is defined according to the difference between the Q function and the value function to quantify the value of a given action in a given state or,

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s). \tag{3.9}$$

3.1.4 Bellman Equations

The Bellman equations establish the theoretical foundation for solving RL problems through the recursive relationship between the value of the current state and all future states. In the Bellman formulation, the value function can be expanded following [34],

$$\begin{aligned} V^{\pi}(s_n) &= \mathbb{E}_{\tau\sim\pi}[R(\tau)|s_0 = s_n] \\ &= \mathbb{E}_{a\sim\pi, s_{n+1}\sim\mathcal{T}}[r_n + \gamma V^{\pi}(s_{n+1})|s = s_n] \\ &= \sum_{a} \pi(a|s_n) \sum_{s_{n+1},r} p(s_{n+1}, r|s_n, a)[r_n + \gamma V^{\pi}(s_{n+1})], \end{aligned} \tag{3.10}$$

where the second line follows from the definition of the value function and the third line is an expansion of the expected value. From Eq. 3.10, we see that the value function is just a sum of the reward from state s_{n+1} weighted by the probability of selecting the action leading to that state and the probability of the environment transitioning to that state.

The Bellman equations allow a clear distinction between model-free and model-based control approaches. When the transition probabilities and reward function are known (model-based), methods like dynamic programming or model-predictive control can be used to solve the Bellman equations iteratively. Additionally, an agent can learn a model directly through interaction with an environment and then use it in planning such as the world models approach proposed by Ha et al. [35]. If the transition probabilities and reward function are not known or there is no interest in learning a model (model-free), then the agent can use techniques such as temporal difference learning to update its value/policy functions.

3.1.5 Temporal Difference (TD) Learning

TD is a method of generating update targets from experiences in the environment for the state and state-action value functions. This can be done at each timestep instead of requiring the completion of an entire episode as in Monte Carlo methods [34]. The TD update equation is given as

$$V^{\pi}(s_n) \leftarrow V^{\pi}(s_n) + \alpha[r_{n+1} + \gamma V^{\pi}(s_{n+1}) - V^{\pi}(s_n)], \tag{3.11}$$

where α is the learning rate. Each action taken returns a reward from the environment and a next state, s_{n+1}, that is then used to update the previous estimate of the value function for the current state, s_n. The term in brackets is called the TD error and is often denoted,

$$\delta_n = r_{n+1} + \gamma V^{\pi}(s_{n+1}) - V^{\pi}(s_n). \tag{3.12}$$

TD learning aims to minimize this error through sufficient state space exploration.

3.1.6 Partial Observability

In the context of the radiation search scenario where measurements are noisy and uncertain, it is more useful to describe the *partially observable Markov decision process* (POMDP). The finite POMDP is defined similarly to the MDP tuple but with two additional terms, $\langle S, A, T, R, \Omega, \mathcal{O} \rangle$ at each time step, n where:

- the observation, $o_n \in \Omega$, in the finite set of observations, and
- the observation probability distribution, $\mathcal{O}(o_n|s_n) = p(o_{n+1}|s_n)$.

The observation and transition probability distributions are considered fixed and unknown. An observation is a function of the true state but is not necessarily representative of the true state due to the stochastic nature of the environment. In this section, we closely follow the notation of Wierstra et al. [36]. A history

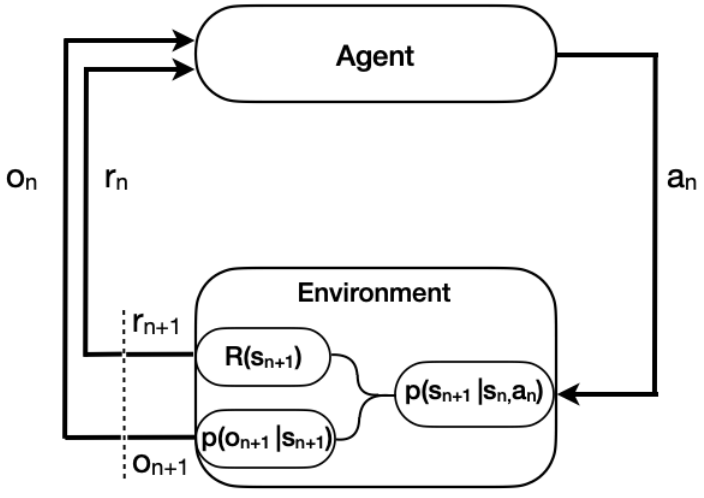


Figure 3.2: POMDP. At every timestep $(n+1)$, the agent receives an observation o_{n+1} and reward r_{n+1} conditioned on the state s_{n+1}. The observation only gives partial information about the state. The state transition and the reward depend on taking action a_n in state s_n.

is a sequence of observations and actions up to a timestep n, defined as $H_n = (o_0, a_0, o_1, a_1, ..., o_{n-1}, a_{n-1}, o_n)$. A successful policy needs to consider H_n to inform its decisions since a single observation does not necessarily uniquely identify the current state. This can be implemented directly by concatenation of all previous observations and actions with the current observation input or through the use of recurrent neural networks as described in 2.3. Figure 3.2 demonstrates the observation dependence on the state s_n, which is not accessible to the agent.

The function $M(H_n)$ provides a sufficient statistic of the past history and serves as the basis for the agent's decision making [36]. This allows the policy to be reformulated as $\pi(a_n|h_n) = p(a_n, M(H_n); \theta)$ where θ is some parameterization and since the policy and environment transitions are stochastic, the history can be written,

$$p(H) = \rho_0(s_0)p(o_0|s_0)\prod_{n=0}^{N-1} p(o_{n+1}|s_{n+1})p(s_{n+1}|a_n, s_n)\pi(a_n|h_n). \tag{3.13}$$

The expected return can then be expressed as

$$J(\pi) = \int_H p(H|\pi)R(H)dH = \mathbb{E}_\pi[R(H)], \tag{3.14}$$

where $R(H)$ is defined the same as Eq. 3.1.

3.2 Deep Reinforcement Learning (DRL)

Classical RL relies on estimates of the value functions updated through repeated experiences in the environment to enact a policy $\pi(a|s)$. Thus, the policy is entirely dependent on the value functions and relies on sufficient coverage of the state space to make accurate estimates of state values to inform the selection of actions. In high dimensional state spaces found in many practical applications, this control proposition quickly becomes infeasible. DRL alleviates this issue by using *neural networks* (NN) to directly approximate the value and policy functions through the collected experiences [37]. Additionally, these separate parameterizations allow decoupling of the action selection from the value functions.

3.2.1 Policy Approximation

In the following sections, we operate under the POMDP assumptions specified in Section 3.1.6. NNs are universal function approximators (section 2.2) and therefore provide a useful and generalizable paradigm across a broad range of tasks. We denote the weights and biases of the networks as θ and ϕ and write the parameterized policy $\pi_\theta(a_n|h_n)$ and value function as $V_\phi^\pi(h_n) = V(h_n;\theta)$. Here, h_n refers to the hidden state of a *recurrent neural network* (RNN) as specified in section 2.3 and serves as the sufficient statistic of the history H_n. The policy output is converted to a distribution over actions with the exponential softmax activation function defined as,

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{3.15}$$

where z is some vector of dimension K. This activation function guarantees that the output probabilities will be in the interval $[0, 1]$.

These parameterizations are amenable to first order optimization methods and simply require a useful objective to be defined. The general gradient ascent framework is defined as,

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta J(\pi_\theta)|_{\theta_k}, \tag{3.16}$$

where α is the learning rate, k is the parameter iterate, and the objective is the expected return of the policy. In practice, the true objective is not readily available and so stochastic estimates are used instead that approximate the true gradient in expectation [34]. As this is an on-policy framework, optimization is only performed from the episodes collected during the parameter iterate θ_k.

3.2.2 Policy Gradient

The gradient in 3.16 can be solved analytically as first done by Williams [38]. Starting with the definition specified in Eq. 3.14 and following [36],

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \int_H p(H|\pi_\theta) R(H) dH \\
&= \int_H \nabla_\theta p(H|\pi_\theta) R(H) dH \\
&= \int_H \frac{p(H|\pi_\theta)}{p(H|\pi_\theta)} \nabla_\theta p(H|\pi_\theta) R(H) dH \\
&= \int_H p(H|\pi_\theta) \nabla_\theta \log\, p(H|\pi_\theta) R(H) dH \\
\nabla_\theta J(\pi_\theta) &= \mathbb{E}_H[\nabla_\theta \log\, p(H|\pi_\theta) R(H)],
\end{aligned} \tag{3.17}$$

where line 2 follows from the fact that the reward $R(H)$ does not depend on the parameters and line 3 to 4 uses the log-derivative trick. Intuitively, each gradient step will be in the direction that increases the log probability of a history weighted by the reward associated with that history. Thus, if a history resulted in a lower or negative reward, the parameters will be updated such that that history becomes less likely.

The gradient of the log probability of a history is easily derived starting from the

definition in Eq. 3.13 and setting the terms that do not rely on θ to zero,

$$\nabla_\theta \log\, p(H|\pi_\theta) = \sum_{n=0}^{N-1} \nabla_\theta \log\, \pi_\theta(a_n|h_n). \tag{3.18}$$

In addition, the cumulative reward over a history is substituted with Eq. 3.2 since the cumulative reward from a time step is only relevant when the reward was consequent to the action. This yields the final gradient equation as

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_H[\sum_{n=0}^{N-1} \nabla_\theta \log\, \pi_\theta(a_n|h_n)\hat{R}_n]. \tag{3.19}$$

This expectation can be approximated through the histories collected by our policy π_θ resulting in the unbiased gradient estimator,

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{M} \sum_{m=1}^{M} \sum_{n=0}^{N-1} \nabla_\theta \log\, \pi_\theta(a_n|h_n^m)\hat{R}_n. \tag{3.20}$$

These gradients are obtained numerically and are easily applied to the NN parameterizations through backpropagation.

3.2.3 Actor Critic (A2C)

A major drawback of the policy gradient approach is the susceptibility to high variance in the gradient estimate due to the computational demand of collecting enough histories to approximate the expectation. Williams proposed the baseline function to reduce the variance of the gradient estimates without adding bias [38]. This is represented in the following formula where $b(h_n)$ is the baseline function,

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_H[\sum_{n=0}^{N-1} \nabla_\theta \log\, \pi_\theta(a_n|h_n)(\hat{R}_n - b(h_n))]. \tag{3.21}$$

An illustrative example of the baseline is to consider an environment where the reward signal is only positive and of varying magnitude. Thus, the gradient step will only be

increasing the parameter weights, albeit by smaller or larger amounts. If a baseline such as the sample mean of the cumulative reward across histories of an iterate were subtracted, this would result in the smaller cumulative rewards being negative thereby performing down-weighting those parameters.

In the A2C framework, the baseline function is chosen to be the value function V_ϕ^π as this captures the expected return from a given state following the current policy. It then becomes immediately clear whether the selected action had a positive or negative impact on the cumulative return and the parameters can be adjusted accordingly. This allows modification of Eq. 3.21 to use the advantage function as follows from Schulman et al. [39],

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \mathbb{E}_H[\sum_{n=0}^{N-1} \nabla_\theta \log\, \pi_\theta(a_n|h_n)(\hat{R}_n - V_\phi^\pi(h_n))] \\
&= \mathbb{E}_H[\sum_{n=0}^{N-1} \nabla_\theta \log\, \pi_\theta(a_n|h_n)(r_n + \sum_{n'=n+1}^{N-1} \gamma^{n'} R(h_{n'}, a_{n'}, h_{n'+1}) - V_\phi^\pi(h_n))] \\
&= \mathbb{E}_H[\sum_{n=0}^{N-1} \nabla_\theta \log\, \pi_\theta(a_n|h_n)(r_n + V_\phi^\pi(h_{n+1}) - V_\phi^\pi(h_n))] \\
&= \mathbb{E}_H[\sum_{n=0}^{N-1} \nabla_\theta \log\, \pi_\theta(a_n|h_n) A^\pi(h_n, a_n)],
\end{aligned} \tag{3.22}$$

where the last line uses the definition of the Q function and Eq. 3.9. The advantage function in Eq. 3.22 is an estimate of the true advantage function and so the bias and variance must be taken into account. In theory, the advantage function should be unbiased if $V_\phi^\pi = V^\pi$, but this is rare in practice.

Schulman et al. propose the following *generalized advantage estimator* (GAE) with parameters γ, λ to control the bias-variance tradeoff,

$$\hat{A}_n^{GAE(\gamma,\lambda)} := \sum_{n'=0}^{N-1} (\lambda\gamma)^{n'} \delta_{n+n'}, \tag{3.23}$$

where δ is the TD error defined in Eq. 3.12. This is an exponentially-weighted average

of the TD error where γ determines the scaling of the value function that adds bias when $\gamma < 1$ and λ that adds bias when $\lambda < 1$ if the value function is inaccurate [39]. This leaves the final A2C gradient used in our algorithm as,

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_H[\sum_{n=0}^{N-1} \nabla_\theta \log\, \pi_\theta(a_n|h_n)\hat{A}_n^{GAE(\gamma,\lambda)}]. \tag{3.24}$$

The value function parameters are updated with stochastic gradient descent on the *mean square error* (MSE) loss between the value function estimate and the empirical returns,

$$\phi_k = \arg\min_\phi \mathbb{E}_{h_n,\hat{R}_n}[(V_\phi(h_n) - \hat{R}_n)^2]. \tag{3.25}$$

3.3 Proximal Policy Optimization (PPO)

A common issue in policy gradient methods is the divergence or collapse of policy performance after a parameter update step. This can prevent the policy from ever converging to the desired behavior or result in high sample inefficiency as the policy rectifies the performance decrease. Schulman et al. proposed the PPO algorithm as a principled optimization procedure to ensure that each parameter update stays within a trust-region of the previous parameter iterate [21]. We chose to use the PPO-Clip implementation of the trust-region because of the strong performance across a variety of tasks, stability and ease of hyperparameter tuning as referenced in [21] and [40].

The PPO-Clip objective is formulated as,

$$\mathcal{L}(\theta_{k+1}, \theta_k) = \mathbb{E}_H[\mathbb{E}_n[\min(r_n(\theta_{k+1}, \theta_k)\hat{A}_n, \text{clip}(r_n(\theta_{k+1}, \theta_k), 1-\epsilon, 1+\epsilon)\hat{A}_n)]]. \tag{3.26}$$

Here, $r_n(\theta_{k+1}, \theta_k) = \frac{\pi_{\theta_{k+1}}(a_n|h_n)}{\pi_{\theta_k}(a_n|h_n)}$, denotes the probability ratio of the previous policy iterate to the proposed policy iterate and ϵ is the clipping parameter that enforces a hardbound on how much the latest policy iterate can change in probability space

reducing the chance of a detrimental policy update. A further regularization trick is early-stopping based on the *approximate Kullback-Leibler divergence* (AKLD) used in [41]. The AKLD is a measure of the difference between two probability distributions and the approximation is the inverse of $r_n(\theta_{k+1}, \theta_k)$ in log space. If the AKLD between the current and previous iterate over a batch of histories exceeds a user-defined threshold, then the parameter updates over that batch of histories are skipped. Algorithm 1 gives an overview of the PPO-Clip.

Algorithm 1: PPO-Clip [41]

Input: init. policy parameters θ_0 and value function parameters ϕ_0, AKLD threshold η

for $k = 0, 1, 2, ...$ **do**

Collect set of trajectories $\mathcal{H}_k = \{H_i\}$ with π_{θ_k} in environment.

