MANCHESTER
1824
The University of Manchester

Autonomous Estimation and Seeking of Multiple Radiation Sources via Particle Filtering and Active Goal-Directed Deep Reinforcement Learning

Third Year Individual Project – Final Report

April 2025

Chun Hei Wong

11092209

Supervisor:

Dr. Zhongguo Li

Contents

Word count: 11884

Abstract

This paper proposes using a hybrid approach that combines particle filtering with goal-directed deep reinforcement learning for the autonomous seeking and estimation of multiple radioactive sources. The objective is to improve both the accuracy and efficiency of the search process by guiding exploration based on source belief estimates. A particle filter is employed to estimate the number, positions, and intensities (equivalent dose rate at a distance of 1 m) of radiation sources in a noisy environment, while a deep Q-network (DQN) agent selects exploratory goals based on a dynamically updated goal probability distribution and eventually seeks out the strongest source.

The agent is incentivised to move towards designated goals which represent informative areas, accelerating convergence and improving estimation quality. Simulation results demonstrate that the proposed method outperforms traditional ϵ-greedy RL in convergence speed and consistency. The method demonstrates reliable convergence for up to four sources, with scalability remaining a challenge for higher source counts. The results confirm the potential of integrating model-based estimation with goal-driven exploration for real-time search tasks in uncertain environments. Future extensions include enhanced parameter tuning and transfer learning to support generalisation across varied scenarios.

Declaration of originality

I hereby confirm that this dissertation is my own original work unless referenced clearly to the contrary, and that no portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Intellectual property statement

Acknowledgements

I would like to express my sincere gratitude to my project supervisor Dr. Zhongguo Li for his invaluable guidance, support, and feedback throughout the course of this project. His insights and encouragement were instrumental in shaping the direction of the research and overcoming key challenges along the way. I am especially thankful for his patience and willingness to assist me with his expertise during every stage of the project.

1 Introduction

1.1 Background and motivation

Ever since the discovery of radioactivity over a century ago, scientific research has pioneered the use of radioactive materials in many fields such as energy, industry, agriculture, national defence, medical treatment and scientific research, which have enabled the development new techniques and technologies [1]. However, radioactive materials produce ionising radiation which can be detrimental to human and animal health at a sufficient level of exposure. Accidents or improper handling relating to radioactive materials can cause widespread casualties and social panic.

The possibility of accident is sufficient to be an important safety concern; according to statistics from the International Atomic Energy Agency (IAEA), as of 31 December 2023, there had been 4243 incidents (unauthorised activities and events out of regulatory control) involving nuclear or other radioactive material since 1993 [2]. Unauthorised activities related to radioactive material means that it is likely not under professional supervision, which can lead to improper handling and a greater risk of accident.

Therefore, it is an important concern to properly deal with these accidents once they occur. More specifically, in the cleanup process, it is important to be able to locate the scattered radioactive material and navigating towards them to remove them.

Due to the risk posed by the radioactive material itself, human operators will have their health endangered in the process of searching and cleaning up. Therefore, this task is most suitable for an autonomous robot to complete.

Autonomous search is the use of a mobile platform, equipped with sensors to seek out and estimate the source(s) of a chemical, biological, or radiation emission. Due to the widespread applications of autonomous search, many approaches have been attempted to solve the problem most effectively. These include but are not limited to, information theory oriented approaches such as Entrotaxis [3], machine learning approaches such as hierarchical control (HC) [4], or even hybrid dual control approaches such as active information-directed reinforcement learning (AID-RL) [5].

A particle filter is a sequential Monte Carlo method used to estimate the state of a system based on incomplete and noisy observations [6]. A particle filter uses a set of many “particles”, where each “particle” represents a possible state of the system, to represent the posterior distribution, the probability distribution resulting from updating a prior distribution (probability distribution before evidence) with new evidence (observation) using Bayes' rule, of a stochastic process given the incomplete and noisy observations. In autonomous search, the set of observations is often incomplete due to the large search space, and measurements are often noisy due to the noise of the sensor used by the mobile platform. Thus, a particle filter is ideal to form an environment belief state, estimating the number of radiation sources along with their strength and location, thereby addressing the estimation aspect of autonomous search.

Reinforcement learning (RL) is one of the three machine learning paradigms, concerned with how an intelligent agent should take actions in a dynamic environment to maximise cumulative reward. Therefore, RL is an ideal approach to solve the seeking problem of autonomous search, as it involves an agent (the mobile platform) in a dynamic environment (the physical space with the radiation source(s)). Using RL, the actions of the mobile platform can be optimised to seek out the source(s) as accurately and efficiently as possible, utilising the observations available to the mobile platform (sensor reading and physical location), as well as the environment belief state from the particle filter. The RL process is performed over many episodes, where at the start of each episode the environment is reset, i.e. the agent is placed back to the initial start location. Each episode is terminated or truncated according to different conditions as defined later in Section 3.8.

This paper introduces an active reinforcement learning algorithm that integrates goal-directed exploration with particle filtering for seeking and estimating multiple radiation sources. While prior approaches have used reinforcement learning and/or particle filtering for the source seeking and/or tracking problem, this paper is the first to combine the methods while leveraging their synergy – enhancing estimation accuracy and navigation efficiency.

1.2 Aims and objectives

The overall aim of the project is, in a high-level simulation, where physical details are abstracted away (mobile platform and radiation sources are both represented as points in space), of a mobile sensor-equipped platform situated in a discrete 2D space with an unknown number of radiation sources, to design a RL based algorithm to estimate the number of radiation sources along with their strength and location, and to navigate to the strongest radiation source using the mobile platform (hereafter known as the final objective). The motivation behind navigating to the strongest radiation source first is rooted in real-life applications; the agent can target the strongest radiation source for clean-up/ removal as priority. The algorithm should also be able to be effectively applied to randomly initialised environments with randomised source parameters and starting locations (for both mobile platform and source). The process encapsulates the design of the simulation itself, the construction, training and tuning of the relevant neural network, as well as adjusting the hyperparameters to improve the effectiveness of the training process and the algorithm.

The objectives to be achieved are split into several stages:

Simulation environment of autonomous search problem and mobile platform:

- Develop a simulation environment in Python to model a discrete 2D space with multiple radiation sources, including an agent which can move around and take radiation intensity measurements.

Source estimation:

- Implement a particle filter to represent the environment belief state of the number of sources and parameters of each source using noisy sensor measurements.
- Develop an algorithm to extract source term estimates from the belief state.

Source seeking:

- Determine most appropriate reinforcement learning algorithm.
- Design a reward structure which guides the agent towards the final objective.
- Run the model with the same starting scenario and tune hyperparameters to improve results.

Evaluation and performance assessment:

- Assess the performance of the model against a set of metrics (can also compare between different approaches).
- Validate the generality of the model by testing on different starting scenarios.

2 Literature review

2.1 Introduction

This section reviews existing literature on autonomous search, specifically source term seeking and/or estimation, and reinforcement learning, with a particular focus on active reinforcement learning where exploration is guided. The aim is to highlight how different methodologies address the challenges of sparse rewards and efficient exploration. This review also considers solutions to filtering problems, in particular particle filtering, which attempt to solve the problem of estimating the state of a system from an incomplete and noisy set of observations.

2.2 Particle filtering

The fundamental ideas behind particle filtering methods, particularly importance sampling, have been known since the 1950s, but were largely overlooked mostly due to the insufficient amount of computing power available at the time [6]. The particle filter approximates the posterior distribution with a set of discrete samples, or “particles”. Each particle represents a possible state of the system, and each is assigned a weight which reflects how well it agrees with the observations made so far.

Particle filters are also known as sequential Monte Carlo methods, which is a term coined by Liu et al. in 1998 [7], to solve the Bayesian filtering problem. The goal of Bayesian filtering is to compute the posterior probability distribution of the state, given all measurements up to a certain time step t [6]. Monte Carlo methods rely on repeated random sampling to approximate numerical results, allowing inference over complex, but often deterministic systems. As a result, unlike traditional filtering techniques such as the Kalman filter, which assume linearity and Gaussian noise, particle filters are well-suited for non-linear, non-Gaussian systems, making them highly applicable to the source estimation problem [6].

As the dimensionality of state space increases, the number of particles required to accurately represent the posterior distribution grows dramatically due to the curse of dimensionality, making particle filtering a computationally intensive method. However, with modern advancements in computational power and parallel processing, these methods have become practical for real-time applications in tracking, robotics, and signal processing [8].

2.3 Deep Q-network (DQN) for RL

Solving a RL task entails finding a policy that achieves lots of reward over the long run. A policy p is defined to be better than or equal to another policy p' if the expected return $\mathbb{E}$ under p is greater than or equal to that under p' for all states s. There is always at least one policy that is better or not worse than any other – this is known as the optimal policy [9].

We denote all optimal policies (either a single policy or multiple policies) by p_*. They share the same optimal state-value function v_* defined as

$$v_*(s) = \max_p v_p(s) \tag{1}$$

for all $s \in S$, where S is the set of all states [9].

Optimal policies also share the same optimal action-value function q_*, expressed as

$$q_*(s,a) = \max_p q_p(s) \tag{2}$$

for all $s \in S$ and $a \in A(s)$, where a is an action and $A(s)$ is the action set for a given state s [9].

The Bellman optimality equation expresses the idea that the value of a state s under an optimal policy p equals the maximum expected return $\mathbb{E}$ from the state, considering the best action a to take. The Bellman optimality equation for q_* is

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a] \tag{3}$$

where R_{t+1} is the immediate reward after taking action a in state s, γ is the discount factor which controls the value of future rewards (where $0 < \gamma < 1$), $\max_{a'} q_*(S_{t+1}, a')$ is the best possible action the agent could take at the next state S_{t+1}. $S_t = s, A_t = a$ sets the premise that the agent started in state s and took action a [9].

In Q-learning, the Bellman optimality equation is used to iteratively update a Q-table that stores estimates of $q(s,a)$ for each state-action pair. When an agent experiences a transition $(s, a, \text{rew}$, *s')*, the Q-value is updated as

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[\text{rew} + \gamma \max_{a'} Q\,(s', a') - Q(s,a)\right] \tag{4}$$

where $0 < \alpha \leq 1$ is the learning rate [9]. If an ideal Q-table is available, the agent can simply select the action a at each state s that carries the maximum Q-value, thereby generating the maximum cumulative reward [5].

In practice, however, this is impractical as action-value pairs are estimated separately for each state-action pair with no generalisation [10]. The search space can be extremely large, and observations which define the reward function R, as discussed in Section 3.3 – 5, are often noisy. As a result, an impractically large number of experience transitions would be required to populate the Q-table sufficiently for reliable exploitation.

In 2013, Mnih et al. published the seminal paper "Playing Atari with Deep Reinforcement Learning" [10], introducing the deep Q-network (DQN) algorithm. The algorithm achieved competence in all seven Atari games tested and outperformed a human expert in three of them.

Deep Q-learning uses a neural network function approximator with weights ω as a Q-network. As neural networks iteratively update their weights ω to minimise a loss function, a Q-network can be trained by minimising a sequence of loss functions $L_t(\omega_t)$ that change at each time step t,

$$L_t(\omega_t) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[\left(y_{target} - Q(s,a;\omega_t)\right)^2\right], \tag{5}$$

where $y_{target} = \mathbb{E}_{s'\sim\mathcal{E}}\left[\text{rew} + \gamma \max_{a'} Q\left(s',a';\omega_{t-1}\right) \,\middle|\, s,a\right]$ is the target Q-value and the agent is interacting with environment $\mathcal{E}$ [10]. Note that the target y_t is updated at every time step t, in contrast to supervised learning, where targets remain static throughout training — a key difference in how neural networks are applied in RL. The target is also typically from the previous $t-1$ time step to help training stability. Mnih et al. [10] also introduced a technique known as experience replay. The agent's experiences at each time step, $e_t = (s_t, a_t, rew_t, s_{t+1})$ are stored into a buffer known as a replay memory $D = e_{t-B}, \ldots, e_t$, where B is the size of the replay memory D. Q-learning updates are done by sampling a random minibatch of b samples, where b is the batch size [10]. As a result, the Q-network is trained similar to the formulation described in (5), but the samples used to compute the target y_t are not from the immediately preceding time step $t-1$. Instead, the loss $L_t(\omega_t)$ is computed over all b sampled experiences in the batch [10].

The soft target update rule, originally proposed in the DDPG algorithm by Lillicrap et al. [11], is used to stabilise the learning process by gradually updating the target network parameters with a mixing factor $\tau \in [0, 1]$ as

$$\phi_{target} \leftarrow \tau\phi + (1-\tau)\phi_{target}, \tag{6}$$

where ϕ denotes the parameters of the Q-network and ϕ_{target} the parameters of the target Q-network.

2.4 Estimation and seeking of radiation sources

Given that the estimation and seeking of radiation sources is a critical task in nuclear materials safety, numerous approaches have been developed to solve it as efficiently as possible.

Totten suggested a multi-agent deep reinforcement learning approach RAD-TEAM, which is model-free [12]. Each agent is equipped with radiation and proximity sensors for source detection and obstacle avoidance, respectively. Two coordination strategies are explored: centralised training decentralised execution (CTDE) where agents form the environment belief state through the use of a global critic and a team reward, and broadcasted training broadcasted execution (BTBE) where agents use an individual critic and reward structure then broadcast findings to other agents to form the environment belief state [12]. In unobstructed environments, four-agent BTBE teams outperform all other compared control strategies for all tests, suggesting the effectiveness of multi-agent BTBE RAD-TEAM methods. Nevertheless, the approach is heavily dependent on team size; in a constrained environment with obstacles, small BTBE teams (one or two agents) as well as all CTDE configurations fail to complete the localisation task [12].

