Article

GLIDE: Multi-Agent Deep Reinforcement Learning for Coordinated UAV Control in Dynamic Military Environments

Divija Swetha Gadiraju 1,†, Prasenjit Karmakar 2,†, Vijay K. Shah 3 and Vaneet Aggarwal 4,*

1 School of Interdisciplinary Informatics, University of Nebraska, Lincoln, NE 68588, USA; dgadiraju@unomaha.edu
2 Department of Computer Science and Engineering, IIT Kharagpur, Kharagpur 721302, India; prasenjitk@cse.iitkgp.ac.in
3 Department of Cybersecurity Engineering, George Mason University, Fairfax, VA 22030, USA; vshah22@gmu.edu
4 School of Industrial Engineering, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA
* Correspondence: vaneet@purdue.edu
† These authors contributed equally to this work.

Abstract: Unmanned aerial vehicles (UAVs) are widely used for missions in dynamic environments. Deep Reinforcement Learning (DRL) can find effective strategies for multiple agents that need to cooperate to complete the task. In this article, the challenge of controlling the movement of a fleet of UAVs is addressed by Multi-Agent Deep Reinforcement Learning (MARL). The collaborative movement of the UAV fleet can be controlled centrally and also in a decentralized fashion, which is studied in this work. We consider a dynamic military environment with a fleet of UAVs, whose task is to destroy enemy targets while avoiding obstacles like mines. The UAVs inherently come with a limited battery capacity directing our research to focus on the minimum task completion time. We propose a continuous-time-based Proximal Policy Optimization (PPO) algorithm for multi-agent Learning In Dynamic Environments (GLIDE). In GLIDE, the UAVs coordinate among themselves and communicate with the central base to choose the best possible action. The action control in GLIDE can be controlled in a centralized and decentralized way, and two algorithms called Centralized-GLIDE (C-GLIDE), and Decentralized-GLIDE (D-GLIDE) are proposed on this basis. We developed a simulator called UAV SIM, in which the mines are placed at randomly generated 2D locations unknown to the UAVs at the beginning of each episode. The performance of both the proposed schemes is evaluated through extensive simulations. Both C-GLIDE and D-GLIDE converge and have comparable performance in target destruction rate for the same number of targets and mines. We observe that D-GLIDE is up to 68% faster in task completion time compared to C-GLIDE and could keep more UAVs alive at the end of the task.

Keywords: UAV swarm; military application; multi-agent reinforcement learning

1. Introduction

Unmanned aerial vehicles (UAVs), are employed extensively in missions involving navigating through unknown environments, such as wildfire monitoring [1], target tracking [2], and search and rescue [3], as they can host a variety of sensors to measure the environment with relatively low operating costs and high flexibility. Most research on UAVs depends on the target model’s accuracy or prior knowledge of the environment [4]. However, this is extremely difficult to achieve in most realistic implementations because environmental information is typically limited. Deep Reinforcement Learning (DRL) has made incredible advances in recent years in many well-known sequential decision-making tasks, including playing the game of Go [5], playing real-time strategy games [6], controlling robots [7], and autonomous driving [8], especially in conjunction with the development of deep neural networks (DNNs) for function approximation [9]. A Deep Reinforcement
Learning (DRL) agent autonomously learns an optimal policy to maximize its rewards through its interaction with the environment. The flight environment for a UAV is usually local or completely unknown during online path planning. Hence, the agent has to react to the dynamic environment using incomplete information, which is the key challenge in UAV action control [10,11]. In [12], DRL was used to solve an online path-planning problem independently on the environment model by trial-and-error interactions. In [12], the model is cooperation-oriented during the algorithm’s online phase, which makes the algorithm robust to selfish exploitation. Model-free RL methods [13] and the Q learning methods [14] have gained recent popularity. We focus on one of the most important applications of UAV, which is target tracking and obstacle avoidance.

In applications of UAVs in military or civil fields such as strike, reconnaissance, rescue, and early warning [1], UAVs need path planning in dynamic environments. This is challenging as the UAV needs to avoid all the obstacles that are not known to the UAV beforehand [15]. A single agent in a dynamic environment might lack the battery capacity to accomplish all the tasks efficiently. Hence, multiple UAVs are employed to coordinate and complete the task. Interestingly, the vast majority of applications in the DRL literature involve the participation of multiple agents, which requires using multi-agent reinforcement learning (MARL). This was achieved by viewing the problem as a temporal correlation and a sequential decision-making optimization problem. This is effectively solved by the DQN. MARL is designed to address the sequential decision-making problem for multiple agents operating autonomously in a shared environment, which together seek to maximize their own long-term return by interacting with the environment as well as other agents [16]. MARL algorithms can be classified into different categories based on the types of situations they handle as fully cooperative, fully competitive, and a mix of the two. This allows a variety of new solutions that build on concepts such as cooperation or competition [16]. However, multi-agent settings also introduce challenges, including inadequate communication, difficulties in reward assignment to agents, and non-stationary environments.

We consider a military environment application for multiple UAVs, where each UAV can act independently of the other agent’s actions. The environment contains a set of targets that the UAVs need to coordinate and destroy. There are several mines in the field that are capable of destroying the UAVs. These mines are placed at any random location that is unknown to the UAVs. We assume that the mines have a sensing radius in which they can detect the UAV and destroy it. In this scenario, the UAV must learn to detect and avoid mines within a minimum distance from the sensing radius of the mine. The goal here for each UAV is to destroy the enemy target by avoiding the mines and reaching the base safely. The challenge for a UAV is that it has no prior knowledge of the location of the mines and has to find a suitable policy to adapt to the dynamic environment. We propose a multi-agent Learning In Dynamic Environments (GLIDE), which uses MARL to control the action of a set of UAVs. In GLIDE, the MARL-based approach aims to find optimal strategies for agents in settings where multiple agents interact in the same environment. In GLIDE, we adopt Proximal Policy Optimization (PPO) to train our agents, which is a well-known state-of-the-art continuous control algorithm. The aim of the UAV fleet is to explore the entire field of operation as quickly as possible so that the task can be completed within a minimum time. We compare the performance of centralized action-control-based GLIDE, called C-GLIDE with decentralized action-control-based GLIDE, called D-GLIDE. The results show that the UAVs can accomplish minimum task completion time with D-GLIDE.

The main contributions of this paper are as follows:

1. The coordinated UAV action control problem is formulated as a Markov Decision Process (MDP) with the action space involving accelerations of all the UAVs. Previous works use fixed-length strides for the movement of UAVs, due to which the UAVs move to a fixed distance at every timestep. In GLIDE, at each timestep, the UAVs can choose to change their acceleration. This gives freedom for the UAVs to move to any amount of distance intended per timestep. We observed that this approach also used lower parameters for training the model, since we just have three continuous values in the x, y, and z
directions. The state space takes into account the global situational information and also the local situation faced by each UAV during the time of operation.

2. We propose two MARL algorithms based on PPO for coordinated UAV action control, namely centralized GLIDE (C-GLIDE) and decentralized GLIDE (D-GLIDE) with a continuous action space. In C-GLIDE, the action control is performed based on the combined state space information available at the base. This keeps all the UAVs updated with global and local information. However, this slightly hampers the task-completion time, whereas D-GLIDE is based on centralized training and decentralized execution. The UAVs have access to their local data and are updated with global information once every few timesteps. This resulted in a faster task-completion time.