Compute $\hat{A}_n^{GAE}$ from V_{ϕ_k}.

if AKLD$(\pi_{\theta_{k+1}}, \pi_{\theta_k}) < \eta$ **then**

Update policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{H}_k|(N-1)} \sum_{H \in \mathcal{H}_k} \sum_{n=0}^{N-1} \min \left(\frac{\pi_\theta(a_n|h_n)}{\pi_{\theta_k}(a_n|h_n)} A^{\pi_{\theta_k}}(h_n, a_n), \right.$$

$$\left. \text{clip}(A^{\pi_{\theta_k}}(h_n, a_n), 1+\epsilon, 1-\epsilon) \right),$$

Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{H}_k|(N-1)} \sum_{H \in \mathcal{H}_k} \sum_{n=0}^{N-1} \left(V_\phi(h_n) - \hat{R}_n \right)^2,$$

end for

3.4 Summary

This chapter introduced concepts of RL that are fundamental to our approach in solving the radiation source search problem. The key goal of RL is learning a policy that maximizes the cumulative reward for a given task and Markovian environment through trial and error. Radiation source search is framed as a POMDP because the agent only receives noisy observations of the state and thus the agent requires

"memory" to be successful. A fundamental component of learning is the value function that provides a measure of the value of states in the environment and thereby enables a principled approach to policy development. By parameterizing the value function and policy with NNs, the RL problem can be solved from the optimization perspective, which leads to the A2C architecture. The next chapter will cover the implementation details of the radiation source search environment, our agent architecture, comparison algorithms, and the metrics of performance.

Chapter 4

Radiation Source Search Environment and Algorithms

This chapter covers the implementation of the central elements of this thesis. Section 4.1 details the simulation environment and how the RL agent was trained within it. Section 4.2 discusses the design decisions associated with the RL agent. Sections 4.3 and 4.4 explain the comparison search algorithms and Section 4.5 compares the computational complexity of each method. For clarity, the term "agent" refers to action controllers in general and will be preceded by the specific controller when relevant.

4.1 Radiation Source Search Environment

The radiation source search environment was fundamental to the training of the policy. The development of the environment required many careful design decisions in an attempt to provide a useful proof of concept for the efficacy of DRL in practical radiation source search contexts. In the remainder of the paper, we assume that a Gamma radiation source has already been detected through some other means and the objective is to now locate it. We also assume an isotropic detector and a constant background rate per episode.

4.1.1 Gamma Radiation Model

Gamma radiation measured by a detector typically comes in two configurations, the total Gamma-ray counts or the Gamma-ray counts across an energy spectrum. The full spectrum is more information rich as radiation sources have identifiable photo-peaks

but is more complex and computationally expensive to simulate. Thus, our localization and search approach uses the gross counts across the energy bins. Caesium-137 was selected as the source of interest since it is commonly used in industry applications and is monoenergetic [42]. We denote the parameter vector of interest as $\mathbf{x} = [\mathcal{I}_s, x_s, y_s]$, where x_s, y_s are the source coordinates in m and $\mathcal{I}_s$ is the source intensity in *counts per second* (cps) at a source-detector distance of 1 m. These quantities are assumed to be fixed for the duration of an episode in this thesis. An observation at each timestep, n, is denoted as $\mathbf{o}_n$, and consists of the measured counts, z_n, detector position denoted $[x_n, y_n]$ [m], and 8 range sensor measurements is generated. The background radiation rate is a constant λ_b [cps]. The following model is used to approximate the mean rate of radiation counts measurements in an unobstructed environment (convex) with an isotropic detector,

$$\lambda_n(\mathbf{x}) = \frac{\mathcal{I}_s * \epsilon * A * \Delta t}{4 * \pi * [(x_s - x_n)^2 + (y_s - y_n)^2]} e^{-\mu x} + \lambda_b, \tag{4.1}$$

where A, ϵ, and Δt, are the detector surface area [m], the detector intrinsic efficiency, and the dwell time [s], respectively. Since we use gross counts, the detector efficiency is assumed to be one and we consider a unit dwell time. The exponential term models the Gamma ray interaction with matter and will be approximated by a binary attenuation model. We model a typical hand-held scintillation detector (a thallium activated cesium iodide detector with scintillator size $2 \times 1 \times 0.5$ in), which allows further simplification of the model to

$$\lambda_n(\mathbf{x}) = \frac{\mathcal{I}_s}{(x_s - x_n)^2 + (y_s - y_n)^2} + \lambda_b. \tag{4.2}$$

Thus, the measurement likelihood function is defined as

$$p(z_n|\theta) = \mathcal{P}(z_n; \lambda_n(\mathbf{x})) = \frac{e^{-\lambda_n(\mathbf{x})} * \lambda_n(\mathbf{x})^{z_n}}{z!}. \tag{4.3}$$

For the simulation environment with obstructions present (non-convex), the following binary attenuation modification was used when the detector does and does not have *line of sight* (LOS):

$$\lambda_n(\mathbf{x}) = \begin{cases} \frac{\mathcal{I}_s}{(x_s - x_n)^2 + (y_s - y_n)^2} + \lambda_b & \text{LOS}, \\ \lambda_b & \text{NLOS}. \end{cases} \tag{4.4}$$

We define the *signal-to-noise ratio* (SNR) as,

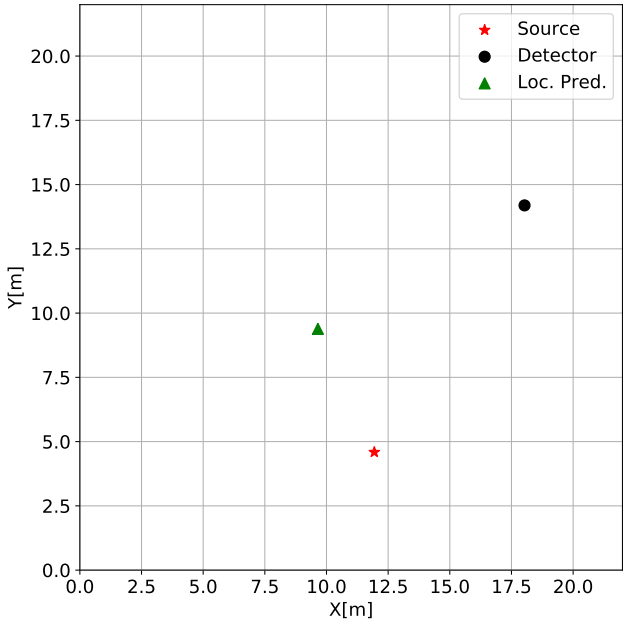
$$\text{SNR} = \frac{\mathcal{I}_s / D_{init}^2 + \lambda_b}{\lambda_b}, \tag{4.5}$$

where D_{init} is the initial Euclidean distance between the source and detector positions. This equation was also used for the non-convex environments to maintain consistency even though it is not strictly true. Figure 4.1 shows a set of randomly sampled episode parameters for convex (4.1*a*) and non-convex (4.1*b*) environments. The environment was implemented using the open-source Gym interface developed by OpenAI [43].

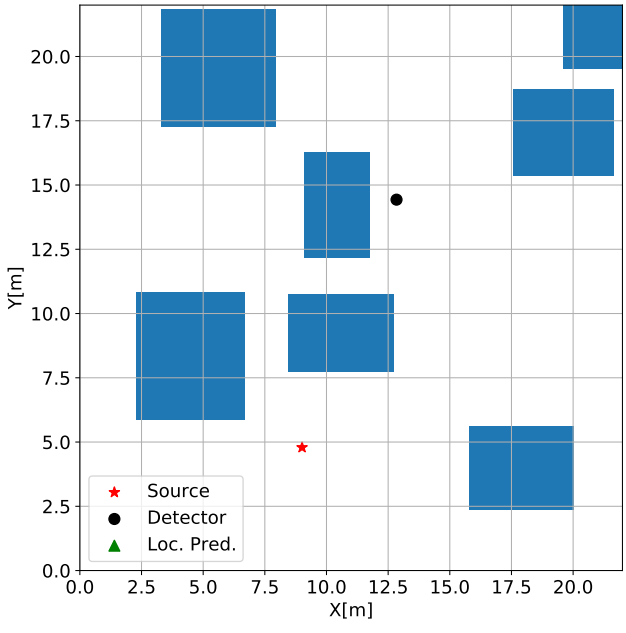
4.1.2 Reward Function

The reward function defines the objective of the RL algorithm and completely determines what will be learned from the environment. Reward is only utilized for the update of the weights during the optimization phase and does not directly factor into the RL agent's decision making during an episode. The reward function for the convex and non-convex environment is as follows,

$$r_{n+1} = \begin{cases} 0.1 & \text{if } \psi_{n+1} < \min_n \psi_n, \\ -0.5 * \frac{\psi_{n+1}}{D_{\text{search}}} & \text{otherwise.} \end{cases} \tag{4.6}$$



(a) Convex environment



(b) Non-convex environment

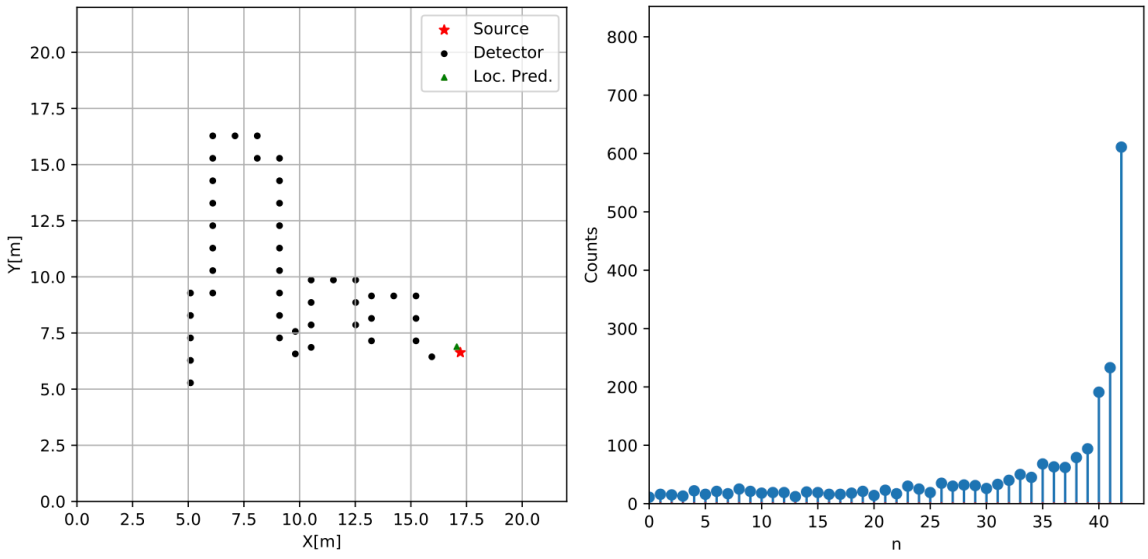
Figure 4.1: A sample of the starting conditions for a convex and non-convex environment. In both environment types, the red star is the source position, the black circle is the detector position, and the green triangle is the agent's prediction of source position. In the non-convex environment, the blue rectangles are obstructions that block line of sight between the source and detector and have to be navigated around.

Here, the source-detector shortest path distance is defined as ψ, and D_{search} defines the largest Euclidean distance between vertices of the search area. The shortest path distance is essential for the non-convex environment and becomes the Euclidean distance when there is LOS. The normalization factor in the negative reward provides an implicit boundary to the search area. This reward scheme incentivizes the RL agent to find the source in the fewest actions possible as the negative reward is weighted more heavily. The reward magnitudes were selected so that standardization was not necessary during the training process as mean shifting of the reward can adversely affect training [44].

The reward function was designed to provide greater feedback for the quality of an action selected by the RL agent in contrast to only constant rewards. For example, in the constant negative reward case, if the RL agent initially takes actions that increase D above the previous closest distance for several timesteps and then starts taking actions that reduce D, it will get the same negative reward even though it is has started taking more productive actions. This distance-based reward gives the RL agent a more informative reward signal per episode during the learning process. Figure 4.2 shows a sample of the RL agent operating within the environment, the radiation measurements it observes, and the reward signal it receives.

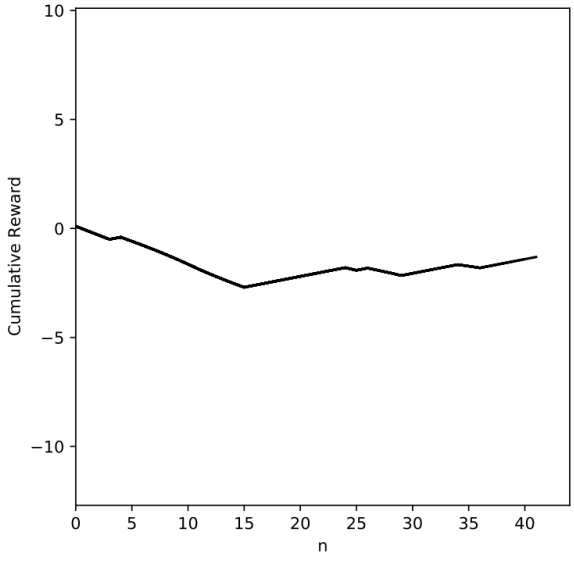
4.1.3 Graph Representation

The shortest path distance and LOS had to be calculated millions of times per training session for the reward function and radiation measurement, respectively. To carry this out efficiently, the environment was mapped to a visibility graph each time the parameters were reset. A visibility graph is an undirected, acyclic graph where each node is a location in $2D$ Euclidean space and each edge determines the accessibility between nodes. Obstructions remove nodes and edges from the set of traversable nodes. We used the open-source C++ VisiLibity1 package by Obermeyer to efficiently



(a) Detector path.

(b) Radiation measurements.



(c) Cumulative reward

Figure 4.2: Example of the RL agent during an episode. Figure 4.2a shows the detector position at each timestep as it moves closer to the source. Figure 4.2b shows the radiation counts measurements at each timestep corresponding with the detector position. Figure 4.2c shows the cumulative reward signal that the RL agent uses during training. The reward signal is only used after an episode for weight updates and during testing the reward signal is not provided.

calculate the shortest path at each timestep with the A^* search algorithm [45]. This results in a time complexity of $\mathcal{O}(|\text{nodes}|^2)$ where $|\cdot|$ is the cardinality of the set of nodes. LOS was determined using ray tracing.

4.1.4 Training Configuration

Table 4.1 shows all the environment parameters for the training of the policy in convex and non-convex scenarios. All the intervals in the table indicate sampling from a uniform distribution per episode. The source and detector starting positions were sampled uniformly from the area dimensions. These area dimensions did not explicitly constrain where the detector could move and were selected to provide a proof-of-concept for smaller-scale application scenarios. The algorithms presented here could be easily scaled to larger area dimensions provided the source intensity was increased proportionally. The source and background rates were considered constant per episode and chosen to provide a range of SNRs with the lowest ratio at 1 and the highest ratio at 2. We focused on optimizing the policy for the more challenging case of lower SNR source search by requiring the initial source and detector Euclidean distance to be at least 10 m. This resulted in a Gamma distribution of initial SNRs as shown in Figure 4.3.

