DREUS: Deep RL-Enabled UAV Swarm for Post-Disaster Surveillance and Survivor Response

***Abstract*—Unmanned aerial vehicles (UAVs) have proven effective across diverse domains due to their flexibility, cost-efficiency, and ease of deployment. On account of their ability to traverse challenging terrains and operate independently of ground-based infrastructure, UAVs are particularly well-suited for disaster response. However, in post-disaster scenarios where traditional networking infrastructure has collapsed, autonomous operation of UAVs becomes inevitable. To maximize the rescue efforts and minimize the loss of life, it is crucial to effectively survey the affected area, locate and identify survivors, and ensure emergency assistance. Moreover, navigation in disaster-affected areas presents unpredictable dynamics such as debris, collapsing structures, etc. Thus, for effective disaster response, UAVs must be trained to achieve specific objectives while navigating dynamic obstacles. Consequently, in this work, we propose DREUS, a deep reinforcement learning (DRL)-based disaster response framework, enabling UAV swarm to conduct surveillance in post-disaster environments as well as identify and assist survivors based on facial features and emotion recognition. The framework incorporates a shared swarm database, which facilitates coordinated exploration by preventing redundant visits and centralizing survivor data for seamless identification and reacquaintance. To optimize UAV performance, DREUS employs federated DRL, promoting collaborative exploration and leverages a significant replay memory buffer (SRMB), expediting individual agent training through prioritizing critical experiences. Survivor movement within the DRL training environment is modeled using real-world 'human mobility in disaster' datasets, enhancing realism and reliability. The framework's effectiveness is assessed through implementation in the OpenAI Gym environment, simulating various disaster scenarios to evaluate its performance comprehensively. The framework is further validated through CoppeliaSim, a realistic simulator incorporating advanced physics, cameras, and GPS functionalities.**

***Index Terms*—Federated learning, deep reinforcement learning, unmanned aerial vehicles, facial recognition, path planning**

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have become integral to modern technology, finding applications across diverse domains due to their versatility, cost-efficiency, and rapid deployment capabilities. Their use spans from agricultural monitoring, infrastructure inspection, and logistics delivery to environmental conservation and urban planning [1]. In recent years, UAVs have gained prominence in disaster response due to their ability to operate autonomously in challenging and hazardous conditions [2]. According to a study by the United Nations Office for the Coordination of Humanitarian Affairs (OCHA), the use of UAVs in disaster scenarios can reduce the time required for damage assessment by up to 50% compared to traditional methods [3], [4]. Moreover, the International Federation of Red Cross and Red Crescent Societies (IFRC) highlights that UAVs significantly improve situational awareness in search-and-rescue missions, enhancing the likelihood of saving lives [5]. Equipped with advanced sensors and cameras, UAVs can navigate areas inaccessible to ground teams, efficiently identify survivors, and provide real-time data for coordinated relief efforts, making them an ideal choice for post-disaster applications.

The autonomous navigation of UAVs, enabling efficient and precise flight paths without human intervention, is essential in disaster environments due to the unpredictable and dynamic nature of such scenarios. Post-disaster areas are often characterized by collapsed infrastructure, debris, and inaccessible terrains, which make manual UAV operation challenging and inefficient. Moreover, the lack of reliable communication networks in disaster-hit areas underscores the need for UAVs to operate independently, enabling coordinated search-and-rescue missions without human intervention. Several studies have explored the use of UAVs for disaster response. For instance, Erdelj et al. proposed a framework for UAV-based communication networks to assist in disaster scenarios [6], emphasizing the importance of autonomous swarm navigation for better coverage and coordination. In a study by Kerle et al., UAVs were employed for damage assessment, highlighting their effectiveness in accessing remote locations quickly [7]. A more recent work by Bai et al. introduced a cooperative UAV framework for monitoring flood-affected regions, demonstrating the potential of UAV swarms to efficiently gather data and streamline response efforts [8]. These studies highlight the vital role UAVs play in disaster scenarios; however, they overlook critical aspects of survivor rescue and prioritization.

To this end, Alawad et al. proposed a system utilizing swarm optimization algorithms to maximize area coverage and victim localization [2]. However, the study mainly focused on optimizing search patterns without integrating real-time survivor identification or swarm-level coordination enhancements. Similarly, the Swarm-Specific Search and Rescue (SS-SAR) framework introduced decentralized UAV swarm coordination for locating and rescuing agents but lacked emphasis on integrating advanced recognition techniques such as emotion or facial analysis for survivor prioritization [9]. Wan et al. presented improved path-planning strategies for UAVs navigating hazardous environments [10]. While effective for terrain challenges, this study did not incorporate collaborative decision-making or a centralized data-sharing approach among UAVs. These works tend to focus on specific components, such as swarm coordination or navigation, and do not fully address the integration of survivor identification with advanced

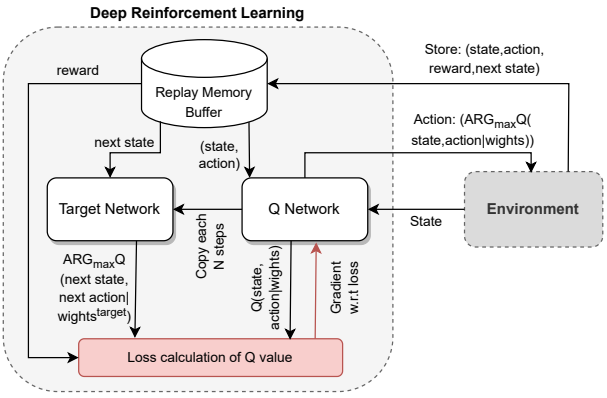


Fig. 1. Deep Reinforcement Learning Architecture.

emotion recognition or the challenges of collaborative UAV swarm operation in a dynamic post-disaster environment.

To address the gaps identified in existing works, we propose DREUS, a comprehensive **D**eep **R**einforcement learning (DRL)-**e**nabled disaster response framework using **U**AV **S**warm. Unlike previous approaches that often neglect survivor-centric rescue activities, DREUS enables UAV swarms to conduct extensive surveillance in post-disaster environments while identifying and reacquainting survivors using advanced facial recognition model (i.e., YOLOv8s [11]) and emotion analysis classifier (i.e., Haar Cascade [12]). By incorporating a shared swarm database, our framework ensures efficient and coordinated exploration, avoiding redundant UAV visits while centralizing survivor data for seamless tracking and prioritization. Furthermore, DREUS leverages federated DRL, which facilitates collaborative training among UAVs, enhancing their decision-making and adaptability in dynamic disaster scenarios. To accelerate the learning process, a significant replay memory buffer (SRMB) is used, that prioritizes critical experiences, ensuring that UAVs can rapidly adapt to high-stakes situations. Realistic modeling of survivor movement, based on human mobility patterns from real-world disaster datasets [13], further enhances the reliability and practicality of our approach. Our contributions are four-fold:

- We propose DREUS, a DRL-based disaster response framework designed to enable UAV swarms to conduct coordinated surveillance and effectively locate survivors in post-disaster environments.
- We incorporate a shared swarm database that prevents redundant UAV visits and centralizes survivor data, enabling coordinated exploration and automated survivor identification and reacquaintance.
- We optimize the training using federated learning (FL)-based collaborative exploration and leverage SRMB to expedite training by prioritizing critical experiences.
- We validate the framework through implementation using OpenAI Gym [14], modeling survivor behavior using real-world human mobility data from disaster scenarios [13] and further evaluate its performance in CoppeliaSim simulator [15].