A method to localise multiple radiation sources using a particle filter (a model-based method) was proposed by T. Lazna and L. Zalud [13]. The number of radiation sources is not known in advance and is included as one of the environment state variables that the particle filter has to estimate. The measurement trajectory that the agent takes in the environment is also information-driven, guiding the agent to take observations that are most informative for refining the source estimate. This strategy reduces the required exploration time by approximately 40%. However, the method struggles to reliably localise low-intensity sources [13]. While the proposed method jointly estimates both source locations and intensities [13], the result section focuses exclusively on the localisation accuracy and trajectory efficiency. The absence of evaluation metrics for count rate estimation leaves the robustness of the full state estimation unverified, particularly in the presence of background radiation and noise.

A method for autonomous search of radioactive sources was proposed by Huo et al. [1]. A partially observable Markov decision process (POMDP)-based search strategy is used, in which the reward function is defined in terms of information gain. The reward strategy enables the agent to autonomously navigate to the position of the radiation source, while ensuring efficient path-taking by maximising the information gained per step. The search algorithm was first simulated, then tested on a robot platform, demonstrating the practical performance of both the navigation and estimation objectives of the algorithm [1]. Nonetheless, the algorithm is evaluated in an environment with only a single radioactive source, limiting its applicability to real-life scenarios where multiple sources may be present.

Hu et al. [4] suggested a hierarchical control (HC) architecture for the detection and navigating to radioactive sources in an unknown environment with obstacles. The detection task, under hierarchical control (HC), is decomposed into two subtasks: navigating out of the blocked area and mapping obstacles, followed by moving towards the radiation source once it is detectable. The two policies are trained using deep reinforcement learning (DRL), and a high-level controller classifies the current stage and switches between the policies via supervised learning. HC is found to outperform all baseline strategies, especially in complex environments with many obstacles. Nevertheless, this method is tested for a single radioactive source and also only focuses on the model-free navigation to the radioactive source without explicitly estimating source parameters.

2.5 Active exploration in RL

In the traditional RL, the ϵ-greedy strategy is commonly used to balance exploration and exploitation, where the ϵ decays over time as the agent learns more about the environment and makes it more likely for the agent to use its learned policy to exploit its knowledge to gain rewards, while performing a random action during exploration. However, the problem of autonomous search is a sparse-reward problem [14]; the search area contains a vast amount of possible states compared to the one state which represents the location of the strongest source. Consequently, the agent is unlikely to come across the strongest source during random exploration, making random exploration an ineffective strategy to train the Q-network.

Recently, there has been an increase in using active reinforcement learning as an attempt to use various techniques to incentivise the agent to explore poorly understood states and actions during exploration. For example, Li et al. [5] implemented an information-directed approach is used where Entrotaxis is employed to motivate the agent to perform efficient exploration.

A survey in active exploration methods by Ladosz et al. [14] gives an overview of goal-based exploration, where states of interest are marked as goals and used to guide exploration. Goal-directed approaches have been applied to a multitude of applications. This includes 2D maze solving, which outperforms state-of-the-art RL algorithms in 2D maze environments [15], as well as playing arcade games (Montezuma's Revenge, Pitfall!, and Private Eye), achieving strong performance and even superhuman performance on Pitfall! [16]. This approach has also been extended to simultaneous localisation and mapping (SLAM) [17], which represents the most closely related application to autonomous search found in existing literature.

Despite the success of goal-based methods across these domains, goal-directed exploration has not yet been applied to the autonomous search problem. This presents a promising direction, as the source locations themselves offer natural candidates for goal selection, potentially enabling more efficient exploration.

2.6 Conclusion

The reviewed literature offers an examination on the motivation and development behind key methods and algorithms such as particle filtering and deep Q-network (DQN), as well as more recent work on the localisation of multiple radiation sources and active reinforcement learning. However, most approaches dealing with the autonomous search of radiation sources treat localisation as the main focus [12, 13], with little or no analysis on the estimation of source strength. Furthermore, most studies only focus on tackling either localisation or seeking as separate tasks, and even when both are considered simultaneously [1], the scope is limited to single-source environments.

As identified in the existing literature, no prior work has integrated goal-directed reinforcement learning with particle filtering for the joint estimation and seeking of multiple radiation sources. This paper addresses that gap by proposing a unified framework that synergistically combines the strengths of both goal-directed deep reinforcement learning (utilising DQN) and particle filtering, aiming to improve the performance of both source estimation and seeking.

3 Methods

3.1 Introduction

The methodology begins with designing the simulation, followed by implementing key features such as algorithm-simulation interfacing and result visualisation to establish a reliable training and evaluation environment. Following the completion of the simulation, a particle filter for source estimation and a RL algorithm based on Deep Q-Network (DQN) with goal-directed exploration and reward-driven exploitation for navigation to the strongest source estimate will be implemented. While the primary objectives of the particle filter and RL algorithm are distinct, they complement each other to enhance overall effectiveness. The exploration strategy helps the particle filter converge efficiently and correctly, while the particle filter's source estimate is used to form the reward structure to facilitate in the training of the Q-network. The model is then trained, and hyperparameters adjusted over the course of training sessions to optimise performance.

3.2 Simulation environment

The simulation is initialised as a discrete 2D gridworld environment representing a 50 m × 50 m search area. Locations are defined using 2D Cartesian coordinates ranging from $(0, 0)$ to $(x_{max}, y_{max}) = (50, 50)$. The location of the agent at time step t is expressed as $l_t = (x_t, y_t)$. The starting scenario initially used for training is detailed in TABLE I below. Note that the agent begins every new episode from the starting location.

TABLE I – Starting agent location and source parameters

	x-coordinate	y-coordinate	Equivalent dose rate at 1 m (mSv/h)
Agent	12	12	N/A
Source 1 (strongest)	35	40	200
Source 2	10	35	140
Source 3	40	10	120

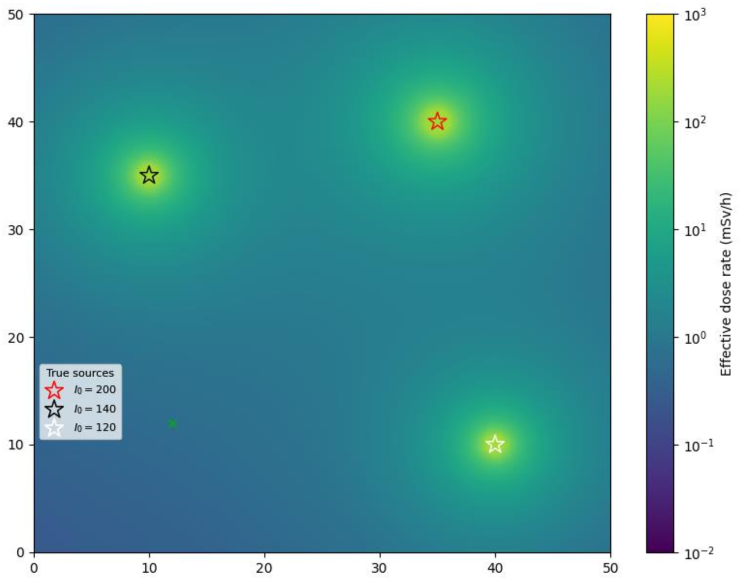


Fig. 1 – Visualisation of simulation via *Matplotlib*

Fig. 1 shows the visualisation of the initial simulation environment with the starting location of agent denoted by a green cross, along with source locations and the distribution of radiation intensity. Sources are numbered in descending order of strength. Note that the colour gradient used to depict the equivalent dose rate is on a logarithmic scale.

The agent must move one unit in an orthogonal direction at each time step t, and is equipped with a radiation sensor (at the same position as the agent) which provides simulated equivalent dose rate (hereafter referred to as intensity) readings given by

$$I_z = I_{true} + u \tag{7}$$

$$I_{true} = \frac{I_1}{{d_1}^2} + \frac{I_2}{{d_2}^2} + \cdots + \frac{I_r}{{d_r}^2} \tag{8}$$

where symbols are defined as follows in TABLE II.

TABLE II – Symbols for simulated radiation intensity calculation

Symbol	Description
r	Total number of radiation sources
I_z	Simulated intensity reading from the sensor
I_{true}	True total intensity contribution from all sources
I_r	Equivalent dose rate at 1 m for source n
d_r	Euclidean distance from source n to agent/sensor
u	Measurement uncertainty modelled as $u \sim N(\mu, \sigma^2)$ where N is the Gaussian distribution.

The measurement uncertainty is modelled as Gaussian noise with $\mu = 0$ and $\sigma = 10\% \times I_{true}$. 10% is chosen as the sensor uncertainty, as many electronic personal dosimeters on the market have approximately 10% of measurement uncertainty [18, 19]. The equivalent dose in mSv is selected as the measure for radiation intensity in this paper because it reflects the potential danger to human operators performing the search task. Note that the agent is assumed to know its location at all times with perfect accuracy as the localization of the agent itself is out of scope of this project.

To validate the robustness of the algorithm, random starting scenarios are initialised with the following constraints: $r \in [1, 5]$, $I_r \in [100, 300]$ mSv/h, and the initial agent position, as well as the position of the sources must be at least 5 m apart. The minimum distance between sources is introduced as a requirement, motivated by how the total measured radiation intensity is simply the superposition of the intensities radiated by individual sources. As a result, sources positioned too closely together become virtually indistinguishable from a single source when observed from a distance. These constraints ensure a reasonable difficulty level for the autonomous search problem and helps the particle filter stabilise and converge to the correct global minimum.

3.3 Formulation of environment belief state via particle filtering

Note that the state considered by the particle filter—used for source estimation—is referred to as the environment state to distinguish it from the state s of the agent which is used in reinforcement learning.

First, the environment state space along with the particles, weights, and observations is defined. Background radiation is assumed to be negligible and is therefore ignored. Let us have a set of observations $Z = \{z_t\}_{t=1}^{S}$, where S is the total amount of steps taken by the agent throughout training. A new observation z_t is made every time step t. Each observation $z_t = (x_z, y_z, I_z)$ consists of the x and y coordinates of the agent at the time of observation, along with the measured equivalent dose rate. As the number of sources r is unknown, it is contained within the environment belief state as one of the state variables and the length of environment state vector θ_t varies in length accordingly [13]. Consequently, let us define environment state vector $\theta_t = (r, x_1, y_1, I_1, \dots, x_r, y_r, I_r)$, where each set of environment state variables (x, y, I) represents the x and y coordinates of the source, and the equivalent dose rate (hereafter referred to as intensity) at a distance of 1 m. Each set of environment state variables (x, y, I) is ordered within environment state vector θ_t such that the value of intensity is in descending order. Note that r must be a whole number while the other environment state variables can hold decimal values. Each particle ${\theta_t}^{(i)}$ contains a belief of the true environment state in the form of environment state vector θ_t, and let there be a set of N particles $\pi_t = \left\{{\theta_t}^{(i)}\right\}_{i=1}^{N}$, where each particle is also assigned a weight ${w_t}^{(i)}$ in the set of N weights $w_t = \left\{{w_t}^{(i)}\right\}_{i=1}^{N}$.

The implementation of a particle filter typically consists of the following steps. Initialisation, prediction, update, and resampling if necessary. Initialisation is only done once, while the other steps are repeated after every new observation.

Initially, the set of particles π_t are initialised to be uniformly distributed across the environment state space. A maximum number of sources r_{max} needs to be set, with the minimum number of sources assumed to be 1 [13]. Each weight ${w_t}^{(i)}$ in w_t is assigned an equal value, and the weights are normalised such that the sum of all weights satisfies $\sum_{i=1}^{N} {w_t}^{(i)} = 1$ as

$$ {w_t}^{(i)} = \frac{1}{N}. \tag{9} $$

This uniform distribution increases the likelihood that the particle filter will converge to the correct minimum. With a sufficient number of N particles, there is a high probability that at least some particles will be near the true state of the environment.

Typically, particle filter implementations involve a prediction step that reflects the state transition between each time step t in dynamic systems. However, the prediction step can be omitted here as the system can be assumed to be static; i.e. the half-life of the radioisotopes acting as radiation sources markedly exceed the duration of the autonomous search, meaning their equivalent dose rate at 1 m stays approximately constant, and that the sources are stationary [13].

Subsequently, the observations at each time step t are used to update the posterior probability using Bayes' theorem as

$$ p(\theta_t|z_t) = \frac{p(z_t|\theta_t) \times p(\theta_t|z_{1:t-1})}{p(z_t|z_{1:t-1})} \text{ [13].} \tag{10} $$

To apply this update recursively to the set of particles π_t and set of weights w_t for every time step t, the likelihood for each particle, $p\left({\theta_t}^{(i)}\middle|z_t\right)$, is computed as

$$ p\left(z_t\middle|{\theta_t}^{(i)}\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(I_z - I_{true})^2}{2\sigma^2}\right) \tag{11} $$

where σ is the measurement uncertainty of the sensor (10% as defined in Section 3.2), I_z is the measured radiation intensity as defined in (7) from Section 3.2, and I_{true} is the true radiation intensity at the agent's current location (x_z, y_z), assuming ${\theta_t}^{(i)}$ represented the true state of the environment, as defined in (8) from Section 3.2. Additionally, the likelihood of a particle ${\theta_t}^{(i)}$ which predicts two sources within 5 m of each other is also equal to 0 (equivalent rule is defined in Section 3.2 when initialising source locations). Thereafter, the likelihood for each particle, $p\left({\theta_t}^{(i)}\middle|z_t\right)$, is used to update the weight from the previous time step, ${w_{t-1}}^{(i)}$, corresponding to each particle, as

$$ {w_t}^{(i)} = p\left(z_t\middle|{\theta_t}^{(i)}\right) \times {w_{t-1}}^{(i)}. \tag{12} $$

Once this has been done to all N of the particles ${\theta_t}^{(i)}$ in the set of particles π_t, the weights are again normalised such that the sum of all weights satisfies $\sum_{i=1}^{N} {w_t}^{(i)} = 1$ as

$$w_t{}^{(i)}{}_{norm} = \frac{w_t{}^{(i)}}{\sum_{i=1}^{N} w_t{}^{(i)}}. \tag{13}$$

As the particles are updated with every observation, the posterior probability distribution is likely to converge to some kind of minima, leading to particle degeneracy (when a few particles dominate the weight distribution). To prevent particle degeneracy, resampling of the particles is performed when the effective sample size N_{eff} drops below a chosen threshold N_{thr} [6]. N_{eff} is defined as

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w_t{}^{(i)})^2} \tag{14}$$

Multinomial resampling is performed by a weighted random sampling of the particles, where the probability that particle i is sampled for the new set of particles $\pi_t{}'$ is its weight $w_t{}^{(i)}$ [20]. This is repeated N times to gather N new particles $\theta_t{}^{(i)\prime}$ for the new set of particles $\pi_t{}'$. The new set of weights $w_t{}'$ is initialised equally and the weights are normalised such that the sum of all weights satisfies $\sum_{i=1}^{N} w_t{}^{(i)} = 1$ as shown in (9).