Detector step size was fixed at 1 m/sample and movement direction in radians was limited to the set, $\mathcal{U} = \{i * \frac{\pi}{4} : i \in [0, 7]\}$. The RL implementation can easily be adapted to handle more discrete directions and variable step sizes or even continuous versions of these quantities. These two constraints were made to limit the computational requirements (Section 4.3) for the comparison algorithm. Maximum episode length was set at 120 samples to ensure ample opportunity for the policy to explore the environment, especially in the non-convex case. Episodes were considered completed if the detector came within 1.1 m of the source or a failure if the number of samples reached the maximum episode length. The termination distance was selected to cover

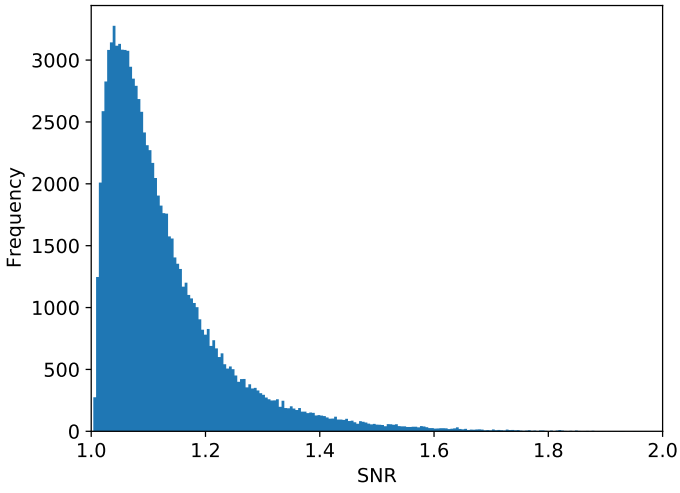


Figure 4.3: Distribution of initial episode SNRs resulting from uniform sampling of the environment parameters shown in Table 4.1. This biased the policy towards operation in the more challenging lower SNR contexts.

a range of closest approaches as the detector movement directions and step size are fixed.

The state space was eleven dimensions to include eight detector-obstruction range measurements for each movement direction. This modeled some range sensing modality such as an ultrasonic or optical sensor. The maximum range was selected to be 1.1 m to allow the controller to sense obstructions within its movement step size. The range measurements were normalized to the interval $[0, 1]$, where 0 corresponds to no obstruction within range of the detector. If the policy selected an action that moved the detector within the boundaries of an obstruction, then the detector location was unchanged for that sample.

4.2 RAD-A2C Implementation

4.2.1 Architecture

The RAD-A2C is composed of a *particle filter gated recurrent unit* (PFGRU), one *control gated recurrent unit* (CGRU) module to encode the inputs over time for action selection, and three linear layers. Figure 4.4 provides an illustration of the information

Parameter	Value
Area Dimensions	22 m $\times$ 22 m
Src. det. initial positions	[-22, 22] m
Src. rate	$[1 \times 10^2, 1 \times 10^3]$ cps
Background rate	[10, 50] cps
State space	11
Action space	8
Max. search time	120 samples
Velocity	1 m/sample
Termination dist.	1.1 m
Min. src.-det. initial dist.	10 m
Number of obstructions	[1,5]
Obstruction dim.	[2,5] m

Table 4.1: Radiation source simulation for convex and non-convex environment parameters. The brackets indicate an interval that was uniformly sampled on a per episode basis. Src. and det. are abbreviations for source and detector, respectively.

flow through the system. At each timestep, the observation is propagated to both the PFGRU and the *actor critic* (A2C) modules. The PFGRU uses a linear layer to regress its mean "particles" onto a source location, which is concatenated with the observation and fed into the A2C. The Actor layer regresses the CGRU hidden state onto a multinomial distribution over actions using a softmax function (Eq. 3.15). The Critic layer regresses the hidden state onto a value prediction. This value prediction is only necessary for the training phase and has no direct impact during inference as covered in Section 4.2. The dotted lines indicate the flow of the error gradients in backpropagation during training.

The RAD-A2C should be easily extendible to other source search scenarios such as a 3D environment, moving sources, using more advanced radiation transport simulators, and selection of detector step size and dwell time. These variations would only require a change in the dimensions of the input and output of the model, possibly of the hidden state size as well, and an appropriate update of the simulation environment/reward function. This is a major advantage of DRL as compared to hand-crafted algorithms. The downside of DRL is the long and computationally intense training costs and

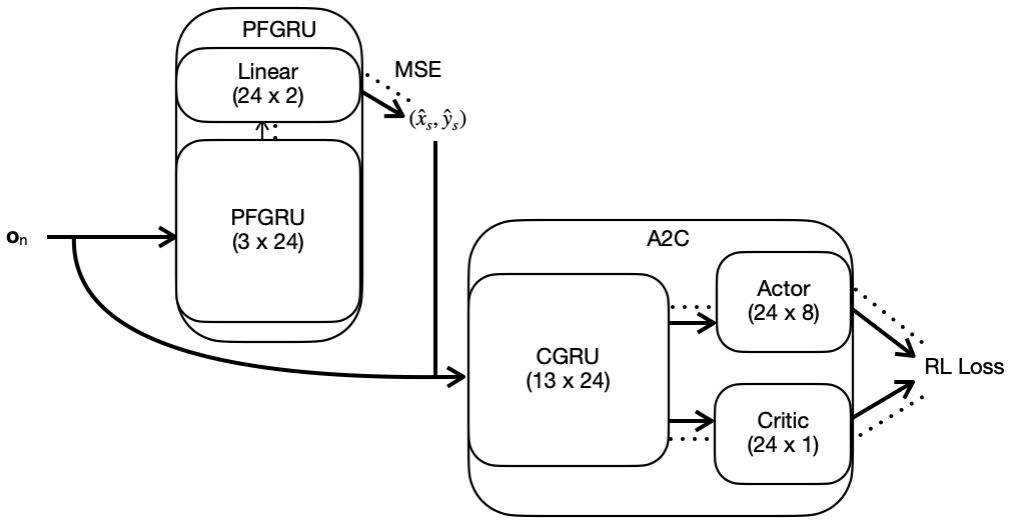


Figure 4.4: RAD-A2C source search architecture where quantities in the parenthesis denote the dimensions. The PFGRU provides a location prediction, denoted $(\hat{x}_s, \hat{y}_s)$, at each timestep, which is concatenated with the observation and fed into the A2C. The CGRU module encodes the inputs over time in its hidden state and the Actor layer selects an action from this hidden state. The Critic layer predicts the expected return from the hidden state and is only needed during training. The dotted lines indicate the gradient flow during backpropagation.

sensitivity to hyperparameters. A weakness of our RAD-A2C implementation is that source intensity is not predicted by the PFGRU as this would require prior knowledge about the upper limit of the intensity. We opted for scenario generalization by performing search without a source intensity estimate. While source intensity is often of interest in radiation source localization scenarios, an additional estimator such as least squares fitting could be used in conjunction with our model. Future work could examine the impact of source intensity prediction on RAD-A2C search performance.

Table 4.2 shows the hyperparameters that resulted in the strongest performance for the RL agent from the parameter sweep. The total training time for a single RL agent running on 10 cores took approximately 60 hours. Section 4.2.3 will go into further detail of the results of the parameter ablation analysis.

Parameter	Value
Epochs	$3,000$
Episodes per epoch	40
Batch Size	10
Tot. weights & biases	$7,443$
CGRU hidden size	(13×24)
PFGRU hidden size	(11×24)
PFGRU particles	40
Learning Rate A2C	3×10^{-4}
Learning Rate PFGRU	5×10^{-3}
Optimizer	Adam
(γ,λ,η)	$(0.99, 0.9, 0.105)$

Table 4.2: Hyperparameter values with the strongest performance for the RL agent from our parameter sweep.

4.2.2 Training

A common technique in DL is to standardize the input data to increase training stability and speed. This is done by subtracting the mean and dividing by the standard deviation per feature across a batch of input data. The RL context does not have easy access to the data statistics since it is collected and processed online. We used a technique proposed by Welford for estimating a running sample mean and variance as follows [46],

$$\begin{aligned} \mu_{n+1} &= \mu_n + \frac{(o_{n+1} - \mu_n)}{n} \\ S_{n+1} &= S_n + (o_{n+1} - \mu_n)(o_{n+1} - \mu_{n+1}) \\ \sigma^2_{n+1} &= \frac{S_{n+1}}{n}, \end{aligned} \tag{4.7}$$

where $\mu_0 = o_0$, $S_0 = 0$. The statistics were updated after each new observation and then standardization was performed.

The estimate of the gradient iterate (Eq. 3.21) is improved by increasing the number of histories being averaged over. Schulman et al. improved training scalability by instantiating copies of the RL agent and environment on different CPU cores to

LR	**Ep. Len. [samples]**	$[2.5^{\text{th}}, 97.5^{\text{th}}]$	**Ep. Comp. %**	$[2.5^{\text{th}}, 97.5^{\text{th}}]$
1×10^{-4}	47	$[22, 88]$	96	$[15, 100]$
3×10^{-4}	39	$[17, 86]$	99	$[65, 100]$
5×10^{-4}	44	$[22, 95]$	96	$[28, 100]$
7×10^{-4}	48	$[20, 91]$	52	$[0, 100]$
9×10^{-4}	46	$[23, 91]$	92	$[22, 100]$

Table 4.3: Median completed episode length and median episode completion percentage of the RAD-A2C relative to the learning rate for the A2C. The percentiles correspond to the 2.5^{th} and 97.5^{th}. All other A2C hyperparameters are as specified in Table 4.2.

parallelize episode collection. Each RL agent computes its parameter gradients after all episodes for an epoch have been collected. The gradients are then averaged across all the cores and a weight update is performed per core. An important distinction in the implementation used here is the environment variation across the CPU cores. All of the sampled quantities (Section 4.1.4) were different per core and fixed per epoch resulting in a more generalized policy. This is because the averaged gradient step will be in the direction that improves performance across a diverse set of environments. Tobin et al. proposed a similar idea called domain randomization that aimed to bridge the gap between RL simulators and reality by introducing extra variability into the simulator [15].

4.2.3 RAD-A2C Ablation Analysis

An ablation analysis was performed over a subset of the RAD-A2C hyperparameters, specifically, A2C *learning rate* (LR) in Table 4.3 and CGRU/PFGRU *hidden state* (H) size in Table 4.4. All other A2C hyperparameters are as specified in Table 4.2. Additionally, we compared agent performance with the PFGRU module and a simple regression GRU module. The fixed assessment set was composed of 100 randomly sampled episodes (convex/non-convex, low/med/high SNRs) with 100 Monte Carlo runs per episode. The metrics of performance across episodes were median completed episode length, median completed episode percentage, and the 2.5^{th} and 97.5^{th} percentiles, more details can be found in Section 5.1.2.

H size	**Ep. Len. [samples]**	$[2.5^{\text{th}}, 97.5^{\text{th}}]$	**Ep. Comp. %**	$[2.5^{\text{th}}, 97.5^{\text{th}}]$
16	38	$[17, 83]$	98	$[58, 100]$
24	39	$[17, 86]$	99	$[65, 100]$
32	40	$[18, 89]$	99	$[52, 100]$
48	42	$[24, 98]$	95	$[29, 100]$

Table 4.4: Median completed episode length and median episode completion percentage of the RAD-A2C relative to the hidden state size for the PFGRU and the CGRU. The percentiles correspond to the 2.5^{th} and 97.5^{th}. All other A2C hyperparameters are as specified in Table 4.2.

The median performance for both metrics was mostly stable across both hyperparameters. This could be due to the randomness of our proposed training procedure that increases the likelihood of taking useful gradient update steps per weight update. The best hyperparameters were a LR of 3e−4 and an H size of 24 evidenced by a high median completion percentage and the tightest percentiles. The LR of 7e−4 highlights the well-known issue in DRL of sensitivity to hyperparameter tuning [44].

Another localization module was considered that used a GRU with a linear layer for source location prediction (REG-GRU) instead of the PFGRU. The REG-GRU is more efficient computationally and decreased the total training time considerably. The best performing hyperparameters from the RAD-A2C model were used to train the REG-GRU variation. This resulted in a median completed episode length of 45 and median episode completion percentage of 95% with 2.5^{th} percentile of 9 and 97.5^{th} percentile of 100. The lack of completion percentage consistency across episodes for the REG-GRU highlights the necessity and capability of the PFGRU.

4.3 Information Driven Controller

Information-driven search is an information-theoretic framework for sequential action selection. This framework endows the controller with the ability to update its path plan as new observations become available as opposed to relying only on whether the target has been detected or not [47]. Information is integrated across time by tracking the posterior probability density of states of interest. This can quickly become

computationally prohibitive and so heuristic methods such as the BPF (Section 2.5.1) are employed. Section 4.3.1 gives an overview of the parameters used in the BPF and the resampling scheme. Section 4.3.2 and 4.3.3 detail the two information metrics from the literature that have been applied in the radiation source search context.

4.3.1 Bootstrap Particle Filter (BPF)

The BPF is typically used to track a dynamic process over time. In our context, the nuclear source intensity and coordinates are fixed throughout an episode. We adapt the BPF for parameter estimation with a random walk process model that has very low variance Gaussian noise. The initial particles were sampled uniformly from fixed intervals as specified in Table 4.5. Eq. 4.2 and Eq. 4.3 are the measurement model and likelihood, respectively. The background rate, λ_b, was considered constant and known.

Sequential importance resampling (SIR) is a technique to combat particle degeneracy and occurs when the number of effective particles drops below a given threshold. We selected the *Srinivasan sampling process* (SSP) resampling proposed by Gerber et al. because of asymptotic convergence of the error variance [48]. Additionally, SSP resampling requires only $\mathcal{O}(N_p)$ operations. See [48] and [49] for more details.

4.3.2 Fisher Information Matrix (FIM)

The FIM is a measure of the information content of a measurement relative to the measurement model. It was first used in optimal observer motion for bearings-only tracking by Hammel et al. [50]. In their implementation, the controller selects the action at each timestep that maximizes the determinant of the FIM (system observability), which is equivalent to minimizing the area of the uncertainty ellipsoids around the state estimates. This arises from the connection between the FIM and the *Cramér-Rao lower bound* (CRB).

The CRB provides a lower bound on the error covariance of an unbiased estimator and is the inverse of the FIM [51]. The FIM is the Hessian of the log-likelihood and is denoted as follows,

$$J_{n+1}(\mathbf{x}) = -\mathbb{E}[\nabla_x \nabla_x^T \ln(p(z_{n+1}|\mathbf{x}))], \tag{4.8}$$

where T denotes the transpose. Morelande et al. derived the closed form FIM for the radiation source localization problem as [5],

$$J_{n+1}(\mathbf{x}) = \frac{\nabla_{\mathbf{x}} \lambda_{n+1}(\mathbf{x}) \nabla_{\mathbf{x}}^T \lambda_{n+1}(\mathbf{x})}{\lambda_{n+1}(\mathbf{x})}, \tag{4.9}$$

where $\lambda_n(\mathbf{x})$ is defined in Eq. 4.2. This resulted in the following gradient for each parameter

$$\begin{aligned} \frac{\delta \lambda_n}{\delta \mathcal{I}_s} &= \frac{1}{(x_n - x_s)^2 + (y_n - y_s)^2}, \\ \frac{\delta \lambda_n}{\delta x_s} &= \frac{2(x_n - x_s)\mathcal{I}_s}{[(x_n - x_s)^2 + (y_n - y_s)^2]^2}, \\ \frac{\delta \lambda_n}{\delta y_s} &= \frac{2(y_n - y_s)\mathcal{I}_s}{[(x_n - x_s)^2 + (y_n - y_s)^2]^2}. \end{aligned} \tag{4.10}$$

Ristic et al. used the BPF particles at each time step to calculate the FIM as follows,

$$J_{n+1}(x_n) \approx \sum_{j=1}^{N_p} J_{n+1}(\mathbf{x}_n^j) w_n^j, \tag{4.11}$$

due to better performance when the posterior is multi-modal [9]. They applied this formulation to action selection in the radiation source search in the following manner,

$$a_{n+1} = \arg\max_{u_{n+1,L}} \left[\sum_{l=n+1}^{L} \mathrm{tr}(J_l(u_l)) \right], \tag{4.12}$$

where L is the number of lookahead steps, tr() is the matrix trace, and u_n is the control vector that determines the detector's next position.