3. We built a simulator for our experimentation called UAV SIM. Our experimentation results show that both these algorithms converge and have a comparable target-destruction rate and mine-discovery rate. With a low number of targets and mines, both C-GLIDE and D-GLIDE perform equally. C-GLIDE is useful for lower number of targets, mines, and UAVs. As the number of targets and mines increases to the maximum limit, D-GLIDE completes the task with up to 68% less time compared to C-GLIDE. We also observe that the target destruction rate of D-GLIDE is up to 21% and up to 42% higher, which indicates that more UAVs are alive with the same number of mines in the field compared to C-GLIDE.

The remainder of this article is organized as follows. Section 2 discussed the related work. The system description and problem formulation are presented in Section 3. A DRL-based path planning algorithm is proposed in Section 4. Section 5 shows the performance evaluation and simulation environment. Section 6 concludes the article with concluding remarks and future research directions.

2. Related Work

UAVs have recently become popular in commercial and many other fields for target identification and detection. Many studies have been conducted concentrating on the applications of UAVs or the use of UAVs to complete specific scientific research tasks. In this section, we present a review of the literature related to our research.

2.1. Path Planning and Action Control

There are many works in path planning using DRL, with applications in drone fleets for delivery, traffic flow control, automated driving, and especially UAVs [16]. The authors of [17] used a dueling double deep Q-networks (D3QN) approach for path-planning UAVs in dynamic contexts. They assume the availability of global situational data for the UAV, which is used for its decision-making and path planning. In [18], the path planning for a cooperative, non-communicating, and homogeneous team of UAVs is formulated as a decentralized partially observable Markov decision process (Dec-POMDP). DDQN with combined experience replay is used to solve the problem. In [19], a study on UAV ground target tracking under obstacle environments using a deterministic policy gradient (DDPG) algorithm is presented. The authors attempt to improve the DDPG algorithm for UAV target tracking. In [10], a 3D continuous environment with static obstacles is built, and the agent is trained using the DDPG algorithm. In [20], an approach is presented to exploit global–local map information that allows trajectory planning. The work tries to scale to large and realistic scenario environments efficiently with an order of magnitude more grid cells compared to previous works in a similar direction. In [21], using the Deep Q Learning method, the UAV executes computational tasks offloaded from mobile terminal users, and the motion of each user follows a Gauss-Markov random model. The authors of [4] proposed an algorithm in which each UAV makes autonomous decisions to find a flight path to a predetermined mission area. Each UAV’s target destination is not predetermined, and the authors discuss how their algorithm can be used to deploy a team of autonomous drones to cover an evolving forest wildfire and provide virtual reality to firefighters. In this work, GLIDE leverages a continuous-time-based algorithm for coordinated UAV control in a
dynamic military environment. Here, we not only consider the global situational data but also the local situation data of each UAV and control the actions of a group of UAVs.

2.2. Multi-Agent Approach for UAV Control

In [22], the UAV control policy is learned by the agent, which generalizes over the dynamic scenario parameters. In [18], a data-harvesting method using path planning from distributed Internet of Things (IoT) devices is presented and uses multiple cooperative, non-communicating, and homogeneous groups of UAVs. In [23], the limitation on communication in UAV is discussed and addressed using DRL for energy-efficient control to improve coverage and connectivity. In [17], path planning for UAVs in dynamic environments with potential threats is considered based on the global situation information using D3QN. In [24], an online distributed algorithm for tracking and searching is proposed. The authors in [25] propose a Geometric Reinforcement Learning (GRL) algorithm for the path planning of UAVs. In this work, we focus on the action control of a group of UAVs in a centralized and decentralized implementation of GLIDE. Our aim is to scan the entire field in the minimum possible time and complete the task.

2.3. PPO Based MARL

PPO is an advantage-based actor–critic algorithm that tries to be conservative with policy updates [26]. In one trajectory, using KL divergence and a clipped surrogate function, it can update multiple steps of update [27]. A review of MARL and its application to autonomous mobility is discussed in [16], where MARL’s state-of-the-art methods are presented along with their implementation details. In [28], to handle the nonlinear attitude control problem, contemporary autopilot systems for UAVs are studied with a DRL controller based on a PPO algorithm. In [27], a long-term planning scenario that is based on drone racing competitions held in real life is discussed. The racing environment was created using AirSim Drone Racing Lab, and the DRL agent was trained using the PPO algorithm. Their result shows that a fully trained agent could develop a long-term planning scenario within a simulated racing track.

A summary of related works is presented in Table 1. In this work, we introduce a novel formulation of the UAV action control problem. Following that, we propose GLIDE, which leverages a continuous-time PPO-based approach for the action control of a group of UAVs in a dynamic environment. A centralized action control algorithm C-GLIDE is compared with a decentralized action control algorithm D-GLIDE. We implemented the simulation environment called UAV SIM for our experimentation. Recent advancements in decentralized multi-agent systems are illustrated by several studies. One of the key approaches that has been used in applications is a mean-field-based approach [29–31], which is a form of centralized training decentralized execution strategy [32], and has been used in many applications including ride-sharing systems [33–36], datacenter resource allocation [37], traffic signal control management [38], and the coordination of robot swarms [39]. In this paper, our decentralized approach uses such an approach. In the next section, the system model will be discussed.

Table 1. A table summarizing related works.

<table>
<thead>
<tr>
<th>Paper</th>
<th>Number of UAVs</th>
<th>Objective</th>
<th>Solution Approach</th>
<th>Environment</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>[10]</td>
<td>Single</td>
<td>Path planning</td>
<td>Centralized DDPG</td>
<td>Three-dimensional continuous environment with aerial obstacles</td>
<td>Reward convergence</td>
</tr>
<tr>
<td>[26]</td>
<td>Multiple</td>
<td>Jointly control multiple agents</td>
<td>Centralized PPO</td>
<td>Military environment</td>
<td>Reward convergence</td>
</tr>
<tr>
<td>[27]</td>
<td>Multiple</td>
<td>Drone racing competition</td>
<td>Decentralized PPO</td>
<td>Environment was created using AirSim</td>
<td>Task completion</td>
</tr>
<tr>
<td>[17]</td>
<td>Multiple</td>
<td>Path planning</td>
<td>Centralized D3QN combined with greedy heuristic search</td>
<td>Military environment developed using STAGE Scenario</td>
<td>Task completion</td>
</tr>
<tr>
<td>[18]</td>
<td>Multiple</td>
<td>Data harvesting</td>
<td>Centralized DQN</td>
<td>Urban city like structure map</td>
<td>Successful landing and collection ratio</td>
</tr>
<tr>
<td>This work, GLIDE</td>
<td>Multiple</td>
<td>Coordinated action control</td>
<td>Centralized and decentralized PPO</td>
<td>Military environment created with our simulator, UAV SIM</td>
<td>Task completion time and reward convergence</td>
</tr>
</tbody>
</table>
3. System Model

