


## Article

# Multi-USV Dynamic Navigation and Target Capture: A Guided Multi-Agent Reinforcement Learning Approach

Sulemana Nantogma <sup>1</sup>, Shangyan Zhang <sup>1</sup>, Xuewei Yu <sup>2,3</sup>, Xuyang An <sup>2</sup> and Yang Xu <sup>1,\*</sup>

<sup>1</sup> School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

<sup>2</sup> China North Artificial Intelligence & Innovation Research Institute, Beijing 100072, China

<sup>3</sup> College of Artificial Intelligence, Nankai University, Tianjin 300350, China

\* Correspondence: xuyang@uestc.edu.cn

**Abstract:** Autonomous unmanned systems have become an attractive vehicle for a myriad of military and civilian applications. This can be partly attributed to their ability to bring payloads for utility, sensing, and other uses for various applications autonomously. However, a key challenge in realizing autonomous unmanned systems is the ability to perform complex group missions, which require coordination and collaboration among multiple platforms. This paper presents a cooperative navigating task approach that enables multiple unmanned surface vehicles (multi-USV) to autonomously capture a maneuvering target while avoiding both static and dynamic obstacles. The approach adopts a hybrid multi-agent deep reinforcement learning framework that leverages heuristic mechanisms to guide the group mission learning of the vehicles. Specifically, the proposed framework consists of two stages. In the first stage, navigation subgoal sets are generated based on expert knowledge, and a goal selection heuristic model based on the immune network model is used to select navigation targets during training. Next, the selected goals' executions are learned using actor-critic proximal policy optimization. The simulation results with multi-USV target capture show that the proposed approach is capable of abstracting and guiding the unmanned vehicle group coordination learning and achieving a generally optimized mission execution.



**Citation:** Nantogma, S.; Zhang, S.; Yu, X.; An, X.; Xu, Y. Multi-USV Dynamic Navigation and Target Capture: A Guided Multi-Agent Reinforcement Learning Approach. *Electronics* **2023**, *12*, 1523. <https://doi.org/10.3390/electronics12071523>

Academic Editor: Mahmut Reyhanoglu

Received: 8 February 2023

Revised: 18 March 2023

Accepted: 20 March 2023

Published: 23 March 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Keywords:** multi-agent reinforcement learning; multi-USV system; cooperative control; target capture; deep RL; unmanned systems

## 1. Introduction

The important technological developments in unmanned vehicles have greatly influenced the number of real-world applications that are suitable for unmanned vehicle deployment. Hence, in the last decade, the application of autonomous unmanned vehicles in military and civilian operations, such as search and rescue, patrolling, monitoring, and surveillance tasks, has witnessed a tremendous increase [1–6]. One key challenge in the effective deployment of autonomous unmanned vehicles is the ability to perform complex group missions, which require coordination and collaboration among multiple vehicles. On the other hand, many real-world tasks require a group of agents to work together. Compared with a single agent, multi-agent systems have the advantages of strong adaptability and fault tolerance. In the robotics and autonomous unmanned vehicles contexts, a single platform or controller of an unmanned vehicle can be regarded as an agent. Through division and cooperation, each platform can perceive the environment and acquire information about the task area quickly and more accurately to accelerate the completion of missions and improve the efficiency of the group.

Unmanned vehicles can be categorized into four types based on their mode of operation. These include the unmanned aerial vehicle (UAV), unmanned ground vehicle (UGV), unmanned surface vehicle (USV), and autonomous underwater vehicle (AUV) [7].

In the last few decades, the robotics community has extended its interest to the study of cooperative decision making and controlling unmanned vehicles. A group of unmanned vehicles working together in a shared environment can be regarded as a multi-agent system. In cooperative multi-agent systems, a collective scheme is required to enable the group of agents to work together to complete a common task and coordinate in time and space.

Traditionally, group mission strategies for multi-robot systems were created using predefined rules and algorithms [8–10], but these methods can be rigid and not easily adaptable to changes in the environment or mission objectives. Consequently, there has been a growing interest in using reinforcement learning techniques that enable multiple robots to learn group mission strategies in a more flexible and adaptable way. To achieve coordination in autonomous multi-robot systems across different application domains, various control architectures have been proposed [11,12]. Deep reinforcement learning (DRL) has emerged as a promising technique in this area [13,14]. DRL has been integrated with multi-agent systems (MAS) to create multi-agent DRL, which has been extensively studied in the literature. For example, the multi-agent deep deterministic policy gradient (MADDPG) [15] has been developed as a centralized training framework with decentralized execution to enable multi-agent teams to cooperate or compete during task execution. The multi-agent hierarchical deep deterministic policy gradient has been introduced as an extension of the MADDPG to address search and exploration problems with numerous robotic agents [16]. Additionally, proximal policy optimization (PPO) and its variants have been proposed and applied to various group missions to generate sophisticated behaviors [17–21]. These and similar frameworks and approaches have been utilized to tackle the challenge of cooperative mission execution with multi-agent systems, such as the multi-agent pursue evasion (MPE) problems, using unmanned vehicles [17,22–24].

The MPE problem is commonly known as target encirclement and capture. When dealing with MPE environments, the targets are evasive and try to avoid capture. They can be faster than the pursuers, and multiple agents are employed to capture an intruder or evader(s) who attempt to evade capture. The main goal of the MPE is to minimize the time required to capture the evasive intruder or target. Due to its practical significance in military applications, such as missile avoidance and interception, surveillance, criminal pursuit, reconnaissance, and rescue operations, several variations of MPE games have been extensively studied [25–29].

Several approaches in the literature assume unstructured environments. For example, the authors in [30] proposed a region-based relay pursuit scheme for pursuers to capture a single evader that adopts the pure evasion strategy. An RL-based method that employs the MADDPG algorithm to realize a pursuit strategy for the MPE game was considered in [31]. However, this approach is not suitable with an increased number of agents. A classical single pursuer and single evader pursuit-evasion game based on Apollonius partitions with Q-learning was studied in [32]. A decentralized learning approach [33] was explored where the pursuers were modeled as learning automata, while a new fuzzy reinforcement learning algorithm was proposed in [23] to solve the MPE with a single superior evader. Additionally, the authors in [34] studied the MPE from the view of the behavior-based control method and adopted a motor schema-based reactive control architecture based on the Apollonius circle to solve the MPE differential game with a superior evader.

In contrast, while some works tackle the MPE problem in a more general form, other studies consider specific applications using unmanned systems with DRL and other frameworks. For instance, a cooperative pursuit of unauthorized UAVs in urban airspace via MARL was presented in [35], Zhang et al. [36] focused on the pursuit-evasion game of a multiquadcopter in an obstacle environment with online motion planning by deep reinforcement learning, an approach using multiple autonomous underwater vehicles in 3D water volumes with and without obstacles was studied in [37], and Wei et al. [29] designed an unmanned target hunting system for a swarm of unmanned underwater vehicles (UUVs) to hunt a target with high maneuverability, while a collaborative strategy of a UAV/UGV heterogeneous system was proposed to derive a pursuit-evasion game

in a complex three-dimensional environment in [38]. In addition, a distributed, partially observable multi-target hunting proximal policy optimization (DPOMH-PPO) algorithm for multi-unmanned surface vehicle (USV) target hunting was proposed in [17], and a DRL approach for multi-vehicle pursuit in urban environments was presented in [24]. The authors in [39] considered a decentralized DRL method for pursuing an omnidirectional target with multiple agents, while a method for enabling a multi-robot system to encircle a target and avoid collisions was studied in [40]. Another proposed approach is a distributed transferable policy network framework based on DRL which employs a graph communication mechanism for robotic agents' interaction to tackle the problem of single-target capture with collision avoidance [41].

However, some of these approaches may not be effective in cases where the target has a strong escape strategy, as they do not consider predicting the target's movement trajectories. Additionally, in many DRL approaches, each agent receives information in a lumped vector without leveraging human knowledge of the target or the natural spatial structures of the environments. This is important because while multi-agent RL can be a very powerful tool, especially for achieving autonomous cooperative behavior of unmanned systems, several challenges exist when utilizing RL in the multi-vehicle cooperative capture of mobile targets in complicated multi-intersection and restricted moving spaces.

The first issue has to do with the state of agents and its effects on learning convergence. It is usually unrealistic to assume a completely observable and noise-free state. This means that aside from the fact that a learning agent might not be able to determine accurately its state, convergence becomes an issue. Additionally, a significantly different state might appear to be very similar. Hence, to facilitate the training and learning of cooperation for multi-robot group missions, designers can use filters of human guidance to help agents estimate the true state of the vehicle or to guide the learning process of the group.