Helferty et al. proposed to use the trace of the CRB as it is a sum of squares of the axes of the uncertainty ellipsoid [52]. This is also known as A-optimality in the

optimal experimental design literature [53]. Ristic et al. maximized the trace of the FIM that should correspond to maximizing the information, however, it is beyond the scope of this work to show the relation between these two criteria. This control strategy will result in the optimal trajectory for minimizing the uncertainty of the estimated quantities given perfect source information (i.e. low or no measurement error). The source information in the nuclear source search context is not perfect due to the stochastic nature of nuclear decay and background radiation. Additionally, the FIM is not well defined for initial search conditions where the background radiation dominates the signal from the source, i.e. when the source-detector distance is large and/or the background rate is high.

4.3.3 Rényi Information Divergence (RID)

Ristic et al. proposed another information-driven search strategy to address the shortcomings of the FIM-based approach. This approach is based upon the RID, also known as α-divergence, a general information metric that quantifies the difference between two probability distributions. In Bayesian estimation, maximizing this difference corresponds to reducing the uncertainty around the state estimates. The use of RID was first proposed in the sensor management context by Kreucher et al. [54]. The RID is defined as,

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1} \ln \left[\int P^\alpha(x) Q^{1-\alpha}(x) dx \right], \tag{4.13}$$

where α specifies the order. In the limit as α approaches one, the RID approaches the Kullback-Leibler Divergence [54].

Ristic et al. adapted the RID for action selection in the nuclear source search context with a BPF [10]. The general flow of the algorithm is to apply an action from the set of actions to get the next potential detector position, calculate the expected

posterior density for that action over a measurement interval, and then select the action that resulted in the greatest RID. The particle approximation of the RID is shown in the following equation,

$$\mathbb{E}[D_\alpha(p(\mathbf{x}^{u_{n+1}}|z), p(\mathbf{x}|z))] \approx \frac{1}{\alpha - 1} \sum_{z=Z_0}^{Z_1} p(z|\mathbf{x}) \ln \left[\frac{p_\alpha(z|\mathbf{x}^{u_{n+1}})}{p(z|\mathbf{x})^\alpha} \right], \tag{4.14}$$

where $\mathbf{x}^{u_{n+1}}$ denotes the change in detector position after taking action u_{n+1}, Z_0, Z_1 is a measurement interval, and $z \in \mathbb{N}$. The density $p_\alpha(z|\mathbf{x}^{u_{n+1}})$ is approximated after filtering the latest measurement and particle resampling as,

$$p_\alpha(z|\mathbf{x}^{u_{n+1}}) = \sum_{j=1}^{N_p} w_n^j p(z|\mathbf{x}_n^{j,u_{n+1}})^\alpha, \tag{4.15}$$

and $p(z|\mathbf{x})$ results from the particle approximation of the marginal distribution of a measurement. Like the FIM, the RID can also be computed for L-step planning.

4.3.4 Hybrid RID-FIM Controller

We propose a hybrid controller that utilizes either the RID or FIM as metrics for action selection. This was motivated in part by the empirical observation that the RID controller would often get stuck oscillating between two positions that were just above our termination criteria for source-detector distance resulting in incomplete episodes. The FIM is a poor control metric when there is little information available as is often the case at the start of a search. The RID is more computationally expensive than the FIM but provides a principled control method across a range of information contexts. Thus, the RID was used for control at the beginning of each episode until the RID reached a sufficient threshold, then the metric was switched over to the FIM for the remainder as shown in Alg. 2.

We decided on myopic (one-step lookahead) planning due to the exponential increase in computational cost inherent to both metric calculations. Additionally,

Parameter	Value
N_p	$6,000$
Process noise XY	15
Process noise $\mathcal{I}_s$	1
Prior XY	$[0, 25]$m
Prior $\mathcal{I}$	$[100, 1000]$cps
Resampling threshold, β	1.0
Lookahead, L	1
Order, α	0.6
Switch threshold, η	0.36
Meas. interval $[Z_0, Z_1]$	$\pm$75 cps

Table 4.5: Parameter values for the BPF and RID-FIM.

many source search scenarios will have high uncertainty in the state estimates for many timesteps so planning far in advance is not advantageous. Myopic search is often sub-optimal but is a fair tradeoff when the problem dynamics are stable [54]. The parameter values for the RID-FIM, as well as the BPF, are detailed in Table 4.5. All parameters were selected by a parameter sweep over a set of 100 randomly sampled episodes where the selection criteria was shortest average episode length and most episodes completed.

Algorithm 2: RID-FIM Controller

Input:$\{\mathbf{x}_0^j, w_0^j\}_{j=1}^{N_p}$, set RID FLAG to 1, switch threshold η, effective particles threshold β, measurement interval $[Z_0, Z_1]$
Receive init. measurement, z_0, perform prediction and filtering of particles
while episode not terminated **do**
  if RID FLAG **then**
    Calculate RID according to 4.14 over $[Z_0, Z_1]$
  else
    Calculate FIM according to 4.11
  end if
  Select action that maximizes information metric
  Receive z_{n+1}, perform prediction and filtering of particles
  if $N_{eff} < \beta * N_p$ **then**
    Resample and reweight particles
  end if
end while

4.3.5 Posterior Cramér-Rao Lower Bound (PCRB)

The BPF is a biased estimator as it only uses a finite number of particles. Thus, the previously defined CRB does not hold [55]. The PCRB provides a lower bound on the *root-mean-square error* (RMSE) performance for a biased estimator. Tichavsky proposed the PCRB for discrete-time nonlinear filtering [56], however, we follow a similar formulation found in Bergman's dissertation [57]. The PCRB is determined recursively in the following manner,

$$\begin{aligned} P_{0|0}^{-1} &= \Sigma^{-1}\Lambda^{-1}\int_x \nabla_x \lambda_0(\mathbf{x})\nabla_x^T \lambda_0(\mathbf{x})d\mathbf{x}, \\ P_{n+1|n+1}^{-1} &= Q_n + R_{n+1} - S_n^T(P_{n|n}^{-1} + V_n)^{-1}S_n, \end{aligned} \tag{4.16}$$

where the terms are S_n, V_n, and Q_n are all the same inverse process noise covariance matrix, denoted as Σ^{-1}. This arises from the fact that our process model is a random walk with Gaussian noise for each state. The term R_n is the FIM defined in Eq. 4.8. The prior, $P_{0|0}$, is a result of the uniform distribution of the particles where Λ is a diagonal matrix of the uniform probabilities for each parameter. More details of the derivation of the PCRB and prior can be found in Theorem 4.5 and Section 7.3, respectively [57].

We average the RMSE and PCRB over the Monte Carlo evaluations resulting in the following formulation,

$$\sqrt{\frac{1}{K}\sum_{i=1}^{K}\left\|\hat{\mathbf{x}}_n^i - \mathbf{x}_n^i\right\|^2} \gtrsim \sqrt{\frac{1}{K}\sum_{i=1}^{K}\mathrm{tr}(P_n^i)}, \tag{4.17}$$

where K is the total number of episodes and $\gtrsim$ denotes that the inequality only holds approximately for finite K [57]. The PCRB provides an indicator of the suboptimality of an estimator and so we use it to directly compare the performance of the A2C with the RID-FIM. This is accomplished by evaluating the A2C with the exact same BPF

estimator used with the RID-FIM for the source location state estimates. Not only can the estimator RMSE be compared against the PCRB, but the PCRBs resulting from both controllers can be compared as well. This will serve as a proxy for the quality of the control path generated by each controller.

4.4 Gradient Search (GS)

We use the simple GS algorithm implemented by Liu et al. [14]. GS relies on sampling the gradient of the radiation field for each search direction at each timestep. This is not an efficient algorithm as the detector must make D moves per action selection but serves as a useful baseline for performance comparison. The action selection is made stochastic by sampling from a multinomial distribution, denoted multi(n,p), over actions with probabilities proportional to the softmax of the gradients to avoid the trapping of local optima. GS is summarized by the following equation,

$$a_{n+1} \sim \text{multi}(|\mathcal{U}|, \text{softmax}([\frac{1}{q}\frac{\delta z_{n+1}}{\delta u_1}, \ldots, \frac{1}{q}\ \frac{\delta z_{n+1}}{\delta u_{|\mathcal{U}|}}])), \tag{4.18}$$

where u is the detector position after action i, σ is the softmax function (Eq. 3.15), and q is a temperature parameter. The temperature parameter was selected by a parameter sweep over a set of 100 randomly sampled episodes where the selection criteria was shortest average episode length and most episodes completed. This was done separately for both the convex and non-convex environments.

4.5 Complexity Analysis

We compare the theoretical computational complexity of the algorithms discussed in the previous sections. The notation is consistent with the rest of the thesis but will be reviewed. N_p refers to the number of BPF particles, N_{gp} refers to the PFGRU particles, M is the number of state dimensions, H is the number of hidden state dimensions for

Module	Operation Complexity $\mathcal{O}(\cdot)$
A2C	$H(H + M + \lvert\mathcal{U}\rvert + 1)$
FIM	$\lvert\mathcal{U}\rvert^L(N_p + N_p M^2)$
RID	$ZN_p\lvert\mathcal{U}\rvert^L$
BPF	$N_p(M + 1)$
PFGRU	$N_{gp}H(H + M + 1)$

Table 4.6: Complexity analysis for the information-driven controller and the RAD-A2C. The analysis is broken up into the controller modules (A2C, FIM, RID) and the localization modules (PFGRU, BPF).

the RL agent, L is the number of lookahead steps, Z is the measurement interval, and $|\mathcal{U}|$ is the number of search directions. Table 4.6 shows the Big-O complexities for the A2C, PFGRU, FIM, RID, and BPF. GS was omitted because it relies on taking multiple measurements for action selection.

Both the FIM and RID are exponential in the number of lookahead steps with a base proportional to the number of search directions. In contrast, the A2C is quadratic in the number of hidden state dimensions and grows linearly with the number of search directions. Additionally, the size of $N_{gp} * H \ll \mathrm{N}_p$. The RID is the most computationally demanding as the Z interval typically has to be sufficiently large to cover the range of possible measurements. The PFGRU is slightly worse in computational complexity than the BPF but has better scaling with state dimensions as $N_{gp} \ll \mathrm{N}_p$. The hybrid RID-FIM controller will have an upper bound on computational complexity equivalent to the RID and a lower bound equivalent to the FIM.

4.6 Summary

This chapter covered the implementation details of the main components of this thesis. The radiation source search environment is composed of a Gamma radiation model, detector dynamics, a distance-based reward signal, and the layout of obstructions (convex vs. non-convex). Source, detector, and background radiation parameters were randomly sampled from uniform intervals for each episode. We will compare our

algorithm against the information-driven hybrid RID-FIM controller with a BPF and GS over a range of SNRs in a convex environment. In the non-convex environment, only the RAD-A2C and GS will be compared. The next chapter will show the results of these evaluations and discuss the implications.

Chapter 5

Results

This chapter presents the experimental results. Section 5.1 details the experimental setup, assumptions, and evaluation criteria. Section 5.2 covers the general results for the *gradient search* (GS), *Rényi information divergence-Fisher information matrix* (RID-FIM), and our deep reinforcement learning architecture (RAD-A2C) in the convex environment and then focuses on the actor critic (A2C) controller and the RID-FIM controller performances relative to the PCRB. Section 5.3 shows the results in the non-convex environment for the RAD-A2C and the GS. Sections 5.2.4 and 5.3.3 discuss each of these experiments and the implications for our hypothesis.

5.1 Experimental Setup

5.1.1 Test Configuration

All search methods were evaluated across a range of SNRs in the convex environment. Only the A2C and GS were compared in the non-convex environment as the BPF measurement and process model do not account for obstructions. The SNRs were broadly grouped into "low", "medium", and "high" group labels, and the environment parameters were randomly sampled as specified in Section 4.1.4 to create a fixed test set. Each SNR label interval was further subdivided into four intervals to ensure a uniform distribution. Table 5.1 summarizes the distribution of the $1,000$ different environment parameter combinations per SNR label. A single combination of randomly sampled environment parameters will be referred to as an episode and the term SNR

SNR Label	Interval	Sub-Interval	Total Episodes
Low	1.0 - 1.2	0.05	1,000
Medium	1.2 - 1.6	0.05	1,000
High	1.6 - 2.0	0.1	1,000

Table 5.1: Distribution of SNRs for the fixed test set, grouped and referred to by the SNR label ("low", "medium", "high"). The interval refers to the SNR interval and the sub-interval is the further division of the SNR interval. This ensures a uniform distribution across the SNR. Each SNR label had a total of $1,000$ episodes.

will be used interchangeably with the SNR group labels. Monte Carlo simulations were performed for all experiments to determine the average performance of the algorithms. Each algorithm was run 100 times per episode and the results averaged.

5.1.2 Metrics

Weighted median completed episode length and median percent of completed episodes served as the main performance metrics. The median was selected because of the skewness of the result distributions, both in terms of the Monte Carlo simulations per episode and across all the episodes. The weighted median was used for the completed episode length with a weighting factor between $1 - 100$, determined by the number of Monte Carlo simulations that were completed by the agent per episode. The completed episode length corresponds to the number of radiation measurements required to come within the episode termination distance of the source before the maximum episode length is reached. This quantifies the agent's effectiveness in incorporating the measurements to inform exploration of the search area. Percent of episodes completed is the more important metric as the priority in radiation source search is mission completion and this works in tandem with the completed episode length to characterize the agent's performance. An ideal agent would have a low median episode length and a high median percent of episodes completed. We also consider the episode *root-mean-square error* (RMSE) of the source location prediction to characterize the estimator performance, but it was of lesser importance than the

aforementioned metrics. The episode RMSE of the source location prediction measures how well the localization module tracks the true source location during an episode.