This section presents the system model for the multi-UAV action control problem. Each UAV scans a large area divided into multiple grids. The grid coordinates are used to simplify the positioning of the UAVs and a set of $K$ targets. A group of UAVs is assigned to accomplish the task in this area. Consider a set of $N$ UAVs that intend to complete the task within a time $T$. The time of operation $T$ consists of several time slots $(1, 2, \cdots, t)$, each of length $\delta$. All the UAVs start from the base and return to the base upon task completion. We assume that all the UAVs have sufficient battery capacity for task completion. We consider a military environment where the UAVs need to navigate and destroy the target. The targets are immobile and are placed in the area of operation. A single UAV would have a limited battery and cannot span the entire area when considering a large field of operation, whereas a group of UAVs can coordinate and accomplish the task of destroying the targets more efficiently. The action control becomes challenging due to the presence of mines, which have a destruction range within which they can destroy the UAVs. Hence, the UAVs need to detect and avoid the mines with a distance greater than the destruction range of the mine. All the UAVs communicate with the base and obtain regular updates about the mines. We assume that the total number of mines is known before starting the task, but the location of the mines is unknown. A UAV, upon detecting a mine, communicates the location of the mine to the base. The base then updates all the UAVs with the coordinates of the mines. Figure 1 represents the system model. All the UAVs have knowledge of all the target positions at the beginning of each experiment. We consider a square grid world of size $g \times g \in \mathbb{N}$ with each cell of size $c$ and the set of all possible positions of UAV, $m$. We assume the continuous movement of UAVs in the environment and incorporate the map-based state space. A comparison of the system model of GLIDE with the works in the literature is presented in Table 2. The details of the UAV simulator are given in the next subsection.

Table 2. A table comparing the system models.

<table>
<thead>
<tr>
<th>Paper</th>
<th>Single or Multiple UAVs</th>
<th>Obstacles and Mines</th>
<th>Assumptions</th>
<th>Environment</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>[40]</td>
<td>Multiple</td>
<td>None</td>
<td>All the UAVs are connected to a cellular network</td>
<td>Ground</td>
<td>Finding the best path</td>
</tr>
<tr>
<td>[24]</td>
<td>Multiple</td>
<td>Preset obstacle areas in grid map</td>
<td>UAVs follow the assigned path</td>
<td>Ground patrol area</td>
<td>Target searching and tracking</td>
</tr>
<tr>
<td>[41]</td>
<td>Multiple</td>
<td>None</td>
<td>All the UAVs are connected to a cellular network</td>
<td>Ground based dynamic users</td>
<td>QoE driven UAVs assisted communications</td>
</tr>
<tr>
<td>This work, GLIDE</td>
<td>Multiple</td>
<td>Mines</td>
<td>UAVs periodically communicate with the base</td>
<td>Military environment created with our simulator, UAV SIM</td>
<td>Target destruction</td>
</tr>
</tbody>
</table>

Figure 1. An illustration of the System Model.
3.1. UAV Simulator

A set of UAVs are considered to move in the environment, which is modeled as a grid of dimensions \( g \times g \) and of height \( h \). All the UAVs start from the base and return to the base upon task completion. Each UAV’s current position in the grid is represented by \( p_i^t = [x_i^t, y_i^t, z_i^t] \), with the altitude ranging from between 0 and \( h \). The operational status \( b_i^t \in \{0, 1\} \), indicates whether the UAV is inactive or active. The environment contains the start and end positions at the base. Each UAV has a battery as its fuel and a radio antenna for communication. The UAV can communicate to the base and internally with other UAVs. There is a base set up, from which the UAVs are deployed, and they are supposed to complete their task and come back to the base, from which the UAVs are monitored. In addition to this, each UAV has sensing capability with a sensing range. When the UAVs are functioning, we consider the following parameters of the environment, which can give the state information: the battery consumption of the UAV; the current UAV location, which is given in coordinates; the coordinates of the target, which needs to be destroyed; and the discovered mines on the field. Other factors that affect UAV performance are the distance traveled, and directional movement is also taken into account. Figure 2 shows an illustration of the UAV simulator with a group of UAVs in the environment with mines and targets. The targets and mines are placed at random locations at the beginning of each episode. The location of targets is known to all the UAVs but the locations of mines are unknown to the UAVs at the beginning of the episode. All the UAVs are aware of the number of targets and number of mines at the beginning. The targets are placed on a ground location with a height of \( h_T \), and mines are just placed on the ground with a firing range. Each UAV has a sensing radius within which it can detect mines and targets. The proposed algorithm uses this information to determine the UAV’s plan of action. We assume the following: (i) Each UAV will move to the position that has been planned for the epoch. (ii) Each UAV can monitor their battery levels and calculate how long it can fly with the remaining battery. (iii) Each UAV communicates information on its own location, as well as the location of the mine if it is discovered. We also assume negligible communication delay between UAVs and the base. In C-GLIDE, the UAVs communicate continuously with the base. In D-GLIDE, the UAVs communicate whenever a mine is discovered; otherwise, the communication is once in every \( K \) timesteps. The grid clearly defines the surveillance area. The simulator code is available on github (https://github.com/GLIDE-UAV/GLIDE (accessed on 22 July 2024)).

![UAV Simulator](image)

**Figure 2.** UAV Simulator with the outdoor environment with UAVs. Mines and targets placed on the ground at random locations.

3.2. Markov Decision Process Model

The objective of this work is to ensure the task completion of target destruction with a group of UAVs in an adaptable and intelligent way, while taking into account various factors like mine avoidance, and distance traveled by each UAV. At each timestep, the input data is analyzed to understand the dynamics of the environment. The environment here is the UAV simulator, which uses a common coordinate system for the operation of
all the agents. The action control is performed based on the requirement of minimum distance traveled and obstacle avoidance for task completion. The optimization problem is formulated as an Markov Decision Process (MDP). The MDP environment is defined by the tuple $< S, A, R, \lambda >$, where $S$ denotes the state space, $A$ is the action space, $R$ represents the reward function, and $\lambda$ is the discount factor ranging from 0 to 1. At each timestep $t$, the agent is provided with a state representation, denoted as $s$, of the environment. The agent then selects an action, denoted as $a$, based on a policy $\pi(a|s)$. Subsequently, a transition to the next state, $s'$, takes place with a probability determined by $P(s'|s,a)$, and the agent receives a reward, $r$, as detailed in the following subsections.