Resampling reduces particle set variance [13]; particles with a high weight are chosen repeatedly for the resampled particle set leading to many identical particles in the new set. This problem is further exacerbated by how the system state is static; there is no prediction step to update the environment belief state of the particles, meaning that the particle filter will rapidly collapse to N identical particles with equal weight $w_t{}^{(i)}$, which is an ineffective representation of the posterior distribution. Therefore, regularisation has to be introduced to increase the particle diversity, by adding a small random perturbation to all environment state variables in each particle $\theta_t{}^{(i)}$. A novel and simple approach to compute the perturbation is introduced in this paper. Regularisation for each source location and intensity belief, (x, y, I), is performed as

$$x' = x + N(\mu, \sigma_{loc}{}^2) \tag{15}$$

$$y' = y + N(\mu, \sigma_{loc}{}^2) \tag{16}$$

$$I' = I + N(\mu, \sigma_I{}^2) \tag{17}$$

where $\mu = 0$, and

$$\sigma_{loc} = \delta_{end} + (\delta_{loc,\ start} - \delta_{end}) \times \exp\left(\frac{-t}{\delta_{decay}}\right) \tag{18}$$

$$\sigma_I = \delta_{end} + (\delta_{I,\ start} - \delta_{end}) \times \exp\left(\frac{-t}{\delta_{decay}}\right) \tag{19}$$

The exponential decay for both location and intensity perturbation, σ_{loc} and σ_I is controlled by the total time steps t taken so far. The parameters $\delta_{loc,\ start}$ and $\delta_{I,\ start}$ determine the initial values of the perturbations, while δ_{decay} controls the exponential decay rate. Lastly, δ_{end} is a constant controlling the asymptotic value of σ_{loc} and σ_I as S greatly exceeds δ_{decay}.

It is important to note that the perturbed particle ${\theta_t}^{(i)\prime}$ is bounded by the limits as stated in the last paragraph of Section 3.2. Any perturbation that is out of bounds is clipped to the limits (search area or intensity range).

A similarly motivated, but modified approach is applied to perturb the number of sources belief r, which is a bit different from the other state variables as it is both restricted to integer values and controls the length of the environment state vector of particle ${\theta_t}^{(i)}$.

$$r' = \begin{cases} r+1 \quad \text{with probability } \dfrac{\sigma_r}{2} \\ r-1 \quad \text{with probability } \dfrac{\sigma_r}{2} \\ r \quad \text{with probability } 1-\sigma_r \end{cases} \tag{20}$$

where σ_r is also scaled by exponential decay similar to above

$$\sigma_r = \delta_{r,\ end} + (\delta_{r,\ start} - \delta_{r,\ end}) \times \exp\left(\frac{-S}{\delta_{decay}}\right) \tag{21}$$

and variables are similarly defined as in (18) and (19) with the exception of $\delta_{r,\ end}$ which is much smaller than δ_{end}. σ_r is analogous to σ_{loc} or σ_I, but in this case is used to perturb the belief in number of sources. r' is also bounded between r_{max} and 1, so if the increase/decrease makes the amount of sources out of bounds, the change is ignored.

When r is decreased by resampling, the source closest to the strongest source is removed, as it is often the case that the particle can erroneously believe that an extra weak source is close to the strong source. This is due to the noisy sensor measurements, leading to intensity contributions from an assumed weaker source which is hard to distinguish from a distance due to contribution being overridden by the strong source. When r is increased by resampling, the extra source parameters (location and strength) are randomised.

Another novel technique introduced is scramble probability $\sigma_s = \sigma_r$, which denotes the probability at which the locations and strengths of sources are shuffled around at random, such that a strength For example, particle ${\theta_t}^{(i)} = (r, x_1, y_1, I_1, x_2, y_2, I_2)$ could be shuffled into ${\theta_t}^{(i)\prime} = (r, x_1, y_1, I_2, x_2, y_2, I_1)$. This technique is implemented due to the observation that a common local minima is when the locations of sources are correct, but strengths are in the wrong order. It is relatively easier to identify the location of a radiation source when close by, as the measured intensity greatly increases according to inverse square law, but pinpointing the exact intensity at 1 m is difficult unless the agent reaches the location of the source term.

After resampling is performed, the environment state variables of each source (x, y, I) in each particle ${\theta_t}^{(i)}$ is again ordered such that intensity I of each source is in descending order.

This exponential decay scaling approach is inspired by a similar equation used in RL to balance exploration and exploitation [9]. It reflects how the posterior probability distribution is expected to converge as the agent experiences more time steps t and gathers more observations. As time progresses, the particle filter's predictions should become more confident, requiring fewer perturbations to refine the environment belief state or escape erroneous local minima.

The implementation of the particle filter (excluding the initialisation step) is summarised in Algorithm 1 below [5].

Algorithm 1: Particle filter to formulate environment belief state.

```
Require: old particles, old weights and current observation $\left\{{\theta_{t-1}}^{(i)},\ {w_{t-1}}^{(i)},\ z_t\right\}_{i=1}^{N}$
1.  for $i = 1, 2, \dots, N$, do
2.      Compute likelihood $p\left(z_t \middle| {\theta_{t-1}}^{(i)}\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(I_z - I_{true})^2}{2\sigma^2}\right)$
3.      Update weight ${w_t}^{(i)} = p\left(z_t \middle| {\theta_{t-1}}^{(i)}\right) \times {w_{t-1}}^{(i)}$
4.  end for
5.  for $i = 1, 2, \dots, N$, do
6.      Normalise weights ${w_t}^{(i)} = \frac{{w_t}^{(i)}}{\sum_{i=1}^{N} {w_t}^{(i)}}$
7.  end for
8.  Calculate effective sample size $N_{eff} = \frac{1}{\sum_{i=1}^{N} \left({w_t}^{(i)}\right)^2}$
9.  if $N_{eff}$ is less than threshold $N_{thr}$ then
10.     Resample $\left\{{\theta_t}^{(i)\prime},\ {w_t}^{(i)\prime}\right\}_{i=1}^{N}$
11.     Apply regularisation perturbance and scramble $\left\{{\theta_t}^{(i)},\ {w_t}^{(i)}\right\}_{i=1}^{N}$
12.     Order environment state variables of source $(x, y, I)$ in descending order of intensity $I$
13. else
```

```
14.    Reuse old particles ${\theta_t}^{(i)} = {\theta_{t-1}}^{(i)}$
15. end if
Ensure: new particles and new weights $\{{\theta_t}^{(i)},\ {w_t}^{(i)}\}_{i=1}^{N}$
```

3.4 Source term estimation via environment belief state

In order to use the environment belief state from the particle filter to produce an estimation of the parameter(s) (intensity and location) of source(s) and number of sources, the mode in number of sources r across all particles, r_{mode} is used. The estimation is computed only at the end of each episode to reduce computational load. The set of particles with number of sources belief $r^{(i)} = r_{mode}$, $\pi_{t,\ mode}$ with number of particles N_{mode} is used for computing the estimate. The mean of the set of particles $\pi_{t,\ mode}$ is core to the source term estimate, and therefore environment state weighted mean $\overline{\theta_E}$ at the end of episode E is expressed as

$$\overline{\theta_E} = \frac{\sum_{i:r^{(i)}=r_{mode}}^{N_{mode}} {w_t}^{(i)} \times {\theta_t}^{(i)}}{\sum_{i:r^{(i)}=r_{mode}}^{N_{mode}} {w_t}^{(i)}} \text{ [13].} \tag{22}$$

The environment state weighted standard deviation s_E is used to assess the confidence associated with the belief state, and is similarly computed only at the end of each episode to reduce computational load. Unlike the weighted mean, the weighted standard deviation of the number of sources r, s_r, is also computed – this time using the set of all particles π_t with N particles and set of all weights ${w_t}^{(i)}$ with N weights – to capture the uncertainty in source count as

$$s_r = \sqrt{\frac{\sum_{i=1}^{N}({w_t}^{(i)} \times (r_i - \overline{r_w})^2)}{\sum_{i=1}^{N} {w_t}^{(i)}}} \tag{23}$$

where $\overline{r_w}$ is the weighted mean of r given by

$$\overline{r_w} = \frac{\sum_{i=1}^{N} {w_t}^{(i)} \times r_i}{\sum_{i=1}^{N} {w_t}^{(i)}}. \tag{24}$$

The weighted standard deviations of each set of source parameters, (x, y, I), is expressed similarly to the environment state weighted mean $\overline{\theta_E}$ and also only considers the set of particles with number of sources belief $r^{(i)} = r_{mode}$, $\pi_{t,\ mode}$. Therefore, the environment state weighted standard deviation s_{θ_E} is expressed as

$$s_{\theta_E} = \sqrt{\frac{\sum_{i:r^{(i)}=r_{mode}}^{N_{mode}} \left({w_t}^{(i)} \times \left({\theta_t}^{(i)} - \overline{\theta_{ep}}\right)^2\right)}{\sum_{i:r^{(i)}=r_{mode}}^{N_{mode}} {w_t}^{(i)}}} \tag{25}$$

where the environment state weighted standard deviation of r in s_{θ_E} (which only considers a subset of particles with the same r) is replaced by s_r as expressed in (20).

To remove large fluctuations in the estimation, the source estimate is taken as a moving average of the environment state weighted mean $\overline{\theta_E}$. Similarly, the moving average of s_{θ_E} is also considered. Therefore, source estimate $\widehat{\theta_E}$ up to episode E is expressed as

$$\widehat{\theta_E} = \frac{1}{M}\sum_{k=0}^{E-1} \overline{\theta_{E-k}} \tag{26}$$

and source estimate standard deviation $\widehat{s_{\theta_E}}$ up to episode E is expressed as

$$\widehat{s_{\theta_E}} = \frac{1}{M}\sum_{k=0}^{E-1} s_{\theta_{E-k}} \tag{27}$$

where M is the size of the moving window, and is a constant except when $E < M$ when episode number E is used as the moving window instead (i.e., mean of all environment state weighted means/ standard deviations up to the current episode). The implementation of the source term estimation using the belief state from the particle filter is summarised in Algorithm 2.

```
Algorithm 2: Source term estimation using belief state
Require: particles and weights (belief state) $\{{\theta_t}^{(i)},\ {w_t}^{(i)}\}_{i=1}^{N}$
    1. Calculate environment state weighted mean $\overline{\theta_E} = \frac{\sum_{i:r^{(i)}=r_{mode}}^{N_{mode}} {w_t}^{(i)} \times {\theta_t}^{(i)}}{\sum_{i:r^{(i)}=r_{mode}}^{N_{mode}} {w_t}^{(i)}}$
    2. Calculate weighted standard deviation of the number of sources
           $s_r = \sqrt{\frac{\sum_{i=1}^{N}({w_t}^{(i)} \times (r_i - \overline{r_w})^2)}{\sum_{i=1}^{N} {w_t}^{(i)}}}$
    3. Calculate environment state weighted standard deviation and replace weighted standard
       deviation of $r$ with $s_r$
           $s_{\theta_E} = \sqrt{\frac{\sum_{i:r^{(i)}=r_{mode}}^{N_{mode}} \left({w_t}^{(i)} \times \left({\theta_t}^{(i)} - \overline{\theta_{ep}}\right)^2\right)}{\sum_{i:r^{(i)}=r_{mode}}^{N_{mode}} {w_t}^{(i)}}}$
```

```
4. Calculate source estimate $\widehat{\theta_E}$ and source estimate standard deviation $\widehat{s_{\theta_E}}$
       $\widehat{\theta_E} = \frac{1}{M}\sum_{k=0}^{E-1}\overline{\theta_{E-k}}$
       $\widehat{s_{\theta_E}} = \frac{1}{M}\sum_{k=0}^{E-1} s_{\theta_{E-k}}$
Ensure: source estimate $\widehat{\theta_E}$ and source estimate standard deviation $\widehat{s_{\theta_E}}$
```

3.5 Reward structure

The source term estimation is used to guide the agent to be closer to the estimated source term, in order to mitigate the sparse reward problem (see Section 2.5) while considering the convergence of the source estimate as to not mislead the agent with false estimates. Consequently, a convergence flag for the strongest source C_1 is defined as

$$C_1 = \begin{cases} \text{True}, & \text{if } \widehat{s_1} < s_{thr} \text{ and } \widehat{s_r} < s_{thr,\ r} \\ \text{False}, & \text{otherwise} \end{cases} \tag{28}$$

where $\widehat{s_1} = (\widehat{s_1}_x,\ \widehat{s_1}_y,\ \widehat{s_1}_I)$ is the set of standard deviations of the state variables of the strongest source extracted from $\widehat{s_{\theta_E}}$ and $s_{thr} = (s_{thr_x},\ s_{thr_y},\ s_{thr_I})$ is the standard deviation threshold for the 2-D coordinates of the source and intensity. Similarly, $\widehat{s_r}$ is the standard deviation for the estimated number of sources $\widehat{s_r}$ and $s_{thr,\ r}$ is the standard deviation threshold for the number of sources. A sufficiently low estimate standard deviation means that the source term estimate is trustworthy, i.e. particles are giving source term estimates with low spread.