This paper presents a human-guided environment transition model approach for coordination training in an environment where the adversaries or task transition model can be observed and modeled. The key idea in this approach is to utilize human knowledge of the observed mission environment to generate a transition model that can be initialized to guide the overall coordination of the multi-agent system in different situations by generating goals that will lead to a successful mission completion of unmanned vehicle groups. More specifically, the contributions of this paper include the following:

- A novel formulation of the MPE problem as a multi-USV operation in a floating city environment that can be applied in target pursuit in multi-intersection urban areas.
- We develop a hybrid deep reinforcement learning framework that leverages heuristic mechanisms to guide group mission learning of multiple unmanned surface vehicles. The proposed framework consists of subgoal generation in the form of navigation targets based on the knowledge of the environment, and a goal selection heuristic model based on immune network models is used to select subgoals for execution by an actor-critic proximal policy optimization model.
- A simulation experiment is conducted to analyze the performance of the proposed framework in realizing the target-capturing behavior of a multi-USV group.

The rest of the paper is organized as follows. Section 2 introduces some key preliminaries related to the MARL model and algorithm. Section 3 describes the multi-agent USV cooperative problem and the target decision model. Next, the proposed approach is presented in Section 4, followed by the experiment and results in Section 5.

## 2. Preliminaries

### 2.1. Multi-Agent Observation Models

The observation and states of agents in decentralized multi-agent systems are localized. In this case, the interaction between agents is defined by a graph  $\mathcal{G} = (V, E)$ , where  $V = \{v_1, v_2, \dots, v_N\}$  is the set of agents and  $E$  is the set of edges representing agent interactions. This interaction  $E \subset V \times V$  is represented by undirected edges in the form of

$\{v_i, v_j\}$ , where agents  $i$  and  $j$  are neighbors. Hence, the set of neighbors of agent  $i$  within which agent  $i$  can sense local information about others is given by

$$N_i(\mathcal{G}) = \{j | \{v_i, v_j\} \in E\} \quad (1)$$

If the set of nodes or edges changes over time, then  $\mathcal{G}$  can be regarded as a dynamic graph. The information shared by agent  $i$  to agent  $j$  is a function of the states of agent  $i$  about agent  $j$ , namely  $o_{ij} = f(s_i, s_j)$ , where  $o_{ij} \in O_i$  and  $s_i, s_j \in S_i$ . Here,  $O_i$  is the complete information set agent  $i$  receives from all its neighbors, and  $S_i$  is the set of states of all neighbor agents of  $i$ . The observation  $o_{ij}$  is only available to agent  $i$  if agent  $j$  is in the neighborhood of agent  $i$ .

## 2.2. Partially Observable Stochastic Games

Multi-agent decision making can generally be classified as partially observable stochastic games (POSGs) since the agents have incomplete or noisy observations. Thus, MARL in multi-robot cooperative tasks, such as target capture, can be modeled as partially observable. The POSG is an extension of the stochastic Markov decision process (MDP), where rewards or punishments and transitions between agents' states are determined by the agents' joint actions, regardless of the possible conflicting goals of agents. In POSGs, partial observation of agents' states is assumed.

**Theorem 1.** A POSG is a tuple  $\langle S, A_1 \dots A_n, O_1 \dots O_n, f, R_1 \dots R_n \rangle$  where the following apply:

- $n$  is the number of agents (controller agents or USVs);
- $S$  is the finite set of states;
- $A_i, i = 1, \dots, n$  are the finite set of actions available to the agents which form the joint action set  $\mathbf{A} = A_1 \times A_2 \times \dots \times A_n$ ;
- $O_i, i = 1, \dots, n$  are the finite set of observations of the agents. The joint observation is denoted as  $\mathbf{o} = \{o_1, \dots, o_n\}$ ;
- $f : S \times \mathbf{A} \times S' \rightarrow [0, 1]$  is the Markovian state transition function, where  $f(s'|s, \mathbf{a})$  denotes the probability of reaching state  $s'$  after the joint action  $\mathbf{a}$  in state  $s$ . The joint action of all agents  $\mathbf{a}$  at a stage  $t$  is denoted as  $\mathbf{a}_t = [a_{1,t}, \dots, a_{n,t}]^T$   $\mathbf{a}_t \in \mathbf{A}, a_{i,t} \in A_i$ ;
- $R_i : S \times \mathbf{A} \rightarrow \mathbb{R}, i = 1, \dots, n$  are the reward functions of the agents.

A reward of agent  $i$  after taking action  $a_{i,t} \in A_i$  in stage  $t$  toward completing task  $c_i$  is denoted as  $r_{i,t+1}$ . The individual controller  $h_i : S \times A_i \rightarrow [0, 1]$  of the agents forms the joint coordination controller (policy)  $\mathbf{h}$ . Since the reward of each agent in a team depends on the joint action, the respective agents' rewards depend on the joint controller:

$$R_i^h(s) = \mathbf{E} \left[ \sum_{t=0}^T \gamma^t r_{i,t+1} | s_0 = s, \mathbf{h} \right] \quad (2)$$

where  $\gamma$  is the discounting factor and  $T$  is the length of the horizon. Because each agent has its own reward function that shows the agent's preferences, POSGs model multi-agent systems that are self-interested. The multi-agent system becomes cooperative when all agents have the same reward function preferences ( $\forall_{i,j} R_i = R_j$ ). In this case, the multi-agent system is modeled as a decentralized, partially observable Markov decision process (Dec-POMDP) [42].

## Proximal Policy Optimization

In recent years, research on policy gradient methods has produced algorithms such as trust region policy optimization [43] methods that show state-of-the-art performance over Q-learning. Among the core algorithms in the policy gradient and actor-critic field is the proximal policy optimization (PPO) algorithm [44,45]. PPO is a minimal version of the trust region policy optimization algorithm that aims to solve the problem of low data utilization of traditional gradient policy algorithms with simpler implementation, execution, and sampling methods. Both trust region policy optimization and PPO use the

actor-critic structure, which combines the pros of traditional value-based and policy-based approaches. The actor-critic model employs two deep neural networks: the actor for taking the action and the critic for handling rewards. Specifically, to promote small policy updates, PPO proposes a clipped surrogate loss function and combines the policy surrogate and a value function error term as shown in Equation (3) [44].

$$L^{CLIP+V+S}(\theta) = \hat{\mathbb{E}}[L^{CLIP}(\theta) - \chi_1 L^V(\theta) + \chi_2 S\pi_\theta] \quad (3)$$

where  $L^{CLIP}$  is a clipped surrogate objective put forward in [44] as shown in Equation (4). Moreover,  $\chi_1$  and  $\chi_2$  are coefficients,  $L^V$  constitutes the squared error loss of the value function, and  $S$  denotes an entropy loss:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}[\min(rt(\theta)\hat{A}_t, \text{clip}(rt(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)] \quad (4)$$

In Equation (4),  $\epsilon$  and  $r_t(\theta)$  denote a hyperparameter and the probability ratio, respectively, where  $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ . This clips the probability ratio  $r$  at  $1 + \epsilon$  or  $1 - \epsilon$  for a positive or negative advantage, respectively, forming the clipped objective after multiplying the advantage approximator  $\hat{A}^t$  [46].

### 2.3. The Artificial Immune Network

The immune network theory is a critical theory of the artificial immune system that exhibits characteristics of adaptivity. The immune network hypothesis, proposed by Jerne [47], is based on Burnet's clonal selection theory. The hypothesis suggests that different types of antibodies and lymphocytes communicate with each other and maintain a stable network equilibrium through an interaction mechanism between network cells. This network is self-regulating through stimulatory and suppressive interactions among B cells, antigen epitopes, antibody idiotope, and antibody paratope. In the robotics domain, a computational model of Jerne's idiotypic network theory was proposed in [48] as a means of inducing adaptive behavior mediation, and it demonstrated some encouraging results.

## 3. Problem Formulation

### 3.1. Description of the Multi-USV Target Capture

We consider a problem where a team of unmanned surface vehicles is deployed in an floating environment to provide internal security by monitoring the activities of other boats within the area. The multi-USV team detects and finds intruder boats and captures them by entrapment or chasing them out of the environment. This scenario is an extension of multi-agent region surveillance and pursuit-evasion, which have been studied in the literature. The environment consists of multiple intersection pathways for easy navigation of USVs in the floating environment.

We assume dual pathways that are narrow and that the USVs cannot make a U-turn on a pathway. We further assume that there are a few exits and entrance points into and out of the environment. Additionally, the environment contains civilian boats that go about their business. We consider the case where an intruder is to be detected and chased by the multi-USV team with the objective to capture or chase the intruder out of the boundary as fast as possible. Here, capture means the pursuing USVs cross the pathway of the intruder boat.