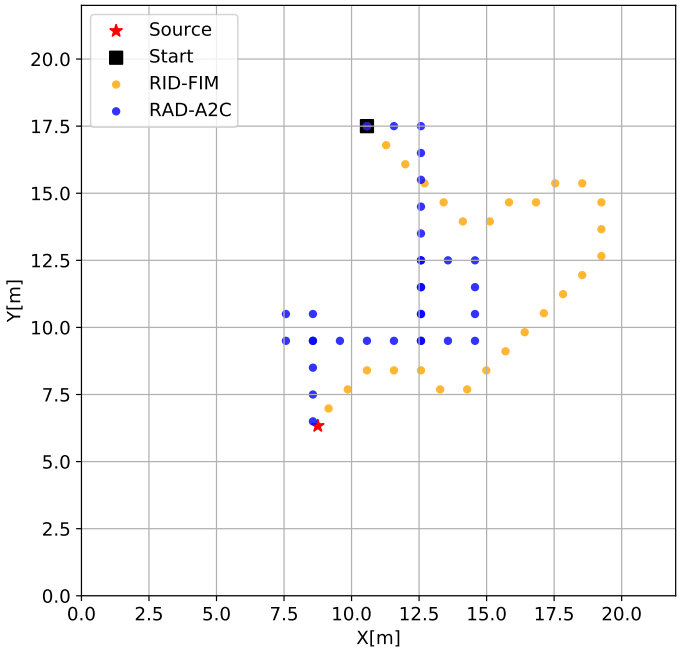
5.1.3 Experiments

Three sets of experiments were run in the radiation source search environment to assess the performance characteristics of our proposed RAD-A2C architecture. The first experiment focused on the comparison of all of the search algorithms discussed in Chapter 4. The second experiment assessed the RID-FIM and A2C action selection quality with BPF performance as a proxy. The final experiment looked at the performance of the GS and RAD-A2C in a non-convex environment where the number of obstructions was varied. We hypothesize that the RAD-A2C will achieve better performance than GS across SNR and environment convexity and similar or worse performance to the RID-FIM in the convex environment.

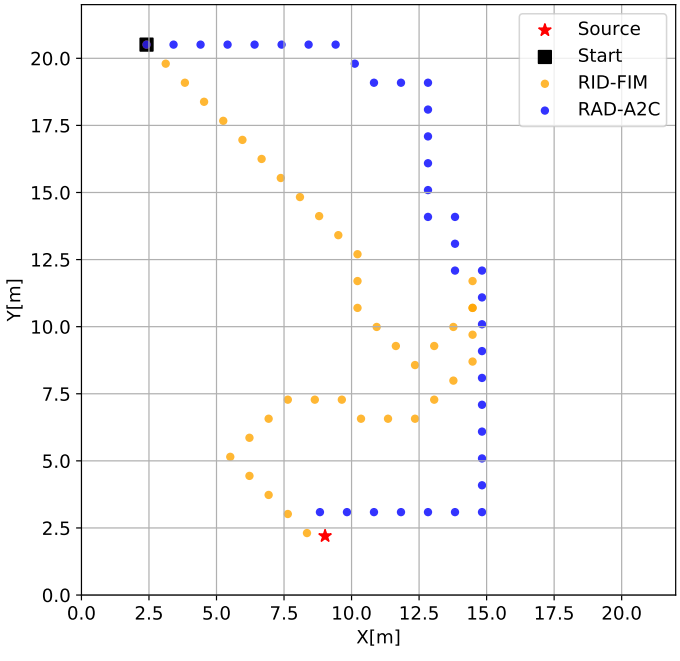
5.2 Convex Environment

5.2.1 Detector Path Examples

Two detector paths for the RAD-A2C and the RID-FIM in two different SNR configurations of the convex environment are shown in Figures 5.1a, 5.1b. The source prediction marker was omitted to reduce clutter. Both algorithms must explore the area as they search for radiation signal above the noise floor. In the high SNR configuration, both algorithms make sub-optimal decisions that move the detector away from the source, a result of the probabilistic nature of the measurement process. However, they both quickly adjust and successfully find the radiation source. The detector starts much further from the source in the low SNR configuration and the detector must take many more actions before picking up any signal. In both scenarios, the RID-FIM makes more diagonal movements relative to the RAD-A2C.



(a) High SNR configuration.



(b) Low SNR configuration.

Figure 5.1: Two detector paths for the RAD-A2C and the RID-FIM in high and low SNR configurations of the convex environment overlayed on a single plot. The black square denotes the detector starting position and the red star represents the radiation source. Both algorithms must explore the area as they search for radiation signal above the noise floor.

Method	Low	Medium	High
PFGRU	5.85 ± 0.26 m	$5.99 \pm .24$ m	$6.17 \pm .23$ m
BPF	$6.44 \pm .47$ m	$6.31 \pm .46$ m	$5.98 \pm .44$ m

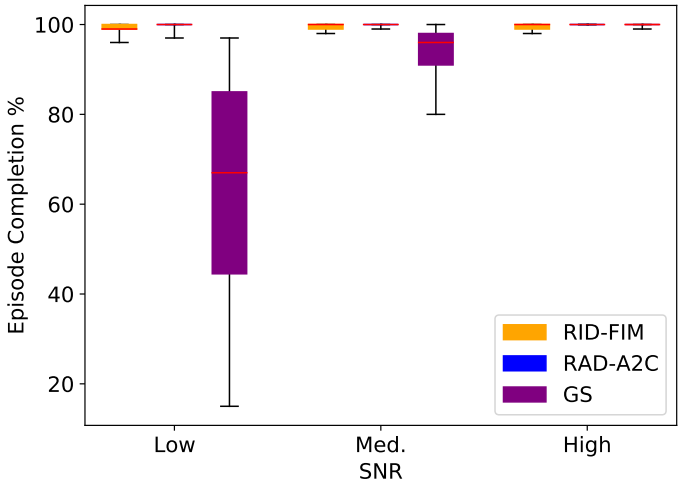
Table 5.2: Mean of the RMSE per episode for source location estimates of the PFGRU and BPF across SNR. The uncertainty is the standard error of the mean. The PFGRU does slightly better than the BPF for the lower SNRs.

5.2.2 Performance

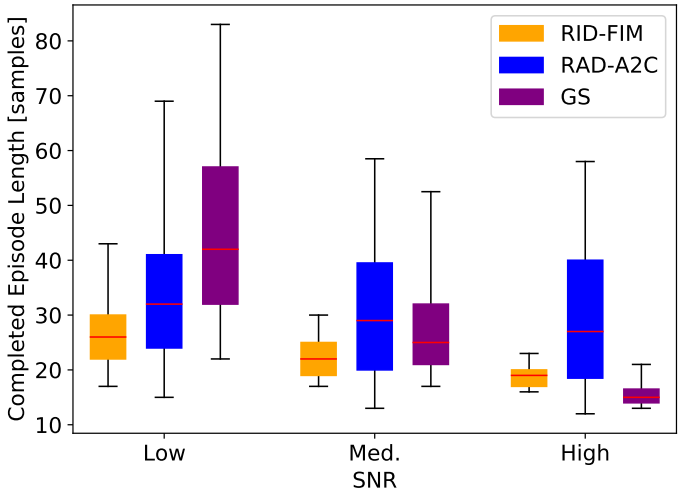
Box plots for the completed episode percentage and completed episode length for all methods in the convex environment are found in Figures 5.2a and 5.2b, respectively. The median is denoted in red, the boxes range from the first to the third quartile and the whiskers extend to the 2.5^{th} and 97.5^{th} percentiles. GS achieved the shortest episode completion length for all experiments at high SNR but performance decreased swiftly at the lower SNR levels. The RID-FIM had a consistent performance with tight boxes for both metrics at all SNRs. The RAD-A2C was the only algorithm to maintain 100% completion for all SNRs with the tradeoff being the longest median episode length for all but one of the SNRs. Figure 5.3 shows the relationship between median episode length to median episode completion. Top-performing search algorithms are located on the far right of the plot and ideally near the bottom. Table 5.2 compares the mean RMSE for the source location prediction from the *particle filter gated recurrent unit* (PFGRU) and the *bootstrap particle filter* (BPF). The PFGRU does slightly better than the BPF for lower SNRs.

5.2.3 BPF Comparison

The RID-FIM and A2C controller are compared directly by replacing the PFGRU in the RAD-A2C with the BPF. This new system will be denoted as BPF-A2C in the following plots. Swapping in the BPF for the PFGRU facilitates in-depth analysis of controller performance through the lens of the BPF performance. The estimator performance depends entirely on the quality of action selection throughout an episode



(a) Completed episode percentage.



(b) Completed episode length.

Figure 5.2: Box plots for the completed episode percentage and completed episode length against SNR in the convex environment. The median is denoted in red, the boxes range from the first to the third quartile and the whiskers extend to the 2.5^{th} and 97.5^{th} percentiles. Figure 5.2b shows the RID-FIM consistently found the source in a short amount of time even as SNR decreased. Figure 5.2a shows the RAD-A2C was the only method that completed 100% of the episodes. GS performance sharply declined for lower SNRs.

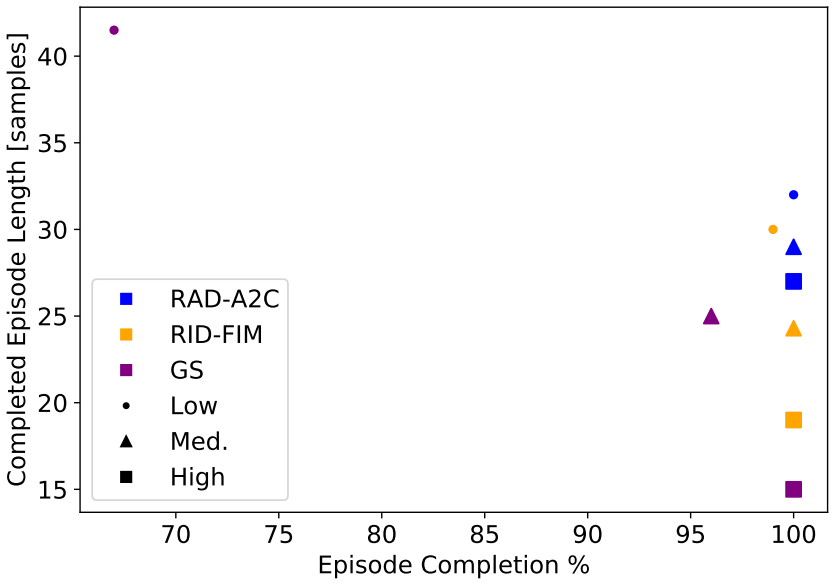


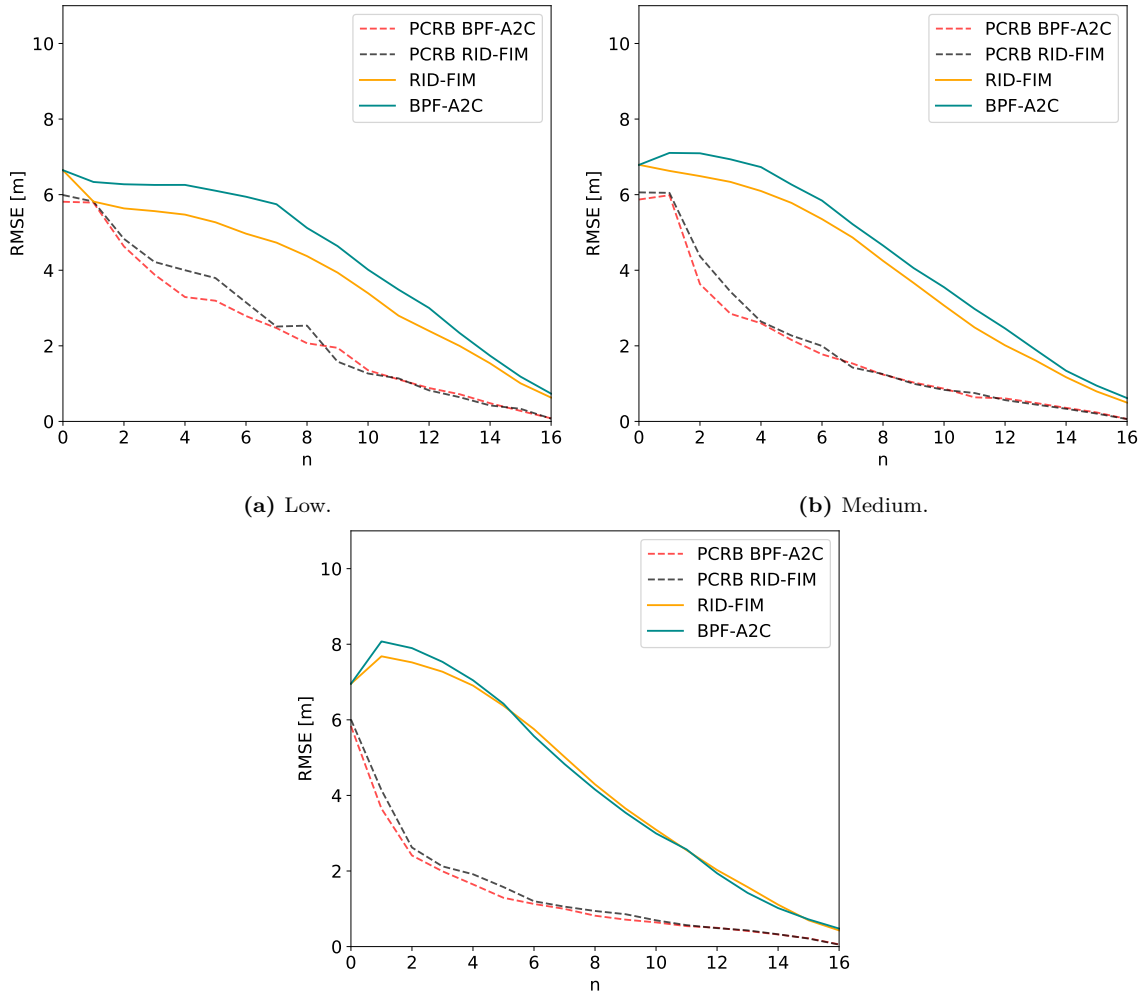
Figure 5.3: Median completed episode length against median completion rate. The marker shapes denote the SNR level and the color denotes the search method. An ideal search algorithm would be located in the bottom right of the plot for all the SNRs.

as this determines what information the estimates will be based on. Thus, we compare the RMSE for the Euclidean distance between the actual and predicted source location at each timestep for three different episode completion lengths across SNR. The trace of the Fisher information matrix per timestep is also compared for a single episode completion length across SNR.

Figures 5.4, 5.5, and 5.6, show the RMSE and *posterior Cramér-Rao lower bound* (PCRB) for the RID-FIM and the BPF-A2C for three different completed episode lengths across SNRs. The PCRB serves as a proxy for the sub-optimality of the controllers due to the use of the same estimator, as explained in Section 4.3.5. Each plot is averaged over at least 200 different episodes and at least 700 total Monte Carlo runs. An episode was only considered for this analysis if the completed episode length was the same for both algorithms in the set of the Monte Carlo runs for that episode. This ensured that RMSEs and PCRBs were only averaged over the same set of episodes.

The specific completed episode lengths were chosen to highlight an interesting variation in estimator performance that was observed across completed episode lengths ranging from $10 - 60$ samples and SNR levels. The RMSE for the RID-FIM is lower or equal to the BPF-A2C at a completed episode length of 17 across SNR. This changes for a completed episode length of 20 where the RID-FIM RMSE is only lower than the BPF-A2C at the lowest SNR. For the completed episode length of 28, the BPF-A2C now has a lower RMSE than the RID-FIM for all SNRs. In all of the plots, the PCRB for the BPF-A2C is slightly lower or equal to the PCRB for the RID-FIM. The PCRB decreases at a faster rate for the high SNR compared to the low SNR. Estimator RMSE consistently approaches the PCRB by the end of an episode. The intersection of the RMSE curves seen in Figure 5.5b was observed at different completed episode lengths (not shown) relative to SNR. This occurred at a completed episode length of 17 for the high SNR, at 20 for medium SNR, and 23 for low SNR. The RMSE initially increased for the high SNR in direct relation with the completed episode length in all the RMSE plots shown.