The remainder of the paper is structured as follows: We go over the preliminary information in Section II. Section III discusses the related works. In Section IV we present our proposed DREUS framework. We go over the frameworks' technical specifics in Section V. In Section VI, we explain the evaluation setup and present the empirical analysis and findings. Finally, we conclude the paper in Section VII.

II. Background

In this section, we present some preliminary concepts that will help explain the framework later.

A. Deep Reinforcement Learning

A promising method for autonomously learning complicated behaviors from limited sensor observations is DRL. While much DRL research focuses on video games and simulated control, it has also demonstrated potential for enabling physical robots to learn real-world skills. Since real-world scenarios closely resemble human learning processes, they provide an ideal testbed for DRL algorithms [16]. Unlike traditional RL, where agents rely on a Q-table, DRL uses a neural Q-network, enabling effective learning in complex, real-world environments. The system diagram of a DRL architecture is presented in Figure 1. For training the Q-network, the agent's experiences are saved as training data samples into a storage called Reply Memory Buffer. To provide stability to the action decision from the Q-network, an additional supplemental neural network is added to the DRL framework, called the target network. The weights from the Q-network are copied to the target network after a certain number of episode steps. The target network predicts the future Q-values for the next states, which are used to calculate the loss of the Q-network's prediction.

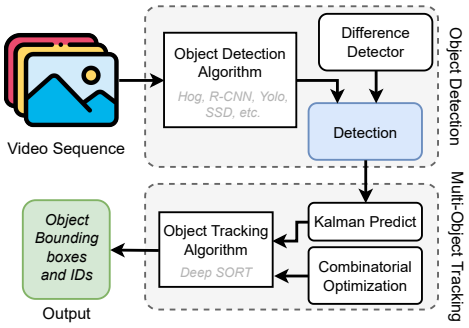


Fig. 2. Basic steps of Multiple Object Tracking (MOT) algorithms.

B. Multiple Object Tracking

Multiple Object Tracking (MOT) is a core computer vision task focused on tracking multiple objects across video frames. Objects are detected and localized using algorithms like HOG, R-CNN, YOLO, or SSD, which generate bounding boxes and assign unique IDs based on accuracy and computational needs (Figure 2). MOT then links objects across frames to maintain identities, even during occlusions or detection loss. Techniques like Kalman filtering predict future positions, while methods like Deep SORT leverage deep learning for robust tracking, effectively handling ID switches and occlusions.

C. Facial Encoding for Emotion Detection

Facial Encoding for Emotion Detection involves analyzing a person's facial expressions and converting them into numerical representations (encodings) that correspond to different emotions, such as happiness, sadness, anger, or surprise. Using machine learning models, facial encoding captures subtle facial features like eyebrow movements, mouth curvature, and eye positioning, which are key indicators of emotional states [17]. Once encoded, these features are compared against known patterns of facial expressions to classify emotions. Deep learning techniques, particularly convolutional neural networks (CNNs), are commonly used for this task, making emotion detection more accurate and applicable in areas like human-computer interaction, customer experience analysis, and mental health monitoring.

D. Federated Learning

Data and computation resources are now often dispersed across end-user devices, different areas, or corporations. Laws or regulations prevent the aggregated or direct sharing of distributed data and computing resources among various areas or organizations for machine learning tasks. FL is an effective approach for utilizing distributed computing and data resources to collaboratively train machine learning models. FL also abides by the rules and regulations to ensure data security and privacy [18]. The basic goal of FL is to do a collaborative on-device training of a single machine learning model without disclosing the raw training data to any other parties [19]. The basic steps of an FL architecture is shown in Figure 3, where the cloud server holds the global machine learning model to be trained. In the first iteration, the random weights of the global model is $Sent$ to the end devices, each having a local model. The end devices $Train$ their respective local models with their private data and then $Submit$ the local model's weight to the global model. Finally, the global model $Aggregates$ the weights and $Updates$ it's weights. The updated global model's weights again $Sent$ to the end devices, each of which $Updates$ their local models and $Trains$ it with the private local data. These steps are repeated till the end of iterations.

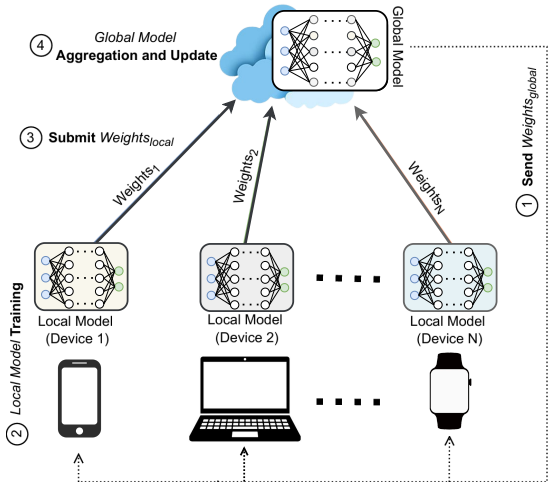


Fig. 3. Federated Learning Architecture.

III. Related Work

The use of UAVs for disaster response has been widely explored in recent years, addressing various challenges such as area coverage, obstacle navigation, and search efficiency. For instance, Gomez et al. present UAV-based systems for post-earthquake assessment, where UAVs autonomously surveyed disaster-stricken areas to generate high-resolution 3D maps for structural damage analysis [20]. In another study, Mohapatra et al. presented UAV-assisted remote sensing frameworks for wildfire detection and damage assessment [21]. UAVs equipped with thermal and infrared cameras effectively detected hotspots and evaluated fire propagation. Munawar et al. proposed a UAV system for flood monitoring using satellite imagery and UAV-collected data to estimate water levels and identify high-risk zones [22]. These works have demonstrated the potential of UAVs for disaster assessment and environmental monitoring; however, overlooked the need for survivor detection and assistance.

Consequently, Oh et al. introduced a UAV system equipped with thermal imaging cameras to locate survivors in post-disaster environments [23]. By detecting body heat signatures, their method enabled efficient identification of survivors. Lygouras et al. proposed a UAV-based search-and-rescue (SAR) system that combined computer vision and machine learning to identify survivors from aerial images [24]. While these systems were effective for detecting survivors, they did not address environmental dynamics or obstacle avoidance, which are critical in disaster scenarios. Zhang et al. developed a UAV-assisted framework for search-and-rescue operations using vision-based algorithms to locate survivors in collapsed structures [25]. Their study demonstrated the feasibility of UAVs in identifying survivors but did not optimize UAV path planning for coordinated exploration.

To address the challenges of dynamic environments and survivor assistance, Wang et al. proposed a UAV path planning algorithm using multiple-waitlist-based task assignment (MWTA) algorithm to navigate disaster environments [26]. In another study, Wu et al. employed a DRL-based method for UAV path-finding in dynamic disaster scenarios and identified optimal routes and targets through training [27]. These approaches fall short in implementing collaborative exploration strategies, advanced survivor identification with emotion detection, and realistic modeling of survivor movements. The proposed DREUS framework addresses these limitations by integrating federated DRL for swarm collaboration, shared databases for survivor tracking, and real-world human mobility datasets to enhance realism, ultimately improving the effectiveness of disaster response efforts.