3.2.1. State

The state reflects the environment at a given time, in this case, the dynamic military environment information at each timestep. The state space, $S$, consists of the target coordinates, mine coordinates, and various attributes of the UAV. We consider a set of $N$ UAVs communicating with a base. Since the mines are capable of destroying the UAV, we consider a binary vector $B^N$ with $N$ elements with each entry, $b_i$, indicating the live status of the UAV i.e., 1 denoting that the UAV is alive and 0 denoting that the UAV is destroyed. The target coordinates are represented in a two-dimensional grid, called the target map denoted by $W^{m \times m}$. If the target is present in the field and is active, then the grid coordinates corresponding to the target locations are set to $+1$. If the target is destroyed by one of the UAVs, then those coordinates are represented by $-1$. Every other situation is represented by 0 in the target map. Similarly, the mine map indicates whether the mine is discovered or destroyed. If the mine is detected and is active, then its corresponding coordinates in the mine map are set to $+1$. If the mine is destroyed by the UAV, it is marked by $-1$ and every other situation is represented by 0. A mine could blast a UAV and destroy itself, such situations where the UAV did not destroy the mine are represented by 0. Note that, we only keep the non-zeros entries of target map and mine map to reduce the size of the state space. The position of each UAV is a three-dimensional vector. Since, we have $N$ UAVs, all the UAV positions are considered in the state space and is represented by $U^{N \times 3}$. The three-directional velocity of each UAV, is also considered in the state space, denoted by $V^{N \times 3}$. Let $M^{m \times m}$ be the mine coordinates. The combined state information is used to choose the actions of each UAV.

$$s_t = \{ \{ x : x \neq 0 \land x \in W^{m \times m} \}, \{ y : y \neq 0 \land y \in M^{m \times m} \}, U^{N \times 3}, V^{N \times 3}, B^N \}$$ (1)

The base has the entire information of the environment, that is, the target locations, detected mine location and state information for each UAV position, velocity and liveliness. The base communicates the state with the UAVs and receives the UAV specific information from each UAV during the communication. In C-GLIDE, the UAVs are in continuous communication with the base whereas in D-GLIDE, the UAVs communicate whenever a mine is discovered or once in every $K$ timesteps.

3.2.2. Action

The objective of an agent is to map the state space $S$ to the action space $A$. A set of $N$ deployed UAVs move within the limits of the grid of the environment. At every timestep $t$, an agent $i \in N$ selects an action $a_i \in A$ for each UAV. The action is the decision to move toward the next target. The position of each UAV $i$ is described using $p_i = [x(i), y(i), z(i)]^T \in \mathbb{R}^3$ with altitude at ground level or any value up to an altitude $h$. The operational status of a UAV is either taking an action or staying in the current position (no movement). The UAV can choose to move toward the next target or stay in the current position. A set of $N$ deployed UAVs move within the limits of the grid of the environment. The action space is continuous acceleration in three directions for each UAV. The initial speed considered for each UAV is 0 m/s. With every timestep, the UAV changes its speed, moves a certain distance, and calculates the velocity for the next timestep. The maximum acceleration for each UAV is 50 m/s$^2$. 
The action set is a choice of acceleration in each direction, i.e., x, y, and z. For instance, with the current UAV position, a particular acceleration can be chosen as \((a_x, a_y, a_z)\) as \((-5, 0, 5)\); then, the UAV would move in the direction of the resultant of the three chosen vectors covering a distance of \(5\sqrt{2}\). Here, the UAV might face an obstacle or reach the end of the grid in the y direction; hence, its acceleration in the y direction is 0. This helps us bind the movement of UAVs within the grid. Using this action space helps the UAV to not only move linearly but also take a curvature motion. Note that \(N\) UAVs are present, so we predict \(N\) resultant acceleration vectors at each timestep.

All the UAVs start at the base with an initial speed of 0 and, depending on the current acceleration, direction of movement towards the target, and time, choose their next acceleration. In C-GLIDE, every agent has a different value of acceleration based on a centralized architecture, whereas in D-GLIDE, every agent chooses a different value of acceleration in a decentralized architecture. These methods are explained in detail in the next section.

3.2.3. Reward Function

The reward function \(r_t\), is based on the state \(s_t\) and the action \(a_t\) at the current timestep \(t\). The DRL algorithm maximizes the discounted future reward, \(R_t\). Based on the current state and action, the agent obtains a reward from the environment. We consider the reward function to be a combination of (a) the proximity-based reward, which takes into account the distance traveled by each UAV; (b) mine detection and avoidance; (c) target destruction; and (d) a liveliness reward. Each reward is normalized between 0 and 1 and added to the total reward.

Proximity Based Reward

Assuming that the UAV scans at a constant distance, we can define a proximity-based reward that will motivate the UAV to move closer to the target and maintain a safe distance from the mines. First, we will discuss avoiding mines and then moving closer to the target. If the UAV moves too close to the mine, it will get destroyed within the destruction range of the mine. If the mine is destroyed by the UAV, the UAV will obtain zero rewards since it does not have to avoid it anymore. Consider a binary variable indicating the liveliness of the mine as \(l_m\), which is set if the mine is alive. If the mine is alive, we calculate the reward based on the minimum distance, \(d_{\text{min}}\), and the least distance, \(d_{l}\). The value of \(d_{l}\) is the destruction range of the mine that needs to be avoided by the UAV to stay safe. The minimum distance \(d_{\text{min}}\) in this case is calculated as the minimum distance between a particular UAV and a detected mine. As the UAV moves closer to this range, it is negatively rewarded. Without the proximity-based reward for the mine, the UAV will not be motivated to maintain distance from the mine. Let the mine proximity reward be represented as \(r_{mp}\) for each UAV and is clipped between 0 and 1.

\[
r_{mp} = \begin{cases} 
-\left(\text{clip}(1/d_{\text{min}}, 0, 1/d_{l}) \times d_{l}\right), & \text{if } l_m = 1 \\
0, & \text{if } l_m = 0 
\end{cases} 
\]

The reward \(r_{mp}\) is calculated at each UAV and is communicated to the base, since we have \(N\) UAVs communicating their reward to the base. At the base, the mine reward is normalized and expressed as,

\[
R_{mp} = \frac{\sum r_{mp}}{N}
\]

Similarly, if a target is destroyed, the agent receives a positive reward; otherwise, it receives a reward based on minimum distance \(d_{\text{min}}\) and the least distance \(d_{l}\). The minimum distance \(d_{\text{min}}\) in this case is calculated as the minimum distance between a particular UAV and a target. The value of \(d_{l}\) is the firing range of a UAV, which indicates the least distance between the UAV and a target so that the UAV can destroy the locked target. The closer the UAV gets to the target, the more it is positively rewarded. A binary variable indicating
the liveliness of the target, \( l_t \), is set if the target is alive. Let the target proximity reward for each UAV be represented as \( r_{tp} \) and clipped between 0 and 1.

\[
 r_{tp} = \left( \text{clip}\left(\frac{1}{d_{\text{min}}}, 0, \frac{1}{d_l}\right) \right) - 1, \quad \text{if } l_t = 1. \tag{4}
\]

All the UAVs communicate the value of \( r_{tp} \) to the base. Since we have \( N \) UAVs, at the base, the target reward is normalized and expressed as

\[
 R_{tp} = \frac{\sum r_{tp}}{N}. \tag{5}
\]

**Target Destruction Reward**

The target destruction reward, \( r_{td} \), is received by an agent when a UAV successfully destroys a target. By doing so, the agent has completed a part of the assigned task and gains a positive reward. Let \( d_t \) be the reward received when a UAV destroys one target from a total of \( S \) targets. In our implementation, the value of \( d_t \) is set to 1. At each UAV, \( r_{td} \), is calculated as follows:

\[
 r_{td} = \begin{cases} 
 d_t, & \text{if } l_t = 1 \\
 0, & \text{if } l_t = 0 
\end{cases} \tag{6}
\]