Figure 5.7 shows the log trace of the Fisher information matrix (Fisher score) for a completed episode length of 19 across SNR averaged over at least 650 episodes. The scores are roughly equal until they reach approximately -5 and then the BPF-A2C method rapidly increases. This occurs at different timesteps relative to SNR and the BPF-A2C exhibits a consistent peak and decline pattern. The RID-FIM has a more steady information gain per timestep and ends up with a higher final score for all SNRs. This pattern was observed across a majority of the completed episode lengths. The sharp increase in the Fisher score for the RID-FIM indicates the sample when enough information is available for the FIM metric to be used for action selection. At shorter completed episode lengths (< 16), the RID-FIM Fisher score is always above the BPF-A2C Fisher score.

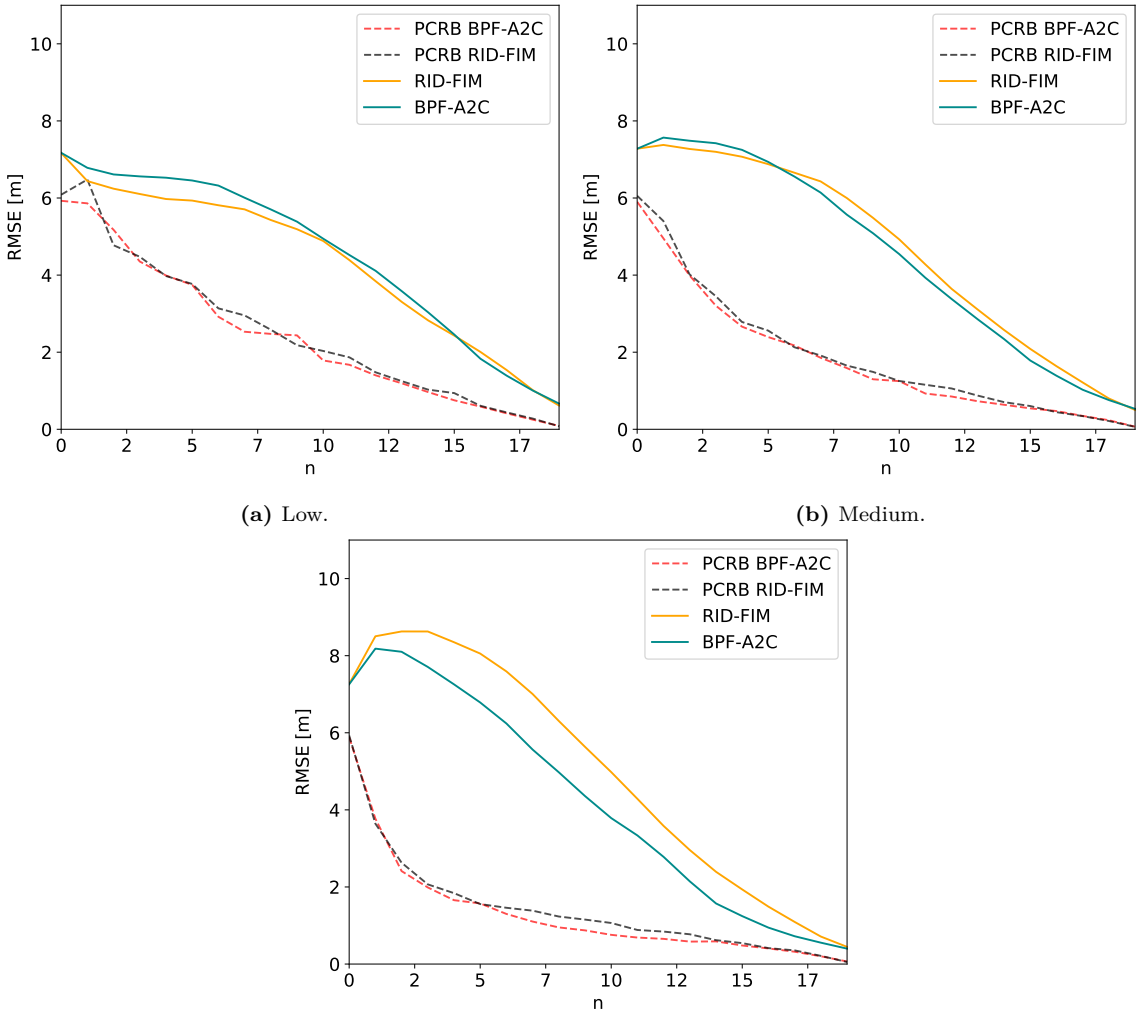


(a) Low.

(b) Medium.

(c) High.

Figure 5.4: Comparison of the Monte Carlo RMSE for BPF estimation of the source location at each timestep for a completed episode length of 17. Each plot contains the BPF PCRB and RMSE for the RID-FIM and BPF-A2C controllers averaged over at least different 200 episodes. (5.4a) is at low SNR, (5.4b) is at medium SNR, and (5.4c) is at high SNR. The RID-FIM has a lower RMSE than the BPF-A2C for the low and medium SNR but the RID-FIM's action selection was solely dependent on potentially spurious BPF state estimates, which allowed the BPF-A2C to match the RID-FIM performance at the high SNR.

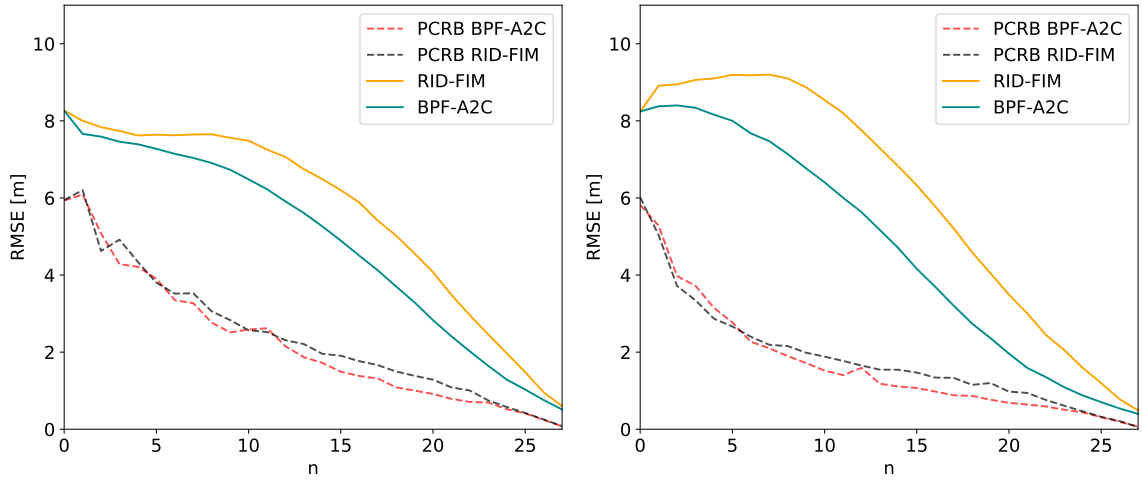


(a) Low.

(b) Medium.

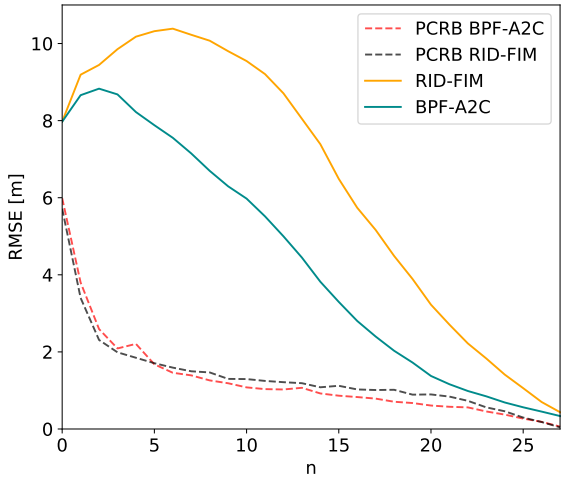
(c) High.

Figure 5.5: Comparison of the Monte Carlo RMSE for BPF estimation of the source location at each timestep for a completed episode length of 20. Each plot contains the BPF PCRB and RMSE for the RID-FIM and BPF-A2C controllers averaged over at least different 400 episodes. (5.5a) is at low SNR, (5.5b) is at medium SNR, and (5.5c) is at high SNR. The RID-FIM has a lower RMSE than the BPF-A2C for the low SNR but the RID-FIM's action selection was solely dependent on potentially spurious BPF state estimates, which caused the BPF-A2C to outperform the RID-FIM at medium and high.



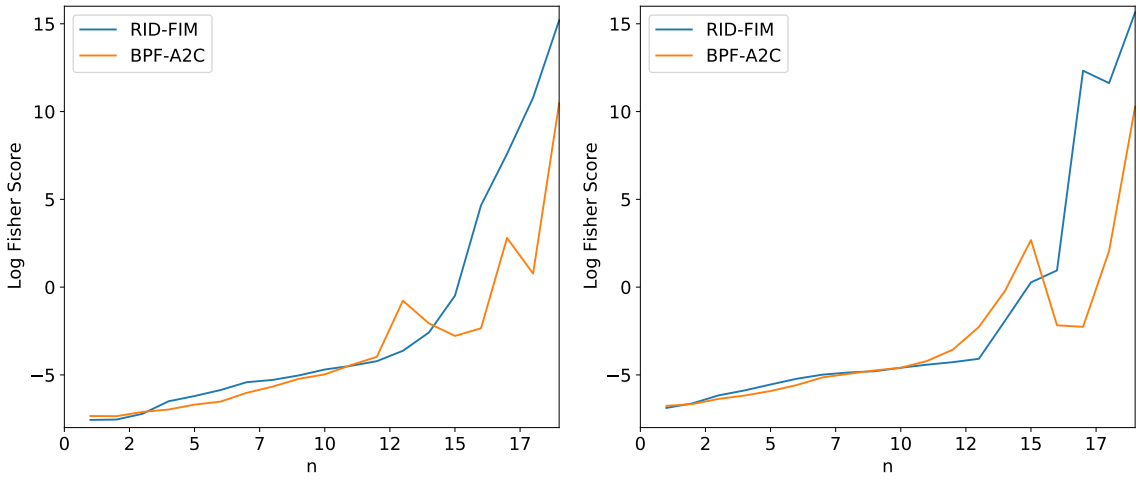
(a) Low.

(b) Medium.



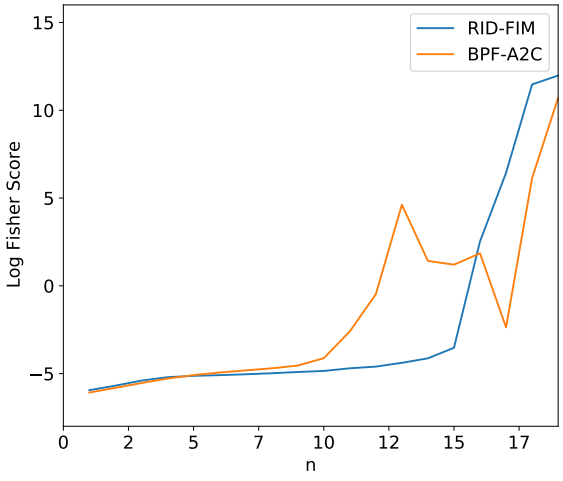
(c) High.

Figure 5.6: Comparison of the Monte Carlo RMSE for BPF estimation of the source location at each timestep for a completed episode length of 28. Each plot contains the BPF PCRB and RMSE for the RID-FIM and BPF-A2C controllers averaged over at least different 650 episodes. (5.6a) is at low SNR, (5.6b) is at medium SNR, and (5.6c) is at high SNR. The BPF-A2C has a lower RMSE than then RID-FIM when the completed episode length was longer due to the RID-FIM's action selection dependence on potentially spurious BPF state estimates.



(a) Low.

(b) Medium.



(c) High.

Figure 5.7: Comparison of the Monte Carlo Fisher information score at each timestep for a completed episode length of 19. Each plot contains the log trace of the Fisher information matrix for the RID-FIM and BPF-A2C controllers averaged over at least different 650 episodes. (5.7a) is at low SNR, (5.7b) is at medium SNR, and (5.7c) is at high SNR. In each of the plots, the sharp increase in the Fisher score for the RID-FIM indicates the sample when enough information is available for the FIM metric to be used for action selection.

5.2.4 Discussion

The results indicate close search performance between the RID-FIM and RAD-A2C algorithms in the convex environment. GS had the shortest episode completion length at high SNR but this required 7 more measurements per action selection. The RAD-A2C showed the best reliability in completing all of the episodes with a minimal spread in the distribution of results but had a greater spread in the completed episode length even at the highest SNR. The longer completed episode length of the RAD-A2C could be due to learned behavior that is advantageous in non-convex environments as the training environment always had obstructions present. The RID-FIM had a tighter and lower distribution of completed episode lengths across the SNRs.

Completion of episodes is the priority in practice as this will eliminate the threat of human harm from nuclear materials. Both algorithms get the job done effectively, however, the RID-FIM has a slightly greater chance of failing when SNR conditions are poor compared with the RAD-A2C. The RID-FIM utilized perfect knowledge of the background rate, which is a reasonable assumption in this particular source search context, however, its performance is likely to be degraded to some degree when it's operating in an uncertain background rate. The RAD-A2C did not receive the true background rate directly but did have prior exposure to the interval of background rates through training as specified in Table 4.1.

The BPF serves as an interesting comparison point between the A2C and RID-FIM controllers. When the completed episode length was short (< 16 samples), the RID-FIM location prediction RMSE was lower than the BPF-A2C and closer to the PCRB at all SNRs. This evidences the effectiveness of information-driven search schemes and the near-optimal performance of the RID-FIM when the BPF does not make spurious estimates. However, the occurrence of the intersection point of the RMSE curves highlights the disadvantage of the RID-FIM's reliance on the estimator

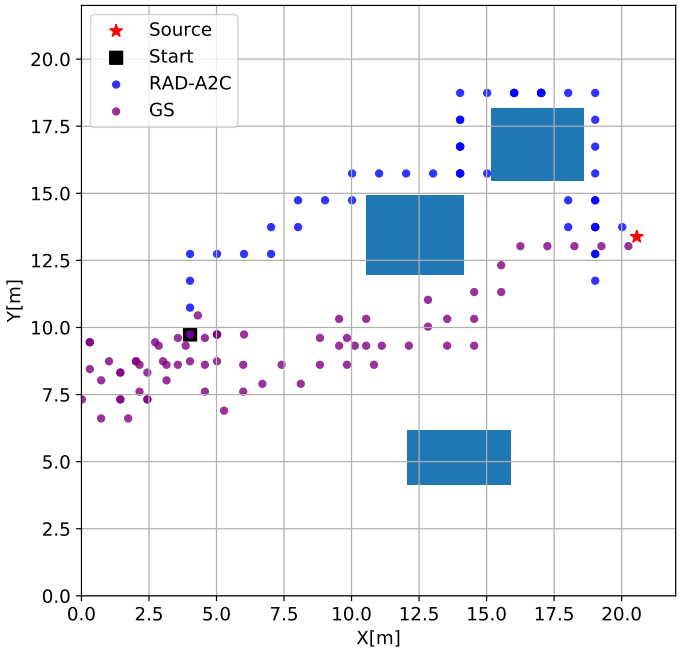
for action selection. If early state estimates are incorrect, this leads the RID-FIM to take more sub-optimal actions until the estimate is corrected. This is evidenced by the longer completed episode lengths $(20, 28)$ that have a greater initial increase in the RMSE as seen in Figures 5.5c and 5.6c. Interestingly, the higher SNR contributes a sharper increase, likely due to the strong radiation measurements being interpreted by the BPF as evidence for the incorrect estimate.

