

## Article

# Generalising Rescue Operations in Disaster Scenarios Using Drones: A Lifelong Reinforcement Learning Approach

Jiangshan Xu, Dimitris Panagopoulos , Adolfo Perrusquía \* , Weisi Guo  and Antonios Tsourdos 

Faculty of Engineering and Applied Sciences, Cranfield University, College Road, Bedford MK43 0AL, UK; jiangshan.xu.290@cranfield.ac.uk (J.X.); d.panagopoulos@cranfield.ac.uk (D.P.); weisi.guo@cranfield.ac.uk (W.G.); antonios.tsourdos@cranfield.ac.uk (A.T.)

\* Correspondence: adolfo.perrusquia-guzman@cranfield.ac.uk

**Abstract:** Search and rescue (SAR) operations in post-earthquake environments are hindered by unseen environment conditions and uncertain victim locations. While reinforcement learning (RL) has been used to enhance unmanned aerial vehicle (UAV) navigation in such scenarios, its limited generalisation to novel environments, such as post-disaster environments, remains a challenge. To deal with this issue, this paper proposes an RL-based framework that combines the principles of lifelong learning and eligibility traces. Here, the approach uses a shaping reward heuristic based on pre-training experiences obtained from similar environments to improve generalisation, and simultaneously, eligibility traces are used to accelerate convergence of the overall approach. The combined contributions allows the RL algorithm to adapt to new environments, whilst ensuring fast convergence, critical for rescue missions. Extensive simulation studies show that the proposed framework can improve the average reward return by 46% compared to baseline RL algorithms. Ablation studies are also conducted, which demonstrate a 23% improvement in the overall reward score in environments with different complexities and a 56% improvement in scenarios with varying numbers of trapped individuals.

**Keywords:** lifelong reinforcement learning; shaping reward heuristic; eligibility traces; State-Action-Reward-State-Action (Sarsa); rescue environments; drones



Academic Editor: Yu-Jun Zheng

Received: 28 April 2025

Revised: 25 May 2025

Accepted: 1 June 2025

Published: 3 June 2025

**Citation:** Xu, J.; Panagopoulos, D.; Perrusquía, A.; Guo, W.; Tsourdos, A. Generalising Rescue Operations in Disaster Scenarios Using Drones: A Lifelong Reinforcement Learning Approach. *Drones* **2025**, *9*, 409. <https://doi.org/10.3390/drones9060409>

**Copyright:** © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Earthquake events have caused significant loss across the world, both for people's lives and infrastructure. These natural disasters, depending of their magnitude, can lead to the collapse of buildings, trapping victims under debris, and modifying a city's ecosystem. Immediate intervention is required to maximise the survivability of victims, as well as anticipate any potential earthquake replica. According to the CATDAT Damaging Earthquakes Database [1], the damage caused by high magnitude earthquakes is severe, such as in Haiti in 2010, where the median cost was around USD 7.8 billion, with an approximate fatality median of 137,000 people. These statistics underscore the need for the development of effective and timely search and rescue (SAR) operations that maximise the likelihood of survivability of victims after an earthquake event.

The incorporation of small autonomous vehicles has been transformative in SAR operations. For example, autonomous vehicles have been used to enter ruins with complex terrain for food delivery and to supply aid or identify targets in post-disaster event zones [2,3]. However, search and rescue missions in post-earthquake environments pose high uncertainty, which complicates the path-finding process to locate trapped victims. Traditional methods [4,5] have typically required prior knowledge of both the environment

and the patient's location [6], which limits their applicability in real-world implementations where such information might be unavailable.

### 1.1. Related Work

In earthquake-induced SAR operations, UAVs are particularly effective in identifying and locating trapped individuals in areas that are otherwise inaccessible to ground teams [7]. For instance, a cost-effective UAV system was developed to generate 3D point clouds from digital imagery, aiding in the detailed mapping of disaster sites [8]. This technology is essential in earthquake settings, where terrain is significantly altered, and immediate identification of potential rescue zones is crucial. UAVs have been applied in other disaster settings, such as floods and tsunamis [9,10], where the versatility of UAVs in SAR missions was demonstrated.

These aforementioned techniques can be directly transferred into earthquake scenarios, where accurate mapping and modelling of the environment are vital for rescue teams to operate effectively. Furthermore, UAVs have been utilised for simulating and predicting the impact of future disasters [11]. These approaches can also be adapted to earthquake settings, enhancing both pre-disaster preparedness and post-disaster response, thereby improving the overall effectiveness of SAR missions.

#### 1.1.1. Path Planning Algorithms

Path planning plays a crucial role in SAR operations, especially in the unpredictable and often hazardous environments caused by earthquakes. Traditional methods, such as Dijkstra's algorithm, have been widely used due to their ability to find the shortest path in a graph, which is an essential requirement for navigating disaster-stricken areas. However, Dijkstra's algorithm is optimised for static environments and struggles to accommodate real-time updates in dynamic settings [12].

To address the limitations of static algorithms,  $A^*$  and its variants have been adopted in SAR operations.  $A^*$  is particularly effective due to its heuristic-based approach, which reduces computational complexity while maintaining efficient path-finding [13]. Notable applications of  $A^*$  include its use in mapping and navigating urban disaster sites where both precision and computational efficiency are crucial. The hierarchical  $A^*$  and Hybrid  $A^*$  algorithms further enhance these capabilities by optimising the search process in complex environments, making them suitable for navigating debris and obstacles typically encountered after an earthquake [14,15].

In dynamic and partially known environments, where landscapes may shift rapidly due to aftershocks or ongoing structural collapses,  $D^*$  and its variants, such as  $D^*$  Lite and Field  $D^*$ , offer more adaptive approaches [16,17]. These algorithms are designed to update paths in real time as new information about the environment is gathered, making them invaluable for SAR missions where conditions evolve rapidly. Field  $D^*$  uses linear interpolation to smooth paths across uneven terrains, improving the mobility of rescue robots in challenging landscapes [17].

Probabilistic approaches such as Rapidly-exploring Random Trees (RRTs) and their optimised versions, like RRT\*, are increasingly being applied in SAR missions. RRTs are particularly useful for exploring large, complex environments where traditional grid-based methods may be insufficient [18]. RRT\*, with its capability to find optimal paths in continuous spaces, has been demonstrated in scenarios requiring real-time navigation through unpredictable terrains, such as those encountered in earthquake-induced ruins [19].

#### 1.1.2. Off-Line Reinforcement Learning (RL)

Off-line reinforcement learning (RL) has gained significant attention for its applicability in scenarios where agents must learn solely from fixed datasets, without access to on-line

exploration. A central challenge in off-line RL is the distributional shift between the behaviour policy used to generate the dataset and the learned policy. This shift can lead to bootstrapping errors, where value estimates for out-of-distribution (OOD) states and actions cause instability during training [20,21]. These errors are particularly detrimental in dynamic environments, such as SAR operations, where the state distribution may change unpredictably.

To mitigate these challenges, various strategies have been proposed. Conservative approaches, such as value regularisation, have been used to limit the Q-value overestimation [22], thereby reducing the likelihood of selecting OOD actions. Model-based approaches aim to estimate environmental uncertainty and incorporate it into policy updates, encouraging more cautious exploration [23]. However, the difficulty of training accurate models in highly dynamic settings remains a significant limitation for these methods.

Another prominent strategy involves constraining the learned policy to stay close to the behaviour policy that generated the dataset [21,24]. This conservative approach reduces the risk of encountering OOD states but may restrict the agent's ability to generalise to novel tasks, particularly in environments that demand exploration beyond the data available. In the context of SAR operations, these off-line RL strategies offer valuable insights but also expose the limitations of traditional methods when dealing with complex and rapidly changing environments.

#### 1.1.3. Meta and Lifelong Reinforcement Learning

Meta-Reinforcement Learning (Meta-RL) [25,26] has garnered significant attention as an approach to improve the adaptability and efficiency of RL agents across various tasks [27,28]. Unlike traditional RL, where agents are trained from scratch for each new environment, Meta-RL algorithms aim to pre-train agents on a distribution of environments, enabling rapid adaptation to new environments with minimal additional training. This framework is particularly advantageous in environments with varying dynamics, as it mitigates the inefficiencies associated with training agents from scratch in each scenario.

Several influential works in Meta-RL have introduced methods that facilitate fast adaptation by leveraging prior experience. Notably, Model-Agnostic Meta-Learning (MAML) [25] and RL<sup>2</sup> [29] are foundational approaches that demonstrate the value of meta-training on a set of tasks, allowing for efficient fine-tuning in novel environments. These methods have proven effective in fields where adaptability is crucial, such as robotics and autonomous navigation.

Lifelong reinforcement learning (LLRL) is an emerging paradigm in which agents continuously learn and adapt over extended periods and across multiple tasks, aiming to accumulate knowledge and improve learning efficiency over time. Unlike traditional reinforcement learning (RL) methods that focus on single-task settings, LLRL emphasises knowledge retention, transfer, and avoidance of catastrophic forgetting, a major challenge in continual learning scenarios [30]. Recent approaches combine memory systems, meta-learning, and curriculum design to enable agents to generalise effectively across task distributions [31,32]. LLRL is particularly relevant for autonomous systems operating in open-world environments, where the ability to adapt to new situations while retaining prior knowledge is critical for long-term autonomy and safety [33]. Advancements in LLRL contribute to building more robust, adaptive, and safe AI systems, especially in domains like robotics, drones, and human-AI interaction.

#### 1.1.4. Fine-Tuning in Reinforcement Learning

Fine-tuning has emerged as a pivotal approach in RL, particularly in scenarios where pre-training on off-line datasets is followed by limited on-line interaction. This method

plays a crucial role in improving sample efficiency and performance in tasks where data collection is costly or risky. Recent research has illuminated both the potential benefits and challenges of fine-tuning policies learned from off-line RL.