In this case, the intruder is assumed to know the positions and detection ranges of the multi-USV team. The intruder boat is also assumed to be faster and have higher maneuverability compared with the multi-USV team. As an important requirement, we develop a competitive intruder's decision-making strategy, which is further used to provide the initial policy for the multi-USV team. Moreover, the intruder boat becomes visible to some of the USVs in the multi-USV team at some time (i.e., some of the USVs detect and obtain the planar coordinates of the intruder boat after a certain moment in time). Figure 1 shows an example mission scenario of USVs (blue) performing the monitoring mission with an intruder boat (red) and civilian boats in the environment. This problem can be directly extended to multiple unmanned ground vehicle target captures in urban city settings.





**Figure 1.** Example floating market or community environment map. The rectangular shapes represent floating structures.

Furthermore, the USVs are equipped with various sensors, including GPS, radar, and communication equipment for intruder and other sensory detection, as well as communication capabilities. However, USVs have limited observation ability and can only detect or communicate within their sensing and communication ranges, denoted as  $R_s$  and  $C_r$ , respectively. The multi-USVs are trained to make their own navigation and maneuvering decisions in a decentralized way (i.e., each USV navigates independently and has information about the current coordinates of the USVs in its neighborhood and the intruder boat through sensing and communication). The exact position of the intruder boat is only known to the USVs that have it within their detection range  $R_s$ .

However, the other USVs in the team can acquire the last location through communication from other USVs. Here, the goal for the multi-USV team is to coordinate their movement to constrain the intruder boat such that its movement can be directed according to the direction the multi-USV team wants, leading to the multi-USV team catching the intruder or the intruder escaping through one of the exit points within the environment. In doing so, the multi-USV team must also avoid collisions with each other and other static structures within the environment. The intruder is said to be captured if it is trapped between two or more of the multi-USV teams. The objective is to train the multi-USV system to perform cooperative intruder boat detection and entrapment within the environment.

### 3.2. The Multi-USV Learning Formulation

Formally, we consider a multi-agent assignment Markov decision process for a team of agents (multi-USV) that must learn to cooperatively perform a given navigation task  $j$  in an adversarial environment  $\mathcal{D} \subset \mathbb{R}^n$ . In executing the task, the agent team selects a navigation destination or goal from a set ( $\mathcal{C}(j) = \{c_1, c_2, c_3, \dots, c_n\}$ ) that is generated as subtasks or goals toward executing  $j$ , based on an expert- or designer-modeled transition function. In this case, agents select and navigate to a location in a dynamic environment with collision avoidance and the minimum time cost. The team MDP can be defined as  $\langle N, O, \mathbf{A}, P, R \rangle$ , where  $N$  is the number of agents,  $\mathbf{A}$  is the set of actions,  $O$  is the global environment observation made up of the local states  $S_N$  of the agents and a goal map, and  $P: S_N \times A_N \times S'_N \rightarrow [0, 1]$  is the transition probability model that enables the transition from  $S_N$  to  $S'_N$ , driven by the joint actions taken by the team  $A_N$ . The navigation targets can be dynamic at different times during the mission due to the dynamics of the environment and task. In this formulation, the actions toward achieving a goal are of the same speed and direction such that  $a_t = [v_t, v_r]$ , where  $a_t$  is the action taken at time  $t$ .

Each agent uses the same policy  $\mu : \mathbb{H} \rightarrow \mathbb{A}$  defined for each subgoal (in this case, a navigation policy), which maps local observations' histories  $\mathbb{H}$  and actions of agents to a new action  $a_t$  to be executed. Hence, agents can share their policy during training. If  $\eta$  is denoted as a horizon of an agent history space, then  $\mathbb{H} = S_n^\eta \times A_n^\eta$ , where  $A_n$  is the local actions of agent  $n$ . The reward function  $R$  of the team depends on the global state and all actions of the team (i.e.,  $R : S_N \times A_N \rightarrow R$ ). Even though this formulation focuses on navigation target selection, the proposed framework can be customized for any scenario where a set of goals can be defined to achieve the cooperative mission.

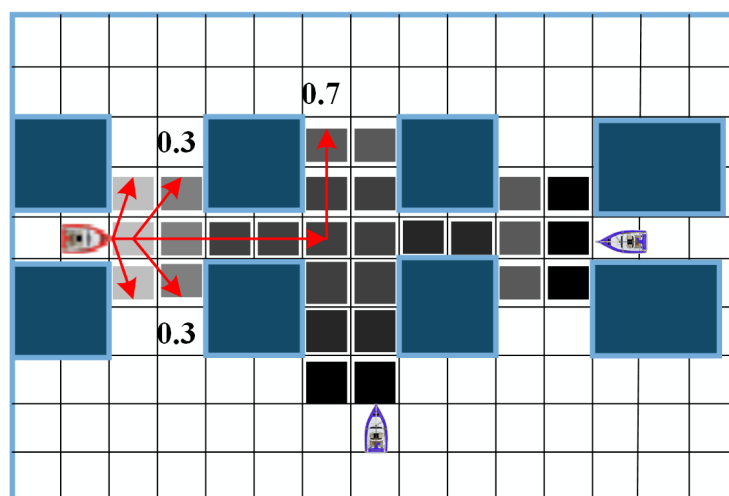
### Reward Function

The reward function provides guidance for the multi-USV system to achieve optimal task performance policies, which are implemented as the training objective. In order to optimize the performance of the multi-USV system, we consider the various situations that USVs will encounter in the mission environment when designing a reward function. Equation (5) shows the shared reward  $R_{USV_i}$  to be maximized by the multi-USV system at time  $t$ . Here,  $r_{dist,i}^t$  is the negative displacements between agent  $i$  and its selected task,  $t_{elapsed,i}^t$  is the time elapsed,  $\gamma$  is a bound constant, and  $c_{collide,i}^t$  is a discounted penalty if agent  $i$  collides, with a penalty factor of  $-0.5$ :

$$R_{USV} = \gamma \frac{1}{t_{elapsed}} + \sum_i^M (c_{collide,i}^t + r_{dist,i}^t) \quad (5)$$

### 3.3. Intruder Boat Policy Strategy

To design a human-competitive policy for the intruder, we developed a probability map model that the intruder used to make its maneuvering decisions. To apply the probability map model for intruder decision making, we divided the monitoring area into equally sized cells and associated each cell with the probability or likelihood of the intruder being captured if it were in that particular cell, thus forming the probability map of the entire monitoring area. Based on this, we designed navigation rules for the intruder boat with which it made its movement decisions. In addition to the strategy for exiting, the direction of movements of the intruder was biased toward less probable cells of capture, as shown in Figure 2. In this strategy, there is a 30% chance of the intruder trying to move out of the community through the closest exit point, with a 70% aim at staying in the environment and evading the multi-USV team. Cells closer to and between structures are riskier, depending on the location and direction of the multi-USV team. The cells size depends on the speed specified for the intruder at the beginning of the simulation.



**Figure 2.** Demonstration of intruder strategy model. The darker the grid, the higher the capture probability.

#### 4. Approach

This study proposes a hybrid multi-agent training and reinforcement learning approach based on a human-guided transition model for cooperative task execution in environments where the adversaries or task transition model can be observed and modeled. The key idea in this approach is to utilize human knowledge of the observed mission environment to generate a transition model that can be initialized to guide the overall coordination of the multi-agent system. This is achieved by generating goals or subtasks that will lead to successful completion of the mission by the multi-agent team.

##### 4.1. Framework Design

The backbone of this framework consists of a goal or subtask generation model designed to provide a goal map to guide the learning model, reducing the coordination problem to goal selection and execution. This methodology is based on the premise that high-level subtasks that lead to the ultimate mission objective at any time during the mission's execution can be observed and abstracted, and a subgoal generation model of such processes can be developed by mapping abstract situations to appropriate subtasks. This idea corresponds well with multi-agent operations in certain environments, such as target pursuit or tracking in urban road networks with multiple intersections.

As shown in Figure 3, the methodology consists of two main stages. In the first stage, observation and domain analysis are performed to identify all potential situations that are disadvantageous to the adversaries, and a transition model for these situations is developed. This is performed based on the knowledge and constraints of the opponents or based on the limitations placed by the structure of the environment. In order to complete the first stage, we need a goal generation model that outputs abstract goals or subtasks whose optimal execution will facilitate successful mission execution for the training agents. In this study, the subtask generation model is represented by a graph of vertices  $G_{C(j)} = \{v_i \in V\}$ , with  $v_i$  representing an intersection and the  $\{v_i, v_j\} \in E$  pathway representing the edges. Thus, depending on the direction, node, or edge where the intruder finds itself, the goal map consisting of the set of nodes connected to the current node or edge can be generated to guide the multi-USV system.