At the base, the target rewards communicated by each UAVs are summed to check if all the targets are destroyed. Since we have \( Z \) targets, the target destruction reward at the base is expressed as

\[
 R_{td} = \frac{\sum r_{td}}{Z}. \tag{7}
\]

**Mine Detection Reward**

A UAV detects a mine within its sensing radius and communicates the location of the mine to the base in addition to avoiding it. The base can then update the rest of the UAVs in the next timestep about the location of the detected mine. A mine detection is denoted by \( m_d \) and is given to each UAV when it detects a mine \( m \) from a total set of \( M \) total mines. In our implementation, the value of \( m_d \) is set to 1.

\[
 r_{md} = \begin{cases} 
 m_d, & \text{if } m \in M \\
 0, & \text{otherwise} 
\end{cases} \tag{8}
\]

The number of mines present in the field is known, but the locations of those mines is unknown. Considering there are \( M \) mines in the field, the mine detection reward calculated at the base can be expressed as follows:

\[
 R_{md} = \frac{\sum r_{md}}{M}. \tag{9}
\]

**Time Based Reward**

The time-based reward \( R_\tau \) is calculated at the base to ensure that all the UAVs are negatively rewarded with each passing timestep. Our aim here is to finish the task as quickly as possible so that we can save the battery consumption of the UAVs. The total time for task completion for all the UAVs is \( T \) and with every timestep \( \tau \), the combined UAVs’ time-based reward is penalized.

\[
 R_\tau = \begin{cases} 
 -1, & \text{if } 0 < \tau < T \\
 0, & \text{otherwise} 
\end{cases} \tag{10}
\]

A time-based reward motivates the fleet of UAVs to finish the task in minimum time; this does not essentially mean that the task aims at keeping the maximum number of UAVs alive. Hence, we introduce a liveliness reward.
Liveliness Reward

We try to keep as many UAVs alive as possible upon task completion. If the UAV is destroyed by the mine, it is rewarded 0. If it destroys the targets and returns to the base, it is alive and is rewarded positively. The reward $R_l$ is calculated at the base as the ratio of the number of UAVs alive, $u_a$, by the maximum number of live UAVs possible, $N$. This is given as follows:

$$R_l = \frac{u_a}{N} - 1 \quad (11)$$

If all the UAVs are alive, then the value of $R_l$ is 0. If none of the UAVs are alive, then $R_l$ is $-1$.

Total Reward

The total reward is the reward given to each UAV but can be calculated at the base if and when the UAVs communicate to the base. We add all the above normalized rewards and train our DRL agent at the base based on the total reward, and this is communicated to each UAV.

$$R = R_{tp} + R_{mp} + R_{td} + R_{md} + R_{\tau} + R_l \quad (12)$$

The frequency of communication with the base is algorithm-specific in GLIDE and is discussed in detail in the next section. The DRL agent is unaware of the environment beforehand and needs to interact with the dynamic environment. It uses its experience of interacting and observing the rewards for various decisions in different states and then learns the optimal policy $\pi^*$ to map the states with their best actions. In a military environment, each UAV has to participate in the task completion process and the number of UAVs is chosen based on the number of targets and area of the field. If a UAV is destroyed by the mine during the episode, it is penalized by the liveliness reward. The subsequent action space would not include the nonalive UAVs.

4. DRL-Based UAV Action Control

The main objective of the DRL agent is to find the best mapping between the state and the action that achieves a maximum discounted average reward. In this work, we leverage the PPO algorithm, which is a policy gradient algorithm. Although conventional policy-gradient algorithms, such as advantage actor–critic (A2C) [26], have successfully produced strong control effects in a variety of decision-making situations, they continue to encounter a number of issues, such as the challenging choice of iteration step size and poor data usage. Hence, a model-free PPO algorithm, which is a state-of-the-art DRL algorithm based on the actor–critic method, is chosen for our approach.

The actor model is tasked with learning the optimal action based on the observed state of the environment. In our scenario, the state serves as input, and the output is the UAV’s movement, represented as an action. The actor model produces an acceleration value that signifies the Q-value associated with the action taken in the preceding state. Conversely, the critic model is designed to assess the actor’s actions and provide feedback. By comparing the critic’s policy with its own, the actor can adjust its policy, aiming for enhancements in decision-making. PPO is centered on enhancing the stability of policy training by constraining the extent of policy changes in each training epoch. This approach minimizes the occurrence of substantial policy updates. Consequently, the policy is updated by evaluating the difference between the current and former policies, utilizing a ratio calculation to quantify the policy change.

The information structure in MARL is more intricate compared to the single-agent context. In this study, agents do not engage in direct information exchange. Instead, each agent formulates decisions by relying on its individual observations and information conveyed by the base. The local observations differ among agents and might encompass global details, such as the collective actions of other agents, owing to the information-sharing structure at the base. We propose two PPO-based multi-agent algorithms, called
continuous centralized GLIDE (C-GLIDE) and continuous decentralized GLIDE (D-GLIDE). Figure 3 provides an illustration of DRL agents’ interaction with the environment.

![Figure 3](https://via.placeholder.com/150)

**Figure 3.** An illustration of DRL agents’ interaction with the environment.

The current state $s_t^i$ of a UAV $i$ is given as input to the actor network and the distribution of actions is obtained. Each agent chooses its action $a_t^i$. The environment gives each agent a reward $r_t^i$ for the action taken, indicating whether the action is beneficial and this reward is used by the critic network. The actor network generates the policy, and the critic network evaluates the current policy by estimating the advantage function $A_t^i$.

$$A_t^i = Q_\pi(s_t, a_t) - V_\pi(s_t)$$ (13)

The policy is modified according to the advantage function. Figure 4 demonstrates the agent environment interaction of a single agent of the PPO algorithm with an actor network and a critic network. During training, a batch of samples is selected from the buffer to update network parameters. In order to improve the sampling efficiency, PPO adopts the importance sampling method. The probability ratio between old policy $\pi_{\theta_{old}}$ and new policy $\pi_{\theta}$ is denoted by $r_t(\theta)$ and is expressed as

$$r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$ (14)

PPO has to restrict the policy’s updating window and uses the clip method to directly limit the update range to $[1 - \epsilon, 1 + \epsilon]$. The loss function of PPO is

$$\mathcal{L}_{clip}(\theta) = \mathbb{E}(\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t))$$ (15)

where $\epsilon$ is a hyper-parameter. The objective function in Equation (15) only uses the policy of the actor network. For the critic network update, another term that uses the value function is needed. The centralized critic networks with weights $\phi$ are at the base. This centralized state consists of a fully observable environment state, which is common to all the agents, and also each agent’s partial local observation. The agent’s behavior optimizes the critic loss $L_{VF}(\phi)$ based on the squared difference between the actual return and the value estimation. The critic estimation bootstraps the next state’s expected value at the end of a mini-batch of size $NT$. This continues until the episode terminates. To minimize the difference between the estimated value and the actual value, the squared loss is used.

$$L_{VF}(\phi) = (V_{\phi} - V_{\text{Target}})^2$$ (16)

An entropy term is added to the objective to encourage exploration. Then, the final objective function for the critic network is
\[ L(\phi) = L_{\text{clip}}(\theta) - c_1 L_{\text{VF}}(\phi) + c_2 S[\pi_\theta(s_t)] \]  

(17)

\( L(\phi) \) is calculated at the base, which is the critic network. In multi-agent DRL, each agent learns by interacting with the environment. The agents learn and optimize their behavior by observing a state \( s_{i,t} \in S \), and each performs an action \( a_{i,t} \in A \) at time \( t \) and receives a reward \( r_i(s_{i,t}, a_{i,t}) \in \mathbb{R} \). The agent goes to the next state \( s_{i,t+1} \). For each agent, the goal is to learn a policy that maximizes their reward. The centralized and decentralized action control of the agents is discussed in detail in the next subsection.