One of the key challenges is that policies initialised via off-line RL often exhibit sub-optimal performance during the initial stages of fine-tuning. Studies have shown that this phenomenon, where policies degrade before improving, occurs across various algorithms [34,35]. To address this issue, several strategies have been proposed. Conservative Q-Learning (CQL) [22] introduces a pessimistic bias to Q-values to prevent overestimation during fine-tuning. Cal-QL [36] enhances this approach by introducing a calibrated value function that balances conservatism and optimism. This calibration ensures that the learned value functions align closely with true returns, mitigating the initial dip in performance seen in other fine-tuning methods. Empirical results from Cal-QL demonstrate its effectiveness in retaining the benefits of off-line pre-training while accelerating and improving the fine-tuning process in on-line interactions.

Table 1 summarises the different methodologies used in SAR operations with their respective limitations.

**Table 1.** Limitations of UAV-Based SAR technologies and algorithms.

| Category                                   | Limitations                                                                                                                                                                                                                                                                        |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Traditional Path Planning (e.g., Dijkstra) | <ul style="list-style-type: none"> <li>- Assumes static environment, not suited for dynamic or evolving disaster scenarios.</li> <li>- High computational cost in large maps.</li> <li>- Inflexible to real-time updates and changes. [12].</li> </ul>                             |
| $A^*$ and Variants                         | <ul style="list-style-type: none"> <li>- Performance degrades in highly cluttered or rapidly changing environments.</li> <li>- Heuristic tuning required for optimality.</li> <li>- Assumes full or near-complete map knowledge.</li> </ul>                                        |
| $D^*$ , $D^*$ Lite, Field $D^*$            | <ul style="list-style-type: none"> <li>- Computational overhead due to real-time replanning.</li> <li>- Linear interpolation (Field <math>D^*</math>) may be less effective in highly irregular terrain.</li> <li>- Limited by the accuracy of sensor data for updates.</li> </ul> |
| RRT/RRT*                                   | <ul style="list-style-type: none"> <li>- Random sampling may be inefficient in highly constrained spaces.</li> <li>- Generated paths may require post-processing to ensure smoothness.</li> <li>- Performance is sensitive to step size and sampling bias.</li> </ul>              |
| Off-line RL                                | <ul style="list-style-type: none"> <li>- Prone to distributional shift and bootstrapping errors [20].</li> <li>- Overly conservative policies may underperform.</li> <li>- Difficult to train robust models in highly dynamic and uncertain environments.</li> </ul>               |
| Meta-RL                                    | <ul style="list-style-type: none"> <li>- Requires access to multiple diverse training environments.</li> <li>- May not generalise well to highly novel tasks outside meta-training distribution.</li> <li>- Meta-training is computationally expensive.</li> </ul>                 |
| Lifelong RL                                | <ul style="list-style-type: none"> <li>- Susceptible to catastrophic forgetting.</li> <li>- Managing memory and balancing transfer vs. interference is difficult.</li> <li>- Lack of standardised benchmarks in open-world SAR environments.</li> </ul>                            |
| Fine-Tuning in RL                          | <ul style="list-style-type: none"> <li>- Initial performance dip after off-line pre-training [34].</li> <li>- Requires careful calibration of value functions.</li> <li>- Data scarcity in online phase limits exploration and learning.</li> </ul>                                |

## 1.2. Contributions

In view of the above, there is a flourishing community dealing with the design of technologies that support rescuers in disaster scenarios, each one with their respective advantages and disadvantages. In this paper, we aim to adopt the benefits of both LLRL with fine-tuning to improve the generalisation of RL algorithms to unseen environments in the field of SAR operations. To this end, we propose a framework that considers the

well-known on-policy Sarsa algorithm enhanced with two main components: (i) a shaping reward heuristic for lifelong learning and generalisation, and (ii) eligibility traces for fast learning. The proposed framework extends the off-line RL capabilities by following the principles of Meta-RL and fine-tuning by pre-training the agent across a diverse set of similar post-earthquake environments, for refinement of the reward function that yields improved policies with better generalisation and adaptability to new situations. This approach not only mitigates the bootstrapping errors common in off-line RL but also leverages lifelong learning to improve performance in dynamic and unpredictable environments. In addition, eligibility traces add to the algorithm a long short-term memory that accelerates learning convergence, which is crucial in SAR operations.

To rigorously evaluate the performance of the proposed algorithm, we design a 2D environment to simulate a rescue scenario. In this environment, the positions of the patients, obstacles, and the initial position of the agent are all randomly generated. These settings replicate real-world challenges, such as the uncertainty of casualty locations and the difficulty of navigating through intricate terrains. Additionally, we tested different search strategies to identify the optimal approach. Experimental results show that, compared to other lifelong RL baselines, SR-Sarsa ( $\lambda$ ) achieves a 50% improvement in performance, and when compared to conventional reinforcement learning methods, the improvement reaches up to 700%.

## 2. Preliminaries

Consider a post-disaster environment modelled as a Markov Decision Process (MDP). The MDP is described by the tuple  $(S, A, P, R, \gamma)$ , where  $S$  is a set of discrete states,  $A$  is the set of discrete actions,  $P(s'|s, a) : S \times A \times S \mapsto [0, 1]$  defines the state transition probability,  $R(s, a, s') : S \times A \times S \mapsto [0, R_{\max}]$  is a user-defined reward function with a maximum value of  $R_{\max}$ , and  $\gamma \in (0, 1]$  is the discount factor with  $s, s' \in S$  and  $a \in A$ . A policy  $\pi : S \mapsto A$  determines the actions followed by the agent given a specific state. The main goal of RL is to learn an optimal control policy  $\pi(a|s)$  that maximises the expected cumulative reward over time by starting from a state  $s_0 \in S$ . This is usually written in terms of a value function  $V^\pi(s)$  or action-value function,  $Q^\pi(s, a)$  as follows:

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_{s' \sim P_{sa}} \left[ \sum_{j=0}^{\infty} \gamma^j r_j \mid s = s_0, \pi \right], \\ Q^\pi(s, a) &= \mathbb{E}_{s' \sim P_{sa}} \left[ \sum_{j=0}^{\infty} \gamma^j r_j \mid s = s_0, a = a_0, \pi \right], \end{aligned} \quad (1)$$

where  $r_j$  is the reward obtained in the step  $j$ ,  $P_{sa}$  is the state transition probability of being in state  $s$  and apply action  $a$ , and  $a_0 \in A$ . The associated Bellman equations are given by

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_{s' \sim P_{sa}} [r(s, a, s') + \gamma V^\pi(s')] \\ Q^\pi(s, a) &= \mathbb{E}_{s' \sim P_{sa}} [r(s, a, s') + \gamma Q^\pi(s', \pi(s'))], \end{aligned} \quad (2)$$

which gives a recursive relation between both the value and action-value functions. Here, the optimal value and action-value functions are computed as

$$\begin{aligned} V^*(s) &= \mathbb{E}_{s' \sim P_{sa}} [r(s, a, s') + \gamma V^*(s')] \\ Q^*(s, a) &= \mathbb{E}_{s' \sim P_{sa}} \left[ r(s, a, s') + \gamma \max_{u \in A} Q^*(s', u) \right]. \end{aligned} \quad (3)$$

### Drone and Environment Modelling

In this implementation, we only focus on a high-level navigation algorithm and assume the stabilisation and compensation of the aerodynamic effects are dealt by an on-board low-level control. The on-board low-level controller is not discussed in this paper; however, the readers can refer to [37–39]. We model the post-earthquake environment as a 2D maze-like environment, as follows:

$$\mathcal{E} = \{(x, y) | x \in [0, W - 1], y \in [0, H - 1]\}, \quad (4)$$

where  $W$  is the width and  $H$  is the height. In this setting, each state in the environment,  $s \in S$ , can be denoted as  $s = (x, y)$ . The actual drone position is given by one grid in the environment. With this in mind, we modelled the action space  $A$  of the drone as the following spanning actions

$$A = \{\text{up, down, left, right}\}. \quad (5)$$

Each action applied by the drone in the environment corresponds to the following changes in its position

$$\begin{aligned} \text{up} : s = (x, y) &\rightarrow s' = (x, y + 1) \\ \text{down} : s = (x, y) &\rightarrow s' = (x, y - 1) \\ \text{left} : s = (x, y) &\rightarrow s' = (x - 1, y) \\ \text{right} : s = (x, y) &\rightarrow s' = (x + 1, y). \end{aligned}$$

Therefore, the state-transition model is described by

$$P(s' | s, a) = \begin{cases} s' = (x, y + 1) & \text{if } a = \text{up} \quad \text{and } (x, y + 1) \text{ is free,} \\ s' = (x, y - 1) & \text{if } a = \text{down} \quad \text{and } (x, y - 1) \text{ is free,} \\ s' = (x - 1, y) & \text{if } a = \text{left} \quad \text{and } (x - 1, y) \text{ is free,} \\ s' = (x + 1, y) & \text{if } a = \text{right} \quad \text{and } (x + 1, y) \text{ is free,} \\ s' = (x, y) & \text{otherwise} \quad (\text{e.g., obstacles or out of the grid}). \end{cases} \quad (6)$$

In the next section, we discuss the proposed methodology for rescue operations in the proposed environment and high-level drone navigation model.

### 3. Methodology

The proposed methodology is summarised in Figure 1. The approach consists of designing a control policy that allows the drone to reach people who need assistance in a rescue environment, e.g., a post-earthquake disaster scenario. Here, we cannot assume prior knowledge of the debris-covered area due to the damage caused by the earthquake, such that the environment is highly modified. In view of this, the proposed approach adopts a lifelong reinforcement concept that aims to use previous experiences obtained from pre-trained models under similar post-earthquake environments to design a shaping reward (SR) heuristic that adapts the reward to the unseen rescue environment.

The pre-training knowledge includes experiences in the form of tuples of state  $s$ , action  $a$ , reward  $r$ , and next state  $s'$  of the trajectories that achieve the best performance in a particular rescue environment. These trajectories are used to construct an SR heuristic that adapts the reward function to better generalise to unseen scenarios using any reinforcement learning method. In this paper, we will focus on tabular reinforcement learning methods.

