Optimizing Plastic Waste Collection in Water Bodies Using Heterogeneous Autonomous Surface Vehicles With Deep Reinforcement Learning

Alejandro Mendoza Barrionuevo, Samuel Yanes Luis, Daniel Gutiérrez Reina, and Sergio L. Toral Marín

***Abstract*—This letter presents a model-free deep reinforcement learning framework for informative path planning with heterogeneous fleets of autonomous surface vehicles to locate and collect plastic waste. The system employs two teams of vehicles: scouts and cleaners. Coordination between these teams is achieved through a deep reinforcement approach, allowing agents to learn strategies to maximize cleaning efficiency. The primary objective is for the scout team to provide an up-to-date contamination model, while the cleaner team collects as much waste as possible following this model. This strategy leads to heterogeneous teams that optimize fleet efficiency through inter-team cooperation supported by a tailored reward function. Different trainings of the proposed algorithm are compared with other state-of-the-art algorithms in three distinct scenarios, one with moderate convexity, another with narrow corridors and challenging access, and the last one larger, more complex and with more difficult to access shape. According to the obtained results, it is demonstrated that deep reinforcement learning based algorithms outperform baselines, exhibiting superior adaptability. In addition, training with examples of actions from other algorithms further improves performance, especially in scenarios where the search space is larger.**

***Index Terms*—Autonomous agents, reinforcement learning, environment monitoring and management, heterogeneous multirobot systems, informative path planning.**

I. Introduction

EVERY year, millions of tons of plastic end up in oceans, rivers and lakes, seriously affecting marine life and aquatic ecosystems. According to a recent report [1], it is estimated that by 2050 there will be more plastic than fish by weight in the oceans if urgent action is not taken. Traditional cleaning methods, such as hand nets and manned boats, have usually been proven to be inefficient and poorly scalable.

This letter introduces an innovative approach that employs a fleet of autonomous surface vehicles (ASVs) designed specifically for the collective task of tracking and removing plastic waste from water bodies. The fleet consists of a heterogeneous group of ASVs, which is divided into two specialized teams: cleaning vehicles and scout vehicles. Unlike homogeneous systems, where all agents share identical roles and capabilities, the proposed heterogeneous configuration leverages the diverse strengths of specialized scouts and cleaners. This allows for more efficient task allocation and higher performance as each team has complementary objectives. These need to cooperate to maximize cleaning efficiency, resulting in a heterogeneous multirobot system (HMRS). Scout vehicles, designed to be lighter and faster, are equipped with cameras capable of detecting and accurately mapping the location of plastic waste over a wide range. On the other hand, cleaner vehicles are slower due to the additional trash collection systems they carry, and are equipped with less advanced cameras to reduce costs. These cameras allow the detection of waste in a significantly smaller area compared to scout vehicles.

To coordinate cooperation between these vehicles and maximize plastic waste collection, this research focuses on the use of deep reinforcement learning (DRL). More specifically, a Deep Q-Learning algorithm is implemented in two deep neural networks (DNNs), each of them shared by the agents of the same team. The main objective is that the scout team provides the most updated contamination model possible, while the cleaner team is in charge of collecting as much trash as possible following this trash-density model. This strategy leads to the development of an informative path planning (IPP) system for heterogenous teams of ASVs, which can be defined as the process of generating paths for agents to gather information from an environment, optimizing certain factors. In this case, the policy must seek to optimize the cooperative efficiency of the cleaning task. Overall, the main contributions of this work are as follows: i) The development of a model-free DRL framework for IPP with heterogeneous fleets of ASVs to have intra-team and cross-team cooperation under realistic constraints. ii) The definition of tailored observation and reward functions to drive the algorithm towards the collective goal. Through state representation and rewards, teams should learn to work cooperatively, since the better one does its task, the easier it will be for the other. iii) To evaluate the performance of the design, a thorough analysis and comparison with other algorithms in the literature is carried out.