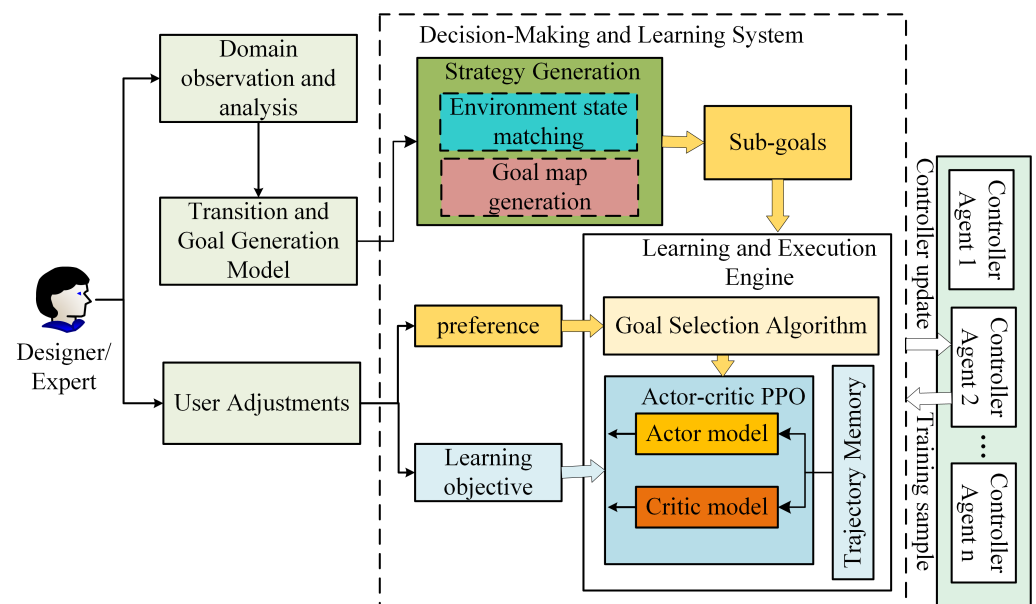


Figure 3. Structure diagram of the proposed framework.

During training, the strategy generation model is executed at certain time steps to produce a goal map or set of subtasks for the decision system to execute or learn to execute. The strategy generator is an intermediate module that encodes the goal generation model



for subgoal generation to guide the decision-making sequence among the multi-agent system to achieve a high degree of team coordination. This module can be replaced by other algorithms or learning algorithms in the absence of an expert-guided task generation model, making it pluggable.

An agent then selects a task from the set of subtasks to execute or learn. In this work, we propose a heuristic-based approach as the transition model encoding for realizing goal selection during training. Our ultimate goal is to learn a coordinated policy for the multi-agent system to perform a cooperative mission, taking advantage of our knowledge of the adversaries or environment constraints to generate abstract goals and leading to a successful mission. The framework implements an actor-critic PPO for learning the selected subtasks from the goal map generation model. The learning objective guides and measures the collective performance of the multi-agent system on the mission. The preference defines constraints or priorities that can be specified for tasks, behaviors, goals, or low-level actions during training.

#### 4.2. Goal Selection Algorithm

In order to realize optimal subgoal selection, we adopted a heuristic immune network model. Prior to the immune process, the state of the agent is used by a strategy generator to generate subgoals. The generated subgoals are then evaluated, and an appropriate subgoal is selected by each agent based on the heuristic immune process. Algorithm 1 presents the dynamic navigation point selection of agents which can be customized to fit other mission scenarios by employing different models to compute the affinity and activation of subgoals.

---

##### Algorithm 1: Subgoal (navigation point) selection algorithm

---

**Data:** Goal map  
**Result:** Optimal target

```

1  $Act_c \leftarrow \emptyset$  { initialize empty list of activation values };
2 for For every navigation position  $j \in goalmap$  do
3    $S_n \leftarrow S$  { get current local state of neighbors };
4    $s_t \leftarrow S_m$  { get own state };
5    $Aff_{N(a)} \leftarrow$  { the set of current neighbors' ( $N(a)$ ) affinity };
6   Determine own affinity  $Aff_a$  using Equation (6);
7   Compute the suitability of selecting  $j$ , i.e.,  $B_{act}$  value of the agent using
   Equation (7);
8    $Act_c \leftarrow B_{act}$  { add to list of activation values };
9  $max_{B_{act}} \leftarrow argmax(Act_c, random(Act_c))$ ;
10 Return the corresponding subgoal of  $max_{B_{act}}$  as the suitable navigation position.
```

---

At every decision step, the goal selection algorithm updates its navigation position preference based on Equation (6) and selects the most attractive goal position. Each agent reevaluates all the available goal positions when the goal map changes and selects the strongest, most suitable subgoal. This process continues until the system terminates. Hence, after receiving the goal map generated from the transition model, the algorithm computes the suitability of each navigation position in the goal map to select the optimal subgoal point:

$$Aff_a = \frac{1}{N(a)} * \sum_{i=1}^{N(a)} \frac{1}{(|\vec{p}_a - \vec{p}_i|)^2} * \epsilon \quad (6)$$

where  $p_a$  represents the relative location of vehicle  $a$ , whose activation is being determined, and  $p_i$  is the positions of other vehicles, while  $\vec{p}_a - \vec{p}_i$  is the orientation from  $a$  to  $i$  and  $\epsilon \in [0.2, 1]$  is a factor that indicates if obstacles are located in the direction of the goal position:

$$B_{act} = (Aff_a + \varepsilon * Aff_{j,a}) - (Aff_{j,a} + \rho * Aff_a) \quad (7)$$

The decision of an agent to move and select a particular goal is determined using Equation (7). In this equation, the first term represents the stimulation effect of the current agent with respect to the target location being considered, while the second term represents the suppressive effects of its neighbors. Here,  $Aff_a$  and  $Aff_{j,a}$  are the affinity of the current agent with respect to its neighbors' ( $N(a)$ ) states, respectively, and vice versa. Additionally,  $\varepsilon$  and  $\rho$  are both scaling factors, where  $\{\varepsilon, \rho \in [0.2, 1]\}$  and  $\rho < \varepsilon$ . The navigation location with the highest activation value is selected, and control is passed to the task execution policy learning.

#### 4.3. Task Execution Policy Learning

After the agent selects a target location, the appropriate policy should be executed to achieve the agent-selected goal. In this case, a navigation policy is implemented, which uses proximal policy optimization [44,45] with the actor-critic method to optimize the navigation goal execution based on the collective experiences of all controller agents. After the target locations are selected by controller agents, each controller agent obtains its local state  $S_n$  and selects an action from its action space  $A_n$ , following the stochastic policy  $\pi_\theta$ . For the navigation policy, the input of the training algorithm consists of the poses and velocities of the agent's physical model. The inputs include some noise to account for the various sensor noises present in the real world. These additional noises are incorporated using a set of dynamic parameters  $\Lambda$  within the range of  $\Gamma \in [0.02, 0.2]$  for the position and orientation noises. Finally, the actor-critic model implements neural networks for learning.

---

#### Algorithm 2: Proximal policy optimization training

---

```

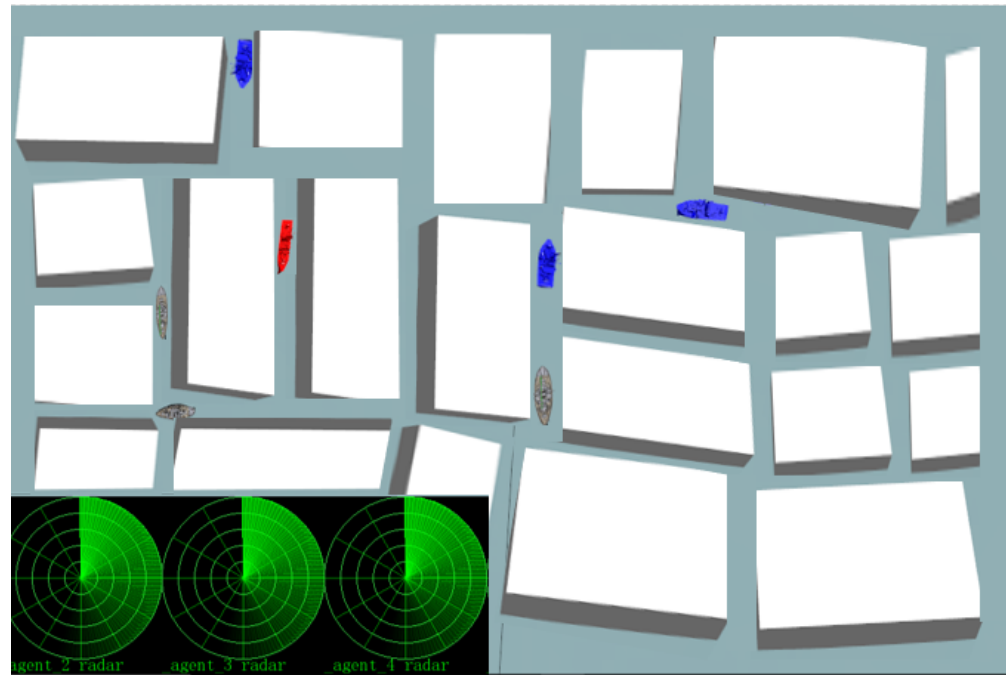
1 Initialize the neural network  $\pi_\theta$  ;
2 Set number of episodes  $K$  ;
3 while current_episode  $\leq$  number_of_episodes do
4   Reset training environment to initial state  $S_0$  ;
5   Uniformly sample the dynamics parameters  $\Lambda$  from a range  $\Gamma$  ;
6   for agent  $i = 1, 2, 3, \dots$  do
7     Get current state of  $i$ ,  $s_t^i$  Get goal position  $s_t^{g,i}$  using Algorithm 1 ;
8     Sample an action from  $A_i$ ,  $a_t^i \leftarrow \pi_\theta(a_t^i | s_t^i)$  ;
9     Collect state  $s_t^i$ , reward  $r_t^i$  and  $a_t^i$  for  $T_i$  time steps ;
10    Compute the advantage estimates  $\hat{A}_1^i, \dots, \hat{A}_{T_i}^i$  ;
11  Optimize surrogate loss with respect to  $\theta$  for  $k$  epochs ;
12  Set  $\theta_{old} \leftarrow \theta$  ;
13  Optimize value loss with respect to  $\Pi$  for number of episodes;
14  Set  $\Pi_{old} \leftarrow \Pi$  ;
```