Figure 4. An illustration of each interaction with the environment.

4.1. C-GLIDE

In a multi-agent system, each agent’s reward depends on both its own actions and those of the other agents. It would be challenging to guarantee the algorithm’s convergence since altering one agent’s policy will have an impact on how other agents choose their optimal course of action and cause erroneous value function estimates. This research adopts a MARL approach to address the issue, as shown in Figure 3. In C-GLIDE, the action control of the UAV fleet is executed centrally. The concept of centralized training and distributed execution in D-GLIDE.

In the continuous centralized GLIDE approach, all the UAVs’ action is considered as a single agent and is controlled by the base. The state space \( s \) is a combined input of all the UAVs’ current positions and the detected mines and the locations of undestroyed targets. The core value function is learned by the critic network. The combined agent in the actor network simply uses its observations to determine the policy. A multi-agent algorithm introduces a scalability issue. Each agent needs to take into consideration the joint action space, whose dimension grows exponentially with the number of agents, in order to address non-stationarity. So, with an increasing number of UAVs, the joint action space grows exponentially. The theoretical study of MARL is complicated by the presence of many agents, particularly the convergence analysis [16]. An approach to addressing scalability is to employ the mean-field regime with a large number of homogeneous agents [42,43]. In mean-field reinforcement learning [43], the interactions within the population of agents are approximated by those between a single agent and the average effect from all the agents. The interplay between the two entities is mutually reinforced. Thus, the impact of each agent on the overall multi-agent system might diminish to the point where all agents are interchangeable or indistinguishable. However, a mean-field quantity, such as the average state or the empirical distribution of states, can accurately describe the interaction with other agents. This greatly simplifies the study because each agent merely needs to discover its optimal mean-field response. Our model considers each UAV to share a common reward function depending only on the local state and the mean field, which encourages cooperation among the agents.
In C-GLIDE in Algorithm 1, the actor network and the critic network are initialized with weights $\theta_0$ and $\phi_0$, respectively, along with a clipping threshold $\epsilon$. The experiment is run for $M$ episodes. In C-GLIDE, all the UAVs have aggregated state and aggregated action. There are $N$ UAVs, so each UAV has an actor network and all the actor networks follow the old policy for $T$ timesteps. The actors collect $D_k$ sets of trajectories with the central actor policy. During each time step, the base updates the UAVs with the current state $s_{t-1}$, and the best action $a^*_t$ is taken by each UAV toward the target while avoiding the mines. If a mine is detected, its location is updated to the base. The rewards and the advantage function are calculated, and the state is updated to the next state. Once every $K$ timesteps, the critic network parameters are updated based on the experience of all the actors stored in a local buffer. The detailed steps of the C-GLIDE algorithm are presented in Algorithm 1.

Algorithm 1 C-GLIDE
1: Initial actor network parameters $\theta_0$, critic network parameters $\phi_0$ and clipping threshold $\epsilon$
2: for episodes $1, 2, \ldots, M$ do
3: for actor $1, 2, \ldots, N$ do
4: Follow central actor policy $\pi_{\theta_{old}}$ for $T$ timesteps
5: Collect $D_k$ set of trajectories
6: for epoch $t := 1, \ldots, T$ do
7: Base updates all UAVs with state $s_{t-1}$
8: if No obstacle in detection range then
9: Take action $a^*_t$ based on $\pi_{\theta_{old}}$
10: else if Mine detected then
11: Choose $a^*_t$ to avoid the mine
12: Update base with the mine location
13: end if
14: Update $s^t_i$ to next state $s^{t+1}_i$
15: Compute the rewards
16: Compute advantages $\hat{A}_1, \ldots, \hat{A}_T$ in (13)
17: end for
18: end for
19: if Timesteps == $K$ then
20: Update $\theta$ by SGD to maximize $\mathcal{L}_{clip}(\theta)$ in (15)
\[ \theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{clip}(\theta) \]
21: Update $\phi$ by SGD and minimize $\mathcal{L}(\phi)$ (17)
\[ \phi_{k+1} = \arg\min_{\phi} \frac{1}{|D|T} \sum_{t \in D} \sum_{t=0}^T (V_{\phi}(s_t) - \hat{R}_t)^2 \]
22: $\theta_{old} \leftarrow \theta$
23: end if
24: end for

4.2. D-GLIDE

D-GLIDE is based on centralized training and distributed execution. A core value function is learned by the critic network. It has access to data from the base, such as other agents’ location information and environmental information. Each agent in the actor network simply uses its own local observations to determine the policy. Assuming that every agent shares the same state space $S$, the algorithm allows any number of agents to be used in task execution. Instead of using the agent’s own value function, this method uses the critic network of each agent to fit the global value function. Only the agent’s policy has to be changed in order to optimize the global value function in this manner.
multi-agent policy gradient may also be obtained directly using the chain rule, similar to the
single-agent deterministic policy gradient. In a dynamic environment containing several
UAVs, for each UAV, \(i\), its observation value at time \(t\) is \(s_i^t\), and the actor network outputs
the mean and variance of the corresponding action probability distribution according
to \(s_i^t\) and then constructs a normal distribution sampling to obtain action. Through the
above methods, UAVs learn the cooperation policy between agents. During the execution
phase, it exclusively depends on its own local perspective to make decisions, resulting in
a collaborative policy that does not rely on communication. Additionally, UAVs with the
same purpose share the same actor network settings to lower the cost of network training.

In D-GLIDE in Algorithm 1, the actor network and the critic network are initialized
and the experiment is run for \(M\) episodes similar to C-GLIDE. In D-GLIDE, the actor
network is decentralized, and there is one critic network that is centralized. Using different
critic networks for each actor network will be noisy. So the training is centralized with one
critic network. Each UAV has an actor network that follows the old policy of collecting \(D_k\)
set of trajectories for \(T\) timesteps. The policy followed by each actor here is a distributed
policy, unlike the centralized policy in C-GLIDE. During each timestep, the base updates
the UAVs with the detected mines. The best action \(a_i^t\) is taken by each UAV towards the
target while avoiding the mines based on the distributed policy. The rewards and the
advantage function are calculated, and the state is updated to the next state. Once every \(K\)
timesteps, the local mini-batch experience samples are used to update the critic network
parameters based on the experience of all the actors. The algorithm procedure is shown in
Algorithm 2.