IV. Proposed Framework

In this section, we discuss the proposed DREUS framework in detail, as depicted in Figure 4. The overall objective of the framework is to survey a post-disaster area, detect survivors (identification), and provide prioritized assistance (through

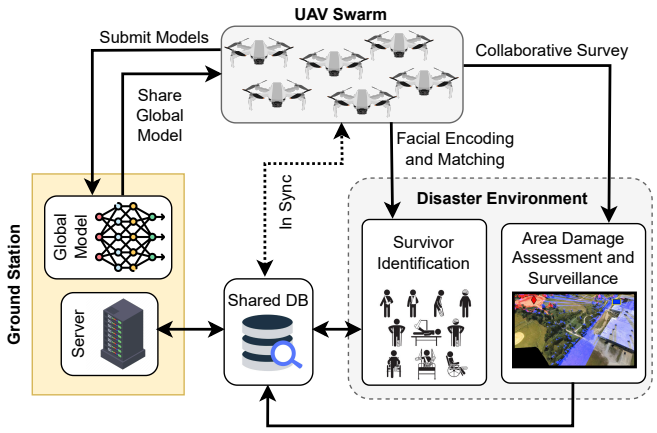


Fig. 4. The proposed DREUS Framework.

alerting rescue teams) using emotion analysis classifiers. In the subsequent sections, we introduce all the proposed modules.

A. Ground Station

The ground station houses two key elements of the framework: (i) the global model for FL and (ii) a server responsible for managing the shared database and its data. This module is vital to the entire framework, as it provides the computational power and intelligence necessary for decision-making. Each UAV in the swarm has its own DRL model that is trained through the UAV's interaction with the environment. Each of these UAVs sends its trained individual DRL models to the ground station consecutively after certain time frames, where the global model aggregates these parameters (i.e., weights and biases) to update itself. The ground station continuously shares the updated global DRL model with the UAV swarm to ensure collaborative exploration experiences by the UAVs. Apart from storing all the information contained in the shared database (e.g., surveillance and survivors), the server shares up-to-date models for face recognition, emotion detection, and damage assessment with swarm through the shared database.

B. UAV Swarm

The UAV swarm plays the central role in conducting various tasks in the disaster environment, including:

- **Coordinated Survey**: The UAVs work coordinately (ensured through reward function, discussed in Section V) to scan areas of the disaster zone for damage assessment, identifying survivors and locate areas of interest.
- **Facial Encoding and Matching**: The UAVs utilize facial encoding algorithms to identify and match survivors, aiding in the tracking of individuals who may be displaced during the disaster.
- **Carrying Shared Database**: A subset of the UAVs considered to be on high altitude platforms (HAPs), hovering at strategic locations and carrying the shared database. The other UAVs are considered to be on low-altitude platforms (LAPs), performing surveillance and survivor tracking.

The UAV swarm stays synchronized with the shared database to ensure surveillance cooperation and access to identified survivors' encodings. It also keeps the UAVs updated with the latest facial and emotion detection models.

C. Shared Database aided Disaster Environment Interaction

A shared database serves as a core repository for all collected data, ensuring that all components of the system are connected and can access updated information as well as models. Specifically, it contains data such as facial recognition matches, survivor locations, and damage assessments, which can be accessed by all UAVs for coordinated actions. The continuous syncing between the UAV swarm, ground station server, and shared database also enables real-time data analysis and model training to improve decision-making. Thus, the collected data through post-disaster zones exploration spans two main areas:

1) *Area Damage Assessment and Surveillance*: The UAVs assess the extent of damage in the affected region, providing aerial imagery and real-time updates to the ground station. This data is crucial for understanding the scale of the disaster and prioritizing rescue efforts.
2) *Survivor Identification and Reacquainting*: The swarm identifies and monitors survivors using advanced recognition and emotion detection algorithms. The system focuses on locating individuals in distress and directing help to them quickly.

As mentioned earlier, the shared database is carried by the HAP UAVs. The DREUS Framework integrates federated DRL, facial recognition machine learning models, and UAV technologies to manage disaster situations, ensuring rapid, large-scale data collection and analysis, enabling efficient rescue missions in post-disaster zones.

V. Technical Details

This section delves into the technical details of DREUS. We begin by formulating the post-disaster response task as a Markov Decision Process (MDP)-based DRL approach. Next, we elaborate on the survivor detection technique, followed by an explanation of the federated DRL model. Lastly, we explore the SRMB module and the design of the environment.

A. Modeling the DRL for Surveillance and Survivor Tracking

We model the problem as a Markov game, which generalizes the MDP. Typically, a DRL-based MDP is represented by a five-tuple comprising states, actions, transition probabilities, rewards, and a discount factor. However, given the complexity of the post-disaster environment in this study, we designed a more sophisticated MDP. The notations used for designing the MDP are defined in Table I. Formally, our MDP is represented by the nine-tuple S, A, T, R, $\pi, \epsilon, \gamma, \phi, \sigma$, with each component described in detail in the following sections:

1) States (S): The set of states encompasses all possible observations an agent can make within the environment. In our partially observable MDP, an agent's observation at a given time step consists of one or more layers of cubes surrounding it. Specifically, if there is one layer, this means the agent perceives 27 cubes around it. This concept effectively simulates the agent's partial view of the environment, enabling it to anticipate obstacles or nearby survivors. The agent can closely monitor any point of interest that enters its surrounding layers. Additionally, we assume that agents can observe the rewards associated with each cell of the two-dimensional plane on the ground (the area being surveyed), with updates on current rewards shared among all agents via the shared database. Knowing high-reward cells at the episode's start, agents plan trajectories toward them while dynamically avoiding obstacles.

2) Actions (A): The set of actions represents the activities an agent can perform to interact with the environment. Since we are modeling a discrete environment made up of cubes, the agent's movements are also discrete (as opposed to continuous angular movements). The agent can execute 11 actions, including moving up, down, forward, backward, left, right, diagonally in four directions, or hovering.

3) Transition (T): Transition probability defines the likelihood of moving from one state to another. In our model, this probability depends on both the current state and the L most recent states, where L is the number of surrounding layers visible to the agent. The agent utilizes information about the L adjacent cubes to select an appropriate action.

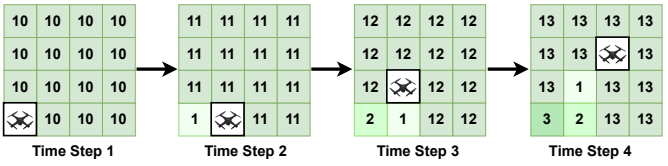


Fig. 5. Simplified reward structure for surveillance.

4) Reward (R): The reward function guides the agent's learning by assigning rewards and penalties. Two kinds of rewards and one penalty are defined in our case. The first reward, illustrated in Figure 5, motivates surveillance by rewarding UAVs for visiting ground cells. Each cell starts with a reward of 10, which resets to zero when visited, discouraging revisits. Rewards for each cell increase by one per time step, encouraging UAVs to prioritize unvisited cells and avoid recently visited cells. The second reward incentivizes survivor detection using our proposed autonomous identification technique (detailed in the following section). If a survivor is detected for the first time, the agent receives twice the reward compared to detecting a previously identified individual. To enforce obstacle avoidance, a penalty comparable in magnitude to the survivor reward is applied when agents encounter obstacles, such as debris or damaged buildings.

5) Policy (π): The policy dictates the agent's behavior by determining which action to take in a given state, based on its exploration of the environment. In our model, the policy is represented by the Q-network's weights and biases, which guide the agent's action for a given observation of the state.