---

Figure 2 shows the summarized algorithm for training the navigation policy. At the beginning of training, the network is initialized. With a given number of episodes, the environment is reset at beginning of an episode. For each episode, the dynamics parameters  $\Lambda$  from a given range  $\Gamma$  are sampled. Next, the current state  $s_t^i$  of the vehicle in the training environment is collected. The target location is obtained by executing Algorithm 1. Based on the current state, the agent's action  $a_t^i \leftarrow \pi_\theta(a_t^i | s_t^i)$  is obtained. Next, to compute the estimated advantage, the state  $s_t^i$ , reward  $r_t^i$ , and  $a_t^i$  are collected. The advantage estimates are performed in a manner similar to the work in [46]. Finally, the surrogate loss and value loss with respect to the vector of parameters  $\theta$  and  $\Pi$  are constructed and updated, respectively [44].

## 5. Experiment and Results

To evaluate this approach, we implemented a simulation system of the multi-USV target capture in a novel floating community. The simulation was a modification of the one presented in [49]. This section describes the details of the training environment set-up and experimental results. Figure 4 shows the simulation training set-up of the multi-USV target capture.



**Figure 4.** Simulation environment set-up.

### 5.1. USV Physical Model

The basic USV model has a diverse set of onboard sensors. Hence, the USV physical model in the simulation environment was equipped with sensors (radar) that were used to map the surrounding environment while executing missions. The basic radar model had a 360 degree field of view with a radius  $r$  which determined its range. Additionally, an onboard inertial measurement unit (IMU) model was used to provide the linear acceleration, orientation, and rotational velocity data and estimate the poses of the USVs. To find the exact positions and orientations of the USVs with respect to the global frame, the USVs each had an onboard global positioning system (GPS). The radar was partitioned into virtual sensors of the same range by assigning a reach for each virtual sensor based on the configuration provided. These virtual sensors returned the relative positions and normalized distance of other objects in the scene. A relational virtual sensor returns information about how a USV is situated to and from other objects, while a velocity sensor returns the velocity of objects within the detection range of the USV. The basic inputs and controls include the heading and velocity controls for motion. The communication of USVs was modeled based on the reach of the communication range.

### 5.2. USV Dynamic Model

Considering the USV movement dimension in six-DOF, the simulated USV model was based on Fossen's six-DOF model for marine vehicles [50]. Figure 5 shows the USV motions in six DOF. This model expresses the resultant movement of a USV as the combined effect of five main forces as shown in Equation (8):

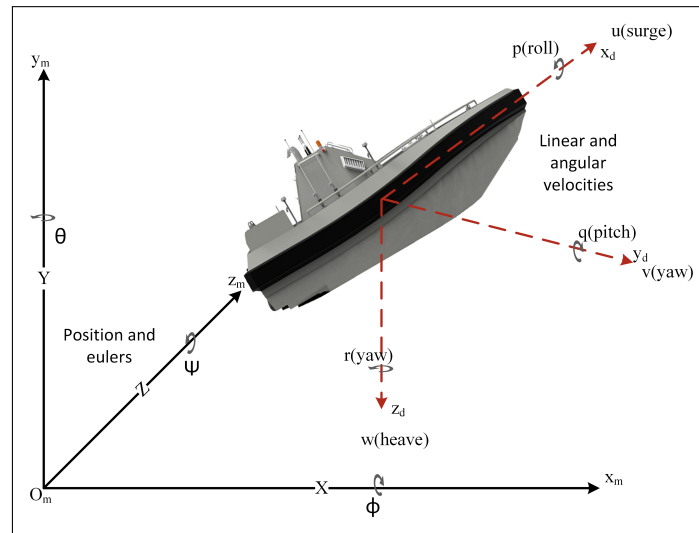
$$\tau_{RF} = \tau_{hsf} + \tau_{hydf} + \tau_{wind} + \tau_{waves} + \tau \quad (8)$$

where  $\tau_{hsf}$  is the hydrostatic forces,  $\tau_{hydf}$  is the hydrodynamic forces,  $\tau$  is the control and propulsion forces, and  $\tau_{wind}$  and  $\tau_{waves}$  are the wind and wave forces, respectively. The kinematic and kinetic model with wind and wave disturbances is defined in Equation (9).

$$\begin{cases} \dot{\eta} = J(\eta)v \\ M_{RB}\dot{v} + C_{RB}(v)v + M_A\dot{v}_r + C_A(v_r)v_r + D(v_r)v_r + g(\eta) = \tau_E + \tau \end{cases} \quad (9)$$

where the following definitions apply:

- $\eta = [x, y, z, \phi, \theta, \psi]^T$  is a vector of the position and Euler angles in the m-frame;
- $v = [u, v, w, p, q, r]^T$  is a vector of the linear and angular velocities in the d-frame;
- $v_r$  is the hydrodynamic terms of the relative velocities vector (i.e., the difference between the vessel velocity relative to the fluid velocity and the marine currents expressed in the reference frame);
- $\tau_E$  represents the forces and moments of environmental disturbances of superimposed wind, currents, and waves;
- The parameters  $J$ ,  $M$ ,  $D$ , and  $C$  are the rotational transformation, inertia, damping, and coriolis and centric fugal matrices, respectively.



**Figure 5.** USV motion in six DOF.

### 5.3. Experimental Settings

The experiment was conducted with teams of two, three, and five USVs in a simulated floating city environment. The initial positions of the teams were spread apart from each other to cover a large area. At the beginning of each episode, the intruder was invisible to the USV group, so the task generation module generated random target points near the intruder. The idea was to detect the intruder if the USVs were to move to these locations. Subsequently, the goal map was generated based on the location of the intruder, as mentioned in Section 4.1, which predicted the possible navigation direction. In the experiment, the USVs could communicate with each other, and the detection range of the USVs was set to 300 m. The length of an episode was set to 1000 simulation steps, and 5000 episodes were run.

The observation space of the agents included the general goal map, the local state of the USV, all the other USVs of the team within the USV communication range, and the distance and headings to the collision points. The action space consisted of a probability distribution over the action space of speed in  $[0, 1]$  and a direction of  $[-1, 0, 1]$ . The actor implemented a three-hidden-layer network. The input state used 3 fully connected hidden layers of 256, 128, and 64 rectifier units. The output layer was a fully connected layer of five units, and different activation functions were used for different parts of the action space.

The critic part was the state value function  $V(s_t)$  and was designed as a two-hidden-layer neural network, including two fully connected layers with 128 and 64 rectifier units. The output layer had only one unit to generate the value with linear activation.

The  $k$  update epochs were set to 20 for the surrogate loss update and 15 for the value loss. Both the surrogate loss and value losses were optimized with the Adam optimizer, with learning rates of 0.0001 and 0.0015, respectively. The parameter for clipping was 0.2.

#### 5.4. Experimental Results

To evaluate the effectiveness of the approach, the experimental results with a multi-USV team against a high-speed intruder on one hand and a slow intruder with the same speed as the multi-USV team on the other hand are presented. The USVs used a fixed speed of  $3 \text{ m}\cdot\text{s}^{-1}$ , while the intruder's speed was set to  $3 \text{ m}\cdot\text{s}^{-1}$  and  $4 \text{ m}\cdot\text{s}^{-1}$  for the slow and fast intruder, respectively. We plotted the average rewards for the different scenario settings.