II. Related Work

This work intersects various fields of study, namely multirobot systems, DRL and IPP for autonomous vehicles. The

Received 19 November 2024; accepted 21 March 2025. Date of publication 28 March 2025; date of current version 9 April 2025. This article was recommended for publication by Associate Editor H. Kasaei and Editor J. Kober upon evaluation of the reviewers' comments. This work was supported in part by the Junta de Andalucía: Consejería de Universidad, and in part by Investigación e Innovación" through the Project Monitorization of Environmental Dangers with Unmanned Surface Agents: (MEDUSA) under Grant PCM_00019. *(Corresponding author: Alejandro Mendoza Barrionuevo.)*

The authors are with the Departamento de Ingeniería Electrónica, Escuela Técnica Superior de Ingeniería, Universidad de Sevilla, 41005 Sevilla, España (e-mail: amendoza1@us.es; syanes@us.es; dgutierrezreina@us.es; storal@us.es).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3555940, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3555940

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

domain of multirobot systems has progressed significantly in recent times, driven by advances in computing and communication technologies [2]. However, despite some research on heterogeneous robot systems [3], most multirobot studies focus on homogeneous configurations. Moreover, although DRL has been applied to multi-agent systems, its use is still more frequent in single-agent systems, as highlighted in surveys [4]. Given the challenge of planning autonomous vehicle trajectories, several algorithms have been employed in IPP [5], such as bio-inspired algorithms [6], Bayesian Optimization [7], or DRL algorithms [8], [9].

Among all of them, as seen in works like [10], DRL has established itself as one of the most effective methods in the field of IPP, including multi-agent systems. In [11], a homogeneous multi-agent system composed of ASVs is presented to efficiently monitor water quality, integrating local Gaussian processes with DRL techniques. In [12], the authors propose a multirobot path planning model for warehouse dispatching system using an enhanced DRL algorithm with prior knowledge and predefined conflict resolution rules. DRL has also been widely applied to other fields within robotics in recent years, in addition to path planning, such as collision avoidance [13], or robotic systems control [14], among others. Additionally, although to a lesser degree, related work on DRL applied to heterogeneous systems can be found. In [15], Dueling Double Deep Q-Learning (DDDQL) is employed with prioritized experience replay to manage fleets of heterogeneous ASVs that carry measurement sensors of different qualities. The purpose is to obtain the best contamination model, but the dynamics of the waste is not addressed. In addition, their fleet heterogeneity focuses on different measurement qualities, so a single neural network shared for all agents is used. In contrast, our method approaches a trash collection problem divided into two teams with marked roles, so two networks are employed, one for each team, shared by the agents belonging to that team.

In [16] an intelligent system for adversarial catching tasks is presented, employing HMRS and multi-agent actor-critic DRL. To enhance cooperation among different types of robot, it introduces asymmetric self-play and curriculum learning techniques. Most studies addressing floating garbage cleaning robots primarily focus on the physical development of prototypes and the mechanical motion control of these systems, such as [17].However, there is a notable gap in research regarding algorithms that guide agents at a high level for detection, positioning and garbage collection tasks.

In [18], a heterogeneous collaborative system is proposed where aerial drones handle waste detection and ASVs perform collection tasks. Aerial drones use a partition-based coverage algorithm, while ASVs are tasked by particle swarm optimization (PSO) and guided by ant colony optimization for path planning. In contrast, our approach leverages DRL framework to autonomously manage decision-making of all type of agents without predefined task assignments, relying on ASVs for both cleaning and waste detection. Unlike [18] which assumes static trash items, one of the main advances introduced with our approach is the simulation of dynamic trash movement by means of environmental currents, which adds realism and complexity to the environments. In addition, their performance evaluation metric focus on total cleanup time, so there is no defined time limit for waste collection as in our approach.

III. Methodology

A. Exploration and Cleaning

This work addresses a waste collection problem in aquatic environments deploying a heterogeneous fleet of ASVs divided into two specialized teams with complementary roles: scouts and cleaners. The scout team consists of smaller, highly mobile and hydrodynamic ASVs equipped with wide-range vision systems, allowing them to cover large areas efficiently. However, these vehicles do not have the ability to collect trash. Their primary task is to identify and map the locations of floating waste, constantly updating a shared model, which is essential for the cleaning team. On the other hand, the cleaner team, comprises ASVs designed specifically to collect the trash. Due to their larger size and the weight of the collection equipment, these vehicles operate at a slower speed. For this reason, and to reduce costs, they are equipped with a simpler mono camera rather than the stereo cameras used by the scout team. This feature limits their detection range to only nearby trash.

To contextualise the future applicability of our approach in the real world, experiments were tested on the detection and geolocation of plastic waste in water using an available ASV, as a scouting agent would do. Trash detection was based on a YOLOv8[1] model, trained on Flow dataset[2] obtaining an accuracy of up to 90%. This field-tested model processes images captured by the stereo vision depth camera to detect trash. Using triangulation and vehicle's GPS and heading data, it obtains global GPS coordinates of the detected waste with a reduced error margin. Although cleaning vehicles lack stereo cameras, they would still be able to update the trash model within their immediate surroundings.