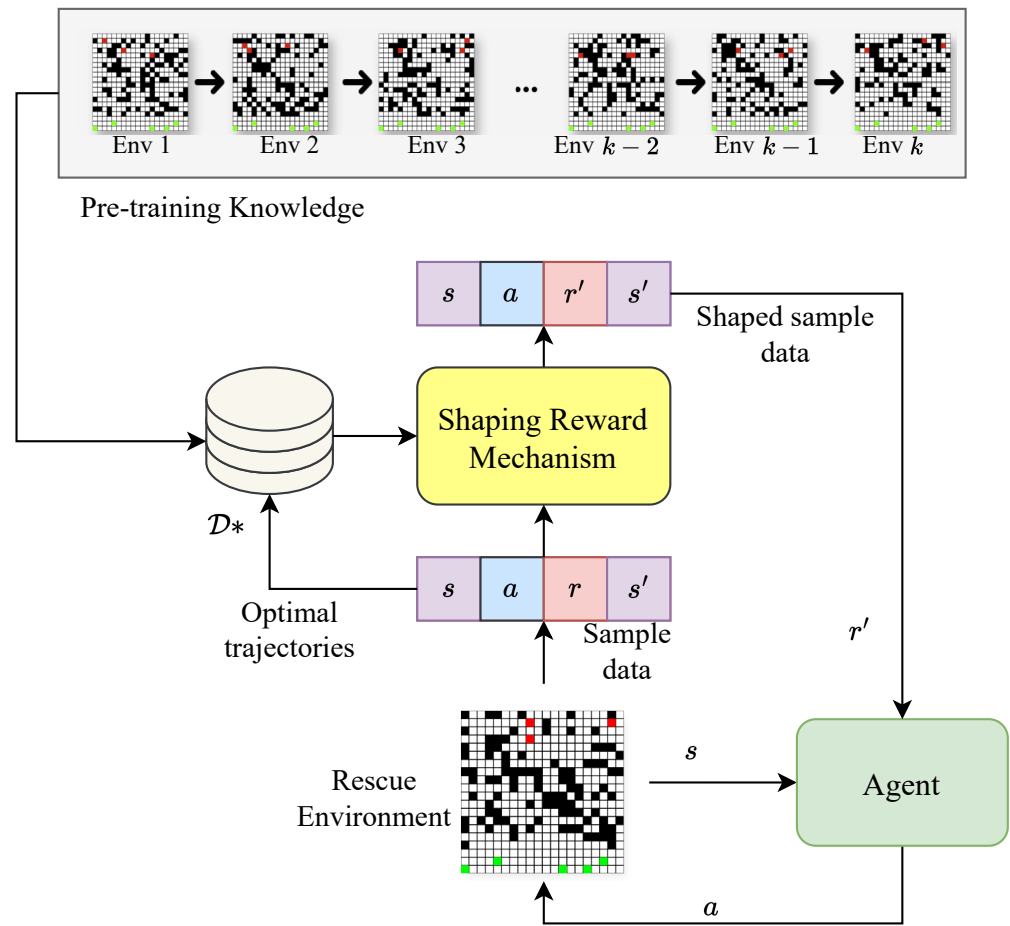


**Figure 1.** High-level diagram of the proposed approach.

### 3.1. Shaping Reward (SR) Heuristic

The shaping reward (SR) heuristic consists of collecting the expert trajectories that give the best performance in a given rescue environment  $k$  as shown in Figure 2. This process is denoted as a pre-training stage.

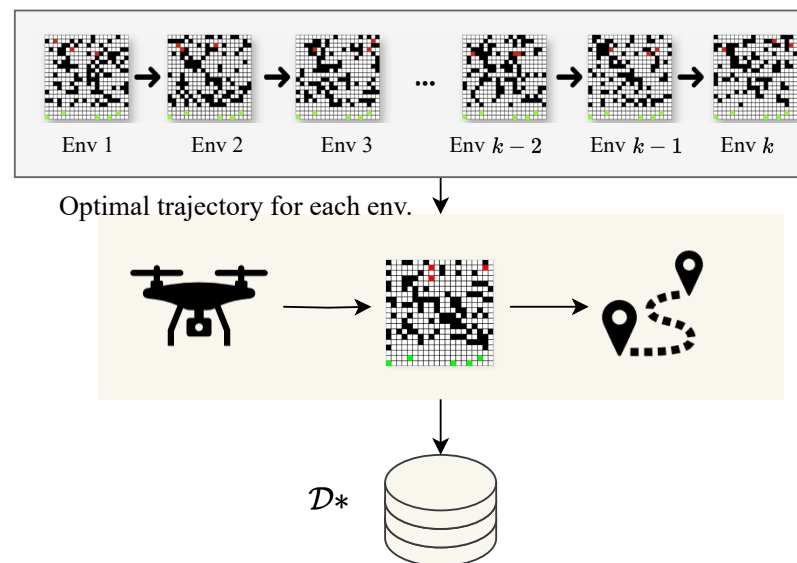


**Figure 2.** Pre-training expert data.

In the pre-training stage, the optimal trajectories for each environment are stored and used to design an SR heuristic for future environments that guides the drone towards more efficient and effective decision-making.

To explain this process, we denote each trajectory data for an environment  $k$  as  $T_k$  and it is represented by two components: (i) trajectory data  $\tau_k^i$  composed by a the sequence of states and actions that the drone receives in each episode  $i$  with  $i = 1, \dots, m$ ; (ii) the cumulative reward or return  $G_k^i$  of the trajectory  $\tau_k^i$  given by  $G_k^i = \sum_{j \geq 0} r_j$ .

Here, we store only the trajectories of each environment that have the highest return since they provide more information on how the drone needs to behave in specific scenarios. So, for each environment  $k$ , we store the optimal trajectories and return denoted as  $\mathcal{D}^* = \{\tau_k^*, G_k^*\}$ . Let us define  $D(s)$  as the number of times that the state  $s$  appears in the dataset, and  $D(s, a, s')$  as the number of times a state transition  $(s, a, s')$  occurs in the optimal trajectory dataset  $\mathcal{D}^*$ .

Then, the SR heuristic is designed as

$$H(s, a, s') = (1 - \gamma)V_{\max} \left( \frac{D(s, a, s')}{D(s)} \right), \quad (7)$$

where  $V_{\max} = \frac{R_{\max}}{1-\gamma}$  is a known upper bound of the value function in terms of the maximum value  $R_{\max}$  of the immediate reward  $r_j$  in each step  $j$  of episode  $i$ . The heuristic  $H(s, a, s')$  is a non-potential based shaping function such that there exists a mapping  $\Phi : S \times A \mapsto \mathbb{R}$ , such that for all  $s, s' \in S$  and  $a \in A$  the following is satisfied

$$H(s, a, s') = \gamma\Phi(s', a) - \Phi(s, a). \quad (8)$$

The incorporation of the SR heuristic can improve optimality of the Q-function. Here, tabular methods use either an off-policy or on-policy target which are, respectively, given by

$$\begin{aligned} Q^*(s, a) &= r(s, a, s') + \gamma Q^*(s', a') \\ Q^*(s, a) &= r(s, a, s') + \gamma \max_u Q^*(s', u). \end{aligned}$$

Consider the on-policy update rule which will be used for the Sarsa algorithm discussed in the next section. If we add the SR heuristic into the on-policy target and apply some simple algebraic manipulations, we give

$$\begin{aligned} Q^*(s, a) - \Phi(s, a) &= r(s, a, s') + \gamma\Phi(s', a) - \Phi(s, a) + \gamma(Q(s', a') - \Phi(s', a)) \\ Q_{\Phi}^*(s, a) &= r_{\text{new}}(s, a, s') + \gamma[Q^*(s', a') - \Phi(s', a)], \end{aligned}$$

where  $Q_{\Phi}^*(s, a) = Q^*(s, a) - \Phi(s, a)$  is the optimal Q-function moved by  $\Phi(s, a)$  and  $r_{\text{new}}(s, a, s') = r(s, a, s') + H(s, a, s')$  is a new reward function. The term  $\Phi(s', a)$  adds a bias term into the update rule that adapts the Q-function based on the pre-training experiences. Notice that a greedy policy satisfies

$$\begin{aligned} a &= \arg \max_{u \in A} Q_{\Phi}^*(s, u) \\ &= \arg \max_{u \in A} [Q^*(s, u) - \Phi(s, u)] \end{aligned} \quad (9)$$

The above means that incorporating the SR heuristic modifies the optimality of the policy and it improves the Q-function to have better generalisation capabilities based on previous experiences collected in similar rescue environments. For the Q-learning target the same process can be applied to yield the same conclusion.

### 3.2. Reinforcement Learning Algorithm

In a rescue scenario, there is a need to reach the target goals as soon as possible such that the drone has to avoid spending too much time in exploring the environment. This means that the drone needs to be capable of recognising previously visited states and which actions were previously selected to accelerate the learning process and avoid long exploration phase. To this end, we will use a long short-term memory known as eligibility traces to enhance the capabilities of tabular methods.

Eligibility traces provide a unique skill to the RL algorithm, considering previously visited states and actions to accelerate learning and exploit more the acquired knowledge [40]. Two different algorithms based on eligibility traces are recognised:  $Q(\lambda)$  and Sarsa ( $\lambda$ ), which are improved versions of both Q-learning and Sarsa algorithms [41] using eligibility traces, respectively. Here, we decide to focus mainly on Sarsa( $\lambda$ ) due to the nature of the rescue problem, which requires on-policy learning rather than off-policy for the proper action selection.

#### 3.2.1. Sarsa ( $\lambda$ ) Algorithm

Sarsa ( $\lambda$ ) updates a Q-table based on an on-policy mechanism and the current temporal difference (TD) error as follows:

$$\begin{aligned} Q(s, a) &\leftarrow Q(s, a) + \alpha \underbrace{(r(s, a, s') + \gamma Q(s', a') - Q(s, a))}_{\text{TD-error}} e(s, a) \\ e(s, a) &\leftarrow \gamma \lambda e(s, a) + 1_{(s_j=s, a_j=a)}, \end{aligned} \quad (10)$$

where  $a'$  is the next action obtained by the selected policy or a GLIE (greedy in the limit with infinite exploration) policy,  $\lambda \in [0, 1]$  is the trace decay factor, and  $e(s, a)$  denotes a specific kind of trace known as cumulative trace since it increases by 1 each time the drone is both at state  $s_j = s$  with action  $a_j = a$ , otherwise it decays the trace by a factor of  $\gamma \lambda$ .