An episode terminates when it reaches t_{max} steps or if the converged strongest source estimate is within 2 m of the agent. Considering the convergence flag C_1, the reward at the last time step t of each episode E, R_{done}, is defined as

$$R_{done} = \begin{cases} 0.175 \times I_{true} & \text{if } \hat{d} \geq 2 \text{ and } C_1 = \text{true} \\ 0.05\left(d_{max} - \hat{d}\right) & \text{otherwise} \end{cases} \tag{29}$$

where $\hat{d}$ is the Euclidean distance between the location of the agent $[x_{done}, y_{done}]$, $d_{max} = \sqrt{50^2 + 50^2}$ is the maximum Euclidean distance possible between the agent and the strongest source, the location estimate of the strongest source $[\widehat{x_{1_{E-1}}},\ \widehat{y_{1_{E-1}}}]$ from $\widehat{\theta_{E-1}}$. Note that the source term estimate from episode $E - 1$ is used, as the estimate from the current episode is only available after the episode has completed. The coefficients 0.175 and 0.05 scale the reward such that the reward for reaching the true strongest source after convergence is approximately 2-4 times larger than reaching the true strongest source before convergence, to encourage the agent to prioritise the final objective after convergence has been reached.

The first part of the piecewise function for R_{done} is to give the agent a big reward for reaching the final objective of being within 2 m of the source estimate. The reward scales with the true intensity recorded so that reaching a false goal is rewarded significantly less (or slightly less if the source estimate has wrongly identified a weaker source as the strongest).

The second part of the piecewise function for R_{done} encourages the agent to be close to the source by giving a reward which reduces with distance from the source. Unlike the reward for reaching the estimated final objective, this is active even before convergence, with the motivation that the source estimate is unlikely to be close to the agent if the agent is far away from any of the sources, and this reward encourages the agent to explore areas away from its starting location. If the agent is already close to the estimated strongest source, this reward encourages the agent stay close to the source to collect observations and verify whether the estimate is correct.

During an episode, negative rewards are used to motivate the agent before the last time step t in an episode, expressed as

$$R_{not\ done} = \begin{cases} -0.0025 \times \widehat{I_{1_{E-1}}} & \text{if } x_{done} \notin [0, 50] \text{ or } y_{done} \notin [0, 50] \\ -0.002 & \text{otherwise} \end{cases} \tag{30}$$

where $\widehat{I_{1_{E-1}}}$ is the estimated intensity of the strongest source from source estimate $\widehat{\theta_{E-1}}$.

The first part of the piecewise function for $R_{not\ done}$ discourages the agent from attempting to exit the search area by giving it a negative reward when it attempts to. It is not physically possible for the agent to exit the search area; the action is rendered ineffective, and the agent stays in place.

The second part of the piecewise function for $R_{not\ done}$ gives the agent a small negative reward for every time step t to encourage the agent to reach the final objective and terminate the episode with a positive reward within the least time steps t per episode.

The combination of R_{done} and $R_{not\ done}$ gives the full reward function R designed in this paper.

3.6 Reward-driven exploitation

As introduced in Section 2.3, the deep Q-network (DQN) algorithm is used to guide the exploitation of rewards by the agent. The state s_t at time step t for input to Q-network is expressed as

$$s_t = y_t \times x_{max} + x_t + 1. \tag{31}$$

This allows for the representation of a maximum of

$$s_{max} = (x_{max} + 1)(y_{max} + 1) = (50 + 1)(50 + 1) = 2601 \tag{32}$$

distinct agent states, with $s_t \in [1, 2601]$, each corresponding to a unique discrete 2D location in the environment. Note the state s_t is different from the environment state considered in particle filtering (see section 3.3), which considers the source quantity/parameters.

The structure of the Q-network is shown in TABLE III below.

TABLE III – Number of nodes in each layer of Q-network

Input layer (nodes)	Hidden layer 1 (nodes)	Output layer (nodes)
2601	2500	4

The Q-network receives the one-hot encoded state vector of size 2601 of as input. For a given state $s_t = n$, the input vector e_s is $e_s \in R^{s_{max}}$ is defined as

$$e_s[i] = \begin{cases} 1 & if\ i = n \\ 0 & otherwise \end{cases} \tag{33}$$

The network outputs a Q-value for each of the 4 discrete actions in action set

$$A(s) = \{\uparrow, \downarrow, \leftarrow, \rightarrow\} \tag{34}$$

Each action moves the agent 1 m in the corresponding direction.

The Q-network is trained by minimising the loss function in (5). To minimise the loss function, the Adaptive Moment Estimation (Adam) optimiser is used. It is one of the most widely used optimisers in deep learning due to its robust performance and fast convergence. It computes individual adaptive learning rates for different parameters from first and second moment estimates of the gradients, obtaining a faster convergence rate than simpler algorithms like stochastic gradient descent (SGD) [21].

Over the course of training, the Q-network is gradually refined, and its approximation converges toward the optimal action-value function $q_*(s, a)$. Once the Q-network is sufficiently accurate, exploiting it to derive a policy becomes straightforward — at time step t, the agent simply selects the action a^* that maximizes $Q(s_t, a)$ at each state s expressed as

$$a_t = a^* = \arg\max_{a \in A(s)} Q^*(s_t, a)$$

This section focuses solely on the exploitation aspect of RL, where decisions are guided by maximising expected reward. The complementary topic of exploration — essential for discovering optimal behaviour — is discussed in the following section.

3.7 Goal-directed exploration

A classic RL strategy is the ϵ-greedy algorithm, where the hyperparameter epsilon $\epsilon \in (0,1)$ is used to balance exploitation and exploration. For each time step t, the agent has probability of ϵ to make a random action a from $A(s)$, in order to explore the environment. This approach is characteristic of model-free methods, where the reward signal is available, but the underlying mechanism for obtaining that reward is not explicitly known or modelled.

In contrast, the source term estimation and seeking problem involves a degree of knowledge about the environment — specifically, estimates of the number and parameters of the radiation sources. This allows us to deviate from purely model-free strategies and instead adopt a model-based or guided approach to improve the efficiency of Q-learning.

The discussion on RL exploration strategies by Ladosz et al. [14] describes goal-based methods, where states of interest are used to guide the agent in exploration.

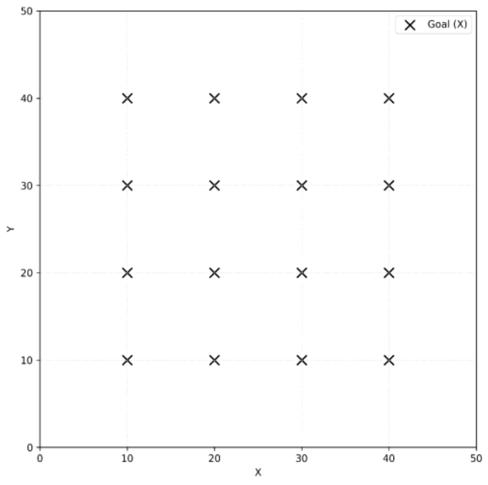


Fig. 2 – Static goals for goal-directed exploration

Fig. 3 (adapted from [14]) shows the set of static goals $G_{static} = \{G_i\}_{i=1}^{n}$ used in this paper, where $G_i = (x_i, y_i)$ and n is the total number of goals, which are used to ensure that the exploration covers the search area adequately. This helps the particle filter avoid convergence to incorrect local minima or escape from existing ones.

A set of static goal probabilities at time-step t, P_{static_t}, corresponds to G_{static}, where each goal $G_i \in G_{static}$ is associated with a probability $P_i \in P_{static_t}$. The set P_{static_t} is initially uniformly distributed and normalised. When a goal G_i is selected, its associated probability P_i is decayed by a factor β, promoting exploration by reducing the likelihood of repeatedly selecting the same goal. After each update, P_{static_t} is normalised to ensure the probabilities sum to 1.

The set of dynamic goals are $G_{dynamic} = \left\{(\widehat{x_{k_{E-1}}},\ \widehat{y_{k_{E-1}}})\right\}_{k=1}^{r}$, where the goals are the estimated location of sources from the previous episode $E - 1$ as described in section 3.4. The source locations are designated as goals, because from (8) the intensity measured at the agent's location I_{true} is dominated by the closest source if the agent is very close to the source. By treating the estimated source positions as goals, the agent is guided toward areas of high radiation intensity. This allows the particle filter's posterior distribution to, more effectively, either correct a previously inaccurate estimation or reinforce an accurate one — both of which bring the agent closer to the actual source. In doing so, the agent is also more likely to receive a higher reward.

Algorithm 3 describes the stochastic process which is used to pick the goal for an episode E at the start of the episode. In this paper, a hybrid goal selection strategy is implemented, where there is a 50% chance of selecting a static goal and 50% chance of selecting a dynamic goal.

Algorithm 3: Hybrid goal selection strategy

```
Require: Static goals $G_{static}$ with probabilities $P_{static}$, Dynamic goals $G_{dynamic}$
    1. Generate random number $f \in [0, 1]$
    2. if $f < 0.5$ then
    3.     Sample selected goal $g = G_i$ from $G_{static}$ with probability $P_{static}$
    4.     Decay the associated probability $P_i$: $P_i \leftarrow \beta \times P_i$
    5.     Renormalise $P_{static}$ such that $\sum_{i=1}^{n} P_i = 1$
    6. else
    7.     Sample goal $g$ uniformly from $G_{dynamic}$
    8. end if
Ensure: static goal probabilities $P_{static}$, selected goal $g$
```

The selected goal g is used for goal-directed exploration by considering goal-directed probability ϱ. The proposed approach is a combination of ϵ-greedy random exploration and navigation towards the goal. Let the agent be at location $l_{t+1} = (x_{t+1}, y_{t+1})$ at time step $t+1$ after an action a is executed. At each time step, the agent selects an action a_t according to

$$a_t \sim \begin{cases} \arg\min\limits_{a \in A(s)} d(l_{t+1}, g), & with\, probability\, \varrho \\ \text{Uniform}(A(s)), & with\, probability\, 1-\varrho \end{cases} \tag{35}$$

where $d(l_{t+1}, g)$ is the Euclidean distance between the predicted next location l_{t+1} and the selected goal g, and $A(s)$ is the set of actions available at state s.

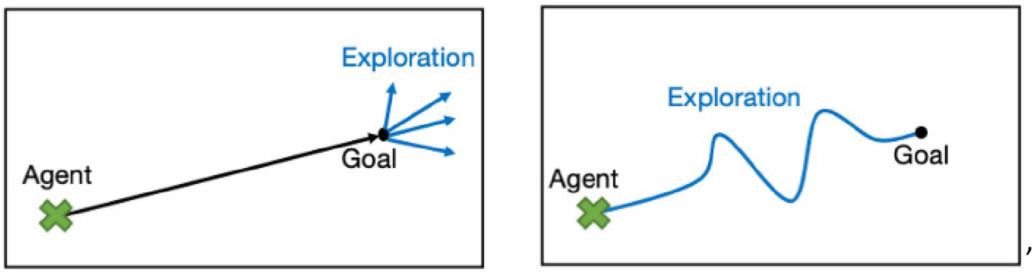


Fig. 3 – Goals to explore from (left) and exploratory goals (right) [14]

The action selection strategy defined in enables the selected goal g to serve both as a destination for exploration and as a source from which further exploration occurs, as in illustrated in Fig. 3. If the maximum number of steps per episode, t_{max}, is set to exceed the number of steps required to reach g—accounting for a probability $1 - \varrho$ of making no progress—the agent can reliably reach the goal before the episode ends. The residual probability $1 - \varrho$ variability in the agent's trajectory, encouraging additional exploration en route. Moreover, upon reaching g, the agent continues to explore the surrounding area, effectively treating g as a starting point for local exploration. Thus, g functions both as an exploratory goal and a goal to explore from.

3.8 Active goal-directed reinforcement learning

The particle filtering and reinforcement learning components work in synergy, providing mutual support. For example, the source estimate produced by the particle filter informs the reward structure, enabling reward-driven exploitation, and provides dynamic goals for goal-based exploration. In turn, the exploratory behaviour guided by goal selection helps improve the accuracy of the particle filter's source estimation.

The overall implementation of active goal-directed reinforcement learning is summarised in Algorithm 4 (roughly follows algorithm structure of AID-RL in [5]). It begins with an initialisation procedure of the particle filter, Q-network, hyperparameters, static goal probabilities, and the initial goal for the first episode. The algorithm then runs episodically, selecting a goal g at the start of every episode, and balances between goal-directed exploration and reward-driven exploitation with ϵ-greedy approach expressed as

$$\epsilon_t = \epsilon_{end} + (\epsilon_{start} - \epsilon_{end}) \times \exp\left(\frac{-t}{\epsilon_{decay}}\right) \tag{36}$$

where ϵ_t is the probability that agent chooses to make an action guided by goal-directed exploration at time step t, and ϵ_{start}, ϵ_{decay}, and ϵ_{end} are hyperparameters that govern the exponential decay of ϵ_t as the learning progresses and the agent learns more about the environment. Consequently, the particle filter and Q-network is updated every time step t, and the source estimate is updated at the end of every episode. The active goal-directed reinforcement learning runs until the maximum number of episodes E_{max}.