B. Environment and Assumptions

The environment is represented as a grid-based map of size $H \times W$ structured as a connected graph, $G = (V, E)$, where each node $v_{i,j} \in V$ corresponds to a specific location of the grid. Thus, the set of nodes can be defined as $V = \{v_{i,j} \mid 1 \leq i \leq H, 1 \leq j \leq W\}$. The set of edges that connect adjacent nodes is denoted as $E \subseteq V \times V$, which represents the possible movements between two adjacent nodes. Here, node adjacency is based on the assumption that the grid is 8 connected, meaning that each node can connect to its surrounding eight neighbors. Therefore, nodes with less than 8 edges indicate the presence of surrounding obstacles. This framework treats time as discrete, meaning that each movement along an edge takes a single time step t to complete. However, scout vehicles operate at twice the speed of cleaners, which means that they move two nodes per time step, while cleaner vehicles move one. To simplify, each mission is set to last a fixed number of steps T, within which the objective is to collect as much trash as possible.

The positions of the fleet of N vehicles can be represented as a set $\mathcal{P} = \{p_n \mid n = 1, 2, \ldots, N\}$, where p_n is the position

[1]https://github.com/ultralytics/ultralytics
[2]https://github.com/ORCA-Uboat/FloW-Dataset

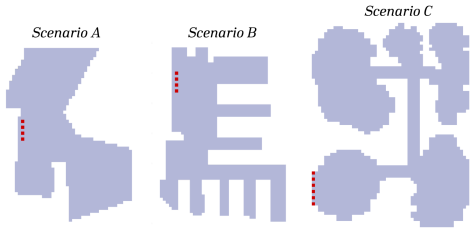

Fig. 1. Representation of the three discretized environment maps which differences in complexity, size, and number of agents. Initial deployment positions are marked in red.

of the vehicle with the n index. Each of the possible positions a vehicle can take is within a node, such as $p_n = v_{i,j} \in V$. The graph can be represented as a matrix $M[i,j] \in \{0,1\}$ of size $H \times W$, where $M[i,j] = 1$ if the node $v_{i,j}$ is navigable, and 0 otherwise.

To train agents in realistic environments, three scenario maps with different complexity are used. They are represented as matrices $M[i,j]$, something commonly found in similar works such as [15], [18]. The first map is shown on the left in Fig. 1, and represents a simpler, more open and with moderate convexity port. In contrast, the second map, shown on the middle, is synthetically created and depicts a typical sport wharf, with narrow lanes and complex pathways, which introduces greater challenges for the algorithm. Lastly, the Scenario C synthetically created, shown on the right, is 35% larger than the previous ones, and features more complex and more difficult to access shapes, as well as randomly generated obstacles with normal distributions in each episode. The scenario map and the boundaries are assumed to be known by the agents. Collisions between vehicles are assumed not possible, since they cannot be in the same node. To ensure a realistic deployment condition, the initial positions of the ASVs will be random within certain safety areas, as seen in Fig. 1. Despite this, the algorithm must optimize the coverage of the map with trajectories adapted to these starting points.

The trash positions are defined as a set B with real coordinates in space, following $B = \{b_k = (x_k, y_k) \mid k = 1, 2, \ldots, K\}$. There, K is the total number of trash elements generated using a normal distribution at the beginning of the episode, and each element $b_k \in B$ represents the exact location (x_k, y_k) where the trash is located in a continuous reference system. At the beginning of each episode, a random point from the visitable nodes is selected as the contamination source, around which trash positions are generated using a multivariate normal distribution. Thus, all the trash distributions are different and are generated at the beginning of the episode. No new items are introduced during the cleaning process. In this framework, the items positions are dynamically updated at each time step to simulate environmental factors based on two components: wind and random fluctuations. Wind is modeled as a constant velocity v_{wind} for all items, and is defined at the start of the episode sampled from a uniform distribution. Random fluctuations are modeled as random variations in the velocity v_{rand} of each item sampled from uniform distributions at each time step. Then, each trash position is updated following: $b_k^{t+1} = b_k^t + \Delta t(w_{wind} \cdot \nu_{wind} + w_{rand} \cdot \nu_{rand})$. Components w_{wind} and w_{rand} are weights set to 1.