Figure 3 shows the magnitude of the TD-error at each state-action pair for a given path for the following tabular RL models (Sarsa, Dyna Q, and Sarsa ( $\lambda$ )).

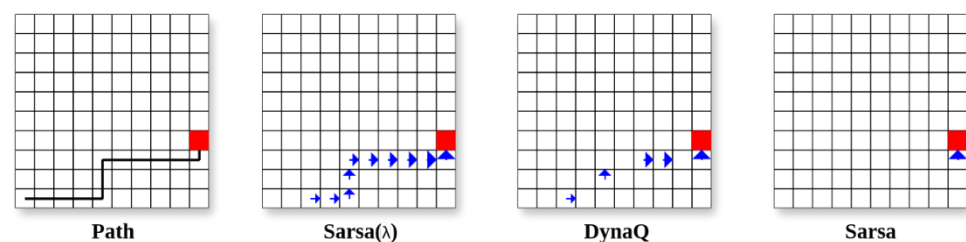


**Figure 3.** Contribution of the TD error for different RL algorithms.

The size of the arrows in the grids in Figure 3 represents the magnitude of the Q-function's update error at each state-action pair. In all grids, the arrows indicate larger update errors along the path, reflecting substantial learning as the RL algorithm propagates the TD error throughout the trajectory. In the last three grids, representing different algorithms or stages of learning, the arrow sizes decrease, suggesting smaller updates as the respective Q-values approach convergence. This visual comparison highlights how Sarsa ( $\lambda$ ) efficiently spreads the learning signal across the state space, leading to faster and more effective convergence compared to traditional tabular RL algorithms. This clearly illustrates that the Sarsa ( $\lambda$ ) algorithm results in more comprehensive updates, particularly in early learning stages, ultimately enhancing the drone's ability to generalise across different states.

### 3.2.2. Sarsa ( $\lambda$ ) with SR Heuristic: SR Sarsa ( $\lambda$ )

In this work, we enhance the generalisation capabilities of Sarsa ( $\lambda$ ) by introducing the SR heuristic into its design. This is conducted by simply injecting the new reward into the Sarsa ( $\lambda$ ) update rule as

$$\begin{aligned} Q(s, a) &\leftarrow Q(s, a) + \alpha(r_{new}(s, a, s') + \gamma Q(s', a') - Q(s, a))e(s, a) \\ e(s, a) &\leftarrow \gamma \lambda e(s, a) + 1_{(s_j=s, a_j=a)}, \end{aligned} \quad (11)$$

By leveraging eligibility traces and the SR heuristic, the drone's generalisation capabilities are enhanced. In sparse environments such as rescue environments, when the drone samples terminal state positions, the Sarsa ( $\lambda$ ) algorithm propagates the one-step TD error across all  $(s, a)$  pairs based on the trace  $e(s, a)$ . Compared to other tabular RL algorithms, eligibility traces accelerate the convergence of the Q-function, whilst the SR heuristic introduces generalisation capabilities to the RL algorithm for similar and unseen environments.

The combined Sarsa ( $\lambda$ ) with SR heuristic is denoted as SR Sarsa ( $\lambda$ ), whose pseudo-code is summarised in Algorithm 1.

---

#### Algorithm 1: SR Sarsa ( $\lambda$ ) algorithm

---

**Require:** Optimal set of trajectories  $\mathcal{D}^*$ , initial Q-function and eligibility trace  $Q(s, a) = 0$  and  $e(s, a) = 0$  for all  $s, a$ , learning rate  $\alpha$ , discount factor  $\gamma$ , eligibility trace decay rate  $\lambda$

- 1: **for** each episode  $i$  **do**
- 2:   Initialise the state  $s$  and reset  $e(s, a)$  for all  $s, a$
- 3:   **for** each step  $j$  **do**
- 4:     Select an action  $a$  using a greedy strategy, e.g.,  $\epsilon$ -greedy.
- 5:     Apply  $a$  and observe next state  $s'$  and reward  $r(s, a, s')$ .
- 6:     Compute the SR heuristic  $H(s, a, s')$  using (7).
- 7:     Compute improved reward  $r_{new}(s, a, s') = r(s, a, s') + H(s, a, s')$
- 8:     Select the next action  $a'$  using a greedy strategy, e.g.,  $\epsilon$ -greedy.
- 9:     Update  $Q(s, a)$  and  $e(s, a)$  using the Sarsa ( $\lambda$ ) update rule (11).
- 10:    Set  $s \leftarrow s'$ , and  $a \leftarrow a'$ .
- 11:   **end for**
- 12: **end for**

---

## 4. Simulation Studies

In this section, we test and compare the proposed SR Sarsa ( $\lambda$ ) algorithm in a post-disaster earthquake scenario. As previously discussed, we focus on high-level drone navigation in a 2D maze-like environment, whilst the low-level controller is assumed to effectively drive the drone into the desired locations. In this environment, the positions of the patients, the obstacles, and the initial position of the drone are randomly generated. These settings mimic the real-world challenges of unknown casualty locations and navigating through intricate terrain. Three baseline algorithms enhanced with the SR heuristic are considered: SR Multi-step (MS) Sarsa, SR Dyna Q, and the state-of-the-art algorithm SR-LLRL based on the Q-learning algorithm [42].

The rescue environment consists of a  $20 \times 20$  grid, which corresponds to 400 cells. We add 100 random obstacles to the environment. The grid has the following color specifications:

- Green grids: Consists of five grids where the drone may start. This emulates the fixed starting location of rescuers.
- Black grids: These grids represent the 100 obstacles generated in the environment.

- Red grids: These represent randomly generated patients, who are typically located far from the drone's initial position, making them difficult to find.

We initiate drone operations from a safe standoff distance, allowing the platform to hover and observe the environment before entering complex or hazardous areas. The variability in each task comes from the different positions of obstacles, patients, and the drone's starting point. After completing a task, the drone faces a new task and uses its prior knowledge to explore the new task and locate patients more quickly. Figure 4 shows a rescue environment sample.

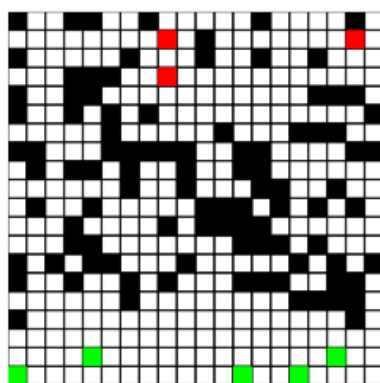


**Figure 4.** Rescue environment sample.

A sparse reward function is designed manually and implemented in all the proposed models to ensure fair comparison. The reward is designed as follows: +2 for reaching each patient, and 0 for each step taken. For each environment configuration, the drone has 200 episodes and 200 steps to interact with it. After completing 200 steps, the drone is reset to the start state, and it starts exploring the task again. Each algorithm uses the following hyperparameter values: learning rate  $\alpha = 0.05$ , exploration rate  $\epsilon = 0.1$ , discount factor  $\gamma = 0.9$ , MS Sarsa steps steps = 5, and trace decay rate  $\lambda = 0.9$ .

We evaluate the performance of each model based on two metrics: (1) the performance across episodes: each environment training consists of 200 episode, such that the score of the  $i$ -th episode is the average score across all the environments in the  $i$ th episode; (2) the performance across environments: each environment consists of 200 episodes, so the average score for the  $i$ th environment is the average across all 200 episodes for that specific environment. The results are shown in Figure 5.

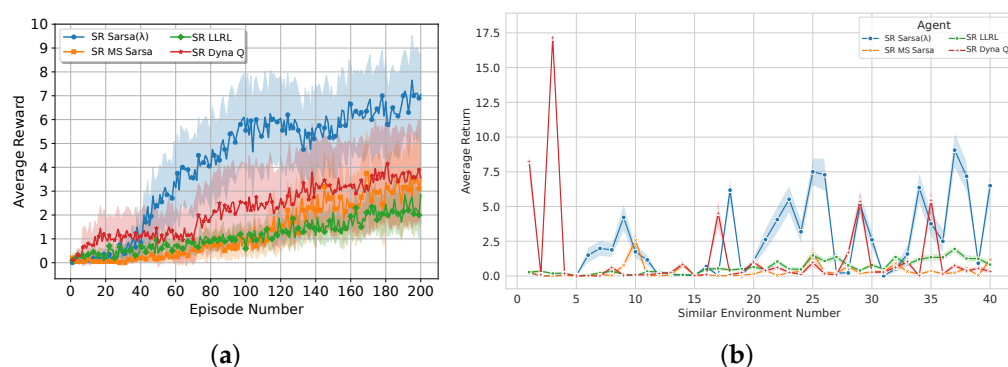


**Figure 5.** Training results of the proposed SR RL algorithms. (a) Average reward for each episode. (b) Average reward for each environment.

The results show that SR Sarsa ( $\lambda$ ) outperforms the other algorithms due to its ability to update all state-action pairs efficiently through the use of eligibility traces. This comprehensive updating mechanism ensures that the drone quickly propagates reward

information across the entire state space, leading to faster convergence and improved control policies. The average reward is almost twice as great as the second-best algorithm, SR Dyna Q. This highlights the robustness and effectiveness of SR Sarsa ( $\lambda$ ) in more complex scenarios, where its ability to efficiently propagate rewards across the state space becomes most beneficial.

SR Dyna Q is the second-best performer algorithm, primarily due to its model-based nature that allows for frequent updates through simulated experiences. This frequent updating enables SR Dyna Q to converge relatively quickly, although not as rapidly as SR Sarsa ( $\lambda$ ). SR MS Sarsa performs reasonably well due to its ability to incorporate more reward terms in the update rule, but it falls short compared with SR Dyna Q and SR Sarsa ( $\lambda$ ) due to its lack of memory.

Finally, SR LLRL performs the worst due to the one-step update of the Q-learning algorithm. This hinders the drone's ability to adapt quickly, resulting in suboptimal policies, particularly in complex environments.