Algorithm 4: Active goal-directed reinforcement learning for autonomous search

```
1.  Initialise learning hyper-parameters
2.  Initialise particle set $\pi_t = \{\theta_t^{(i)}\}_{i=1}^{N}$ uniformly across environment state space, and
    weight set $w_t = \{w_t^{(i)}\}_{i=1}^{N}$ with equal and normalised weights
3.  Initialise Q-network weights randomly
4.  Set goal for episode $E = 1$ as (25, 25) (middle of search area)
    **Episodic learning:**
5.  **for** episode $E = 1 : E_{max}$ **do**
6.    Select a goal $g$ using Algorithm 3 (except for episode $E = 1$)
7.    **for** step $j = 1 : t_{max}$ **do**
8.      Update the particle filter (belief state) using Algorithm 1
9.      **if** random value $\in [0, 1] > \epsilon_t$ **then**
10.       Choose action $a_t$ based on reward-driven exploitation:
              $a_t = a^* = \arg\max_{a \in A(s)} Q^*(s_t, a)$
11.     **else**
12.       Choose action $a_t$ based on goal-directed exploration:
              $a_t \sim \begin{cases} \arg\min_{a \in A(s)} d(s_{t+1}, g), & with\ probability\ \varrho \\ \text{Uniform}(A(s)), & with\ probability\ 1 - \varrho \end{cases}$
13.     **end if**
14.     Execute action $a_t$
15.     Collect reward $R_{t+1}$ according to reward function $R$ described in section 3.5
```

```
16.         Update Q-network by minimising a sequence of loss functions:
                $L_t(\omega_t) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[\left(y_t - Q(s,a;\omega_t)\right)^2\right]$
            using experience replay and Adam optimiser
17.         **if** $R = R_{done}$, i.e. termination condition is met, **then**
18.             **break**
19.     **end for**
20.     Extract source term estimate from environment belief state using Algorithm 2
21. **end for**
```

4 Results and discussion

4.1 Introduction

The results section begins by detailing the hyperparameter configurations used for training the Q-network, the particle filtering algorithm, and the goal-directed exploration strategy. The computational hardware used to obtain these results is also specified. These parameters are manually tuned to achieve a reasonably well-performing model. Results are first presented for the initial environment configuration used during model validation.

The goal-directed exploration strategy is compared against a traditional ϵ-greedy approach employing random exploration, in order to evaluate whether goal-directed behaviour provides a measurable performance advantage.

4.2 Hyperparameter configuration and computational hardware

All simulations and training were run on a local machine equipped with an Intel i7-11800H CPU, 16 GB of RAM, and an NVIDIA RTX 3060 (Mobile) GPU. The simulation is also compatible with Google Colab, where the free GPU runtime typically provides a 2-core Intel Xeon CPU, approximately 13 GB RAM, and a Tesla T4 GPU. However, due to runtime limitations on the free tier, the runtime usually terminates in 2-3 hours. Therefore, the Google Colab free tier is not suitable for running the full simulation (Google Colab Pro would be suitable but requires paying for extra time) which requires approximately 8 hours to complete 2000 episodes.

The hyperparameter configuration for the particle filter, source estimation, reward structure and goal-driven exploration is shown TABLE IV below.

TABLE IV – Values of hyperparameters for particle filter, source estimation, reward structure and goal-driven exploration

Symbol	Parameter description	Value
N	Number of particles	2500
r_{max}	Maximum number of sources	5
$\delta_{loc,\ start}$	Initial value of location estimate perturbation	0.5
$\delta_{I,\ start}$	Initial value of intensity estimate perturbation	0.5
δ_{decay}	Decay rate of perturbations (Higher values result in slower decay)	40000
δ_{end}	Final value of source estimate perturbation	0.05
$\delta_{r,\ start}$	Initial probability of source number estimate perturbation	0.004
$\delta_{r,\ end}$	Minimum probability of source number estimate perturbation	0.0004
M	Window size for moving average used to compute source estimate	25
s_{thr}	Standard deviation threshold for source parameter estimates	6.0
$s_{thr,\ r}$	Standard deviation threshold for source number estimates	0.6
t_{max}	Maximum number of steps per episode	150
ϱ	Goal-directed probability	0.5

The hyperparameter configuration for reward-driven exploitation and active goal-directed reinforcement learning is shown in TABLE V below.

TABLE V – Values of hyperparameters for reward-driven exploitation and active goal-directed reinforcement learning

Symbol	Parameter description	Value
b	Sampled batch size for experience replay	32
γ	Discount factor	0.99
ϵ_{start}	Starting exploration rate	1.0
ϵ_{decay}	The rate at which ϵ_t decays over time (Higher values result in slower decay)	80000
ϵ_{end}	Minimum exploration rate	0.0
τ	Mixing factor	0.005
α	Learning rate	0.001

4.3 Performance evaluation and baseline comparison

To compare the behaviour of both active goal-directed RL and the baseline strategy using ϵ-greedy with random exploration, the path taken by the agent at the beginning, half-way through training (1000 episodes), and at the end of training is shown and analysed accordingly.

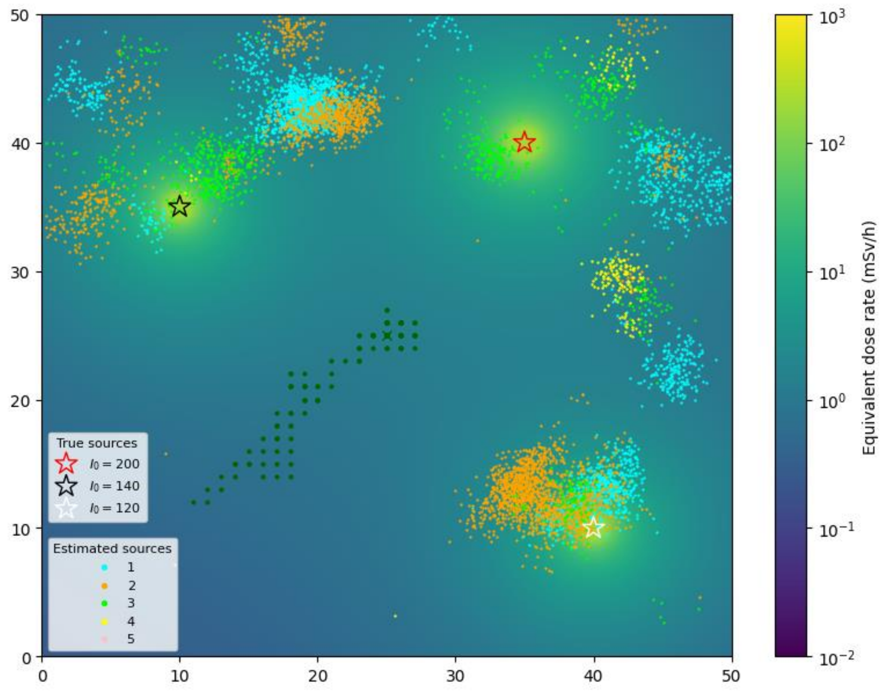


Fig. 4 – Path at the first episode for active goal-directed RL

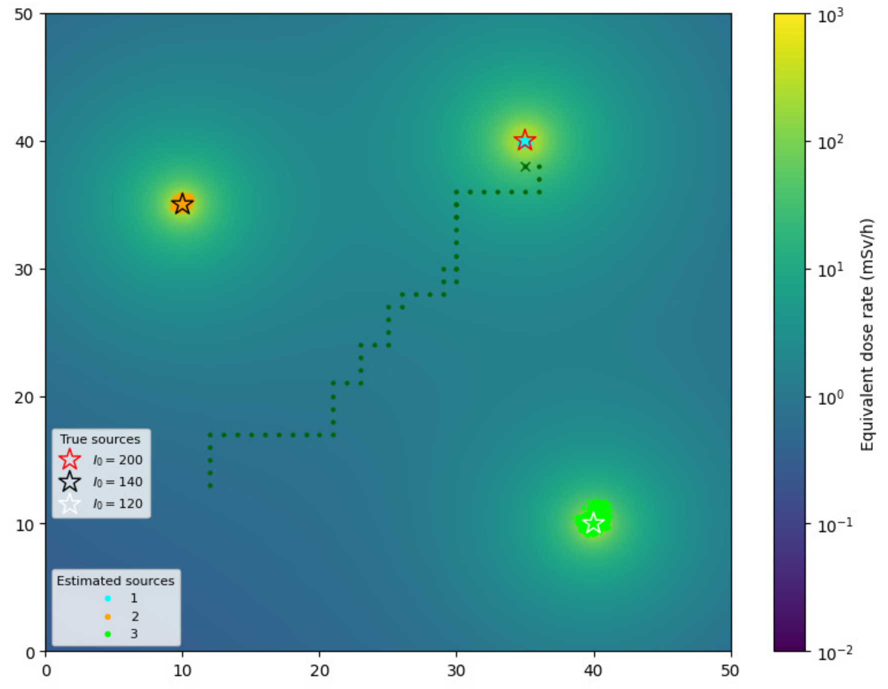


Fig. 5 – Path at the 1000th episode for active goal-directed RL

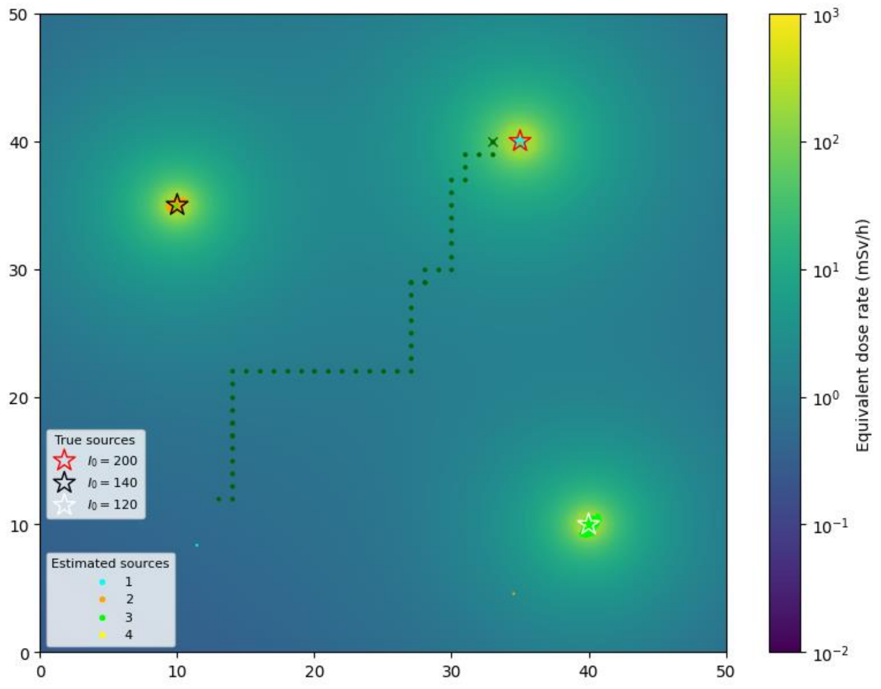


Fig. 6 – Path at the last episode (2000th) for active goal-directed RL

Dark green dots represent the search trajectory; dark green cross denotes the current location of the agent; stars denote the true location of sources and smaller coloured dots represent the particle filter, where each colour represents the belief of a distinct source. A colour gradient represents the radiation distribution in the environment; note that a logarithmic scale is used. Note that the agent starts each episode from (12, 12).

As shown in Fig. 4, the agent begins exploration with no prior knowledge of the source locations. It can be seen that agent combines random exploration with heading to the initial goal of (25, 25) which is set at the first episode. The estimated source positions from the particle filter are scattered and noisy, with several false positives present particularly in areas in away from the agent. This demonstrates the ability for the particle filter to quickly deduce correctly that the sources are far away from the agent, even if the localisation is not complete. The agent is unable to locate the strongest source and the episode truncates at t_{max}.

By mid-training in Fig. 5, the agent has localised all of the sources with reasonable confidence, with estimates converging to the true positions. Note that S_3, which is far away from the agent, is localised with less confidence, which confirms the idea suggested in section 3.7 that being in close proximity with a source improves the prediction of said source. The agent's trajectory is more directed at the final objective of the strongest source, and the episode terminates successfully.

At the end of training in Fig. 6, the agent has localised all of the sources with even higher confidence, and the situation is similar to that of Fig. 5. The agent now heads towards the strongest source using the shortest path possible and the episode terminates successfully.