Therefore, the trash distribution map can be defined as a matrix Y of size $H \times W$ where each value $Y[i,j]$ indicates the number of trash positions from B within the corresponding node area at each time step. Thus, $Y[i,j] = |\{b_k \in B \mid (x_k, y_k) \in \text{Area}(i,j)\}|$, where $\text{Area}(i,j)$ represents the area of the cell (i,j) in M matrix. Consequently, $Y[i,j] = 0$ implies that the cell [i,j] is free of trash, while a positive value of $Y[i,j]$ indicates the presence of trash items. When a cleaner agent passes through a cell, all trash items that are placed inside the area of the cell are collected and removed from the B set. It is assumed that the cleaner agents have unlimited carrying capacity, and that the collection process is handled internally by the vehicle's local autopilot and it does not impact the decisions of the proposed algorithm. The amount of trash collected by a cleaner vehicle from P is defined by the number of trash items located in the same cell as the agent at time t, i.e., $C(p_n, Y) = Y[i,j] \mid v_{ij} = p_n$.

Due to differences in camera quality between the two teams, the range of coverage they can observe is different. It can be defined Θ as the set of nodes around a vehicle within its field of view, which is determined by a radius ρ, following: $\Theta(p_n) = \{v \in V \mid \|v, p_n\| < \rho\}$. The stereo vision of scout agents enables them to estimate trash positions within a wide range of coverage such that the radius ρ is set to six nodes. The YOLOv8 model processes stereo RGB images captured by the scouts, allowing for high-confidence detection within this radius, as the position of each trash item can be triangulated accurately based on the depth information. In contrast, cleaner agents are equipped with simpler, nonstereo cameras, which restrict their detection capability. Without depth perception, cleaners can only detect trash in immediate proximity. Thus, their vision is limited to a radius ρ of one node around their position, allowing them to detect and clean only nearby trash items. This means that the trash model can be defined as a matrix $\hat{Y}[i,j]$ initialised to 0, where each value is equal to the value of $Y[i,j]$ if an agent p_n is at a distance equal or less than ρ. Thus, the model is updated following: $\hat{Y}[i,j] \leftarrow Y[i,j] \iff v_{ij} \in \Theta(p_n)$. Both scouts and cleaners are capable of updating this model, although scouts can do so more effectively due to their advanced vision capabilities.

The contamination map is updated at each time step and shared among all agents through a central server, with communication and data processing delays and range being ignored. This share facilitates a cooperative task approach, allowing each agent to benefit from collective knowledge of the environment and improving overall efficiency and coordination. Based on the range of coverage, a matrix of covered nodes $U[i,j]$ can be defined as a matrix initialised to 0. This matrix is updated at each time step, and each position at a distance equal or less than ρ from the position of a vehicle becomes 1. This can be expressed as $U^{t+1}[i,j] \leftarrow \min(1, U^t + \Theta(\mathcal{P}))$, where each node takes the value of 1 if it has been covered at least once during the episode.

C. Deep Reinforcement Learning

1) State Representation: The framework has been formulated as a Markov Decision Process (MDP), in which each state s represents the current configuration of the agents and the

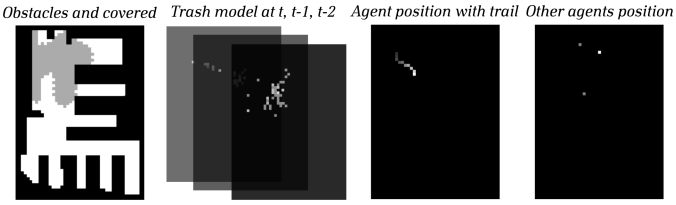

Fig. 2. Example of a state representation, composed of six image-like matrices. They are the input to the neural network.

environment. Actions are taken within a set of possible moves $a \in A$, producing a transition from one state to another. For each transition, a reward r will be obtained as a result of the action taken, quantitatively representing how positive that action has been for the overall objective. The MDP can be adapted to the multi-agent case by setting it as partially observable, so that the partial state is only accessible by observing agent n through the observation function, such that $o_n = \mathcal{O}(s)$. The observation function, $\mathcal{O}(s)$, is a process to map raw input data from a state s to a format that can be interpreted by the agent.