Figures 6–8 show the average episode rewards for two, three, and five USVs and one intruder during training. The performance of the algorithm clearly improved over the course of training within a short period of time, as shown in the training graphs. From these graphs, it is evident that the approach can easily guide the multi-USV team not only to learn a navigation policy but also to dynamically assign the goals generated by the strategy generator to the USVs. Moreover, the results also show the influence of the intruder type, since there was a different average reward return with different speeds for the intruder.

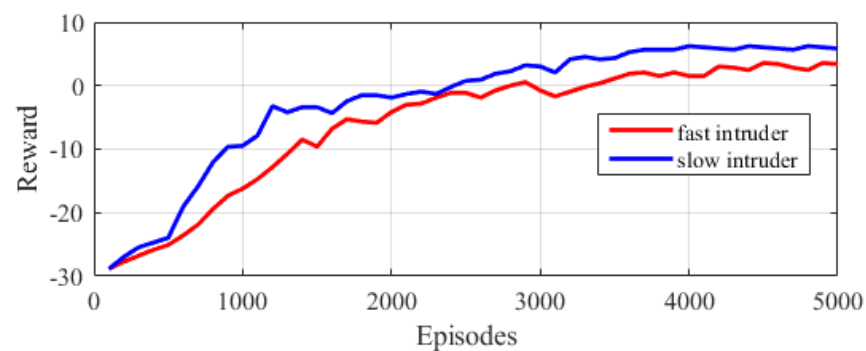


Figure 6. Average episode rewards for two USVs and one intruder.

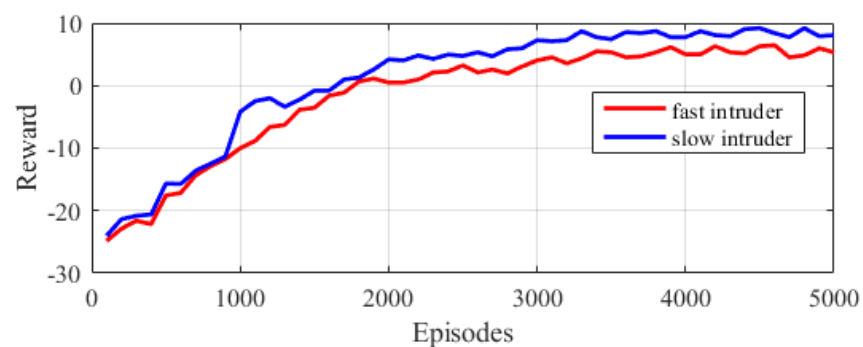
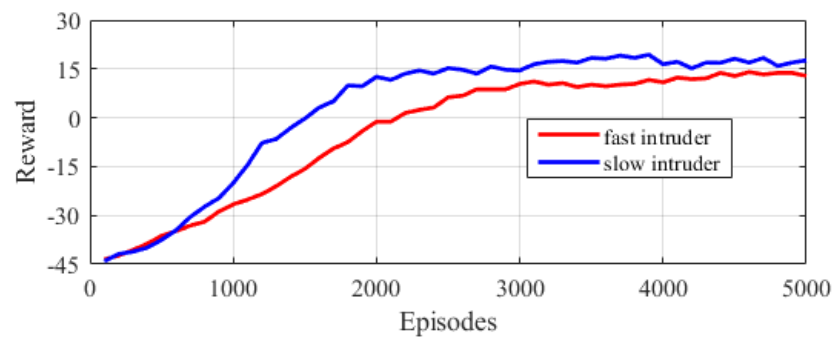


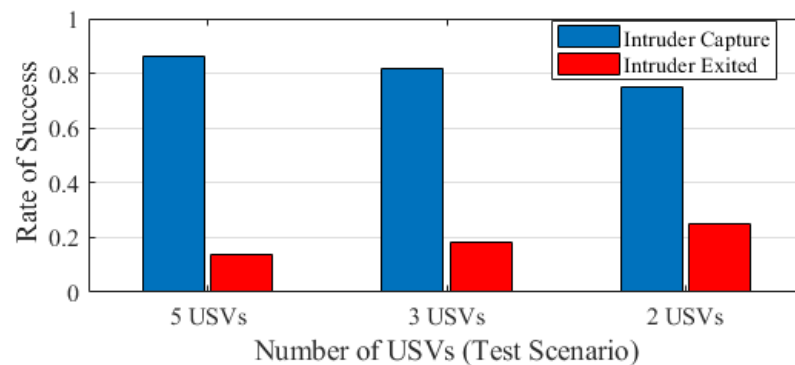
Figure 7. Average episode rewards for three USVs and one intruder.



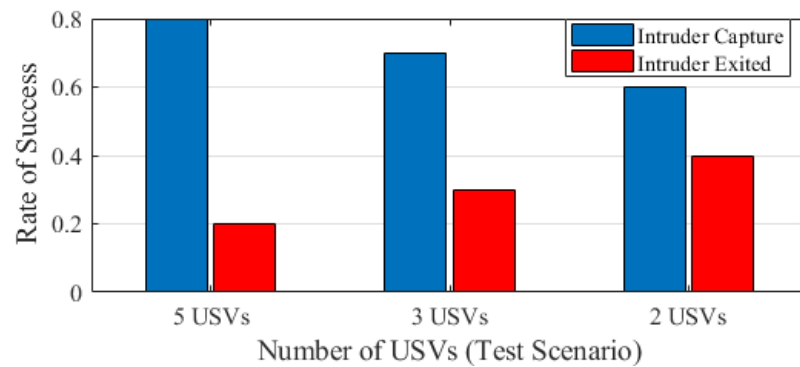


**Figure 8.** Average episode rewards for five USVs and one intruder.

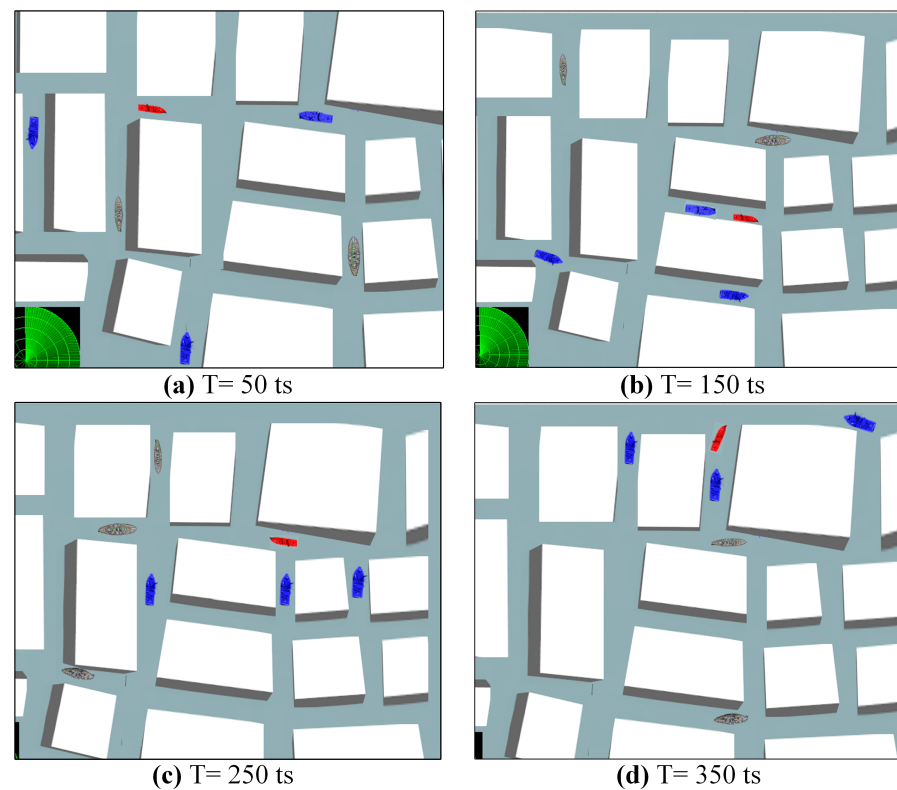
In order to test the performance of the final controllers, 100 simulations with the best-performing controllers for two scenarios with different navigation target speeds were run. In this experiment, the USVs used a fixed speed of  $5 \text{ m}\cdot\text{s}^{-1}$ , while the intruder speed was set to  $5 \text{ m}\cdot\text{s}^{-1}$  and  $7.5 \text{ m}\cdot\text{s}^{-1}$  for the slow and fast intruder, respectively. The detection radius for the former was 200 m, while 350 m was used for the latter. Here, the higher speed made it more difficult for the USVs to make turns at intersections, even if the evaluation and training environments were similar. Figures 9 and 10 show the success rate of the USVs capturing the intruder as opposed to the intruder exiting the environment. The intruder was said to exit if it was not captured within the simulation episode length. Despite the higher speed of the USVs, the system still achieved about 86% and 77% successful capture rates with 5 and 2 USV teams, respectively, as can be seen in Figure 9. On the other hand, there is a decline in capture performance in Figure 10, with 80% and 60% success rates for a similar number of USVs, which can be attributed to the higher speed of the intruder boat. From these graphs, it is evident that more USVs led to more successful captures. In addition, the speed of the intruder affected the success rate of the coordination team. Figure 11 shows a sample capture process of three USVs at increasing time steps. Here, the speed was  $3 \text{ m}\cdot\text{s}^{-1}$  for both the USVs and intruder with a 200 m detection radius. Figure 11a shows the configuration of USVs after 50 ts with a USV to the front (USV1), back (USV2) and right (USV3) of the intruder boat. In Figure 11b, USV1 executes a direct chase, while USV2 and USV3 execute left and right turns, respectively. In Figure 11c, USV3 is now performing the chase, USV2 makes a left turn at the next intersection to cover one of the possible exits of the intruder while USV2 makes a left turn at the second intersection ahead it. Finally, after 350 ts, the target is capture since all the exit points are covered by the team as shown in Figure 11d. From the trajectories of the USVs, it can be found that the USVs were able to capture the target within 350 simulation time steps.