TABLE I
LIST OF NOTATIONS

Symbol	Definition
S	Set of states
s	Current state
s'	Next state
A	Set of actions
R	Reward function
r	Current reward
L	Agent's observable number of layers
K	FL model update interval (episodes)
N	Target model update interval (steps)
B	Batch size of Q-network training samples
T	Transition probability
π	Policy
ϵ	Exploration parameter
γ	Discount factor
ϕ	Sorting parameter
σ	Sampling rate
δ	ϕ value increment interval

6) Exploration Parameter (ϵ): The exploration parameter controls the agent's action strategy. Initially set near 1, ϵ encourages random actions to aid learning. Over time, ϵ decreases, reducing randomness and prompting the agent to exploit the Q-network by leveraging learned knowledge.

7) Discount Factor (γ): The discount factor (γ) balances the agent's focus on immediate versus future rewards, ranging from 0 to 1. A value of γ close to 0 makes the agent shortsighted, focusing only on immediate rewards. Conversely, a future reward R that occurs after N steps will be discounted by a factor of R^N as γ approaches 1, indicating a greater concern for long-term gains.

8) Sorting Parameter (ϕ): This parameter governs the insertion of experiences into the SRMB. Its value ranges from 0 to 1, where a value of zero indicates that only experiences in which an agent encounters a survivor or obstacle are stored in SRMB, disregarding layer encounters. The parameter can be incrementally increased to one over L steps, where L represents the number of layers. Let δ represent the incremental change in the ϕ value, which can be calculated using the following equation:

$$\delta = \frac{\phi_{max} \text{ - } \phi_{min}}{L} \tag{1}$$

Here, ϕ_{max} is set to 1 and ϕ_{min} to 0. An initial increment from 0 indicates that only experiences involving a survivor or obstacle in the nearest layers are stored. As the parameter approaches 1, all experiences, including those involving goals or obstacles in the farthest surrounding layers, are also stored.

9) Sampling Rate (σ): This parameter regulates the sampling rate of experience memories used for batch-wise training of the Q-Network. It can range from 0 to 1, where a value of 0 means all samples are drawn from the general Replay Memory Buffer, while 1 indicates all samples come from the SRMB. We test various σ values to evaluate SRMB's impact on DREUS's performance.

B. Autonomous Survivor Identification and Emotion Detection

In addition to post-disaster zone surveillance, the primary goal of the DREUS framework is real-time survivor detection and emotion classification. As swarm UAVs navigate the affected area, they continuously execute human detection algorithms using data from their camera sensors. Detected survivors' encoded features are stored in a shared database, ensuring centralized access for coordinated operations. The system then utilizes an emotion recognition classifier to assess the emotional states of the survivors, enabling prioritization of assistance based on urgency. Another key feature is the reacquaintance of previously identified survivors after a specified time interval. To avoid continuous tracking, which could degrade overall surveillance performance, a carefully balanced time threshold is implemented. This threshold is set high enough to prevent UAVs from persistently following an individual but low enough to minimize the risk of a survivor moving too far from their last identified location. This balance ensures efficient resource allocation while maintaining the safety and visibility of survivors.

This module, consisting of multiple subsystems, is orchestrated by a **Queue manager** that serves as the communication backbone. Utilizing a queue-based architecture, it optimizes inter-process communication and supports parallel computing, enhancing both scalability and responsiveness.

1) Human Detection Subsystem: This subsystem employs the YOLOv8s model by Ultralytics [28] to detect human presence in RGB images extracted from video feeds. This model was selected for its balance of computational speed and detection accuracy, making it well-suited for real-time applications. To maintain robust tracking, the Supervision library is integrated to streamline communication between detection and tracking modules. The ByteTracker algorithm [29] is utilized to provide continuous updates on the spatial positions of detected individuals. Detection outputs, including bounding box coordinates, confidence scores, and class identifiers, are relayed to ByteTracker for precise and real-time monitoring.

2) Face Detection and Emotion Recognition Subsystem: This subsystem is designed to identify and assess the emotional states of individuals detected in the environment.

Face Detection: It employs the Haar Cascade Classifier [12] from OpenCV for its computational efficiency and effectiveness in detecting faces in real-time. Detected faces are extracted using bounding box coordinates provided by the detection module. To prevent redundant processing, the system compares extracted facial features to a database of existing detections, eliminating duplicate identifiers.

Emotion Recognition: Emotion analysis is performed using a custom-designed CNN. The CNN features multiple convolutional layers for feature extraction, followed by fully connected layers tailored to classify facial expressions into seven categories: *angry, disgust, fear, happy, neutral, sad,* and *surprise*. The model is implemented and trained using TensorFlow/Keras, leveraging its optimization capabilities to ensure efficient training and inference.

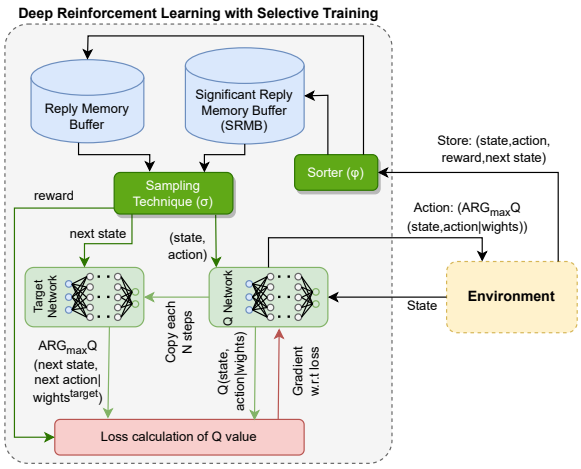


Fig. 6. Improved Deep Reinforcement Learning Architecture with SRMB.

The system's queue-based command structure enables seamless interaction between human detection, face processing, and emotion recognition. Such integration allows for real-time identification and emotional assessment of individuals, which is critical for effective decision-making in disaster response scenarios.

C. Federated Learning for Swarm Intelligence

FL is primarily used to train machine learning models from distributed data sources while ensuring data privacy. We found that FL's inherent distributed learning structure enhances the performance of swarm-based learning models, as the global model benefits from the diverse experiences of different swarm agents. Similar to standard FL, our proposed federated DRL (FDRL) follows four repeating steps, as shown in Figure 3. At the start of the first episode, a global Q-network (at ground station) is initialized with random weights and biases and distributed to all agents. Each agent then initializes its local Q-network model using the received weights and biases. After every K episodes, agents submit the current weights and biases of their Q-networks to the ground stations global model, where an aggregator processes the received models. We experimented with three different aggregation algorithms, which are discussed later in this section. The global model is then updated with the aggregated weights and biases and redistributed to all agents at the start of the $(K+1)$-th episode. The agents, in turn, update their local Q-networks by copying the weights and biases from the global model. Algorithm 1 presents the steps involved in the proposed FDRL.

1) FedSGD: Federated stochastic gradient descent (FedSGD) [30] averages the gradients from a randomly selected subset of swarm UAVs to perform a gradient descent step for updating the global model.
2) FedAvg: Federated averaging (FedAvg) [31] updates the global model by averaging the weights and biases of all local Q-networks.
3) FedMA: Federated match averaging (FedMA) [32] merges nodes with similar weights in each layer of the network, updating the global model based on these.