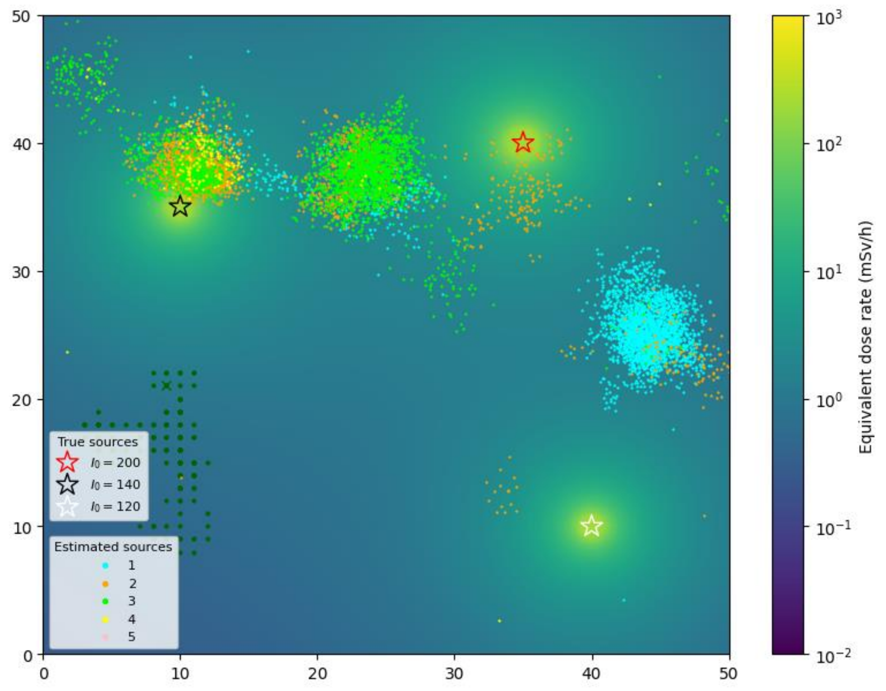


Fig. 7 – Path at the first episode for ϵ-greedy with random exploration

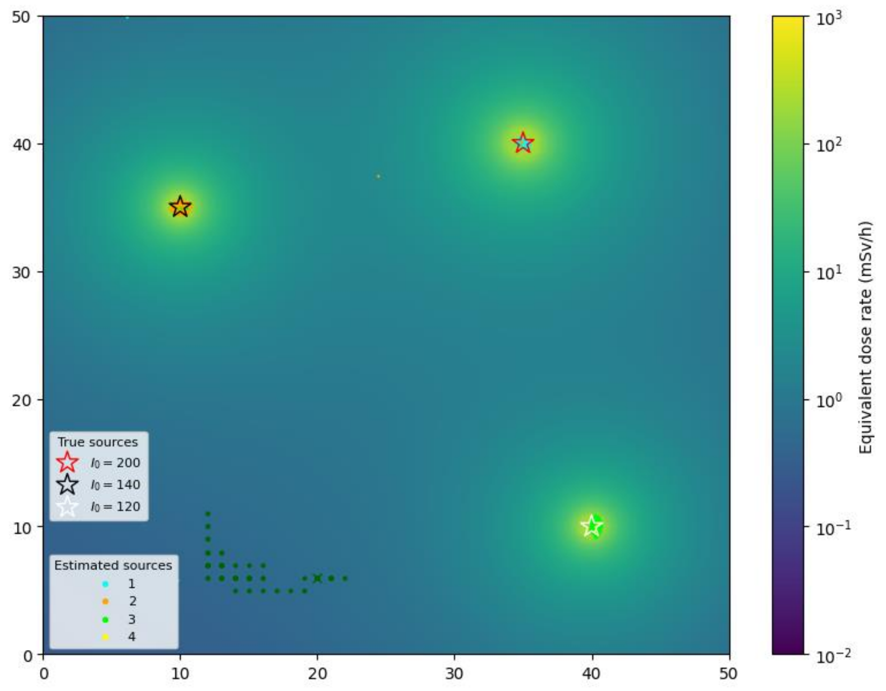


Fig. 8 – Path at the 1000th episode for ϵ-greedy with random exploration

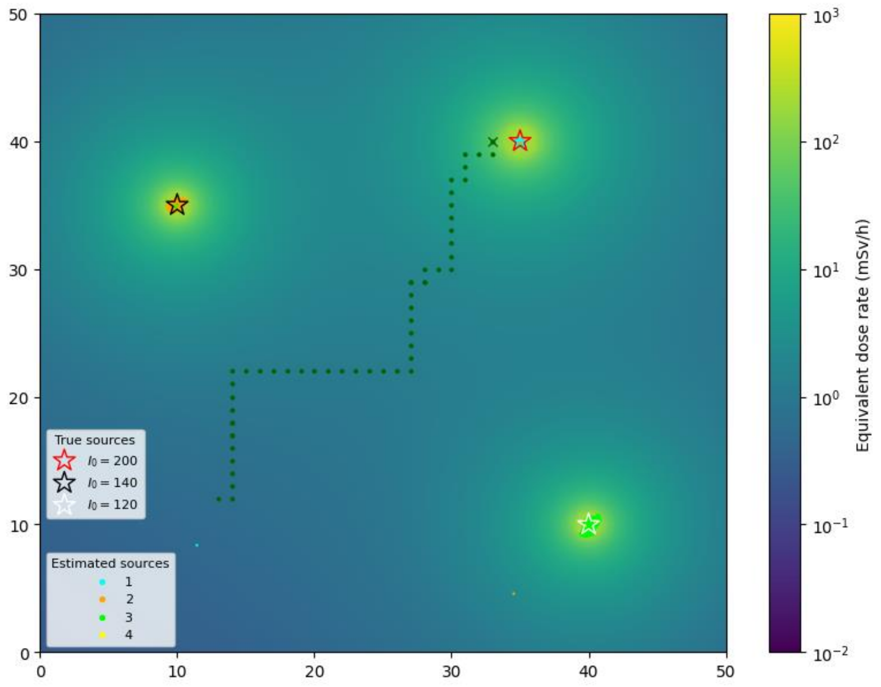


Fig. 9 – Path at the last episode (2000th) for ϵ-greedy with random exploration

Fig. 7, Fig. 8, and Fig. 9 use the same visual format as explained for Fig. 4, Fig. 5, and Fig. 6 earlier.

As shown in Fig. 7, the agent begins exploration with no prior knowledge of the source locations. It can be seen that the random exploration by the agent has led it to stay far away from the sources. The estimated source positions from the particle filter are scattered and noisy, and it can be argued that the estimate is worse than that of the one depicted in Fig. 4, as false positives are much more pronounced; it takes more time for the algorithm to escape false minima than to converge to a true minima from a random distribution.

By mid-training in Fig. 8, the situation is very similar to that of Fig. 5, but without goal-directed exploration the policy to navigate to the strongest source has not been learned yet and the agent hovers in proximity to the starting location.

At the end of training in Fig. 9, the situation is equivalent to the one in Fig. 6, showing that ϵ-greedy with random exploration is still effective but takes a larger number of episodes E to complete source seeking.

The performance of active goal-directed RL is evaluated using a set of metrics designed to firstly benchmark both source estimation. The results are then compared to the baseline strategy using ϵ-greedy with random exploration. The experiments are conducted with the starting agent and environment parameters as specified in section 3.3. To ensure reproducibility, simulation and training for both approaches are repeated three times. Convergence for the weaker sources

(S_2 and S_3) is determined in a similar manner to the convergence for S_1 described in (28) with the convergence of S_2 and S_3 corresponding to convergence flags C_2 and C_3 respectively. Estimate errors are computed by taking the mean of estimate error after convergence.

TABLE VI – Averaged estimation results: values after the $\pm$ represent the standard deviation from all performed simulated environments (Table structure inspired by [13])

Source	Metric assessed	Active goal-directed RL	ϵ-greedy RL with random exploration
S_1	First converged episode	93.5 ± 31.5	132 ± 36.4
	Localisation error (m)	0.143 ± 0.061	0.141 ± 0.042
	Intensity estimate error (mSv/h)	2.356 ± 1.188	2.554 ± 1.142
S_2	First converged episode	214 ± 32.5	249 ± 31.2
	Localisation error (m)	0.137 ± 0.032	0.168 ± 0.045
	Intensity estimate error (mSv/h)	1.359 ± 0.332	0.982 ± 0.415
S_3	First converged episode	232.5 ± 27.5	248 ± 29.3
	Localisation error (m)	0.340 ± 0.069	0.344 ± 0.039
	Intensity estimate error (mSv/h)	1.984 ± 0.230	4.140 ± 0.215
Rate of convergence (over 3 trials)		100%	66.6%
Number of sources			
	Estimate error	0.00135 ± 0.00012	0.00075 ± 0.00032

Note that the error in results exclude the first trial for ϵ-greedy RL with random exploration as the first trial did not converge at all. We can observe that although the estimate errors are similar for both approaches, active goal-directed RL consistently achieves estimate convergence faster.

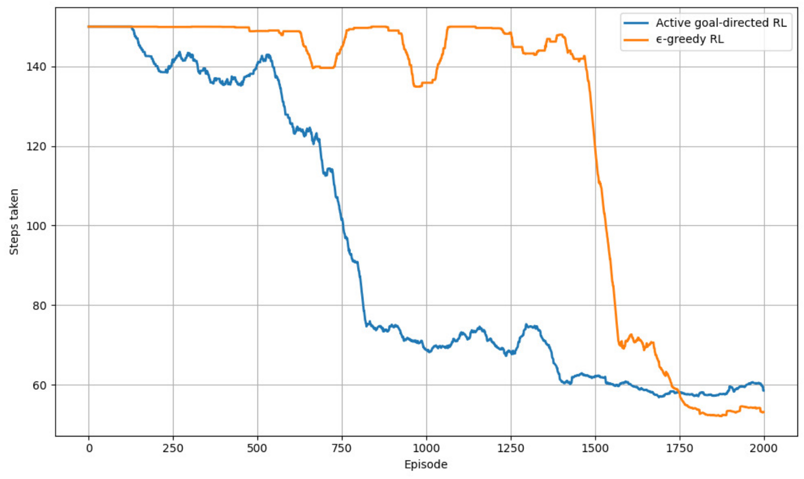


Fig. 10 – Moving average of episode length for both approaches

A plot of the steps taken per episode over the course of episodic learning highlights the significantly faster convergence for finding the optimal source seeking policy of the active goal-directed RL approach.

4.4 Effect of source quantity on convergence

To test the robustness of the proposed goal-directed approach, the amount of sources in the environment were varied and randomised (keeping within constraints stated in section 3.2 to limit problem complexity), and the first episode to converge is tabulated below. The simulation was run once per no. of source(s). Note that a different benchmark is used to determine convergence – convergence is defined to be when estimation error is consistently below $1\ \mathrm{m}$. This change was due to the observation that the original convergence flags C_n only work well with the three sources from the initial environment and estimation errors still showed large fluctuation after all convergence flags were set to true (which is the episode where the error estimate starts to be plotted), especially for larger sources. The rate of epsilon decay also had to decrease for higher number of sources in an attempt to reach convergence; $\epsilon_{decay} = 120000$ was used for 4 sources and $\epsilon_{decay} = 200000$ was used for 5 sources.

TABLE VII – Time taken for estimate convergence vs. Number of sources

Number of sources	Approximate convergence episode (all errors < $1\ \mathrm{m}$)	Notes
1	~200	Already converged when plot begins
2	~250	Already converged when plot begins
3	~600-700	Source 3 takes longer to drop under $1\ \mathrm{m}$
4	~1000+	Extremely large errors initially; late convergence
5	Never converges (> 2000)	Extremely large errors throughout 2000 episodes

4.5 Summary

The proposed active goal-directed reinforcement learning (RL) algorithm was evaluated through a series of simulation trials and benchmarked against a baseline strategy using ϵ-greedy RL.

Estimation performance was assessed using time to convergence, and parameter estimation accuracy. Source seeking efficiency was measured via the steps taken per episode.

Across trials, the goal-directed method consistently enabled the agent to converge to source estimates earlier than the random exploration strategy, with convergence occurring an average of 30 episodes earlier and reflecting a 16.5% decrease in time to convergence. Both methods achieve similarly accurate source estimation, with all localisation errors being below 0.5% of the environment's diagonal distance ($\sqrt{50^2 + 50^2}\ \mathrm{m} \approx 70.71\ \mathrm{m}$), and intensity estimate errors being below 4% of the source strength. It is also important to note that goal-directed RL achieves more reliable convergence, converging 100% of the time in the 3 trials while random exploration ϵ-greedy RL only converged in 2 of the trials, with a 66% rate of convergence.

While the ϵ-greedy RL requires approximately 1600 episodes before abruptly discovering an efficient path to the strongest source, the active goal-directed RL agent showed a more gradual but overall faster decline in the steps taken, already having learned a relatively efficient path at approximately episode 800. Considering distance travelled as a metric for the efficiency of source seeking, these results suggest that active goal-directed RL is approximately twice as fast as ϵ-greedy RL in finding an efficient policy for source seeking.

In addition to the standard trials, the number of sources in the environment was varied from one to five to assess how the source count affects convergence time. Results show that as number of sources increase, convergence is significantly delayed, with environments containing four sources requiring over half the training time to converge. The five-source scenario failed to converge within 2000 episodes.

These results confirm that incorporating goal-based exploration enhances the efficiency of the RL algorithm by guiding the agent toward meaningful regions of the environment, improving both the speed and reliability of source estimation. However, in environments with more than three sources, the performance of the algorithm quickly deteriorates.

5 Conclusions and future work

5.1 Conclusions

This project presented a novel method for the autonomous localisation and estimation of multiple radiation sources by integrating particle filtering with goal-directed deep reinforcement learning.

The proposed approach leverages the complementary strengths of both methods: the particle filter offers robust estimation of source parameters based on noisy measurements, while the reinforcement learning agent actively guides exploration using a goal-based policy informed by the current belief state derived from the particle filter.

Simulation results demonstrate that the goal-directed approach enables slightly faster convergence for source estimation, and significantly more efficient source seeking compared to traditional ϵ-greedy random exploration strategies. In particular, goal-directed behaviour led to earlier identification of sources, which allowed the goal-directed RL to exploit an accurate belief state and source estimate to accelerate the learning process. The particle filter was also shown to benefit from directed exploration, as it avoided local minima and false positives more effectively when the agent was incentivised to visit high-information regions.

The results validate the effectiveness of combining model-based estimation with active learning and exploration. This synergy holds promise for real-world applications in nuclear safety and radiological accident mitigation.

5.2 Future work

Several directions can be pursued to further enhance and generalise the proposed framework:

Improved hyperparameter tuning: While the model demonstrates robust performance with manually tuned parameters, more rigorous optimisation methods such as Bayesian optimisation or population-based training could be employed to systematically improve performance. This would improve the overall performance of the framework.

Transfer learning and scalability: The current framework is trained from scratch in a specific environment. Transfer learning could be explored to enable the reuse of trained policies or belief models in similar but unseen environments, reducing retraining time and improving sample efficiency. This is particularly relevant in real-world applications where training opportunities may be limited.

These future developments would contribute to making the system more adaptable, data-efficient, and deployable in more complex and dynamic real-world settings.