#### 4.1. Impact of the Shaping Reward Heuristic

We compare the proposed SR Sarsa( $\lambda$ ) against non-lifelong reinforcement learning models to show how the SR heuristic enhances the overall performance of RL models. We compare the proposed approach against traditional tabular methods such as Dyna Q, Sarsa ( $\lambda$ ), and MS Sarsa. Figure 6 shows the comparison results.

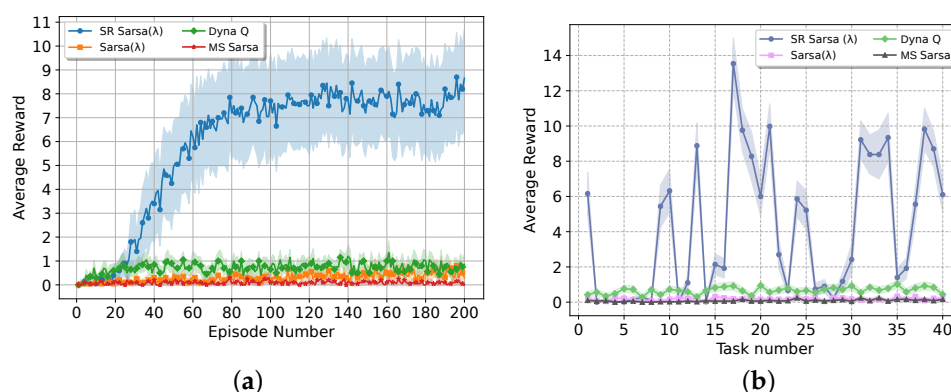


**Figure 6.** Impact of the SR heuristic compared with non-lifelong reinforcement learning models. (a) Average reward for each episode. (b) Average reward for each environment.

The results highlight that none of the traditional tabular methods managed to reach a reward value of 1 across all episodes, whilst the SR Sarsa ( $\lambda$ ) outperform them with a reward of approximately 8.6. This shows the benefits of the proposed SR heuristic in the overall performance improvement of traditional RL methods. On the other hand, we can observe that the performance of the Sarsa ( $\lambda$ ) improves as the number of training environments increases, whilst the traditional models do not show a significant improvement. This difference highlights the critical impact of the pre-training phase in the SR Sarsa ( $\lambda$ ) algorithm, where the drone's prior knowledge and experience allow for more efficient learning and adaptation to new tasks, leading to superior performance.

#### 4.2. Ablation Studies

We perform ablation studies to verify the robustness of the proposed approach. To this end, we consider the robustness of the approach in the following scenarios: (i) different eligibility traces mechanism, i.e., either cumulative or replacing traces; (ii) changes in the number of obstacles (environment complexity); (iii) changes in the number of rescued individuals; and (iv) the pre-training strategy. Each ablation study is discussed as follows:

- (i) Eligibility traces update mechanism: We consider both cumulative and replacing traces to observe how the RL algorithms behave in a lifelong rescue scenario.
- (ii) Environment complexity: We consider five scenarios based on the number of obstacles. We consider 100, 80, 60, 40, and 20 obstacles, respectively. This study allows us to better understand how the complexity of the environment affects the overall performance of the approach.
- (iii) Number of rescued individuals: We vary the number of trapped individuals to assess the generalisation capabilities of the approach for multi-target problems.
- (iv) Pre-training strategy: We explore two different discovery strategies to identify the most efficient search strategy in a rescue scenario. We consider the following strategies:
  - Breadth-first exploration: Where the drone has a small fixed number of steps to explore. If it is not successful, another drone begins the search from the initial position.
  - Depth-first exploration: Where the drone has a larger number of steps to explore the environment without a reset.

#### 4.2.1. Cumulative Traces vs. Replacing Traces

We first analyse the update mechanism of the eligibility traces of the SR Sarsa ( $\lambda$ ). We consider two different updates: cumulative traces and replacing traces.

On the one hand, cumulative traces has the following update mechanism

$$e(s, a) \leftarrow \gamma \lambda e(s, a) + 1. \quad (12)$$

Here, every time a state-action pair  $(s, a)$  is visited, then the corresponding trace is incremented by 1, and the trace associated with non-visited states is decayed with a factor of  $\gamma \lambda$ . This method allows for the accumulation of traces over multiple visits, meaning that the more frequently a state-action pair is visited, the larger its trace becomes. This cumulative effect can lead to a stronger emphasis on frequently visited state-action pairs during learning.

On the other hand, replacing trace follows the following update rule:

$$e(s, a) \leftarrow 1. \quad (13)$$

In this case, whenever a state-action pair  $(s, a)$  is visited, the trace is reset to 1, regardless of its previous state. This replacing method effectively limits the trace to a maximum value of 1, ensuring that recent visits have a stronger impact on learning, while older traces decay as usual according to  $\gamma \lambda$ .

The comparison experiments are designed to assess the convergence speed, stability of learning, and the overall performance of the drone. Key metrics included the number of episodes required to converge to an optimal control policy and the quality of the policies learned in terms of success rate.

Figure 7 shows the results of both updating mechanisms. The cumulative operator proved to be more effective in scenarios requiring long-term learning and consistent performance. It demonstrated the ability to recover from fluctuations and continued to improve over time, ultimately achieving a higher final reward. This makes the cumulative operator particularly well-suited for stable environments where sustained performance is crucial.

On the other hand, the replacing operator exhibits instability after a certain number of episodes. The reward drops, and it does not recover, which indicates that this update rule struggles to maintain consistent performance in the long term. However, its ability to quickly adapt to changes suggests that replacing traces could be more advantageous

in dynamic environments where rapid responses are essential, even if this comes at the expense of stability.

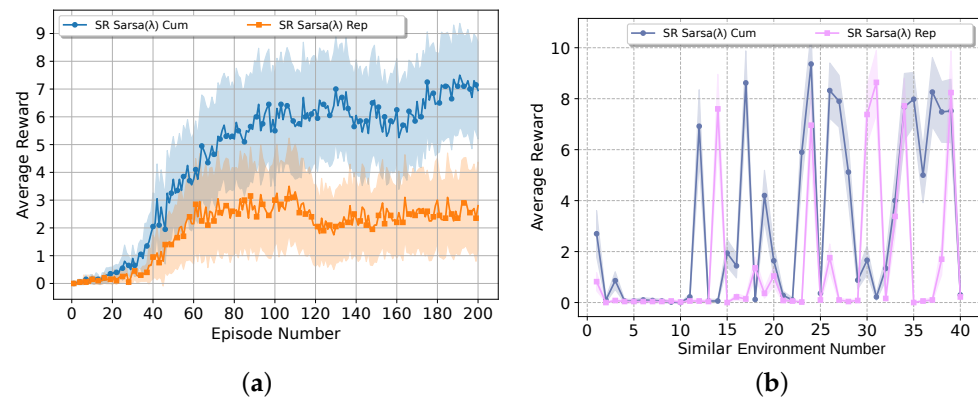


**Figure 7.** Average reward for cumulative and replacing traces operator. (a) Average return per episode. (b) Average return per environment.

The experimental findings suggest that while both traces have their strengths, the cumulative trace is generally preferable for tasks that demand robustness and consistent improvement. In contrast, the replacing traces might be better suited for scenarios where quick adaptation is prioritised.

#### 4.2.2. Impact of the Obstacle Complexity

Figure 8 and Table 2 show the ablation studies results by varying the number of obstacles in the rescue environment.

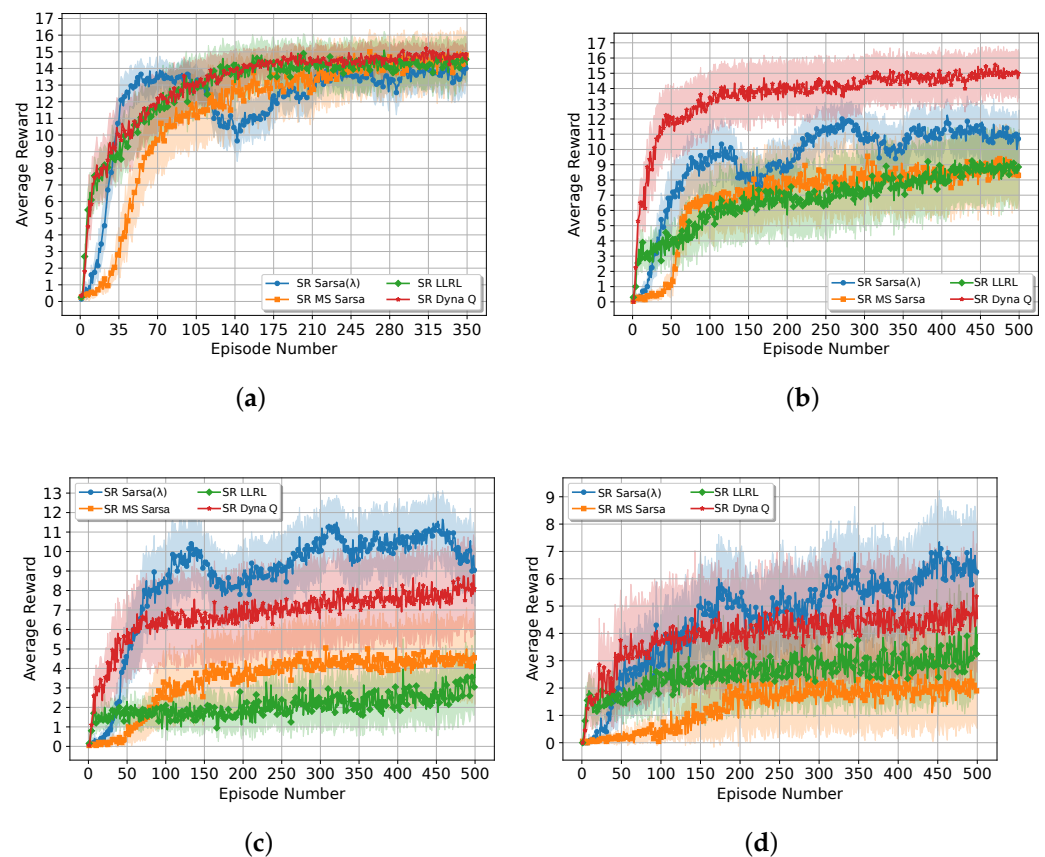


**Figure 8.** Average reward of lifelong RL algorithms in presence of obstacle variations: (a) 20 obstacles; (b) 40 obstacles; (c) 60 obstacles; (d) 80 obstacles.