Algorithm 1: Proposed Federated DRL Algorithm

Set $\{A, R, B, N, \epsilon, \gamma, \phi, \sigma\}$ for DRL
Set $\{K, Algo^{Aggregate}\}$ for FL
Set $\{L, Limit^{ENV}\}$ for S
for *each iteration* $\in MaxIterations$ **do**
  Initialize($Agents, Survivor, Obstacle^{Stat}, Obstacle^{Dyn}$)
  for *each episode* $\in MaxEpisodes$ **do**
    CreateENV($Survivor, Obstacle^{Stat}, Obstacle^{Dyn}$)
    if $episode == 0$ **then**
      Initialize($globalModel$)
      Agents $\leftarrow$ **Send**($globalModel^{w,b}$)
    end
    for *each step* $\in MaxSteps$ **do**
      for *each agent* $\in Agents$ **do**
        if $rand(0,1) \geq \epsilon$ **then**
          $action \leftarrow agent.QNet(s)$
        end
        else
          $action \leftarrow rand(A)$
        end
        $r, s' \leftarrow Perform(action, s)$
        $agent.Memory \leftarrow \phi(action, s, r, s')$
        if $Size(agent.Memory) \geq B$ **then**
          $B \leftarrow \sigma(agent.Memory)$
          $agent.QNet.train(B)$
        end
        if $agent.loc == Goal.loc$ **then**
          $Update(\epsilon)$
        end
        if $step\%N == 0$ **then**
          $agent.TagetNet \leftarrow agent.QNet$
        end
      end
    end
    for *each agent* $\in Agents$ **do**
      $B \leftarrow \sigma(agent.Memory)$
      $agent.QNet.train(B)$
    end
    if $episode\%K == 0$ *and* $episode \geq 1$ **then**
      localModels $\leftarrow \{\}$
      for *each agent* $\in Agents$ **do**
        localModels $\leftarrow$ **Submit**($agent.QNet$)
      end
      $globalModel \leftarrow$**Update**(**Aggregator**(localModels))
      Agents $\leftarrow$ **Send**($globalModel^{w,b}$)
    end
  end
end

We experiment using each of the algorithms (discussed in Section VI) to choose the optimal one for DREUS.

D. Selective Training with SRMB

In this section, we discuss the proposed SRMB concept, designed to accelerate the training of the Q-network. Typically, as discussed in Section II, the training data for the Q-network in DRL consists of experience tuples gathered from the agent's exploration. At each time step, the agent's current state, action, reward, next state, and a completion flag (indicating goal achievement) are stored as a tuple in a memory buffer. This buffer has a fixed size, and after each episode, a random batch of tuples is selected for training the Q-network. However, random sampling does not always guarantee that the selected tuples will significantly aid the learning of intelligent behavior.

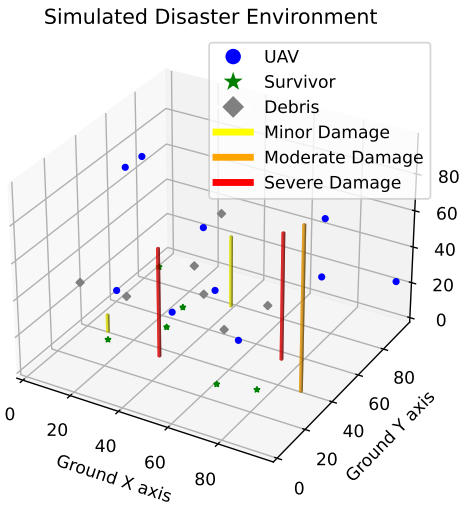


Fig. 7. The simulated environment for training the UAV swarm.

To address this, we propose SRMB, which enhances model training by prioritizing the most important memory tuples. Each agent will utilize both the standard reply memory buffer and the SRMB, with the impact of SRMB controlled by a sampling rate parameter (σ) and a sorting parameter (ϕ). The roles of σ and ϕ were previously discussed. Figure 6 illustrates the block diagram of the improved DRL training using SRMB.

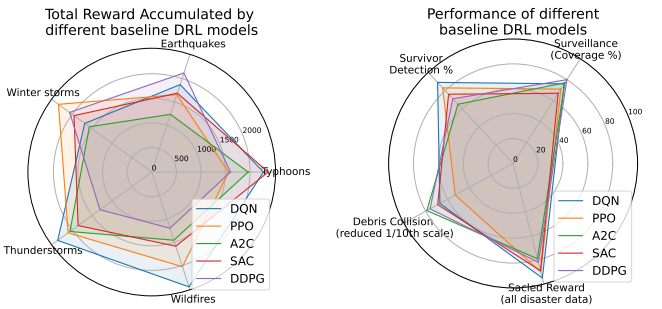
1) Sorter: In contrast to standard DRL, where memory tuples are directly stored in the reply memory buffer, in our approach, all memory tuples first pass through the sorter. The sorter categorizes the tuples based on the ϕ parameter and stores the most relevant ones in the SRMB. The remaining tuples are sent to the Reply Memory Buffer.

2) Sampling Technique: This module prepares the training batch for the Q-network. Based on the sampling rate σ, it selects $B \times \sigma$ samples from the SRMB, where B is the size of the training batch. The remaining $B \times (1 - \sigma)$ samples are drawn from the Reply Memory Buffer.

E. Modeling the Post-Disaster Scenario

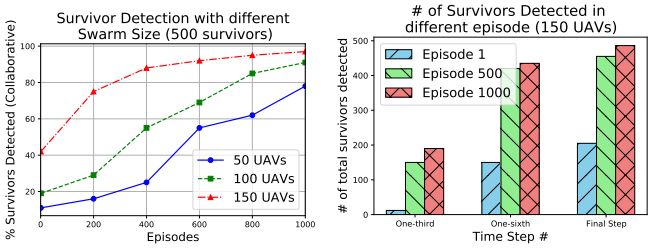
To simulate a realistic post-disaster environment for the DRL training, a 3D environment was developed using OpenAI Gym that accurately represents the challenges a UAV swarm might encounter during the post-diaster response. As presented in Figure 7, the environment incorporates damaged buildings, survivors, and dynamically moving obstacles (debris), each contributing to a more complex and varied decision-making landscape for the UAV swarm.

Damaged Buildings: Represented as vertical towers, buildings have varying levels of damage, categorized as minor, moderate, or severe. This categorization influences UAV behavior, as the reward for detecting these buildings depends on the severity of the damage. The buildings are distributed across different heights and locations in the environment, simulating real-world structural damage patterns.



(a) (b)

Fig. 8. Comparison of various DRL models from the OpenAI Gym baselines library, evaluated in training environments: (a) using human mobility datasets from different disasters to analyze reward gains, and (b) using all datasets to compare performance metrics.



(a) (b)

Fig. 9. Experimentation with the survivor detection performance: (a) percentage of survivors detected over training episodes and (b) number of survivors detected at each step of an episode.

Survivors: The movement of survivors in the simulated environment is modeled using human mobility data derived from real-world disaster scenarios [13]. This data, collected from geolocated Twitter activity during natural disasters, provides a realistic basis for simulating survivor behavior. The dataset includes movement patterns across various disaster types, such as typhoons, earthquakes, winter storms, thunderstorms, and wildfires, ensuring that the framework is adaptable to any kind of disaster. Survivors' movements are informed by the power-law dynamics observed in human mobility, reflecting resilience and perturbations due to disaster events. This comprehensive dataset enables the framework to incorporate realistic and diverse mobility behaviors, representing both steady-state and disrupted conditions. Consequently, UAVs trained in this environment are better equipped to adapt to a wide range of disaster scenarios. Survivor movements are constrained to the XY plane and simulate injuries or restricted mobility caused by entrapment, but their trajectories follow patterns observed in the dataset, enhancing the authenticity of the simulation.