In contrast, the A2C module of the BPF-A2C selects its actions from the location prediction and the measurement directly. Thus, when the SNR is high, the RMSE intersection point occurs at an earlier completed episode length (17 samples) because the A2C factors in measurement information at each timestep, rather than strictly following the possibly incorrect location prediction as the RID-FIM must do. This also explains why the BPF-A2C has lower RMSE at longer completed episode lengths as seen in Figure 5.6. The intersection point occurred at longer completed episode lengths for lower SNR because it takes the A2C longer to come across informative measurements that can correct the spurious BPF state estimates.

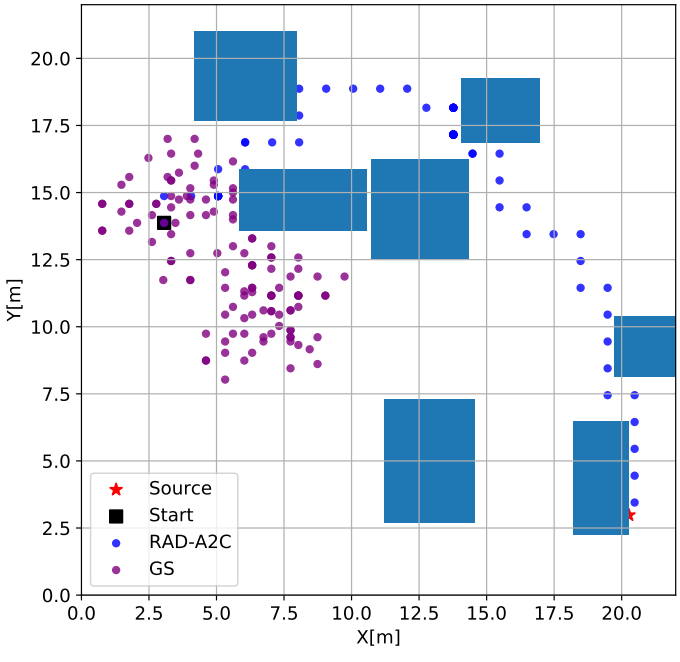
5.3 Non-convex Environment

5.3.1 Detector Path Examples

Two detector paths for the RAD-A2C and the GS in two non-convex environments with three and seven obstructions are shown in Figures 5.8a and 5.8b, respectively. The GS takes many more samples to find a radiation gradient in the three obstruction environment but eventually finds the source. Gradient information is extremely sparse in the seven obstruction environment and thus the GS only moves randomly. The RAD-A2C can avoid the obstructions and find the source in both situations, even moving diagonally between two obstructions in Figure 5.8b. As in the convex environment, the majority of the RAD-A2C movements are in the cardinal directions.



(a) Three obstructions.



(b) Seven obstructions.

Figure 5.8: Two detector paths for the RAD-A2C and the GS in three and seven obstruction configurations of the non-convex environment overlayed on a single plot. The black square denotes the detector starting position, the blue rectangles represent obstructions that block radiation propagation, and the red star is the radiation source. Both algorithms must explore the area as they search for radiation signal above the noise floor.

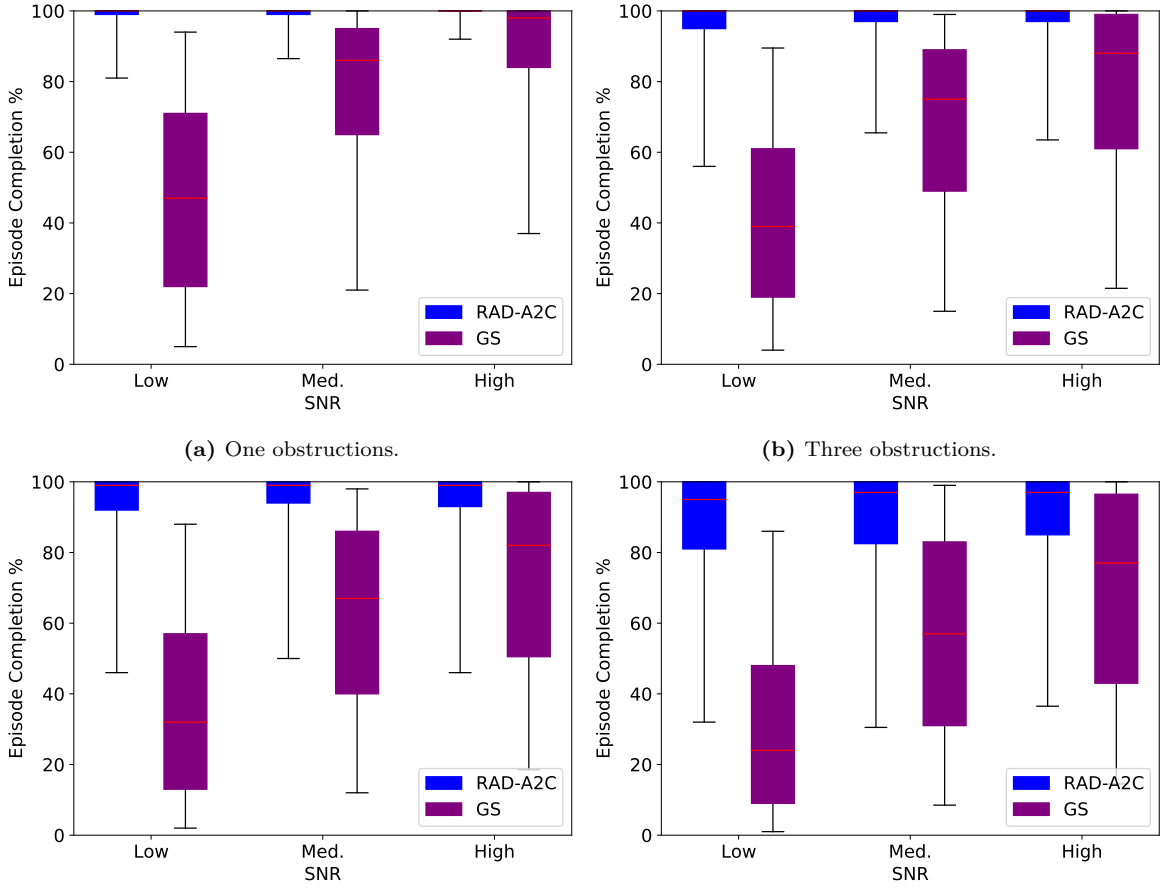
Obstructions	Low	Medium	High
1	7.15 ± 0.28 m	$7.38 \pm .28$ m	7.77 ± 0.30 m
3	$7.19 \pm .31$ m	$7.30 \pm .30$ m	7.75 ± 0.32 m
5	$6.90 \pm .30$ m	$7.18 \pm .29$ m	$7.53 \pm .32$ m
7	$7.01 \pm .31$ m	$7.32 \pm .32$ m	$7.67 \pm .34$ m

Table 5.3: Mean of the RMSE per episode for source location estimates of the PFGRU across SNR for different number of obstructions. The uncertainty is the standard error of the mean. The RMSE is consistent per SNR across number of obstructions and gets worse for higher SNR.

5.3.2 Performance

Box plots for the episode completion percentage and completed episode length against SNR for both methods in the non-convex environment are found in Figures 5.9 and 5.10, respectively. Figures 5.9a and 5.10a are results with one obstruction, Figures 5.9b and 5.10b are results with three obstructions, Figures 5.9c and 5.10c are results with five obstructions, and Figures 5.9d and 5.10d are results with seven obstructions. The median is denoted in red, the boxes range from the first to the third quartile and the whiskers extend to the 2.5^{th} and 97.5^{th} percentiles.

Across obstruction number, the RAD-A2C maintains above 95% of episode completion even at low SNR. The distribution of the RAD-A2C episode completion gets larger as the number of obstructions increases. GS has $> 85\%$ episode completion when there are less than 7 obstructions at high SNR but sees a sharp decrease in performance as the SNR level decreases. Even at high SNR, GS only completes 77% of episodes when 7 obstructions are present. GS also has significant spread in the first and third quartile for most of the completed episode non-convex experiments. The RAD-A2C median for completed episode length increases by approximately 10 samples from a single obstruction to seven obstructions. The first and third quartiles for completed episode length also increase as the number of obstructions increase. Table 5.3 shows the PFGRU source location prediction RMSE. This shows consistency across obstruction number and SNR level.



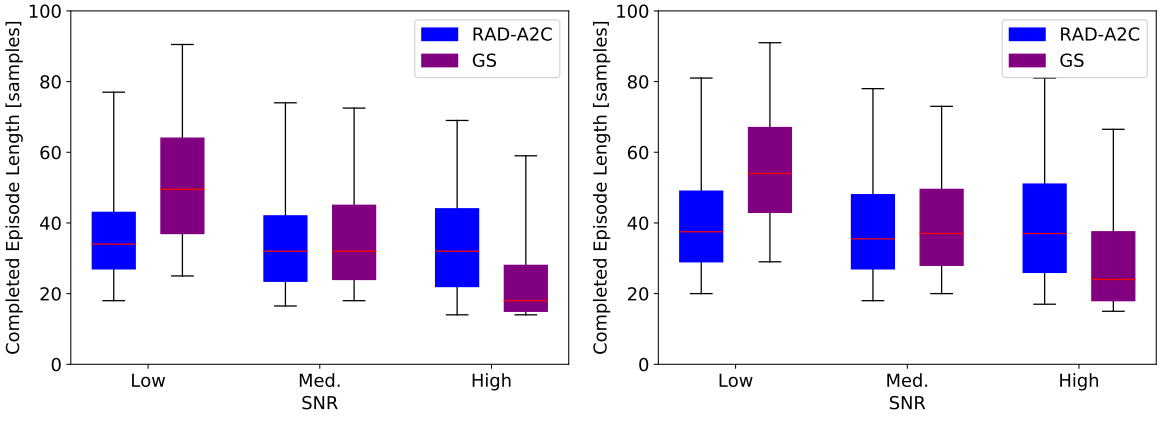
(a) One obstructions.

(b) Three obstructions.

(c) Five obstructions.

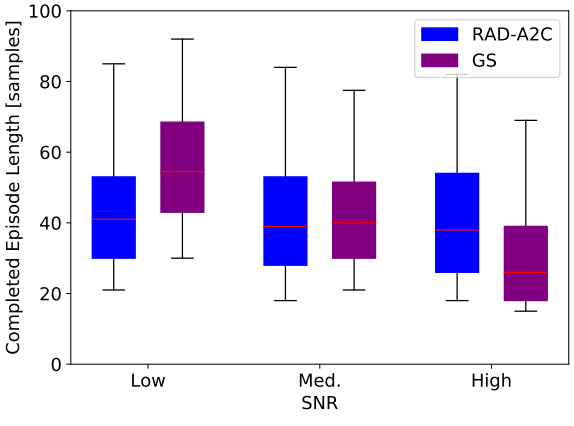
(d) Seven obstructions.

Figure 5.9: Box plots for the completed episode percentage against SNR in the non-convex environment, where each plot corresponds to a different number of obstructions in the environment. The median is denoted in red, the boxes range from the first to the third quartile and the whiskers extend to the 2.5^{th} and 97.5^{th} percentiles. Figure 5.9a was for a single obstruction, Figure 5.9b was for three obstructions, Figure 5.9c was for five obstructions, and Figure 5.9d was for seven obstructions. GS episode completion deteriorates with increasing number of obstructions while the RAD-A2C maintains greater than 95% median episode completion.

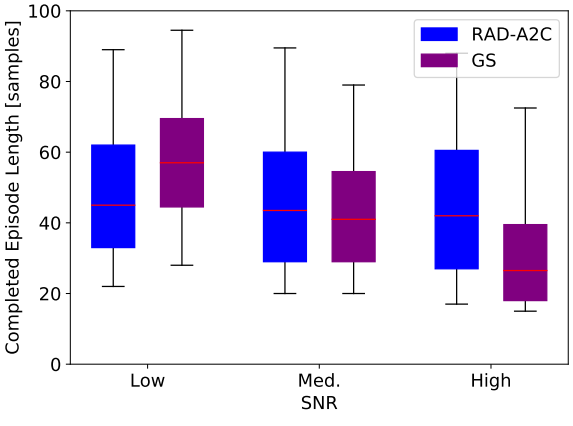


(a) One obstructions.

(b) Three obstructions.



(c) Five obstructions.



(d) Seven obstructions.

Figure 5.10: Box plots for the completed episode length against SNR in the non-convex environment, where each plot corresponds to a different number of obstructions in the environment. The median is denoted in red, the boxes range from the first to the third quartile and the whiskers extend to the 2.5^{th} and 97.5^{th} percentiles. Figure 5.10a was for a single obstruction, Figure 5.10b was for three obstructions, Figure 5.10c was for five obstructions, and Figure 5.10d was for seven obstructions. The RAD-A2C maintains a low completed episode length across the varying number of obstructions and SNR while GS performance deteriorates.

5.3.3 Discussion

The results showcase the strong performance of the RAD-A2C in the non-convex environment. Surprisingly, the episode completion percentage did not decrease substantially in the seven obstruction configuration and the median completed episode length did not increase drastically. This demonstrates the algorithm's ability to generalize as it was only trained on up to five obstructions per environment. The RAD-A2C is not simply a gradient search algorithm as the non-convex environment has many areas with no gradient information as evidenced by the ineffectiveness of the GS. It is curious that the source location prediction RMSE is consistent across SNR and number of obstructions. Greater insights would likely be revealed using a higher resolution metric such as RMSE across timestep. Overall, these results support our hypothesis that the RAD-A2C is an effective search algorithm for both convex and non-convex environments.

5.4 Summary

This chapter presented the results of the radiation source search experiments in both a convex and non-convex environment for the GS, RID-FIM, and RAD-A2C. The GS had strong performance when the SNR was high but quickly lost efficacy with decreasing SNR. The RID-FIM typically required fewer measurements to complete episodes but had a slightly greater chance of not completing all of the episodes at lower SNRs. The RAD-A2C consistently completed all episodes albeit at the cost of taking more measurements. Guaranteed episode completion is arguably the most important criteria for radiation source search applications and so we find our proposed algorithm is superior.

Estimator performance served as another lens to compare the search algorithm performance. The same BPF was used for both controllers (RID-FIM, A2C) so that

the RMSE and PCRB for the location prediction could be compared. We found that on average, the BPF RMSE was lower for the longer episode lengths when the A2C was the controller as it was able to factor in measurements to its action selection, as opposed to the RID-FIM which selected actions solely on the BPF location prediction. The RID-FIM's action selection scheme is well-motivated but is susceptible to incorrect state estimates from the estimator.

In the non-convex environment, the RAD-A2C completed greater than 95% of episodes over a range of obstructions and SNRs. There was very little gradient information available in the environments with more obstructions and thus the GS algorithm completed a much lower percentage of episodes. The RAD-A2C demonstrated generalizability as it was able to maintain a high completion percentage in a seven obstruction environment that it had never been trained on.