The results highlight some interesting performance trends. On the one hand, the SR Sarsa ( $\lambda$ ) shows the best performance in the presence of environments with high complexity by leveraging the SR heuristic and the long short-term memory provided by the traces for efficient navigation. However, the efficiency of the SR Sarsa ( $\lambda$ ) slightly decreases for environments with low complexity. Here, the traces (depending on whether they are cumulative or replacing, which are discussed in future subsections) need to be explored more to cover all the state's possibilities.

**Table 2.** Ablation studies with different number of obstacles.

| Number of Obstacles | SR Sarsa ( $\lambda$ ) | SR Dyna Q | SR MS Sarsa | SR LLRL |
|---------------------|------------------------|-----------|-------------|---------|
| 100                 | 7.56                   | 3.24      | 1.32        | 1.39    |
| 80                  | 6.28                   | 5.47      | 2.98        | 3.31    |
| 60                  | 9.12                   | 8.12      | 4.62        | 3.13    |
| 40                  | 10.87                  | 15.03     | 8.36        | 8.91    |
| 20                  | 13.12                  | 16.11     | 15.94       | 13.98   |

Conversely, the SR Dyna Q and SR LLRL show better results in environments with low complexity. Here, the off-policy nature of these algorithms helps to accelerate learning compared with the SR MS Sarsa. However, as the complexity of the environment increases, the drones experience a performance degradation, whilst the SR Sarsa ( $\lambda$ ) is consistent and improves its performance.

#### 4.2.3. Impact of Trapped People Count

We assess the robustness of the proposed lifelong RL algorithms based on the number of trapped individuals or target goals, which play a critical factor that influences the overall performance of RL algorithm in rescue operations.

For this particular experiment, we consider a maximum number of three trapped individuals. The key goal of this study is to show the robustness of the algorithms to changes in the number of target goals in terms of efficiency (time) and success rate. Figure 9 and Table 3 summarise the obtained results.

The results in Figure 9 show different performance patterns across all algorithms as the number of trapped individuals increases. We can observe that the SR Sarsa ( $\lambda$ ) becomes more efficient as the complexity increases due to its SR heuristic.

**Table 3.** Average reward with different number of trapped people.

| Trapped Count | SR Sarsa ( $\lambda$ ) | SR Dyna Q | SR MS Sarsa | SR LLRL |
|---------------|------------------------|-----------|-------------|---------|
| 1             | 4.52                   | 2.31      | 2.17        | 2.63    |
| 2             | 9.54                   | 2.48      | 4.37        | 7.09    |
| 3             | 7.62                   | 5.62      | 3.98        | 4.62    |

Notice that SR MS Sarsa and SR Dyna Q show an approximate linear increase in search time as the number of trapped people increases. SR Dyna Q struggles with the two-trapped-individuals case, where its performance drops before recovering slightly at the three-trapped-individuals case. SR Sarsa ( $\lambda$ ) demonstrates consistent performance across different trapped counts, showing the highest average rewards. The SR LLRL shows similar performances in all cases, showing that it struggles with the increase in the target patients.

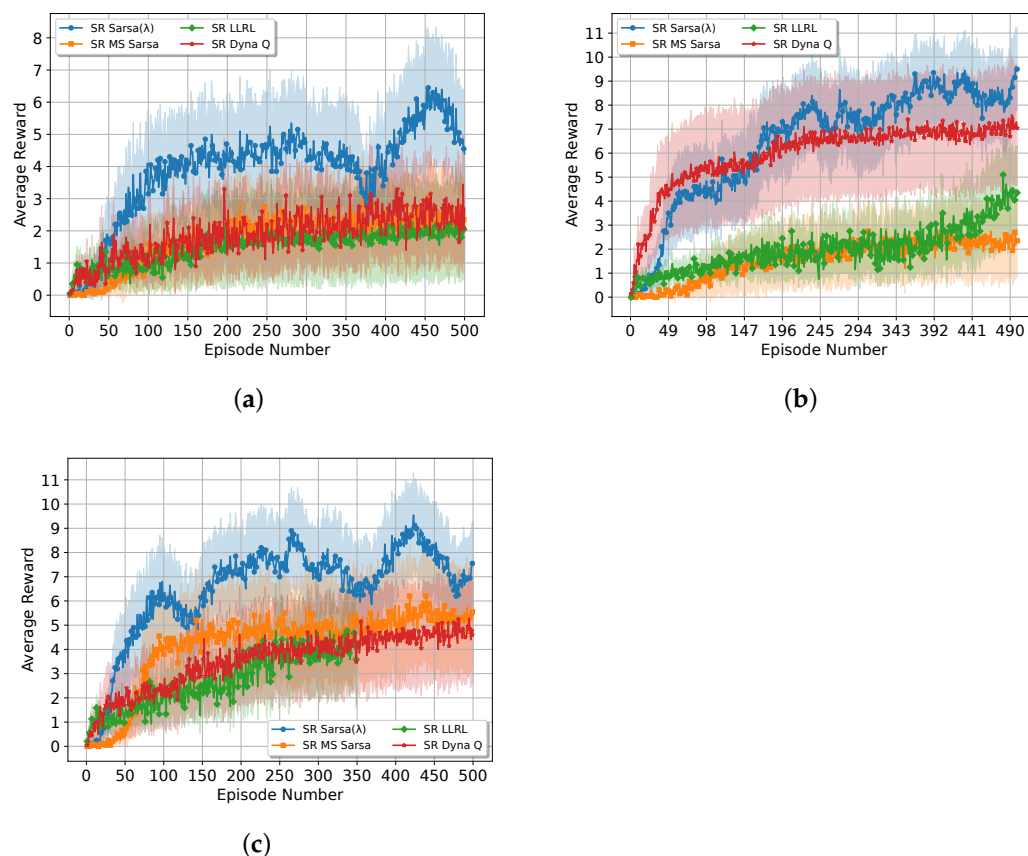


**Figure 9.** Average reward of the RL algorithms under different number of patients: (a) 1 patient; (b) 2 patients; (c) 3 patients.

#### 4.2.4. Efficiency of Different Exploration Strategy

We compare the efficiency of the proposed approach under two different exploration strategies: (i) breadth-first and (ii) depth-first in a rescue task context. The rescue task involves searching for trapped patients in earthquake environments, where the patient's location is unknown, and the environment is dynamic.

The experiments are conducted in dynamic earthquake environments, where the drone has to locate a varying number of trapped patients. The evaluation focused on two key metrics:

1. Time required to find the first trapped patient: This metric reflects how quickly the drone can locate the first trapped individual in the environment.
2. Total number of trapped patients found: This metric measures the overall effectiveness of the exploration strategy in finding all the trapped individuals.

The experimental results are shown in Tables 4 and 5. The results highlight specific differences in the performance of the breadth-first and depth-first exploration strategies when evaluated across varying levels of pre-training.

**Table 4.** Depth-first exploration performance metrics.

| Performance                            | Pre-Training Environment Number |          |          |          |          |
|----------------------------------------|---------------------------------|----------|----------|----------|----------|
|                                        | $T = 0$                         | $T = 10$ | $T = 20$ | $T = 30$ | $T = 40$ |
| Steps required to rescue first patient | 2751.3                          | 2908.5   | 1562.5   | 1751.5   | 1251.7   |
| Patient number found rate              | 0.90                            | 0.97     | 0.97     | 1.0      | 1.0      |

**Table 5.** Breadth-first exploration performance metrics.

| Performance                            | Pre-Training Environment Number |          |          |          |          |
|----------------------------------------|---------------------------------|----------|----------|----------|----------|
|                                        | $T = 0$                         | $T = 10$ | $T = 20$ | $T = 30$ | $T = 40$ |
| Steps required to rescue first patient | 7180.2                          | 8195.5   | 13,560.5 | 6453.3   | 4830.9   |
| Patient number found rate              | 0.63                            | 0.67     | 0.60     | 0.73     | 0.67     |

The breadth-first strategy generally excels in quickly locating the first trapped patient. For instance, without pre-training ( $T = 0$ ), the time required to find the first patient is 7180.2 steps. However, with increased pre-training ( $T = 40$ ), this time improves to 4830.9 steps. Despite this, the breadth-first strategy tends to cover a broader area more rapidly due to frequent resets, which mimics the effect of deploying multiple drones simultaneously. However, this approach leads to a lower overall number of patients found, with the patient number found hovering between 0.63 and 0.67, showing only modest improvement with pre-training. On the other hand, the depth-first strategy, though initially slower in locating the first patient (2751.3 steps at  $T = 0$ ), shows a significant reduction in time with pre-training, reaching 1251.7 steps at  $T = 40$ . This method allows the drone to explore the environment more thoroughly without frequent resets, leading to more successful patient rescues. The number of patients found increases steadily with pre-training, reaching a perfect score of 1.0 at  $T = 30$  and  $T = 40$ , demonstrating the effectiveness of the strategy in comprehensive search scenarios.

#### 4.3. Dynamic Obstacles

We further test the proposed approach under dynamic obstacles. In this experiment, we fix the number of obstacles to 50. We conducted four experiments: (1) pre-training in dynamic environments and testing in different dynamic environments; (2) pre-training in dynamic environments and testing in static environments; (3) pre-training in static environments and testing in dynamic environments (depth-first); and (4) pre-training in static environments and testing in dynamic environments (breadth-first).

The results of these experiments are summarised in Tables 6–9. According to the time taken to find the first person in the four experiments, the efficiency of the SAR operation is improved slightly with the increase in pre-training tasks. However, pre-training in a dynamic environment can not significantly improve the model exploration efficiency with the increase in tasks.

**Table 6.** Pre-training in dynamic environments—testing in dynamic environments.

| Performance                            | Pre-Training Environment Number |          |          |          |          |
|----------------------------------------|---------------------------------|----------|----------|----------|----------|
|                                        | $T = 0$                         | $T = 10$ | $T = 20$ | $T = 30$ | $T = 40$ |
| Steps required to rescue first patient | 917                             | 1002.5   | 1018.3   | 878      | 1304     |
| Patient number found rate              | 0.97                            | 0.97     | 0.97     | 1.0      | 0.93     |

The results show that the proposed SR Sarsa ( $\lambda$ ) can also improve dynamic efficiency in highly complex dynamic environments. Taking breadth-first as an example, drones pre-trained in static environments can also improve performance in more complex dynamic environments, which reflects the high generalisation of the proposed model.