Debris: A unique addition to the environment is the inclusion of randomly moving debris that poses a dynamic obstacle for the UAVs. Unlike the fixed-position damaged buildings, debris moves in all directions (including along the Z-axis), adding another layer of complexity. Collisions with debris also result in penalties, simulating the need for UAVs to avoid moving objects during the search mission.

The environment is designed to replicate the complexity of real-world disaster recovery operations, combining static obstacles (buildings) with dynamic ones (debris) to challenge UAVs in balancing exploration, survivor detection, and obstacle avoidance. A differential penalty system encourages safe navigation, while randomly moving debris simulates the unpredictability of post-disaster conditions.

VI. Evaluation

In this section, we evaluate the DREUS framework through a series of experiments. First, we justify the selection of DQN as our DRL model by comparing it with other DRL alternatives. Next, we assess the model's capabilities in surveillance, obstacle avoidance, and survivor detection. Subsequently, we analyze performance improvements achieved by optimizing SRMB parameters, followed by an evaluation of face and emotion detection models. Finally, we investigate FL aggregation techniques and validate the framework's real-world performance using CoppeliaSim.

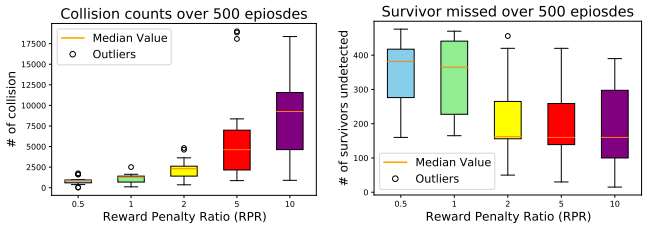
A. Comparison of DRL Models Across Disaster Scenarios

In this section, we analyze the performance of various baseline DRL models (available in OpenAI Gym) across different disaster scenarios (Figure 8). As shown in Figure 8(a), the total reward accumulated by DQN, PPO, A2C, SAC, and DDPG models, when trained using human mobility dataset in disaster environments, exhibit varying performance across typhoons, earthquakes, winter storms, thunderstorms, and wildfires. Among the models, DQN and PPO consistently achieve higher rewards, demonstrating superior learning capabilities and adaptability across all disaster types. DQN, in particular, accumulates the highest rewards, indicating its robustness in handling complex and dynamic environments. SAC and A2C show moderate performance, while DDPG lags behind, accumulating the lowest rewards overall.

Figure 8(b) shows the performance of baseline DRL models across key metrics: surveillance coverage, survivor detection, debris collisions (scaled by 1/10th), and scaled rewards using all disaster datasets. DQN achieves the highest performance, with close to 80% surveillance coverage and 90% survivor detection, while maintaining the lowest debris collisions. PPO follows closely with comparable results. In contrast, SAC and A2C achieve moderate performance, with coverage and detection rates around 70-75%, while DDPG performs the worst, achieving under 60% in both coverage and detection.

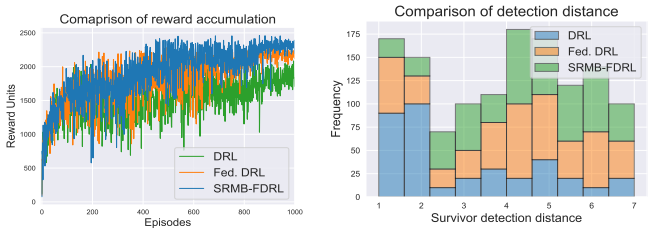
B. Survivor Detection Performance

In this section, we analyze the survivor detection performance of the model. Figure 9 shows a strong correlation between swarm size and detection efficiency. As shown in the Figure 9(a), larger swarms consistently outperform smaller ones in both learning speed and detection rates. The 150 UAV swarm rapidly improves, reaching over 80% detection by the 600th episode and surpassing 90% by the 1000th, showcasing effective area coverage and collaboration. In comparison, the 100 UAV swarm achieves nearly 70% detection, while the 50 UAV swarm progresses slower, reaching around 60% by episode 1000. These results indicate that smaller swarms require more training to optimize their search strategies.



(a) (b)

Fig. 10. Experimentation with the collision avoidance performance through box-plots: (a) collision count distribution with different RPR and (b) the distribution of survivor detection failure with different RPR.



(a) (b)

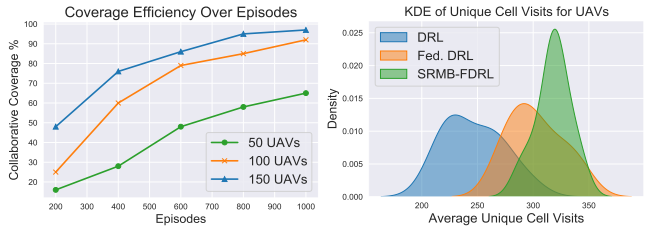
Fig. 11. Caparison of the different DRL techniques in terms of (a) reward accumulation throughout the training episode, (b) intelligent behavior observation via detection distance.

Figure 9(b) highlights detection progress for the 150 UAV swarm at Episodes 1, 500, and 1000. Initially, the UAVs detect few survivors, reflecting a lack of optimized strategies. By Episode 500, detection improves significantly, with over 400 survivors identified at the final step, showcasing enhanced collaboration and spatial coverage. By Episode 1000, performance nears optimal levels, detecting nearly all 500 survivors. However, the rate of improvement tapers off between Episodes 500 and 1000, indicating diminishing returns as the model converges toward optimal strategies.

C. Collision avoidance with Penalty Trade-off

We investigate the impact of different Reward Penalty Ratio (RPR) values on UAV swarm collision avoidance, as shown in Figure 10, and analyze their effect on survivor identification to estimate an optimal RPR. Figure 10(a) shows that lower RPR values (0.5, 1, and 2) result in fewer collisions, with lower medians and compact interquartile ranges. This indicates that UAVs can explore effectively with minimal concern for collisions. As RPR increases to 5 and 10, collision counts rise significantly, with the median for RPR=10 reaching approximately 12,500 and a much wider interquartile range, reflecting high variability. Excessive penalties make UAVs overly cautious, leading to ineffective navigation and increased collisions due to restricted movement. While outliers are present at lower RPR values, their frequency and spread grow at higher RPR values. The results suggest that moderate RPR values strike a better balance between exploration and collision avoidance, whereas high penalties cause suboptimal behavior.

Figure 10(b) shows the survivor detection failure counts over 500 episodes for different RPR values, revealing their impact on UAV swarm performance. Lower RPR values (0.5, 1) result in higher detection failures, with counts often exceeding 400. For RPR = 0.5, variability is significant (160–476), indicating that UAVs prioritize collision avoidance over accurate detection. As RPR increases to 2, detection failures decrease, with fewer outliers and counts dropping as low as 50. This suggests a better balance between collision avoidance and survivor detection. At higher RPR values (5, 10), failure counts remain low, but UAVs may overly focus on detection at the expense of collision avoidance. Overall, moderate RPR values, such as 2, achieve the best balance between efficient navigation and survivor detection.