**Figure 9.** Success rate of USVs when intruder and USV speeds are same (USV speed =  $5 \text{ m}\cdot\text{s}^{-1}$ , USV detection radius = 200 m, and intruder speed =  $5 \text{ m}\cdot\text{s}^{-1}$ ).



**Figure 10.** Success rate of USVs in capturing intruder (USV speed =  $5 \text{ m}\cdot\text{s}^{-1}$ , USV detection radius = 350 m, and intruder speed =  $7.5 \text{ m}\cdot\text{s}^{-1}$ ).



**Figure 11.** Sample capture process of three-USV team at different time steps (ts) with speed of  $3 \text{ m}\cdot\text{s}^{-1}$  for both USVs and intruder and a 200 meter detection radius.

## 6. Conclusions

This paper presents a training mechanism for a multi-agent controller that utilizes a human-guided task generation model for generating abstract subgoals based on the observed situations of an environment or opponents. This approach is applied in a novel floating multi-intersection environment for multiple unmanned surface vehicles (USVs), dynamically selecting a navigation goal generated based on the knowledge of the environment layout and learning navigation control for target capturing. To provide a realistic and competitive intruder policy, a probability model used by the intruder was adopted to make its maneuvering decisions. The multi-USV system employs an immune network and dynamics to achieve two main objectives: collaborative task assignment and proximal policy optimization (PPO) for navigation policy learning of the USV team. The performance of the system during training showed an improvement in the rewards received within a few episodes of the simulation. The evaluation results with 2, 3, and 5 USVs show that the

approach can be applied for multi-agent navigation and target capture in multi-intersection environments and is capable of achieving a success rate of over 80% with 5 USVs and over 60% with 2 USVs. Thus, the proposed method has the potential for practical trajectory design and application in a multi-vehicle environment with multiple intersections and, by extension, in multi-vehicle scenarios where the environment and task structure can be leveraged to guide the training and execution process. For instance, this approach can be directly applied to target pursuit in an urban city environment by simply replacing the USVs with UGVs or UAVs.

However, this approach assumes independent subgoals that are performed concurrently by the multi-agent system. Additionally, the experiments conducted were limited to a single intruder that adopted a somewhat regularized escape strategy. Hence, the proposed approach needs further investigation and experiments in different and more practical applications with other types of unmanned vehicles where the generated tasks are coupled and need to be performed sequentially.

**Author Contributions:** Conceptualization, S.N. and S.Z.; methodology, S.N.; software, S.N., X.Y. and X.A.; validation, X.Y. and X.A.; formal analysis, S.N. and S.Z.; investigation, S.N.; resources, Y.X.; writing—original draft preparation, S.N.; writing—review and editing, S.N. and S.Z.; visualization, S.Z.; project administration, Y.X. and S.N. All authors have read and agreed to the published version of the manuscript.

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

**Data Availability Statement:** Data are contained within the article.

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

## References

- Berns, K.; Nezhadfar, A.; Tosa, M.; Balta, H.; Cubber, G.D. *Unmanned Ground Robots for Rescue Tasks*; IntechOpen Limited: London, UK, 2017. [CrossRef]
- Martz, J.; Al-Sabban, W.; Smith, R.N. Survey of unmanned subterranean exploration, navigation, and localisation. *IET Cyber Syst. Robot.* **2020**, *2*, 1–13. [CrossRef]
- Winston, C. Autonomous Vehicles Could Improve Policing, Public Safety, and Much More. Available online: <https://www.brookings.edu/blog/techtank/2020/08/25/autonomous-vehicles-could-improve-policing-public-safety-and-much-more/> (accessed on 7 February 2023).
- Wolf, M.; Rahmani, A.; de la Croix, J.; Woodward, G.; Hook, J.V.; Brown, D.; Schaffer, S.; Lim, C.; Bailey, P.; Tepsuporn, S.; et al. CARACaS multi-agent maritime autonomy for unmanned surface vehicles in the Swarm II harbor patrol demonstration. In *Proceedings of the Unmanned Systems Technology XIX, Anaheim, CA, USA, 11–13 April 2017*; Karlens, R.E., Gage, D.W., Shoemaker, C.M., Nguyen, H.G., Eds.; International Society for Optics and Photonics; SPIE: Washington, DC, USA, 2017; Volume 10195, pp. 218–228.
- Maritime Executive, T. [Video] Demonstration of Autonomous Vessel Operations. Available online: <https://maritime/-executive.com/article/video-demonstration-of-autonomous-vessel-operations> (accessed on 7 February 2023).
- Eshel, T. Unmanned Boats Demonstrate Autonomous Swarm, Gunnery Support Techniques—Defense Update. Available online: [https://defenseupdate.com/20141006\\_usv\\_demo.html](https://defenseupdate.com/20141006_usv_demo.html) (accessed on 7 February 2023).
- Zhang, T.; Li, Q.; Zhang, C.S.; Liang, H.W.; Li, P.; Wang, T.M.; Li, S.; Zhu, Y.L.; Wu, C. Current trends in the development of intelligent unmanned autonomous systems. *Front. Inf. Technol. Electron. Eng.* **2017**, *18*, 68–85. [CrossRef]
- Howard, A.; Parker, L.E.; Sukhatme, G.S. The SDR Experience: Experiments with a Large-Scale Heterogeneous Mobile Robot Team. In *Proceedings of the Experimental Robotics IX, New York, NY, USA, 2–4 March 2006*; Ang, M.H., Khatib, O., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 121–130.
- Liu, Y.; Song, R.; Bucknall, R.; Zhang, X. Intelligent multi-task allocation and planning for multiple unmanned surface vehicles (USVs) using self-organising maps and fast marching method. *Inf. Sci.* **2019**, *496*, 180–197. [CrossRef]
- Xue, K.; Huang, Z.; Wang, P.; Xu, Z. An Exact Algorithm for Task Allocation of Multiple Unmanned Surface Vehicles with Minimum Task Time. *J. Mar. Sci. Eng.* **2021**, *9*, 907. [CrossRef]
- Antonyshyn, L.; Silveira, J.; Givigi, S.; Marshall, J. Multiple Mobile Robot Task and Motion Planning: A Survey. *ACM Comput. Surv.* **2023**, *55*, 1–35. [CrossRef]
- Cortés, J.; Egerstedt, M. Coordinated Control of Multi-Robot Systems: A Survey. *SICE J. Control Meas. Syst. Integr.* **2017**, *10*, 495–503. [CrossRef]
- Balhara, S.; Gupta, N.; Alkhayyat, A.; Bharti, I.; Malik, R.Q.; Mahmood, S.N.; Abedi, F. A survey on deep reinforcement learning architectures, applications and emerging trends. *IET Commun.* **2022**, early access. [CrossRef]