In this approach, the state will be an image-like representation, as it has been proven successful performance in conjunction with convolutional DNNs [11], [15]. As illustrated in Fig. 2, this representation is composed of six min-max normalized matrices of size $H \times W$. The first corresponds to a matrix with 0 for non-navigable positions, 1 for navigable positions, and 0.5 for navigable positions that have been covered. This matrix can be obtained by $M[i,j] - 0.5 \cdot U[i,j]$, and provide agents relevant spatial information about explored and unexplored areas. The next three matrices correspond to the trash model matrix $\hat{Y}[i,j]$ at time instants t, $t-1$, and $t-2$, which enables agents to predict trends in the movement of trash. The fifth matrix consists of zeros, except for the ten previous positions visited by the observing agent. This forms a trail with ten values from 0 to 1, creating a fading effect on the recent path of the agent to retain in short-term memory. The last matrix consists of zeros, except for the position of other agents. The positions occupied by the scout agents are set to 0.5, and 1 for those with cleaner agents. This enables more informed decisions based on the positioning of other agents.

2) Q-Function Optimization: DRL is a type of machine learning in which agents learn to make decisions by interacting with an environment in search of an optimal policy π^*. Through trial and error, that policy should maximize cumulative reward over time, estimated by the Q-values. In this work Double Deep Q-Learning (DDQL) [19] will be employed, a technique used in DRL to improve learning by applying two estimators of the action-value function to mitigate the overestimation of Q-values: Q-network and the Q-target. During the training process, the Q-network updates its parameters iteratively by taking steps to reduce the Bellman error [20], while the Q-target is periodically updated with the weights of the Q-network. Collected experiences will be stored in a prioritized experience replay memory [21]. Taking samples from this memory, batches are formed to train and update the weights of the network by backpropagation, taking stochastic gradient descent steps toward the direction that reduces the loss $\mathcal{L}$. The loss function is constructed from the quadratic difference between the predicted Q-value from the Q-network and the target Q-value,

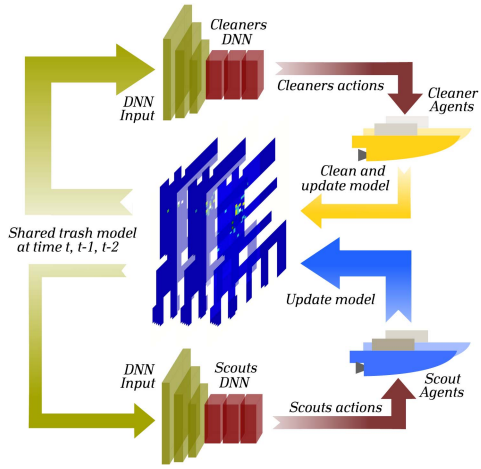

Fig. 3. Conceptual diagram of the framework presented in this work. The nexus of cooperation between the two teams is the trash model, which is the input for the DNN of each team. The scout team must provide the updated locations of the waste so that cleaners can collect it. Wider arrows indicate more influence.

following:

$$\mathcal{L}(\boldsymbol{\theta}) = \left[R + \gamma \cdot Q^{target}\left(s', \underset{a'}{\operatorname{argmax}}(Q(s';\boldsymbol{\theta})); \boldsymbol{\theta}^{-}\right) - Q(s,a;\boldsymbol{\theta})\right]^2 \quad (1)$$

where R is the immediate reward of the transition, γ is the discount factor that weights future rewards, $\boldsymbol{\theta}$ are the trainable weights of the Q-network DNN, and $\boldsymbol{\theta}^{-}$ are the frozen parameters of the Q-target DNN. In this multi-agent setup, two DNNs are employed, one shared for each team. Their architecture is similar to that used in [11]. It consists of a first feature extractor of the state representation, formed by three convolutional layers. The output is fed into three fully connected layers, followed by a Dueling Q-Network structure [22], which decomposes the Q-value estimation into two streams: the value function $V(s)$ and the advantage function $A(s,a)$.

Reward functions tailored to the specific role of each team have also been designed. Within each team, agent experiences are interchangeable, which enables the share of the prioritized experience replay memory. Additionally, a safety coordination constraint method is employed to prevent collisions, inspired by works like [11]. As seen in Fig. 3, the success of this system is based on cooperation between the two teams. The scout vehicles play a crucial role by providing accurate and real-time updates on trash locations. In turn, cleaners rely on this information to navigate efficiently to these locations. The better the scouts perform their mapping tasks, the easier it is for the cleaners to complete their work.