(a) (b)

Fig. 12. Assessing the surveillance performance of the UAVs via (a) comparing the collaborative coverage percentage with different swarm sizes and (b) comparing the KDE curves using different DRL models.

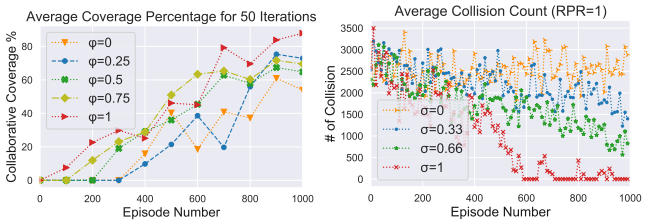
D. Assessment of Improved DRLs

In this section, we compare our improved DRL approaches with conventional DRL in terms of reward accumulation and intelligent behavior (Figure 11). As shown in Figure 11(a), general DRL has slower learning, lower rewards, and high variability, plateauing after 600 episodes. Federated DRL improves performance through distributed learning, achieving higher, more stable rewards. SRMB-FDRL performs best, with faster convergence, higher peak rewards, and stable learning due to the significant reply memory buffer. Figure 11(b) compares survivor detection distances for the three methods. SRMB-FDRL (green) achieves the highest frequency of detections at greater distances (4-7), reflecting intelligent detection behavior. General DRL (blue) concentrates detections at close ranges (1-2), indicating less efficient strategies. Federated DRL (orange) shows intermediate performance, improving over DRL but falling short of SRMB-FDRL. This highlights SRMB-FDRL as the most effective method for enabling agents to detect survivors efficiently from safer distances.

E. Evaluation of the Surveillance Performance

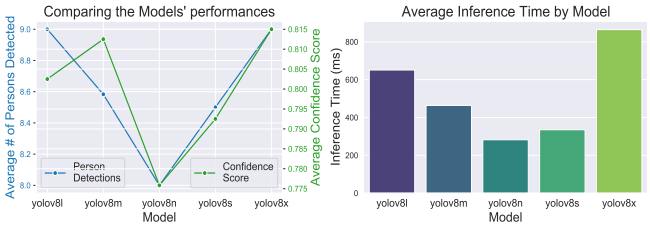
In this section, we evaluate the surveillance efficiency of the DREUS framework, as shown in Figure 12. From Figure 12(a), it is observed that coverage improves with larger UAV swarms. By episode 1000, the 150 UAV group achieves 97% area coverage, compared to 92% for 100 UAVs and 65% for 50 UAVs. While larger swarms offer better collaboration and exploration, gains diminish after 800 episodes as coverage plateaus. The results demonstrate strong scalability and efficiency, with the most significant improvements occurring early in training.

Figure 12(b) compares surveillance performance across general DRL, Federated DRL, and SRMB-FDRL using KDE



(a) (b)

Fig. 13. Experiment with SRMB parameters for determining optimal values of σ and ϕ: (a) average coverage percentage for the agents with variable ϕ parameter values with σ=0.7 and L=4, for 50 iterations, and (b) average collision count for the agents with variable σ values with ϕ=1 and RPR=1.



(a) (b)

Fig. 14. Comparing different YOLO models with respect to (a) the average number of persons detected and the corresponding confidence scores, and (b) average inference time in milliseconds (ms).

curves of unique cell visits. General DRL shows the widest, flattest distribution, indicating inconsistent coverage. Federated DRL improves coverage with a narrower, higher peak, showing better UAV collaboration. SRMB-FDRL achieves the most efficient performance, with a sharp, focused peak reflecting consistent and intelligent exploration. Overall, SRMB-FDRL significantly enhances surveillance efficiency compared to the other methods.

F. Experimenting with the SRMB parameters

In this section, we analyze the SRMB parameters ϕ and σ. Figure 13(a) shows the collaborative coverage percentage for different ϕ values. Higher ϕ values generally improve coverage, with ϕ = 0.75 achieving the best performance, reaching 80% coverage after episode 200. Having ϕ = 0.5 and ϕ = 1, DREUS performs well but lags slightly, while lower values (ϕ = 0, 0.25) incur slower growth, indicating inefficient memory tuple selection. Optimal performance is achieved with ϕ between 0.5 and 0.75. Figure 13(b) evaluates collision counts for varying σ values, which control sampling between SRMB and regular buffers. Higher σ reduces collisions, with σ = 1 showing the steepest drop, nearing zero collisions by episode 400. Lower σ values, like σ = 0, result in consistently high collision counts. Intermediate values (σ = 0.33, 0.66) show gradual improvement, with σ = 0.66 striking a balance by significantly reducing collisions after 400 episodes.

G. Detection Performance and Inference Efficiency

Several detection models were evaluated for our disaster response task, but the YOLO architecture by Ultralytics [28] was chosen for its high-speed, real-time analysis. Unlike traditional region-based methods, YOLO uses a single-pass approach for object predictions, greatly improving processing speed. While models like Faster R-CNN and RetinaNet may excel in specific cases, YOLO's speed makes it ideal for real-time tasks. We selected YOLOv8 for its strong performance, accuracy, and extensive community support. The YOLOv8 models offer various weights, ranging from lightweight models like YOLOv8n (3.2 million parameters) to larger ones like YOLOv8x (70 million parameters). Testing on a custom dataset (Figure 14) showed that larger models (YOLOv8l and YOLOv8x) achieved higher accuracy but took significantly longer to process images, making them unsuitable for real-time use. Lightweight models like YOLOv8n and YOLOv8s balanced speed and accuracy effectively. Figure 14(a) compares person detection counts (blue line) and confidence scores (green line). While models detecting more individuals may show lower confidence, balancing detection count and confidence is crucial for accuracy and reliability.

Figure 14(b) highlights inference times, a key factor for real-time performance. Shorter times ensure system responsiveness, critical for applications like surveillance. Although higher-accuracy models may require longer processing times, balancing accuracy and speed is essential for these scenarios. Since YOLOv8m achieves good detection with high confidence, having a reasonable inference time, we choose this model to present a human detection case study in Figure 15.

Fig. 15. Person detection with confidence score using YOLOv8m.

H. Experiment with Different Aggregator Models

Table II presents the comparative analysis of the aggregator algorithms, with different K values. We start by setting the K value to 1, meaning the global model will be updated after every episode. Then we increase the K value and found that setting this parameter's value to 6 incurs the highest average reward after 1000 episodes. Also, the $FedAvg$ algorithm was found to be the most fitting option for this application.

TABLE II
DREUS'S PERFORMANCE W.R.T. AGGREGATION ALGORITHMS

Algorithm	***Total Reward after 1000 episodes***				
	K**=1**	K**=3**	K**=6**	K**=9**	K**=12**
FedSGD	832	1578	1892	1209	782
FedAvg	1578	1892	2378	2128	2067
FedMA	1123	1592	1928	1683	1389

I. Case study with a Graphic Simulator

We use the CoppeliaSim Robotics Simulator [15] to run simulations for the 3D disaster environment and observe how well the DREUS performs in real-world situations. The simulator incorporates a physics engine to replicate real-world

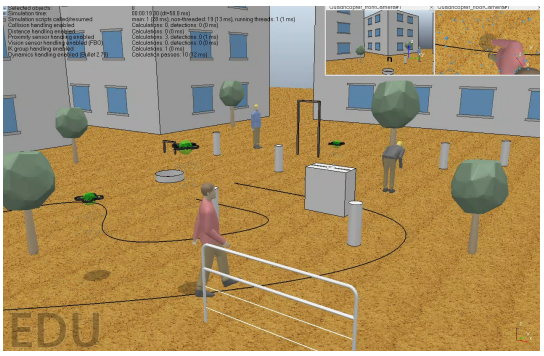


Fig. 16. Case study of DREUS using CoppeliaSim Robotics Simulator.

dynamics, such as wind and collisions. In the simulation, survivors are represented by standing or moving humanoid models amidst damaged buildings, while obstacles include rubble, broken structures, and scattered furniture (Figure 16). The UAVs demonstrate intelligent behavior, effectively navigating the complex environment and adapting trajectories.