**Table 7.** Pre-training in dynamic environments—testing in static environments.

| Performance                            | Pre-Training Environment Number |          |          |          |          |
|----------------------------------------|---------------------------------|----------|----------|----------|----------|
|                                        | $T = 0$                         | $T = 10$ | $T = 20$ | $T = 30$ | $T = 40$ |
| Steps required to rescue first patient | 2118.8                          | 2295.0   | 2970.2   | 2468.8   | 2247.0   |
| Patient number found rate              | 0.97                            | 0.93     | 0.97     | 1.0      | 0.97     |

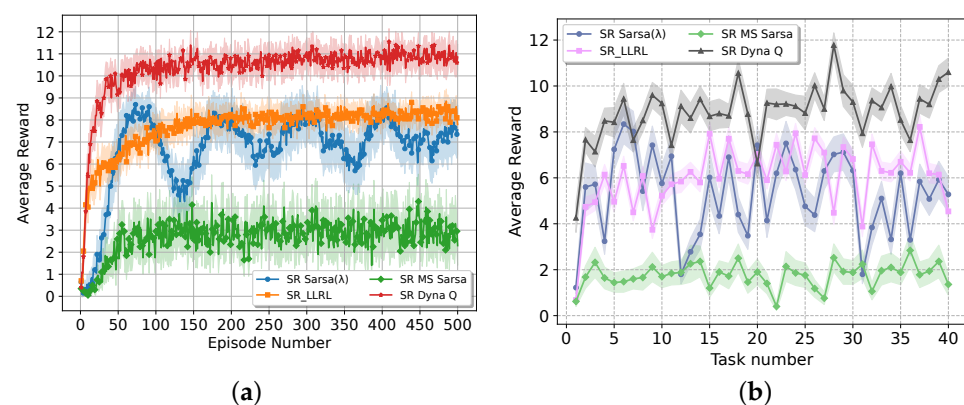
**Table 8.** Pre-training in static environments—testing in dynamic environments (depth-first).

| Performance                            | Pre-Training Environment Number |          |          |          |          |
|----------------------------------------|---------------------------------|----------|----------|----------|----------|
|                                        | $T = 0$                         | $T = 10$ | $T = 20$ | $T = 30$ | $T = 40$ |
| Steps required to rescue first patient | 963.5                           | 1214.1   | 975.5    | 818.5    | 874.9    |
| Patient number found rate              | 1.0                             | 0.93     | 0.97     | 1.0      | 0.97     |

**Table 9.** Pre-training in static environments—testing in dynamic environments (breadth-first).

| Performance                            | Pre-Training Environment Number |          |          |          |          |
|----------------------------------------|---------------------------------|----------|----------|----------|----------|
|                                        | $T = 0$                         | $T = 10$ | $T = 20$ | $T = 30$ | $T = 40$ |
| Steps required to rescue first patient | 2966.5                          | 1690.1   | 2358.3   | 2422.7   | 2413.1   |
| Patient number found rate              | 1.0                             | 0.90     | 0.93     | 0.97     | 0.97     |

Figure 10 shows the comparison results of the proposed approach against other RL algorithms using 40 pre-trained dynamic environments. The results show that both SR Dyna Q and SR LLRL exhibit a better average return compared with the proposed SR Sarsa( $\lambda$ ). This result is expected since the long short-term memory advantage of Sarsa( $\lambda$ ) is lost due to the changes in the obstacles. However, the overall performance is still competitive.

**Figure 10.** Results of the proposed approach under dynamic obstacles. (a) Average return per episode. (b) Average return per environment.

#### 4.4. Comparisons with Traditional Baseline Algorithms

The key non-RL algorithms used in the literature for SAR operations include PID controllers enhanced with Q-learning, Dijkstra,  $A^*$ ,  $D^*$ , Rapidly-exploring Random Trees (RRT), and Artificial Potential Fields (APFs) algorithms (see Appendix A). Here, only the

RRT and APFs algorithms comply with the assumptions of the proposed work, i.e., both the environment and the locations of the targets are unknown. Therefore, we only compare these algorithms against the proposed approach. To conduct fair comparisons, we used the following as metrics: (i) the number of steps needed to find the first patient and (ii) the patient's coverage within 10,000 steps; both metrics in static and dynamic environments. The results are summarised in Table 10.

**Table 10.** Comparison results against non-RL algorithms. Best results are in bold.

| Environment | Metric                          | Algorithm   |        |                        |
|-------------|---------------------------------|-------------|--------|------------------------|
|             |                                 | RRT         | APF    | SR Sarsa ( $\lambda$ ) |
| Static      | Steps to find the first patient | 2621.0      | 10,000 | <b>1251.7</b>          |
|             | Patients' coverage              | 66.67%      | 0%     | <b>100%</b>            |
| Dynamic     | Steps to find the first patient | 842.0       | 10,000 | <b>818.5</b>           |
|             | Patients' coverage              | <b>100%</b> | 0%     | <b>100%</b>            |

The results show that the RRT algorithm performs well in static and dynamic environments, such that it can effectively find the patients and adapt to environmental changes. In a static environment, it takes an average of 3000 to 6000 steps to find the first patient, and the average search and rescue coverage is about 70%. Conversely, the APF algorithm fails to find the patients and exhibits poor performance. This happens because the potential field does not explicitly use goal coordinates; instead, it relies on the combination of attractive forces towards unexplored regions and repulsive forces from obstacles. This limits the ability of the drone to target specific goal locations directly. The proposed SR Sarsa ( $\lambda$ ) in both static and dynamic environments outperforms the RRT algorithm in time and rescue coverage.

## 5. Conclusions

This paper proposes a lifelong reinforcement learning algorithm for rescue operations in post-earthquake disaster environments. The approach is based on a shaping reward heuristic that updates the reward function to generalise better to unseen environments. A Sarsa ( $\lambda$ ) algorithm is used as the main reinforcement learning algorithm to combine adaptation provided by the SR heuristic and long short-term memory given by the eligibility trace mechanism. A 2D maze-like rescue environment is designed which considers different numbers of obstacles and trapped people in different locations. The results of the simulation studies demonstrate that the proposed SR Sarsa ( $\lambda$ ) can successfully navigate and locate targets without prior knowledge of casualty locations. Additionally, the proposed method shows robustness across different environmental complexities and varying numbers of trapped individuals. According to the results from different exploration strategies, breadth-first search effectively covers most areas, making it easier to find more trapped people. The proposed approach is limited to discrete state-action spaces, which limits its generalisation to large or continuous state spaces. This can affect the performance of the shaping reward heuristic, which might require a different treatment. Additionally, a current limitation of the approach is that it assumes perfect low-level control tracking, neglecting sensor noise or uncertainties. These factors should be considered to increase the robustness and resilience of the approach. Future work will consider the implementation of the approach in a more realistic environment scenario to model other factors within the environment and drone sensing technologies, e.g., sensor noise, obstacles with different shapes, and 3D

environments. In addition, the extension of the approach to deep reinforcement learning approaches with first responder feedback is a topic of our further work.

**Author Contributions:** Conceptualisation, J.X. and A.P.; methodology, J.X. and D.P.; software, J.X.; validation, J.X., D.P., A.P., and W.G.; formal analysis, W.G.; investigation, A.P.; resources, A.T.; writing original draft preparation, A.P. and A.T.; visualisation, D.P.; supervision, A.P. and W.G.; project administration, A.T.; funding acquisition, A.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The original contributions presented in this study are included in the article material, and further inquiries can be directed to the corresponding author.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

|          |                                               |
|----------|-----------------------------------------------|
| APF      | Artificial Potential Field                    |
| CQL      | Conservative Q-Learning                       |
| GLIE     | Greedy in the limit with infinite exploration |
| LLRL     | Lifelong Reinforcement Learning               |
| MAML     | Model-Agnostic Meta-Learning                  |
| MDP      | Markov Decision Process                       |
| MS Sarsa | Multi-step Sarsa                              |
| OOD      | Out-of-distribution                           |
| RRT      | Rapidly-exploring Random Trees                |
| RL       | Reinforcement Learning                        |
| SAR      | Search and Rescue                             |
| SR       | Shaping Reward                                |
| TD       | Temporal Difference                           |
| UAV      | Unmanned Aerial Vehicle                       |

## Appendix A. Artificial Potential Field Implementation

Let  $\vec{F}_{\text{total}}$  denote the total force applied to an autonomous platform based on a potential field approach

$$\vec{F}_{\text{total}} = \vec{F}_{\text{attr}} + \vec{F}_{\text{rep}}$$

The attractive force  $\vec{F}_{\text{attr}}$  encourages movement toward open (unexplored) space. It is defined as

$$\vec{F}_{\text{attr}} = \frac{1}{N} \sum_{(dx, dy) \in \mathcal{N}} (1 - W(x + dx, y + dy)) \cdot (dx, dy)$$

where

- $\mathcal{N} = \{(-1, -1), (-1, 0), \dots, (1, 1)\}$  is the 8-connected neighborhood.
- $W(x, y) = \begin{cases} 1 & \text{if cell } (x, y) \text{ is an obstacle} \\ 0 & \text{otherwise} \end{cases}$
- $N = \sum_{(dx, dy) \in \mathcal{N}} (1 - W(x + dx, y + dy))$  is the number of free neighbors.

The repulsive force  $\vec{F}_{\text{rep}}$  pushes the agent away from nearby obstacles, and is defined as

$$\vec{F}_{\text{rep}} = \sum_{(x_o, y_o) \in \mathcal{W}} \mathbb{I}(d < d_{\text{thresh}}) \cdot \frac{1}{d^2} \cdot \frac{(x - x_o, y - y_o)}{d}$$

where

- $\mathcal{W}$  is the set of obstacle coordinates.
- $d = \sqrt{(x - x_o)^2 + (y - y_o)^2}$  is the Euclidean distance to obstacle  $(x_o, y_o)$ .
- $d_{\text{thresh}}$  is a threshold distance for repulsion (we used five units as threshold).
- $\mathbb{I}(d < d_{\text{thresh}})$  is an indicator function that equals 1 if  $d < d_{\text{thresh}}$ , and 0 otherwise.