References

[1] J. Huo, M. Liu, K. A. Neusypin, H. Liu, M. Guo, and Y. Xiao, "Autonomous Search of Radioactive Sources through Mobile Robots," *Sensors,* vol. 20, no. 12, p. 3461, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/12/3461.

[2] "IAEA Incident and Trafficking Database (ITDB) Factsheet," International Atomic Energy Agency (IAEA), 2024. [Online]. Available: https://www.iaea.org/sites/default/files/24/05/itdb_factsheet_2024.pdf

[3] M. Hutchinson, H. Oh, and W.-H. Chen, "Entrotaxis as a strategy for autonomous search and source reconstruction in turbulent conditions," *Information Fusion,* vol. 42, 2018/07/01, doi: 10.1016/j.inffus.2017.10.009.

[4] H. Hu, J. Wang, A. Chen, and Y. Liu, "An autonomous radiation source detection policy based on deep reinforcement learning with generalized ability in unknown environments," *Nuclear Engineering and Technology,* vol. 55, no. 1, 2023/01/01, doi: 10.1016/j.net.2022.09.010.

[5] Z. Li, W.-H. Chen, J. Yang, and Y. Yan, "AID-RL: Active information-directed reinforcement learning for autonomous source seeking and estimation," *Neurocomputing,* vol. 544, 2023/08/01, doi: 10.1016/j.neucom.2023.126281.

[6] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman filter : particle filters for tracking applications* (Artech House radar library). Boston, MA: Artech House, 2004, pp. xiii, 299 p.

[7] J. S. Liu and R. Chen, "Sequential Monte Carlo Methods for Dynamic Systems," *Journal of the American Statistical Association,* vol. 93, no. 443, 1998-9-1, doi: 10.1080/01621459.1998.10473765.

[8] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking | IEEE Journals & Magazine | IEEE Xplore," *IEEE Transactions on Signal Processing,* vol. 50, no. 2, 2002, doi: 10.1109/78.978374.

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning : an introduction*, Second edition. ed. (Adaptive computation and machine learning series). Cambridge, Massachusetts: The MIT Press, 2018, pp. xxii, 526 pages.

[10] V. Mnih *et al.*, "Playing Atari with Deep Reinforcement Learning," 2013/12/19, doi: 10.48550/arXiv.1312.5602.

[11] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015/09/09, doi: 10.48550/arXiv.1509.02971.

[12] B. S. Totten, "Multi-Agent Deep Reinforcement Learning for Radiation Localization," 2023, doi: 10.15760/etd.3652.

[13] T. Lazna and L. Zalud, "Localizing Multiple Radiation Sources Actively with a Particle Filter," 2023/05/24, doi: 10.48550/arXiv.2305.15240.

[14] P. Ladosz, L. Weng, M. Kim, and H. Oh, "Exploration in Deep Reinforcement Learning: A Survey," 2022/05/02, doi: 10.48550/arXiv.2205.00824.

[15] G. Matheron, N. Perrin, and O. Sigaud, "PBCS : Efficient Exploration and Exploitation Using a Synergy between Reinforcement Learning and Motion Planning," 2020/04/24, doi: 10.48550/arXiv.2004.11667.

[16] E. Z. Liu *et al.*, "Learning Abstract Models for Strategic Exploration and Fast Reward Transfer," 2020/07/12, doi: 10.48550/arXiv.2007.05896.

[17] R. Cimurs, I. H. Suh, and J. H. Lee, "Goal-Driven Autonomous Exploration Through Deep Reinforcement Learning," 2021/03/12, doi: 10.48550/arXiv.2103.07119.

[18] Ludlum Measurements, "Electronic Personal Dosimeter," Model 23 Series, May 2024.

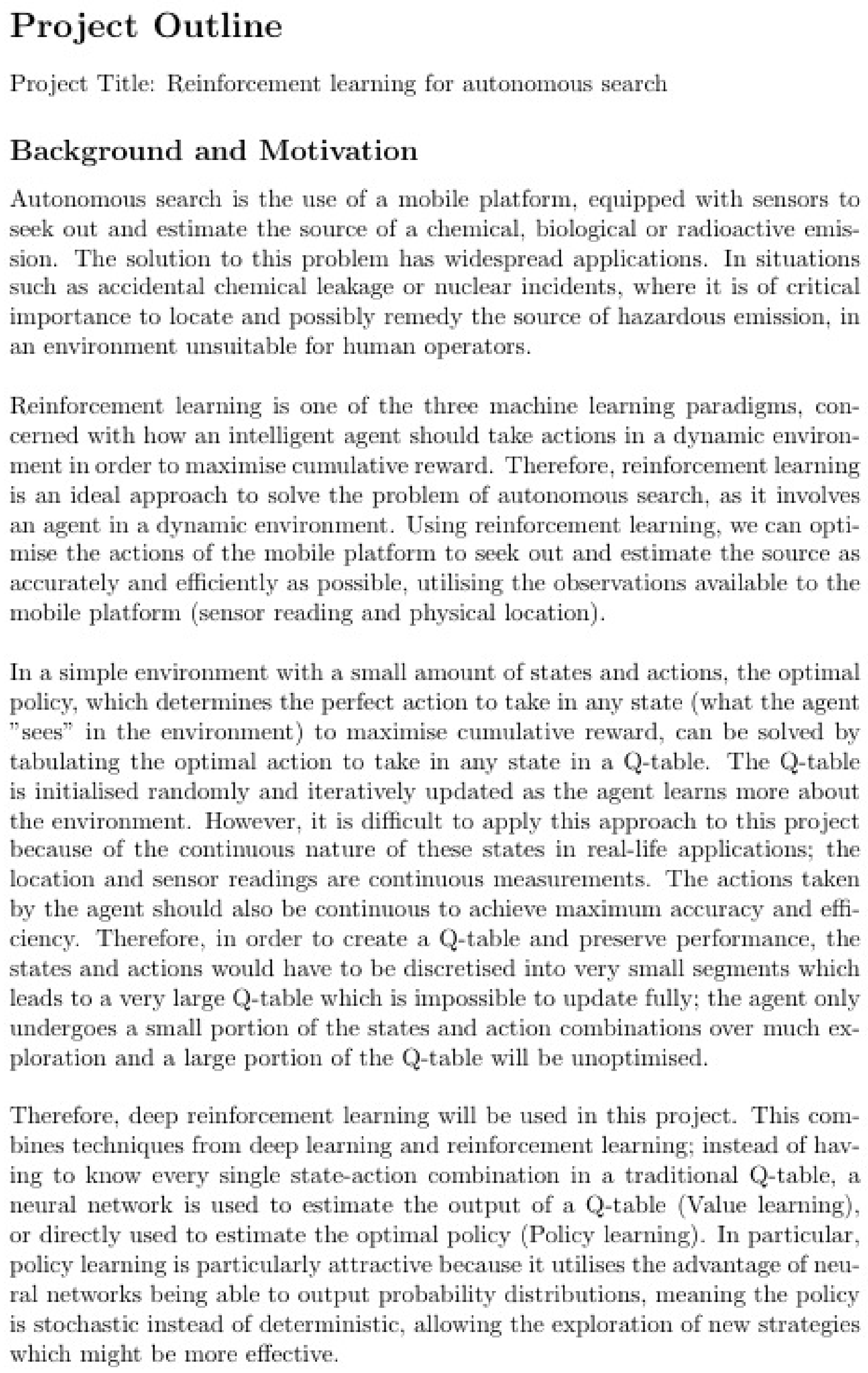
[19] Mirion Technologies, "Personal Electronic Dosimeter," DMC 3000, December 2022.

[20] J. Elfring, E. Torta, R. v. d. Molengraft, J. Elfring, E. Torta, and R. van de Molengraft, "Particle Filters: A Hands-On Tutorial," *Sensors 2021, Vol. 21, Page 438,* vol. 21, no. 2, 2021-01-09, doi: 10.3390/s21020438.

[21] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2014/12/22, doi: 10.48550/arXiv.1412.6980.

Appendices

A – Project Outline



Project Outline

Project Title: Reinforcement learning for autonomous search

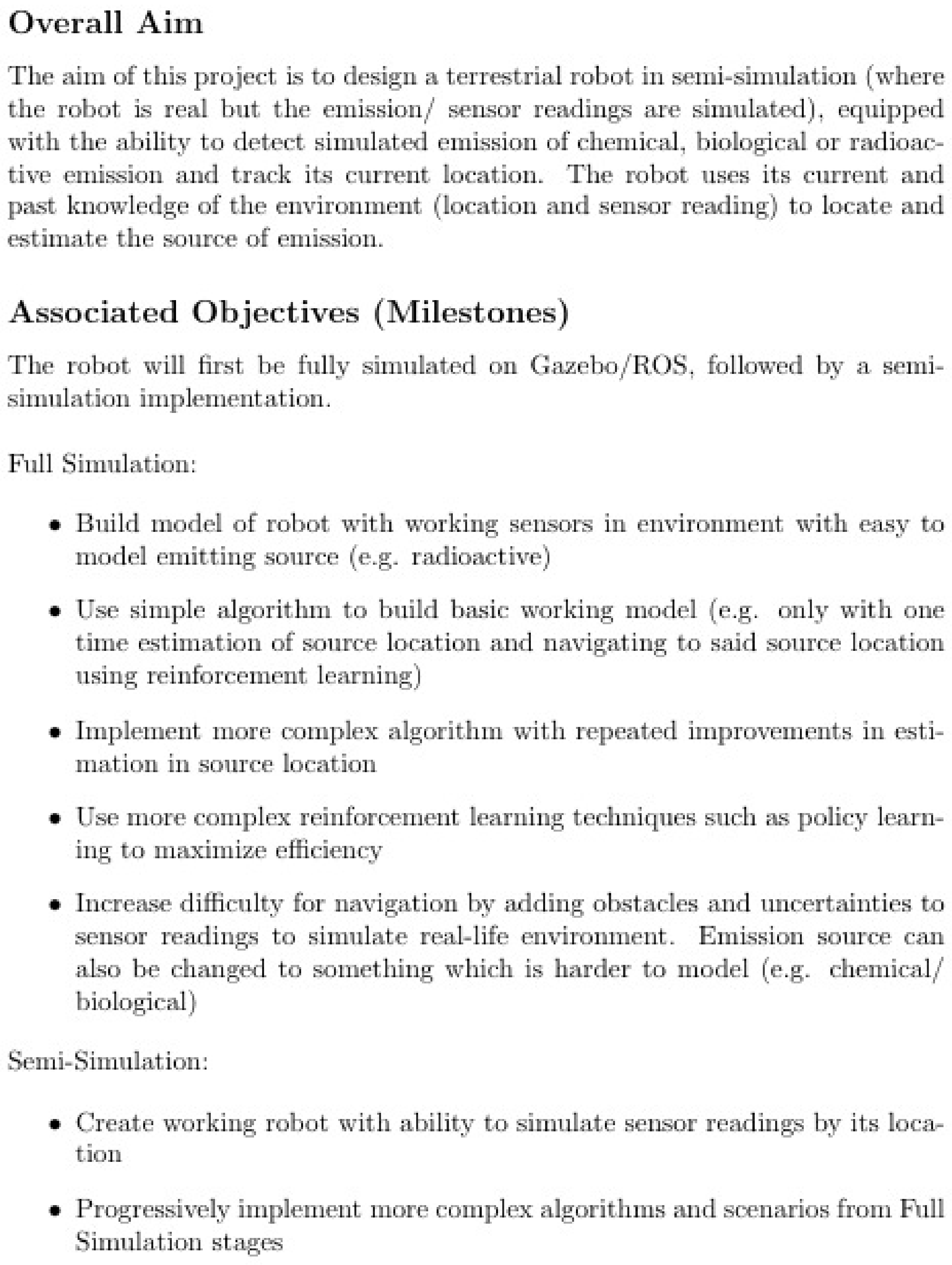
Background and Motivation

Autonomous search is the use of a mobile platform, equipped with sensors to seek out and estimate the source of a chemical, biological or radioactive emission. The solution to this problem has widespread applications. In situations such as accidental chemical leakage or nuclear incidents, where it is of critical importance to locate and possibly remedy the source of hazardous emission, in an environment unsuitable for human operators.

Reinforcement learning is one of the three machine learning paradigms, concerned with how an intelligent agent should take actions in a dynamic environment in order to maximise cumulative reward. Therefore, reinforcement learning is an ideal approach to solve the problem of autonomous search, as it involves an agent in a dynamic environment. Using reinforcement learning, we can optimise the actions of the mobile platform to seek out and estimate the source as accurately and efficiently as possible, utilising the observations available to the mobile platform (sensor reading and physical location).

In a simple environment with a small amount of states and actions, the optimal policy, which determines the perfect action to take in any state (what the agent "sees" in the environment) to maximise cumulative reward, can be solved by tabulating the optimal action to take in any state in a Q-table. The Q-table is initialised randomly and iteratively updated as the agent learns more about the environment. However, it is difficult to apply this approach to this project because of the continuous nature of these states in real-life applications; the location and sensor readings are continuous measurements. The actions taken by the agent should also be continuous to achieve maximum accuracy and efficiency. Therefore, in order to create a Q-table and preserve performance, the states and actions would have to be discretised into very small segments which leads to a very large Q-table which is impossible to update fully; the agent only undergoes a small portion of the states and action combinations over much exploration and a large portion of the Q-table will be unoptimised.

Therefore, deep reinforcement learning will be used in this project. This combines techniques from deep learning and reinforcement learning; instead of having to know every single state-action combination in a traditional Q-table, a neural network is used to estimate the output of a Q-table (Value learning), or directly used to estimate the optimal policy (Policy learning). In particular, policy learning is particularly attractive because it utilises the advantage of neural networks being able to output probability distributions, meaning the policy is stochastic instead of deterministic, allowing the exploration of new strategies which might be more effective.