VII. Conclusion

In this work, we presented DREUS, a robust DRL-based framework designed to enhance UAV swarm operations in post-disaster environments. By integrating a shared swarm database, the framework enables coordinated exploration and seamless survivor identification, reducing redundancy and improving efficiency. The adoption of federated DRL facilitates collaborative learning across the swarm, while the SRMB expedites individual agent training through prioritized experiences. Realism in the training process is further ensured by modeling survivor movement with real-world human mobility datasets in disaster scenarios. Comprehensive evaluations in OpenAI Gym and CoppeliaSim validate the framework's ability to navigate complex environments, locate survivors, and deliver effective assistance. These results demonstrate the potential of DREUS to advance disaster response capabilities, paving the way for intelligent, autonomous UAV systems that can adapt to the challenges of real-world scenarios.

References

[1] G. Muchiri and S. Kimathi, "A review of applications and potential applications of uav," in *Proceedings of the Sustainable Research and Innovation Conference*, 2022, pp. 280–283.

[2] W. Alawad, N. B. Halima, and L. Aziz, "An unmanned aerial vehicle (uav) system for disaster and crisis management in smart cities," *Electronics*, vol. 12, no. 4, p. 1051, 2023.

[3] UN-OCHA, "From digital promise to frontline practice," https://www.unocha.org/publications/report/world/digital-promise-frontline-practice-new-and-emerging-technologies-humanitarian-action.

[4] Y. Karaca, M. Cicek, O. Tatli, A. Sahin, S. Pasli, M. F. Beser, and S. Turedi, "The potential use of unmanned aircraft systems (drones) in mountain search and rescue operations," *The American journal of emergency medicine*, vol. 36, no. 4, pp. 583–588, 2018.

[5] IFRC, "Drones – uncrewed aerial vehicles (uav) team," https://go.ifrc.org/surge/catalogue/other/uav.

[6] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the sky: Leveraging uavs for disaster management," *IEEE Pervasive Computing*, vol. 16, no. 1, pp. 24–32, 2017.

[7] N. Kerle, F. Nex, M. Gerke, D. Duarte, and A. Vetrivel, "Uav-based structural damage mapping: A review," *ISPRS international journal of geo-information*, vol. 9, no. 1, p. 14, 2019.

[8] Y. Bai, K. Asami, M. Svinin, and E. Magid, "Cooperative multi-robot control for monitoring an expanding flood area," in *2020 17th International Conference on Ubiquitous Robots (UR)*. IEEE, 2020.

[9] A. Phadke and F. A. Medrano, "Increasing operational resiliency of uav swarms: An agent-focused search and rescue framework," *Aerospace Research Communications*, vol. 1, p. 12420, 2024.

[10] Y. Wan, Y. Zhong, A. Ma, and L. Zhang, "An accurate uav 3-d path planning method for disaster emergency response based on an improved multiobjective swarm intelligence algorithm," *IEEE Transactions on Cybernetics*, vol. 53, no. 4, pp. 2658–2671, 2022.

[11] Yolov8, "Ultralytics yolov8," https://docs.ultralytics.com/models/yolov8/.

[12] OpenCV, "Git-repo," https://github.com/opencv/opencv/tree/-master/data/haarcascades.

[13] Q. Wang and J. E. Taylor, "Patterns and limitations of urban human mobility resilience under the influence of multiple types of natural disaster," *PLoS one*, vol. 11, no. 1, p. e0147299, 2016.

[14] OpenAI, "Gym is a standard api for reinforcement learning, and a diverse collection of reference environments," https://www.gymlibrary.dev/index.html.

[15] CoppeliaSim, "Create. compose. simulate. any robot." https://www.coppeliarobotics.com/.

[16] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, 2021.

[17] S. K. Khare, V. Blanes-Vidal, E. S. Nadimi, and U. R. Acharya, "Emotion recognition and artificial intelligence: A systematic review (2014–2023) and research recommendations," *Information fusion*, vol. 102, p. 102019, 2024.

[18] J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, and D. Dou, "From distributed machine learning to federated learning: A survey," *Knowledge and Information Systems*, pp. 1–33, 2022.

[19] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1342–1397, 2021.

[20] C. Gomez and H. Purdie, "Uav-based photogrammetry and geocomputing for hazards and disaster risk monitoring–a review," *Geoenvironmental Disasters*, vol. 3, pp. 1–11, 2016.

[21] A. Mohapatra and T. Trinh, "Early wildfire detection technologies in practice—a review," *Sustainability*, vol. 14, no. 19, p. 12270, 2022.

[22] H. S. Munawar, F. Ullah, S. Qayyum, S. I. Khan, and M. Mojtahedi, "Uavs in disaster management: Application of integrated aerial imagery and convolutional neural network for flood detection," *Sustainability*, vol. 13, no. 14, p. 7547, 2021.

[23] D. Oh and J. Han, "Smart search system of autonomous flight uavs for disaster rescue," *Sensors*, vol. 21, no. 20, p. 6810, 2021.

[24] E. Lygouras, N. Santavas, A. Taitzoglou, K. Tarchanidis, A. Mitropoulos, and A. Gasteratos, "Unsupervised human detection with an embedded vision system on a fully autonomous uav for search and rescue operations," *Sensors*, vol. 19, no. 16, p. 3542, 2019.

[25] D. Zhang, Y. M. Shiguematsu, J.-Y. Lin, Y.-H. Ma, M. S. Al Maamari, and A. Takanishi, "Development of a hybrid locomotion robot for earthquake search and rescue in partially collapsed building," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2019, pp. 2559–2564.

[26] B. Wang, Y. Sun, D. Liu, H. M. Nguyen, and T. Q. Duong, "Social-aware uav-assisted mobile crowd sensing in stochastic and dynamic environments for disaster relief networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 1070–1074, 2019.

[27] C. Wu, B. Ju, Y. Wu, X. Lin, N. Xiong, G. Xu, H. Li, and X. Liang, "Uav autonomous target search based on deep reinforcement learning in complex disaster scene," *IEEE Access*, 2019.

[28] Ultralytics, "Yolo." https://github.com/ultralytics.

[29] ——, "Bytetracker," https://docs.ultralytics.com/reference/-trackers/bytetracker/.

[30] H. Yuan and T. Ma, "Federated accelerated stochastic gradient descent," *Advances in Neural Information Processing Systems*, 2020.

[31] S. Ek, F. Portet, P. Lalanda, and G. Vega, "Evaluation of federated learning aggregation algorithms: application to human activity recognition," in *2020 ACM International Symposium on Wearable Computers*, 2020.

[32] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," *arXiv preprint arXiv:2002.06440*, 2020.