## References

1. Daniell, J.E.; Khazai, B.; Wenzel, F.; Vervaeck, A. The CATDAT damaging earthquakes database. *Nat. Hazards Earth Syst. Sci.* **2011**, *11*, 2235–2251.
2. Zhu, X.; Wang, Y.; Regan, D.; Sun, B. A quantitative study on crucial food supplies after the 2011 tohoku earthquake based on time series analysis. *Int. J. Environ. Res. Public Health* **2020**, *17*, 7162.
3. Gruffeille, C.; Perrusquía, A.; Tsourdos, A.; Guo, W. Disaster area coverage optimisation using reinforcement learning. In Proceedings of the 2024 International Conference on Unmanned Aircraft Systems (ICUAS), Crete, Greece, 4–7 June 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 61–67.
4. La, H.M. Multi-robot swarm for cooperative scalar field mapping. In *Robotic Systems: Concepts, Methodologies, Tools, and Applications*; IGI Global: Hershey, PA, USA, 2020; pp. 208–223.
5. La, H.M.; Sheng, W.; Chen, J. Cooperative and active sensing in mobile sensor networks for scalar field mapping. *IEEE Trans. Syst. Man, Cybern. Syst.* **2014**, *45*, 1–12.
6. El Debeiki, M.; Al-Rubaye, S.; Perrusquía, A.; Conrad, C.; Flores-Campos, J.A. An Advanced Path Planning and UAV Relay System: Enhancing Connectivity in Rural Environments. *Future Internet* **2024**, *16*, 89.
7. Panagopoulos, D.; Perrusquía, A.; Guo, W. Selective Exploration and Information Gathering in Search and Rescue Using Hierarchical Learning Guided by Natural Language Input. *arXiv* **2024**, arXiv:2409.13445.
8. Neitzel, F.; Klonowski, J. Mobile 3D mapping with a low-cost UAV system. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2012**, *38*, 39–44.
9. Feng, Q.; Liu, J.; Gong, J. Urban flood mapping based on unmanned aerial vehicle remote sensing and random forest classifier—A case of Yuyao, China. *Water* **2015**, *7*, 1437–1455.
10. Rezaldi, M.Y.; Yoganingrum, A.; Hanifa, N.R.; Kaneda, Y.; Kushadiani, S.K.; Prasetyadi, A.; Nugroho, B.; Riyanto, A.M. Unmanned aerial vehicle (Uav) and photogrammetric technic for 3d tsunamis safety modeling in cilacap, indonesia. *Appl. Sci.* **2021**, *11*, 11310.
11. Marfai, M.A.; Fatchurohman, H.; Cahyadi, A. An evaluation of tsunami hazard modeling in Gunungkidul Coastal Area using UAV Photogrammetry and GIS. Case study: Drini Coastal Area. In Proceedings of the E3S Web of Conferences, Târgoviște, Romania, 6–8 June 2019; EDP Sciences: Les Ulis, France, 2019; Volume 125, p. 09005.
12. Gbadamosi, O.A.; Aremu, D.R. Design of a Modified Dijkstra’s Algorithm for finding alternate routes for shortest-path problems with huge costs. In Proceedings of the 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS), Lagos, Nigeria, 18–21 March 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
13. Zhang, Y.; Tang, G.; Chen, L. Improved A\* algorithm for time-dependent vehicle routing problem. In Proceedings of the 2012 International Conference on Computer Application and System Modeling, Taiyuan, China, 27–29 July 2012; Atlantis Press: Dordrecht, The Netherlands, 2012; pp. 1341–1344.
14. Cheng, L.; Liu, C.; Yan, B. Improved hierarchical A-star algorithm for optimal parking path planning of the large parking lot. In Proceedings of the 2014 IEEE International Conference on Information and Automation (ICIA), Hailar, China, 28–30 July 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 695–698.
15. Sedighi, S.; Nguyen, D.V.; Kuhnert, K.D. Guided hybrid A-star path planning algorithm for valet parking applications. In Proceedings of the 2019 5th International Conference on Control, Automation and Robotics (ICCAR), Beijing, China, 19–22 April 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 570–575.
16. Koenig, S.; Likhachev, M. D\* lite. In Proceedings of the Eighteenth National Conference on Artificial Intelligence, Edmonton, AB, Canada 28 July–1 August 2002; pp. 476–483.
17. Ferguson, D.; Stentz, A. Field D\*: An interpolation-based path planner and replanner. In Proceedings of the Robotics Research: Results of the 12th International Symposium ISRR, San Francisco, CA, USA, 12–15 October 2005; Springer: Berlin/Heidelberg, Germany, 2007; pp. 239–253.
18. LaValle, S.M.; Kuffner, J.J., Jr. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400.

19. Souissi, O.; Benatitallah, R.; Duvivier, D.; Artiba, A.; Belanger, N.; Feyzeau, P. Path planning: A 2013 survey. In Proceedings of the 2013 International Conference on Industrial Engineering and Systems Management (IESM), Rabat, Morocco, 28–30 October 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1–8.
20. Kumar, A.; Fu, J.; Soh, M.; Tucker, G.; Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 11784–11794.
21. Fujimoto, S.; Meger, D.; Precup, D. Off-policy deep reinforcement learning without exploration. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; PMLR: Breckenridge, CO, USA, 2019; pp. 2052–2062.
22. Kumar, A.; Zhou, A.; Tucker, G.; Levine, S. Conservative q-learning for offline reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1179–1191.
23. Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J.Y.; Levine, S.; Finn, C.; Ma, T. Mopo: Model-based offline policy optimization. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 14129–14142.
24. Wu, Y.; Tucker, G.; Nachum, O. Behavior regularized offline reinforcement learning. *arXiv* **2019**, arXiv:1911.11361.
25. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017; PMLR: Breckenridge, CO, USA, 2017; pp. 1126–1135.
26. Gupta, A.; Mendonca, R.; Liu, Y.; Abbeel, P.; Levine, S. Meta-reinforcement learning of structured exploration strategies. *Adv. Neural Inf. Process. Syst.* **2018**, *31*.
27. Liu, E.Z.; Raghunathan, A.; Liang, P.; Finn, C. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. In Proceedings of the International Conference on Machine Learning, Virtual Event, 18–24 July 2021; PMLR: Breckenridge, CO, USA, 2021; pp. 6925–6935.
28. Rakelly, K.; Zhou, A.; Finn, C.; Levine, S.; Quillen, D. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; PMLR: Breckenridge, CO, USA, 2019; pp. 5331–5340.
29. Wang, J.X.; Kurth-Nelson, Z.; Tirumala, D.; Soyer, H.; Leibo, J.Z.; Munos, R.; Blundell, C.; Kumaran, D.; Botvinick, M. Learning to reinforcement learn. *arXiv* **2016**, arXiv:1611.05763.
30. Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 3521–3526.
31. Silver, D.; Singh, S.; Precup, D.; Sutton, R.S. Reward is enough. *Artif. Intell.* **2021**, *299*, 103535.
32. Johnson, E.C.; Nguyen, E.Q.; Schreurs, B.; Ewulum, C.S.; Ashcraft, C.; Fendley, N.M.; Baker, M.M.; New, A.; Vallabha, G.K. L2explorer: A lifelong reinforcement learning assessment environment. *arXiv* **2022**, arXiv:2203.07454.
33. Parisi, G.I.; Kemker, R.; Part, J.L.; Kanan, C.; Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Netw.* **2019**, *113*, 54–71.
34. Kostrikov, I.; Fergus, R.; Tompson, J.; Nachum, O. Offline reinforcement learning with fisher divergence critic regularization. In Proceedings of the International Conference on Machine Learning, Virtual Event, 18–24 July 2021; PMLR: Breckenridge, CO, USA, 2021; pp. 5774–5783.
35. Xiao, C.; Wang, H.; Pan, Y.; White, A.; White, M. The in-sample softmax for offline reinforcement learning. *arXiv* **2023**, arXiv:2302.14372.
36. Nakamoto, M.; Zhai, S.; Singh, A.; Sobol Mark, M.; Ma, Y.; Finn, C.; Kumar, A.; Levine, S. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Adv. Neural Inf. Process. Syst.* **2024**, *36*.
37. Guerrero-Sánchez, M.E.; Hernández-González, O. Dynamics and Control of UAVs. *Machines* **2024**, *12*, 749.
38. Kownacki, C. Control and Position Tracking for UAVs. *Appl. Sci.* **2024**, *14*, 1909.
39. Cetinsaya, B.; Reiners, D.; Cruz-Neira, C. From PID to swarms: A decade of advancements in drone control and path planning—A systematic review (2013–2023). *Swarm Evol. Comput.* **2024**, *89*, 101626.
40. Yu, W.; Perrusquia, A. *Human-Robot Interaction Control Using Reinforcement Learning*; John Wiley & Sons: Hoboken, NJ, USA, 2021.
41. Perrusquia, A.; Yu, W.; Li, X. Multi-agent reinforcement learning for redundant robot control in task-space. *Int. J. Mach. Learn. Cybern.* **2021**, *12*, 231–241.
42. Chu, K.; Zhu, X.; Zhu, W. Accelerating Lifelong Reinforcement Learning via Reshaping Rewards. In Proceedings of the 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 17–20 October 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 619–624.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.



**DSPACE**

<https://dspace.org/>

## **Generalising rescue operations in disaster scenarios using drones: a lifelong reinforcement learning approach**

**Xu, Jiangshan; Panagopoulos, Dimitris; Perrusquía, Adolfo; Guo, Weisi; Tsourdos, Antonios**

**2025-06-01**

MDPI

<https://doi.org/10.3390/drones9060409>

Xu J, Panagopoulos D, Perrusquía A, et al., (2025) Generalising rescue operations in disaster scenarios using drones: a lifelong reinforcement learning approach. Drones, Volume 9, Issue 6, June 2025, Article number 409