Algorithm 2 D-GLIDE

1: Initial actor network parameters \(\theta_0\), critic network parameters \(\phi_0\) and clipping threshold \(\epsilon\)
2: for episodes 1, 2, \cdots, \(M\) do
3:     for actor 1, 2, \cdots, \(N\) do
4:         Follow distributed actor policy \(\pi_{\theta,old}\) for \(T\)
5:         Collect \(D_k\) set of trajectories
6:     for epoch \(t := 1, \cdots, T\) do
7:         Base updates all UAVs the detected mines
8:         if No obstacle in detection range then
9:             Take action \(a_i^t\) based on \(\pi_{\theta,old}\) from state \(s_i^t\)
10:        else if Mine detected then
11:            Choose \(a_i^t\) to avoid the mine
12:            Update base with the mine location
13:        end if
14:         Update \(s_i^t\) to next state \(s_i^{t+1}\)
15:         Compute the rewards
16:         Compute advantages \(\hat{A}_1, \cdots, \hat{A}_T\) in (13)
17:     end for
18:     end for
19:     if Timesteps == \(K\) then
20:         Update \(\theta\) by SGD to maximize \(L_{clip}(\theta)\) in (15)
21:             \(\theta_{k+1} = \arg\max_\theta L_{clip}(\theta)\)
22:     end if
23:     Update \(\phi\) by SGD and minimize \(L(\phi)\) (17)
24:             \(\phi_{k+1} = \arg\min_\phi \frac{1}{|D|T} \sum_{\tau \in D} \sum_{t=0}^T (V_{\phi}(s_t) - \hat{R}_t)^2\)
25:     \(\theta_{old} \leftarrow \theta\)
26: end for
5. Results

In this section, the convergence analysis of the proposed C-GLIDE and D-GLIDE algorithms is presented. The D-GLIDE is analyzed for its effectiveness in various scenarios with respect to the C-GLIDE implementation. Then, a summary of the analysis is provided based on the impact of several hyper-parameters on both C-GLIDE and D-GLIDE algorithms.

5.1. Simulation Setting

We consider a field size of 1000 m $\times$ 1000 m within which the UAVs can fly up to a maximum altitude of 500 m. In this implementation, by default, there are four UAVs at the base and four mines that are placed at unknown locations in the ground as enemy defenses to protect four strategic targets from the attacks of the UAVs. Unless otherwise stated, the aforementioned scenario is considered as the default setting. The destruction range of the UAV and the mine and the detection range of the UAV, along with other environment-specific parameters, are mentioned in Table 3.

The proposed C-GLIDE and D-GLIDE algorithms are implemented on a workstation with a 48 core Intel Xeon CPU, 32 GB DDR4 primary memory, and 4 $\times$ Nvidia RTX 2080 Ti with 12 GB video memory for running CUDA-accelerated Tensorflow. The GPUs are interlinked by the NVlink interface and communicate with the CPU by the PCIe $\times$ 16.0. We used software packages based on Python3.8.12, Tensorflow v1.14.0, stable-baselines v2.10.2, and gym v0.19.0 to implement the C-GLIDE and D-GLIDE algorithms. The hyper-parameter values are described in Table 4.

### Table 3. Environmental parameters.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length $l$</td>
<td>1000 m</td>
</tr>
<tr>
<td>Breadth $b$</td>
<td>1000 m</td>
</tr>
<tr>
<td>Height $h$</td>
<td>500 m</td>
</tr>
<tr>
<td>Height of the Target $h_T$</td>
<td>40 m</td>
</tr>
<tr>
<td>Radar range of the mine $D_1$</td>
<td>50 m</td>
</tr>
<tr>
<td>Detection range of UAV $d_1$</td>
<td>300 m</td>
</tr>
<tr>
<td>Destruction range of UAV $d_2$</td>
<td>100 m</td>
</tr>
<tr>
<td>Maximum speed of the UAV $v_{max}$</td>
<td>90 m/s</td>
</tr>
<tr>
<td>Maximum acceleration of the UAV $a_{max}$</td>
<td>50 m/s²</td>
</tr>
</tbody>
</table>

### Table 4. PPO Hyper-parameters.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neurons in hidden layer 1 $H_1$</td>
<td>256</td>
</tr>
<tr>
<td>Neurons in hidden layer 2 $H_2$</td>
<td>256</td>
</tr>
<tr>
<td>Relay memory size $</td>
<td>B</td>
</tr>
<tr>
<td>Minibatch size $</td>
<td>b_m</td>
</tr>
<tr>
<td>Learning rate $\alpha$</td>
<td>$2.5 \times 10^{-4}$</td>
</tr>
<tr>
<td>Discount factor $\gamma$</td>
<td>0.99</td>
</tr>
<tr>
<td>GAE parameter $\lambda$</td>
<td>0.95</td>
</tr>
<tr>
<td>Activation function</td>
<td>ReLU</td>
</tr>
<tr>
<td>Clip range $\epsilon$</td>
<td>0.2</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>Epochs $\eta$</td>
<td>10</td>
</tr>
<tr>
<td>Total episodes $\mathcal{E}$</td>
<td>$5 \times 10^5$</td>
</tr>
</tbody>
</table>
5.2. Convergence Analysis

In this subsection, a comparison between the proposed D-GLIDE and C-GLIDE is presented to highlight the benefits of decentralization. In Figure 5, the episode reward stabilizes as the D-GLIDE agent interacts with the environment for around 25 million time slots. However, that is not the case for the C-GLIDE algorithm in all scenarios. We observe that both algorithms perform well when there are fewer enemy defenses or strategic targets to destroy, as shown in Figure 5a,c, because the scope of collaboration among individual UAVs is limited and less essential to destroy all targets. Thus, the UAVs do not necessarily have personalized strategies, enabling the C-GLIDE algorithm to comprehend a general policy.

![Figure 5](attachment:image.png)

Figure 5. Convergence of the C-GLIDE and D-GLIDE algorithm under various settings—(a) 1 target; (b) 6 targets; (c) 1 mine; (d) 6 mines; (e) 1 UAV; (f) 6 UAVs—where other objects are as per the default scenario.

However, as the number of mines or targets grows, the more the UAVs have to work together and divide the undertaken task among themselves to achieve the objective in a minimum time, which requires personalized learning in the UAVs. Therefore, D-GLIDE learning helps in such cases as each UAV can optimize its strategy without affecting the others. We show the benefits of decentralization in Figure 5b,d where the C-GLIDE implementation cannot facilitate personalized UAV behavior due to a common set of hidden layers and underperforms compared to the distributed implementation. Furthermore, we observe from Figure 5e,f that increasing the number of UAVs in the environment reduces the complexity of the resultant strategy due to redundant UAVs. Hence, both algorithms perform equivalently as we increase the number of UAVs beyond the number of targets or mines. Please note that C-GLIDE learning becomes challenging as we increase the environment’s targets, mines, and UAVs.

In summary, the D-GLIDE algorithm scales well with the increasing number of objects in the environment and encourages UAV-specific policy optimization for improved performance in contrast to the C-GLIDE implementation.