14. Wang, X.; Wang, S.; Liang, X.; Zhao, D.; Huang, J.; Xu, X.; Dai, B.; Miao, Q. Deep Reinforcement Learning: A Survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–15. [\[CrossRef\]](#)
15. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 6382–6393.
16. Setyawan, G.E.; Hartono, P.; Sawada, H. Cooperative Multi-Robot Hierarchical Reinforcement Learning. *Int. J. Adv. Comput. Sci. Appl.* **2022**, 13, 2022. [\[CrossRef\]](#)
17. Xia, J.; Luo, Y.; Liu, Z.; Zhang, Y.; Shi, H.; Liu, Z. Cooperative multi-target hunting by unmanned surface vehicles based on multi-agent reinforcement learning. *Def. Technol.* **2022**, in press. [\[CrossRef\]](#)
18. Bansal, T.; Pachocki, J.; Sidor, S.; Sutskever, I.; Mordatch, I. Emergent Complexity via Multi-Agent Competition. *arXiv* **2017**, arXiv:1710.03748. Available online: <https://arxiv.org/abs/1710.03748> (accessed on 7 February 2023)
19. Zhao, W.; Chu, H.; Miao, X.; Guo, L.; Shen, H.; Zhu, C.; Zhang, F.; Liang, D. Research on the Multiagent Joint Proximal Policy Optimization Algorithm Controlling Cooperative Fixed-Wing UAV Obstacle Avoidance. *Sensors* **2020**, 20, 4546. [\[CrossRef\]](#)
20. Han, R.; Chen, S.; Hao, Q. Cooperative Multi-Robot Navigation in Dynamic Environment with Deep Reinforcement Learning. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 448–454. [\[CrossRef\]](#)
21. Long, P.; Fan, T.; Liao, X.; Liu, W.; Zhang, H.; Pan, J. Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 6252–6259. [\[CrossRef\]](#)
22. Wen, J.; Liu, S.; Lin, Y. Dynamic Navigation and Area Assignment of Multiple USVs Based on Multi-Agent Deep Reinforcement Learning. *Sensors* **2022**, 22, 6942. [\[CrossRef\]](#) [\[PubMed\]](#)
23. Awheda, M.D.; Schwartz, H.M. Decentralized learning in pursuit-evasion differential games with multi-pursuer and single-superior evader. In Proceedings of the 2016 Annual IEEE Systems Conference (SysCon), Orlando, FL, USA, 18–21 April 2016; pp. 1–8. [\[CrossRef\]](#)
24. Yuan, Z.; Wu, T.; Wang, Q.; Yang, Y.; Li, L.; Zhang, L. T3OMVP: A Transformer-Based Time and Team Reinforcement Learning Scheme for Observation-Constrained Multi-Vehicle Pursuit in Urban Area. *Electronics* **2022**, 11, 1339. [\[CrossRef\]](#)
25. Liang, L.; Deng, F.; Lu, M.; Chen, J. Analysis of Role Switch for Cooperative Target Defense Differential Game. *IEEE Trans. Autom. Control* **2021**, 66, 902–909. [\[CrossRef\]](#)
26. Li, W. A Dynamics Perspective of Pursuit-Evasion: Capturing and Escaping When the Pursuer Runs Faster Than the Agile Evader. *IEEE Trans. Autom. Control* **2017**, 62, 451–457. [\[CrossRef\]](#)
27. Li, Y.; He, J.; Chen, C.; Guan, X. Intelligent Physical Attack Against Mobile Robots With Obstacle-Avoidance. *IEEE Trans. Robot.* **2023**, 39, 253–272. [\[CrossRef\]](#)
28. Xu, Y.; Yang, H.; Jiang, B.; Polycarpou, M.M. Multiplayer Pursuit-Evasion Differential Games With Malicious Pursuers. *IEEE Trans. Autom. Control* **2022**, 67, 4939–4946. [\[CrossRef\]](#)
29. Wei, W.; Wang, J.; Du, J.; Fang, Z.; Jiang, C.; Ren, Y. Underwater Differential Game: Finite-Time Target Hunting Task with Communication Delay. In Proceedings of the ICC 2022—IEEE International Conference on Communications, Seoul, Republic of Korea, 16–20 May 2022; pp. 3989–3994. [\[CrossRef\]](#)
30. Pan, T.; Yuan, Y. A Region-Based Relay Pursuit Scheme for a Pursuit-Evasion Game With a Single Evader and Multiple Pursuers. *IEEE Trans. Syst. Man Cybern. Syst.* **2023**, 53, 1958–1969. [\[CrossRef\]](#)
31. Ye, J.; Wang, Q.; Ma, B.; Wu, Y.; Xue, L. A Pursuit Strategy for Multi-Agent Pursuit-Evasion Game via Multi-Agent Deep Deterministic Policy Gradient Algorithm. In Proceedings of the 2022 IEEE International Conference on Unmanned Systems (ICUS), Guangzhou, China, 28–30 October 2022; pp. 418–423. [\[CrossRef\]](#)
32. Wang, Q.; Wu, K.; Ye, J.; Wu, Y.; Xue, L. Apollonius Partitions Based Pursuit-evasion Game Strategies by Q-Learning Approach. In Proceedings of the 2022 41st Chinese Control Conference (CCC), Hefei, China, 25–27 July 2022; pp. 4843–4848. [\[CrossRef\]](#)
33. Sidney N Givigi, J.; Schwartz, H.M. Decentralized strategy selection with learning automata for multiple pursuer–evader games. *Adapt. Behav.* **2014**, 22, 221–234. [\[CrossRef\]](#)
34. Wang, H.; Yue, Q.; Liu, J. Research on Pursuit-evasion games with multiple heterogeneous pursuers and a high speed evader. In Proceedings of the 27th Chinese Control and Decision Conference (2015 CCDC), Qingdao, China, 23–25 May 2015; pp. 4366–4370. [\[CrossRef\]](#)
35. Du, W.; Guo, T.; Chen, J.; Li, B.; Zhu, G.; Cao, X. Cooperative pursuit of unauthorized UAVs in urban airspace via Multi-agent reinforcement learning. *Transp. Res. Part Emerg. Technol.* **2021**, 128, 103122. [\[CrossRef\]](#)
36. Zhang, R.; Zong, Q.; Zhang, X.; Dou, L.; Tian, B. Game of Drones: Multi-UAV Pursuit-Evasion Game With Online Motion Planning by Deep Reinforcement Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–10. [\[CrossRef\]](#)
37. Özkahraman, Ö.; Ögren, P. 3D Pursuit-Evasion for AUVs. *Researchgate* **2018**, Available online: [https://www.researchgate.net/publication/327903971\\_3D\\_Pursuit-Evasion\\_for\\_AUVs](https://www.researchgate.net/publication/327903971_3D_Pursuit-Evasion_for_AUVs) (accessed on 7 February 2023).
38. Liang, X.; Wang, H.; Luo, H. Collaborative Pursuit-Evasion Strategy of UAV/UGV Heterogeneous System in Complex Three-Dimensional Polygonal Environment. *Complexity* **2020**, 2020, 7498740. [\[CrossRef\]](#)
39. de Souza, C.; Newbury, R.; Cosgun, A.; Castillo, P.; Vidolov, B.; Kulić, D. Decentralized Multi-Agent Pursuit Using Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2021**, 6, 4552–4559. [\[CrossRef\]](#)

40. Ma, J.; Lu, H.; Xiao, J.; Zeng, Z.; Zheng, Z. Multi-robot Target Encirclement Control with Collision Avoidance via Deep Reinforcement Learning. *J. Intell. Robot. Syst.* **2020**, *99*, 371–386. [[CrossRef](#)]
41. Zhang, T.; Liu, Z.; Wu, S.; Pu, Z.; Yi, J. Multi-Robot Cooperative Target Encirclement through Learning Distributed Transferable Policy. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8. [[CrossRef](#)]
42. Bernstein, D.S.; Zilberstein, S.; Immerman, N. The Complexity of Decentralized Control of Markov Decision Processes. In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, San Francisco, CA, USA, 30 June–3 July 2000; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2000; pp. 32–37.
43. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust Region Policy Optimization. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015.
44. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347. Available online: <https://arxiv.org/abs/1707.06347> (accessed on 7 February 2023).
45. Heess, N.M.O.; Dhruva, T.; Sriram, S.; Lemmon, J.; Merel, J.; Wayne, G.; Tassa, Y.; Erez, T.; Wang, Z.; Eslami, S.M.A.; et al. Emergence of Locomotion Behaviours in Rich Environments. *arXiv* **2017**, arXiv:1707.02286. Available online: <https://arxiv.org/pdf/1707.02286.pdf> (accessed on 7 February 2023).
46. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Harley, T.; Lillicrap, T.P.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1928–1937.
47. Jerne, N.K. Towards a network theory of the immune system. *Ann. D'Immunologie* **1973**, *125C*, 373–389. PMID: 4142565. [[PubMed](#)]
48. Farmer, J.; Packard, N.H.; Perelson, A.S. The immune system, adaptation, and machine learning. *Phys. Nonlinear Phenom.* **1986**, *22*, 187–204. [[CrossRef](#)]
49. Nantogma, S.; Pan, K.; Song, W.; Luo, R.; Xu, Y. Towards Realizing Intelligent Coordinated Controllers for Multi-USV Systems Using Abstract Training Environments. *J. Mar. Sci. Eng.* **2021**, *9*, 560. [[CrossRef](#)]
50. McCue, L. Handbook of Marine Craft Hydrodynamics and Motion Control [Bookshelf]. *IEEE Control Syst. Mag.* **2016**, *36*, 78–79. [[CrossRef](#)]

**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.