Chapter 6

Conclusion & Future Work

6.1 Summary

This work demonstrated the efficacy of *deep reinforcement learning* (DRL) for sequential decision making in the radiation source search context. The challenges associated with active nuclear source search and a few of the methods that have been proposed in the literature were covered in Chapter 1. Liu et al. proposed the double deep Q learning DRL implementation for radiation source search [14]. Their implementation was limited to more constrained scenarios (fixed background rate, fixed environment layout). Landgren proposed and experimentally evaluated a *reinforcement learning* (RL) approach that used a multi-armed bandit framework for decisions making to locate a radiation source/s with very few prior assumptions [13]. The drawback of this algorithm is the computation cost at each timestep scales linearly with the discretized occupancy map (all accessible areas around the robot), which can become enormous for large search areas.

Chapter 2 introduced the fundamentals of *deep learning* (DL) and the network architectures that were key to our approach. It also gave a brief overview of state space tracking and the differentiable *particle filter gated recurrent unit* (PFGRU) proposed by Ma et al. [19], which improved the performance of our method at the cost of longer training time.

Chapter 3 reviewed the main concepts of RL and the application of DL to RL resulting in DRL. We used *proximal policy optimization* (PPO), a stochastic, on-policy

DRL framework to train our model as it has shown robustness and less susceptibility to hyperparameter tuning across a variety of applications. Radiation source search can be viewed as a *partially observable Markov decision process* (POMDP) that necessitates an agent that has some form of memory to track state over timesteps.

Chapter 4 outlined the implementation details of two of our main contributions, namely the simulated radiation source search environment, our proposed architecture (RAD-A2C) and the training scheme. The chapter also covered the comparison algorithms. Memory was incorporated into our architecture through the use of the *gated recurrent unit* (GRU). A hybrid information-driven controller with a bootstrap particle filter (RID-FIM) served as the primary comparison in the convex environment. Gradient search (GS) served as the primary comparison algorithm in the non-convex environment. A complexity analysis showed that the RID-FIM is exponential in the number of lookahead steps with a base proportional to the number of search directions while the RAD-A2C is quadratic in the number of hidden state dimensions and grows linearly with the number of search directions.

Chapter 5 presented the results of the search algorithms in the convex and non-convex environments across SNR. The RID-FIM had the lowest and most consistent episode length at the medium and low SNR but had a slightly greater deviation in the episode completion percentage relative to the RAD-A2C. The RAD-A2C was the only method to complete 100% of the episodes at all SNRs. This performance carried over to the non-convex environment where the RAD-A2C maintained a greater than 95% episode completion, even in a challenging scenario it had never been exposed to.

6.2 Future Work

6.2.1 Application to Source Search Variations

As mentioned in Section 4.2, the RAD-A2C formulation has the potential to be applied to other variations of the radiation source search. These include moving

and/or shielded nuclear sources, spatially varying background rates, utilizing an attenuation model for different environment materials, locating an unknown number of multiple sources, and a larger, more complex urban environment such as the one used by Hite et al. [6]. A classification layer could also be added to the A2C module that is trained on detecting whether a source is present or not and how many sources are present. Noise could be added to the other dimensions of the observation vector such as the detector coordinates and/or the obstruction range measurements. In theory, the majority of these cases only require modification of the simulation environment, clever shaping of the reward signal, and hyperparameter sweeps for the model parameters.

Our proposed algorithm could be trained in a more realistic environment and gamma sensor simulation such as the one utilized used for a single UAV source search by Baca et al. [58]. They developed a realistic gamma radiation simulation plugin for the Gazebo/ROS environment. Gazebo is a realistic open-source robotics simulator [59]. This plugin could then be easily interfaced with our DRL algorithm using the OpenAI_ROS Gym developed by Ezquerro et al. that seamlessly connects Gazebo and OpenAI Gym interfaces [60].

6.2.2 Neuromorphic Adaptation

Neuromorphic computing aims to emulate biological neural networks through hardware implementations to enable greater power efficiency [61]. This can be enacted on a variety of different hardware components; one such example is the nano-device known as a memristor. The memristor is a circuit element that can maintain a state through a hysteric current-voltage relationship [62]. Memristors can be arranged in crossbar arrays and "programmed" to perform matrix/vector operations via tuning of the junction conductances. This makes algorithms that rely on neural networks amenable to a neuromorphic architecture conversion by tuning the conductances to represent the weights of the network [63]. Figure 6.1 shows a sample of a feedforward neural

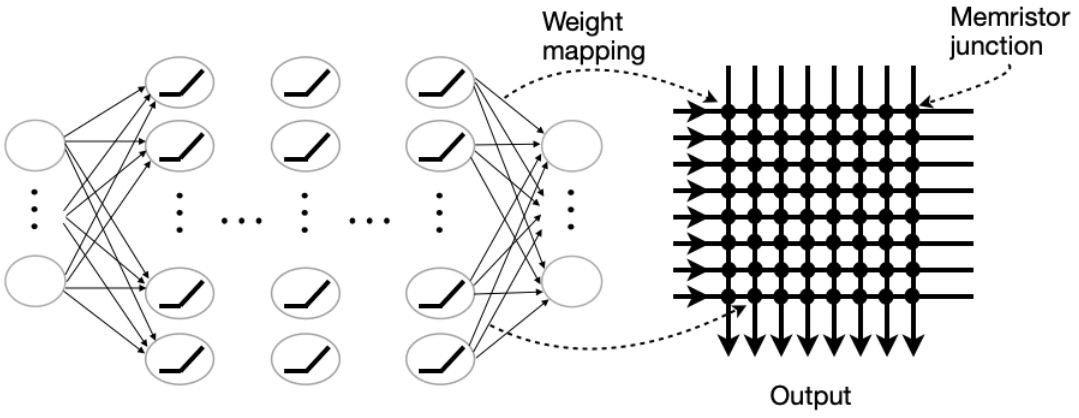


Figure 6.1: A cartoon of a feedforward neural network neuromorphic mapping. The feedforward neural network (left) has its weights and biases directly mapped onto a memristor crossbar array (right). The conductance at each of the nodes of the crossbar array is tuned to according to the weight value and this allows matrix vector multiplication operations [63]. This results in increased computational efficiency as memristors are extremely power efficient [64]

network weights being mapped to a memristor crossbar array. Direct mapping to hardware drastically lowers the power consumption necessary to perform the same operation on standard Von Neumann architectures [64].

Our RAD-A2C implementation worked specifically with recurrent neural networks that are just compositions of weight matrices as covered in 2.3. Liu et al. proposed a memristor-based *long short-term memory network* (LSTM) that achieved approximately 92.9% on the MNIST digit classification [65]. The LSTM requires more parameters than the GRU and the authors used a hidden state size of 32 with an input vector size of 28. Our architecture only used a hidden state size of 24 and an input vector of size 11. Thus, it should be relatively straightforward to perform a neuromorphic mapping of our architecture to memristor crossbar arrays.

6.3 Concluding Remarks

DRL has shown incredible demonstrations of performance across many disparate disciplines [4],[66],[67]. The framework of learning directly through experience mimics that by which humans operate and has great potential for development of novel

solutions to complex problems. Radiation source search has high variation in the situations that occur in practice and algorithms proposed in the literature often have many limiting prior assumptions. Our work in this thesis demonstrated the effectiveness of DRL for radiation source search, specifically through the RAD-A2C architecture, and created an open-source radiation simulation Gym environment for future DRL investigations. This provides a useful starting point for further research studies that can apply different DRL implementations and architectures, as well as modify the environment for variations on nuclear source search scenarios.

Bibliography

[1] A. Sieminski *et al.*, "International energy outlook," *Energy Information Administration (EIA)*, vol. 18, p. 2, 2014.

[2] U. N. S. C. on the Effects of Atomic Radiation, *Sources and Effects of Ionizing Radiation: Sources*, vol. 1. United Nations Publications, 2000.

[3] G. F. Knoll, *Radiation Detection and Measurement*, ch. 3, pp. 73–76. John Wiley & Sons, 2010.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of Go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[5] M. Morelande, B. Ristic, and A. Gunatilaka, "Detection and parameter estimation of multiple radioactive sources," in *2007 10th International Conference on Information Fusion*, pp. 1–7, IEEE, 2007.

[6] J. Hite and J. Mattingly, "Bayesian Metropolis methods for source localization in an urban environment," *Radiation Physics and Chemistry*, vol. 155, pp. 271–274, 2019.

[7] D. Hellfeld, T. H. Joshi, M. S. Bandstra, R. J. Cooper, B. J. Quiter, and K. Vetter, "Gamma-ray point-source localization and sparse image reconstruction using Poisson likelihood," *IEEE Transactions on Nuclear Science*, vol. 66, no. 9, pp. 2088–2099, 2019.

[8] R. Cortez, X. Papageorgiou, H. Tanner, A. Klimenko, K. Borozdin, and W. Priedhorsk, "Experimental implementation of robotic sequential nuclear search," in *2007 Mediterranean Conference on Control & Automation*, pp. 1–6, IEEE, 2007.

[9] B. Ristic and A. Gunatilaka, "Information driven localisation of a radiological point source," *Information fusion*, vol. 9, no. 2, pp. 317–326, 2008.

[10] B. Ristic, M. Morelande, and A. Gunatilaka, "Information driven search for point sources of gamma radiation," *Signal Processing*, vol. 90, no. 4, pp. 1225–1239, 2010.

[11] B. Ristic, M. Morelande, and A. Gunatilaka, "A controlled search for radioactive point sources," in *2008 11th International Conference on Information Fusion*, pp. 1–5, IEEE, 2008.

[12] R. B. Anderson, M. Pryor, A. Abeyta, and S. Landsberger, "Mobile robotic radiation surveying with recursive Bayesian estimation and attenuation modeling," *IEEE Transactions on Automation Science and Engineering*, pp. 1–15, 2020. Early Access.

[13] P. C. Landgren, *Distributed Multi-agent Multi-armed Bandits*. PhD thesis, Princeton University, 2019.

[14] Z. Liu and S. Abbaszadeh, "Double Q-learning for radiation source detection," *Sensors*, vol. 19, no. 4, p. 960, 2019.

[15] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, IEEE, 2017.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[19] X. Ma, P. Karkus, D. Hsu, and W. S. Lee, "Particle filter recurrent neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 5101–5108, 2020.

[20] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems*, pp. 1008–1014, Citeseer, 2000.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[22] C. Galleguillos and S. Belongie, "Context based object categorization: A critical survey," *Computer Vision and Image Understanding*, vol. 114, no. 6, pp. 712–722, 2010.

[23] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational Intelligence and Neuroscience*, vol. 2018, 2018.

[24] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[25] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[26] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[27] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, *Backpropagation: The basic theory*. 1995.

[28] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[29] I. Sutskever, *Training recurrent neural networks*. PhD thesis, University of Toronto, 2013.

[30] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[31] C. Olah, "Understanding LSTM networks." `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015.

[32] J. S. Bayer, *Learning sequence representations*. PhD thesis, Technische Universität München, 2015.

[33] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[34] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 2018.

[35] D. Ha and J. Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.

[36] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber, "Recurrent policy gradients," *Logic Journal of the IGPL*, vol. 18, no. 5, pp. 620–634, 2010.

[37] G. Tesauro, "Temporal difference learning and TD-gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[38] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[39] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[40] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, *et al.*, "What matters in on-policy reinforcement learning? a large-scale empirical study," *arXiv preprint arXiv:2006.05990*, 2020.

[41] J. Achiam, "Spinning Up in Deep Reinforcement Learning." `https://spinningup.openai.com/en/latest/`, 2018.

[42] A. M. Beigzadeh, M. R. R. Vaziri, Z. Soltani, and H. Afarideh, "Design and improvement of a simple and easy-to-use gamma-ray densitometer for application in wood industry," *Measurement*, vol. 138, pp. 157–161, 2019.

[43] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," *arXiv preprint arXiv:1606.01540*, 2016.

[44] J. Schulman, "The nuts and bolts of deep RL research." `http://joschu.net/docs/2017-rldm.pdf`.

[45] K. J. Obermeyer and Contributors, "VisiLibity: A C++ library for visibility computations in planar polygonal environments." `http://www.VisiLibity.org`, 2008. R-1.

[46] B. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[47] L. D. Stone, "OR forum—what's happened in search theory since the 1975 Lanchester prize?," *Operations Research*, vol. 37, no. 3, pp. 501–506, 1989.

[48] M. Gerber, N. Chopin, N. Whiteley, *et al.*, "Negative association, ordering and convergence of resampling methods," *Annals of Statistics*, vol. 47, no. 4, pp. 2236–2260, 2019.

[49] A. Srinivasan, "Distributions on level-sets with applications to approximation algorithms," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pp. 588–597, IEEE, 2001.

[50] S. Hammel, P. Liu, E. Hilliard, and K. Gong, "Optimal observer motion for localization with bearing measurements," *Computers & Mathematics with Applications*, vol. 18, no. 1-3, pp. 171–180, 1989.

[51] H. L. Van Trees, *Detection, estimation, and modulation theory, part I: detection, estimation, and linear modulation theory*. John Wiley & Sons, 2004.

[52] J. P. Helferty and D. R. Mudgett, "Optimal observer trajectories for bearings only tracking by minimizing the trace of the Cramér-Rao lower bound," in *Proceedings of 32nd IEEE Conference on Decision and Control*, pp. 936–939, IEEE, 1993.

[53] L. Pronzato, "Optimal experimental design and some related control problems," *Automatica*, vol. 44, no. 2, pp. 303–325, 2008.

[54] C. Kreucher, K. Kastella, and A. O. Hero Iii, "Sensor management using an active sensing approach," *Signal Processing*, vol. 85, no. 3, pp. 607–624, 2005.

[55] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, vol. 2. Nob Hill Publishing Madison, WI, 2017.

[56] P. Tichavsky, C. H. Muravchik, and A. Nehorai, "Posterior Cramér-Rao bounds for discrete-time nonlinear filtering," *IEEE Transactions on Signal Processing*, vol. 46, no. 5, pp. 1386–1396, 1998.

[57] N. Bergman, *Recursive Bayesian estimation: Navigation and tracking applications*. PhD thesis, Linköping University, 1999.

[58] T. Baca, P. Stibinger, D. Doubravova, D. Turecek, J. Solc, J. Rusnak, M. Saska, and J. Jakubek, "Gamma radiation source localization for micro aerial vehicles with a miniature single-detector compton event camera," *arXiv preprint arXiv:2011.03356*, 2020.

[59] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Sendai, Japan), pp. 2149–2154, Sep 2004.

[60] R. Téllez, A. Ezquerro, and M. Á. Rodríguez, *ROS Manipulation in 5 days: Entirely Practical Robot Operating System Training*. Independently published, 2017.

[61] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[62] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[63] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, *et al.*, "In-memory computing with memristor arrays," in *2018 IEEE International Memory Workshop (IMW)*, pp. 1–4, IEEE, 2018.

[64] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose, and R. W. Linderman, "Memristor crossbar-based neuromorphic computing system: A case study," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 10, pp. 1864–1878, 2014.

[65] X. Liu, Z. Zeng, and D. C. Wunsch II, "Memristor-based LSTM network with in situ training and its applications," *Neural Networks*, vol. 131, pp. 300–311, 2020.

[66] M. AlQuraishi, "Alphafold at casp13," *Bioinformatics*, vol. 35, no. 22, pp. 4862–4865, 2019.

[67] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.