5.3. Effectiveness Analysis

In this subsection, we discuss the effectiveness of the proposed solution in various scenarios and analyze the scalability of the system with an increasing number of targets, mines, and UAVs in the simulated area of operation.
5.3.1. Increasing Targets

Here, we analyze the effect of increasing the number of targets from one to six, keeping the mines and UAVs constant as per our default scenario. From Figure 6a, we observe that both the distributed and C-GLIDE algorithm destroys all the targets in the environment with four deployed UAVs; thus, the algorithms are scalable with the number of targets in the environment. However, a significantly lower number of mines are triggered by the UAVs in the case of the distributed algorithm as compared to the C-GLIDE; see Figure 6b. As a result, UAVs are less intercepted, and therefore, more UAVs are alive at the end of the episode, as shown in Figure 6c.

![Graphs showing correlation between total targets and destroyed targets, live UAVs, and wall time.](image)

**Figure 6.** Correlation among increasing numbers of strategic targets versus (a) destroyed targets; (b) destroyed mines; (c) live UAVs; (d) wall time, or time taken at the end of each episode.

With increasing targets, the UAVs need to collaborate effectively among themselves and learn both group and individual skills to destroy all targets in the minimum amount of time. The C-GLIDE algorithm suffers from increasing targets as it has common hidden layers to process the input state of the environment, limiting the scope of individual skill adoption by the UAVs. However, in the distributed algorithm, each UAV optimizes a separate neural network, and thus the UAVs do not interfere with each other’s learning, enabling a higher degree of personalizing. We further observe a steady increase in the time taken to destroy all targets with the increase in the number of targets as the number of UAVs is fixed. However, the distributed algorithm outperforms the C-GLIDE and takes significantly less time to destroy the targets, as shown in Figure 6d.

In summary, the number of targets influences the degree of collaboration among the UAVs that is required to destroy all the targets in a minimum time and also increase the chance of UAV interception by the mines as the UAVs have to traverse more of the operational area to reach each target.

5.3.2. Increasing Enemy Mines

As the number of mines grows, the probability of UAV interception increases, which results in the destruction of all four UAVs from time to time. In Figure 7a, we observe
that some targets are missed as the number of mines in the environment represents more than four deployed UAVs. However, the median destroyed targets is very close to four, which shows the scalability of both the algorithms with increasing mines. The distributed algorithm outperforms the C-GLIDE by a slight margin in destroying targets. Moreover, the C-GLIDE algorithm performs poorly in terms of avoiding mines, resulting in UAVs getting destroyed during the operation, as compared to the D-GLIDE. This is depicted in Figure 7b.

Subsequently, for the distributed algorithm, more UAVs are alive after destroying all the targets. In Figure 7c, we see a downward trend in live UAVs as a number of mine increases. This is because the location of the mines is not known beforehand; thus the UAVs have to detect and dodge the mines dynamically during their lifetime. More mines mean there is a larger probability of UAV interception, resulting fewer alive UAVs after destroying all targets. Moreover, from Figure 7d, we observe a steady increase in wall time with more mines as it reduces the possible safe paths from a position and may force a UAV to take long alternate paths to reach the same destination, in contrast to the scenario with no mines. The distributed algorithm takes significantly less time to destroy all the targets in the environment.

In short, increasing the mines increases the hardness of the joint path planning of the UAVs as there will be fewer safe trajectories for each UAV. However, the distributed algorithm scales pretty effectively with four deployed UAVs with enemy mines from one to six.

5.3.3. Increasing UAVs

Similarly in this subsection, the number of UAVs is varied in the simulated area of operation from one to six, keeping mines and targets constant as per the default scenario. We observe in Figure 8a that some targets may remain alive with one UAV, where all targets are destroyed every time with six UAVs. Hence, destroying targets becomes easier as we increase the number of UAVs in the environment. However, the probability of being
intercepted by the mines increases with more UAVs while traversing the area of operation. Therefore, more mines are destroyed, as shown in Figure 8b.

The C-GLIDE algorithm suffers from inadequate capacity to facilitate personalized UAV behavior and thus performs relatively badly compared to the distributed algorithm in avoiding interception by the mines. As a result, fewer UAVs are alive after destroying the targets; see Figure 8c for more details. Interestingly, from Figure 8d, we see a steady decrease in the wall time or time required to destroy the targets with an increase in the number of UAVs. The UAVs must learn group strategy along with individual skills to reduce the wall time (e.g., as a group, they have to explore the operational area to discover the hidden mines, and thereafter, individual UAV must select an appropriate target and move in the shortest safe trajectory to reach that target, resulting in the destruction of all targets in minimum time). The D-GLIDE algorithm adapts more group and personalized skills over time. It outperforms the C-GLIDE algorithm in terms of the time taken to destroy the targets.

In summary, increasing the number of UAVs helps in destroying strategic targets; however, the UAVs must collaborate effectively in order to avoid mines and divide the tasks among themselves for completion in a minimum amount of time.

![Figure 8.](image)

5.3.4. Exploring the Area of Operation

In this subsection, we analyze the behavior of the D-GLIDE and the C-GLIDE algorithm in exploring the operational area to find hidden mines with an increasing number of UAVs, targets, and mines individually, keeping others as per the default scenario. As seen in Figure 9a,b, fewer mines are discovered by the UAVs as the number of targets decreases. Therefore, both the algorithms can prioritize destroying the targets in minimal time (see Figure 6d) when the target count is lower and compromise on discovering the mines, resulting in less exploration over the area of operation. However, as the number of targets increases, the UAVs take more time and discover the hidden mines to accomplish the safest path planning to destroy the targets.
We vary the number of mines in the area of operation and observe that the UAVs find most of the mines before destroying the targets. Discovering the mines becomes crucial as the mines are increased; otherwise the UAVs may be intercepted. Next, we increase the number of UAVs and observe that most of the mines are discovered during the operation. In summary, exploring the operational area is crucial in discovering the mines and therefore helps the UAVs to plan the safest paths to the targets and destroy them in minimal time.

![Figure 9. Exploring the operational area to find mines with an increasing number of targets, mines, and UAVs—(a) D-GLIDE (b) C-GLIDE.](image)

6. Conclusions

In this work, we discussed a DRL-based approach for the action control of a group of UAVs. We used MARL for the collaborative movement of the UAV fleet in a dynamic military environment. We proposed two MARL algorithms called C-GLIDE and D-GLIDE. We developed a simulator called UAV SIM, in which the mines are placed at random locations unknown to the UAVs at the beginning of each episode. The performance of the proposed scheme is presented, and the results show that the D-GLIDE approach significantly outperforms the C-GLIDE approach.

We acknowledge multiple limitations of this work for direct applications, which are the subject of future work, including modeling the full complexity of real military environments, complex battery models, and varying communication speeds between UAVs and the central base.

**Author Contributions:** Methodology: D.S.G., P.K., V.K.S., V.A.; investigation: D.S.G., P.K.; writing—original draft preparation: D.S.G., P.K.; writing—review and editing: V.K.S., V.A.; supervision: V.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** D.S. Gadiraju’s work was supported in part by the Science and Engineering Research Board of India via the Overseas Visiting Doctoral Fellowship. This work was performed when D.S. Gadiraju was an Overseas Visiting Doctoral Fellow at Purdue University. V. Aggarwal acknowledges research award from Cisco, Inc.

**Data Availability Statement:** Simulator is provided at Github. The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

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

**References**


---

*Information 2024, 15, 477*


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.