Overall Aim

The aim of this project is to design a terrestrial robot in semi-simulation (where the robot is real but the emission/ sensor readings are simulated), equipped with the ability to detect simulated emission of chemical, biological or radioactive emission and track its current location. The robot uses its current and past knowledge of the environment (location and sensor reading) to locate and estimate the source of emission.

Associated Objectives (Milestones)

The robot will first be fully simulated on Gazebo/ROS, followed by a semi-simulation implementation.

Full Simulation:

- Build model of robot with working sensors in environment with easy to model emitting source (e.g. radioactive)
- Use simple algorithm to build basic working model (e.g. only with one time estimation of source location and navigating to said source location using reinforcement learning)
- Implement more complex algorithm with repeated improvements in estimation in source location
- Use more complex reinforcement learning techniques such as policy learning to maximize efficiency
- Increase difficulty for navigation by adding obstacles and uncertainties to sensor readings to simulate real-life environment. Emission source can also be changed to something which is harder to model (e.g. chemical/ biological)

Semi-Simulation:

- Create working robot with ability to simulate sensor readings by its location
- Progressively implement more complex algorithms and scenarios from Full Simulation stages

B - Health & Safety Risk Assessment

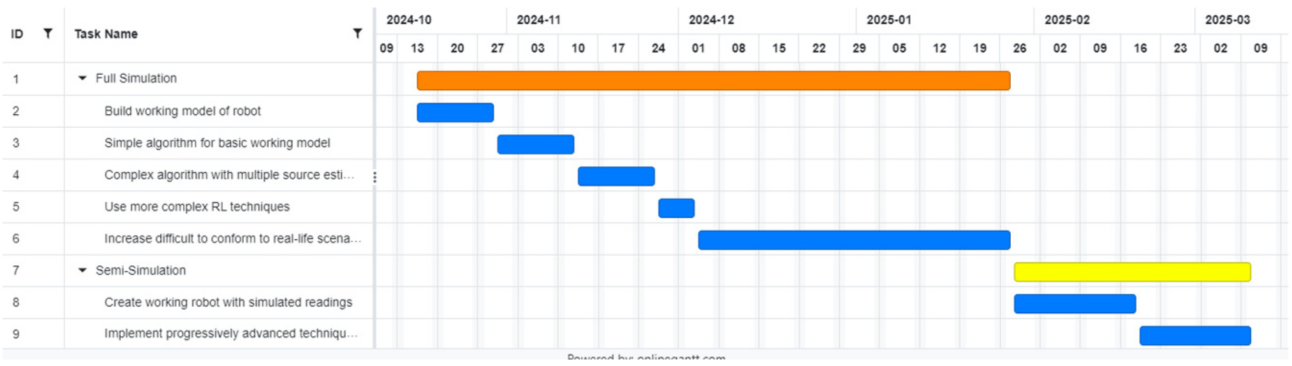
MANCHESTER
1824
The University of Manchester

Risk Assessment for Individual Project Activities

Date: 19/10/2024	**Assessed by:** Chun Hei Wong	**Checked/ Validated* by:**	**Approved by:**	**Building / Location:**	**Assessment ref no:**	**Expiry date:**
Task / premises: (7) Developing reinforcement learning based algorithm to perform autonomous search for radioactive sources, working from home or university study space with laptop.						

Activity	**Hazard**	**Who might be harmed and how?**	**Measures to control risk**	**Risk rating**	**Result**
Extended use of laptop/ computer	Eye strain, repetitive strain injury (RSI), fatigue	User (myself)	• Take regular breaks • Ensure ergonomic setup • Adjust screen brightness	Low	A
Running long simulations while charging laptop Li-ion battery	Overheating, fire risk	User, flatmates	• Ensure adequate ventilation • Always supervise when simulation is being run and laptop is being charged	Low	A
Data loss due to file corruption or crashes	Loss of progress	User	• Use version control (e.g. Git and GitHub) • Regularly back up data • Save data or plots during training	Low	A

C - Initial Project Plan



ID	Task Name
1	Full Simulation
2	Build working model of robot
3	Simple algorithm for basic working model
4	Complex algorithm with multiple source esti...
5	Use more complex RL techniques
6	Increase difficult to conform to real-life scena...
7	Semi-Simulation
8	Create working robot with simulated readings
9	Implement progressively advanced techniqu...

D - Code repository on GitHub

https://github.com/ChunHeiWongIvan/Reinforcement_Learning

Autonomous Radiation Source Localisation and Estimation

This repository implements a simulation-based framework for the autonomous localisation and strength estimation of multiple radioactive sources in a 2D environment. The approach combines particle filtering with deep Q-learning (DQN), using a goal-directed reinforcement learning strategy to actively guide exploration and improve source estimation.

Core Features

- Particle filter for estimating number, location, and intensity of sources.
- Goal-directed deep Q-learning (DQN) to actively navigate the agent toward informative areas in the environment.
- Simulation outputs include:
  - Radiation heatmaps
  - Agent movement visualisation
  - Estimation error plots
  - Source strength and count estimation graphs

Repository Structure

- radiation_STE_discrete.py “ Main simulation script using goal-directed DQN and particle filtering.
- particle_filter.py “ Particle filter implementation (inspired by Roger R. Labbe Jr.'s Kalman and Bayesian Filters in Python).
- radiation_discrete.py “ Environment and agent/source class definitions.

Running the Simulation

Requirements

1. Python 3.9+
2. PyTorch
3. NumPy
4. Matplotlib
5. Statsmodels

To Run

Execute the main simulation file:

python radiation_STE_discrete.py

Training progress and plots will be displayed live (if displayPlot is set to True) and results will be saved in the multi_source_results/ directory.

Notes

- The code auto-detects GPU or MPS acceleration (Mac) if available.
- Simulations may take several hours depending on the hardware. A Colab-compatible version can be adapted.

Output

All key performance metrics, including source localisation error, source count estimates, and intensity estimate error are visualised and exported as:

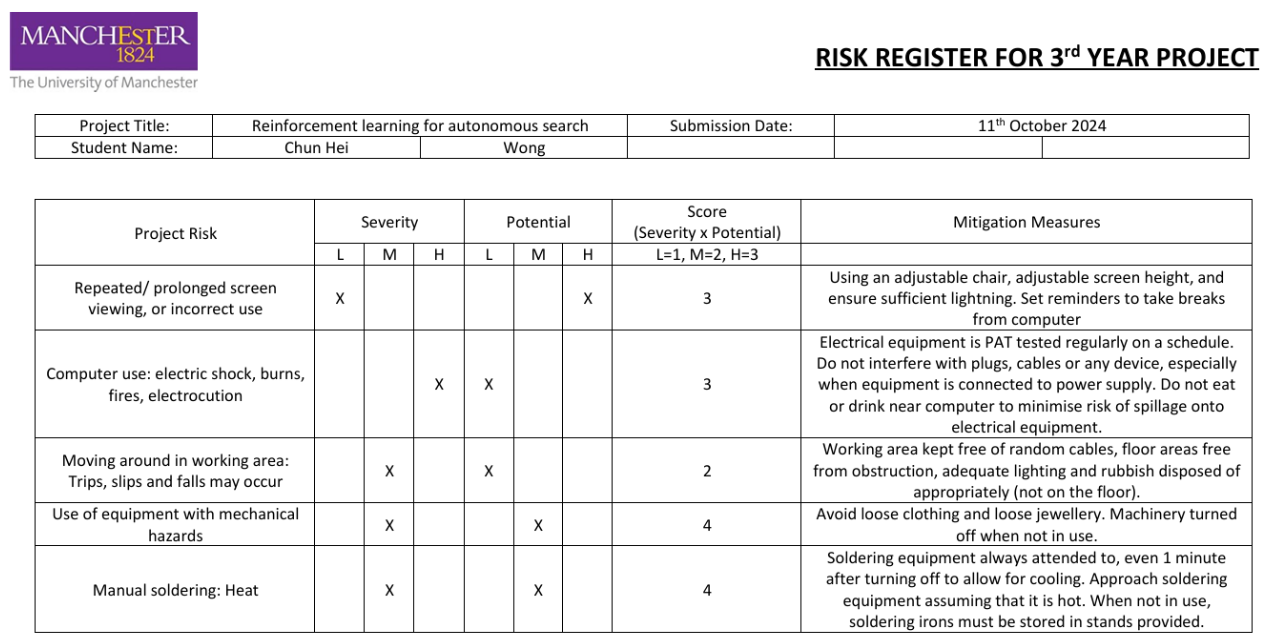
- distance_plot.pkl / dist.png
- length_plot.pkl
- est_strengths.png
- num_sources.png
- estimation_error.png

Future Work

The simulation can be extended to:

1. Support adaptive epsilon-decay strategies.
2. Improve parameter tuning via Bayesian optimisation.
3. Introduce transfer learning across different environments.
4. Evaluate robustness under randomised agent and environment initialisation.

E - Risk Register



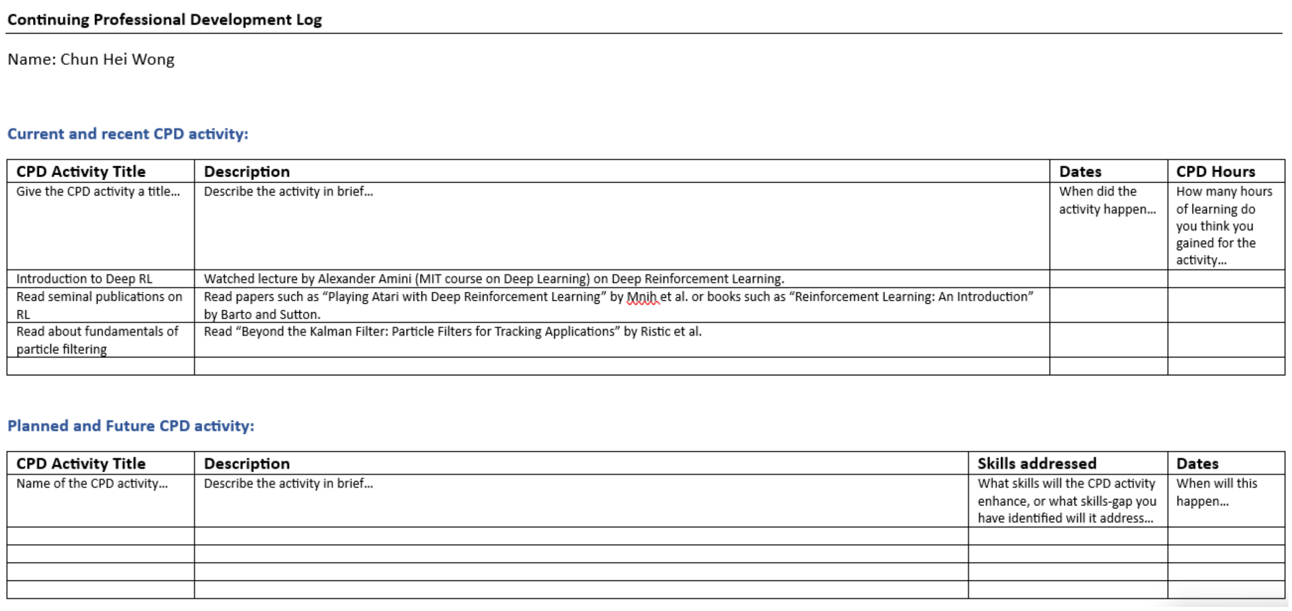
MANCHESTER 1824
The University of Manchester

RISK REGISTER FOR 3rd YEAR PROJECT

Project Title:	Reinforcement learning for autonomous search		Submission Date:	11th October 2024	
Student Name:	Chun Hei	Wong			

Project Risk	Severity			Potential			Score (Severity x Potential)	Mitigation Measures
	L	M	H	L	M	H	L=1, M=2, H=3	
Repeated/ prolonged screen viewing, or incorrect use	X					X	3	Using an adjustable chair, adjustable screen height, and ensure sufficient lightning. Set reminders to take breaks from computer
Computer use: electric shock, burns, fires, electrocution			X	X			3	Electrical equipment is PAT tested regularly on a schedule. Do not interfere with plugs, cables or any device, especially when equipment is connected to power supply. Do not eat or drink near computer to minimise risk of spillage onto electrical equipment.
Moving around in working area: Trips, slips and falls may occur		X		X			2	Working area kept free of random cables, floor areas free from obstruction, adequate lighting and rubbish disposed of appropriately (not on the floor).
Use of equipment with mechanical hazards		X			X		4	Avoid loose clothing and loose jewellery. Machinery turned off when not in use.
Manual soldering: Heat		X			X		4	Soldering equipment always attended to, even 1 minute after turning off to allow for cooling. Approach soldering equipment assuming that it is hot. When not in use, soldering irons must be stored in stands provided.

F - Continuing Professional Development (CPD)



Continuing Professional Development Log

Name: Chun Hei Wong

Current and recent CPD activity:

CPD Activity Title	Description	Dates	CPD Hours
Give the CPD activity a title…	Describe the activity in brief…	When did the activity happen…	How many hours of learning do you think you gained for the activity…
Introduction to Deep RL	Watched lecture by Alexander Amini (MIT course on Deep Learning) on Deep Reinforcement Learning.		
Read seminal publications on RL	Read papers such as "Playing Atari with Deep Reinforcement Learning" by Mnih et al. or books such as "Reinforcement Learning: An Introduction" by Barto and Sutton.		
Read about fundamentals of particle filtering	Read "Beyond the Kalman Filter: Particle Filters for Tracking Applications" by Ristic et al.		

Planned and Future CPD activity:

CPD Activity Title	Description	Skills addressed	Dates
Name of the CPD activity…	Describe the activity in brief…	What skills will the CPD activity enhance, or what skills-gap you have identified will it address…	When will this happen…