3) Reward Function: The reward function is a crucial element that numerically evaluates the optimality of the actions taken in terms of a final objective. This indicates how beneficial or detrimental a choice is in a specific state. This makes it one of the most important components in DRL, since it is the only feedback available to the agent to assess the value of its actions. Some good reward design principles are explored in [23], [24].

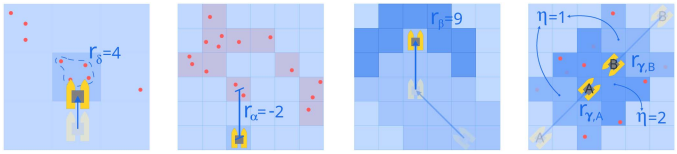

Fig. 4. Visual representations of the terms used in the reward functions.

For example, it is stated that penalizing each step taken tends to induce faster learning compared to simply rewarding the achievement of the goal. In addition, rewards should gradually increase as the agent approaches the goal, providing a progressive incentive. Following these recommendations, to avoid sparse rewards, the proposed reward functions for each type of agent are designed with these principles in mind, and are defined as follow:

$$R_C(s,a_n) = c_\alpha \cdot r_{\alpha,n} + c_\delta \cdot r_{\delta,n}$$
$$R_S(s,a_n) = c_\alpha \cdot r_{\alpha,n} + c_\beta \cdot r_{\beta,n} + c_\gamma \cdot r_{\gamma,n} \quad (2)$$

where $R_C(s,a_n)$ and $R_S(s,a_n)$ are the reward functions of the cleaner and scout team respectively, and c_α, c_β, c_γ and c_δ are parameters to be assessed in order to obtain an effective learning during training.

Both teams share a common penalization $r_{\alpha,n}$, in terms of the negative distance to the nearest known trash to the agent n if the agent has not cleaned in that step. This can be expressed as: $r_{\alpha,n} = -\min(\|p_n, B\|) \iff C(p_n, Y) > 0$. For the cleaner team, each agent is also rewarded for the number of trash items collected, following: $r_{\delta,n} = C(p_n, Y)$.

For the scout team, there will be two additional rewards to enhance exploration and exploitation. To boost an initial exploration phase, they receive a reward $r_{\beta,n}$ for covering non-covered areas, following: $r_{\beta,n} = \sum U^t - U^{t-1}$. This reward will diminish once most of the map has been explored, thus encouraging the rapid identification of high-contamination zones. The last reward $r_{\gamma,n}$ is defined by the changes produced in the model, calculated as the mismatch between two models within the coverage area. This is expressed as: $r_{\gamma,n} = \sum_{ij} \frac{\hat{Y}^t_{ij} - \hat{Y}^{t-1}_{ij}}{\eta(v,\mathcal{P})}$, where $\eta(v,\mathcal{P})$ is the number of vehicles p_n simultaneously covering a node v_{ij}, defined as: $\eta(v,\mathcal{P}) = |\{p_n \in \mathcal{P} \mid \|v, p_n\| < \rho\}|$. Examples of the proposed reward terms are shown in Fig. 4, where it can be observed how agents move into uncovered areas to obtain r_β rewards, or that the η value changes when the coverage areas of two agents overlap.

IV. Experiments

In this section, the evaluation of the proposed system is carried out, as well as the definition of the experiments performed. Since cleanup conditions can vary significantly, three evaluation scenarios have been proposed: Scenario A, characterized by its moderate convexity and ease of going between points in a straight line, Scenario B, representing a typical sport wharf design with narrow corridors and challenging access between points, and Scenario C, which represents a 35% larger map than the previous ones, with more complex areas and more difficult to access shapes, as well as randomly generated obstacles in each episode. These differences can be seen in Fig. 1. Scenarios A and B are represented by matrices of $H = 62 \times W = 46$ (2852 nodes), while the dimensions of Scenario C are $H = 62 \times W = 46$ (3850 nodes). For each scenario, a comparison will be made between some variations of DRL approach and five other state-of-the-art benchmark algorithms from the literature. Specifically, the algorithms employed correspond to lawn mower, random walker, PSO [6] and two greedy. One Greedy algorithm employs Euclidean distance to closest trash items, the other one employs A-Star path planning to find the path to the closest trash item without colliding with boundaries. All algorithms introduce a mechanism by which cleaning agents prioritize moving to a position containing trash whenever such an action is possible in the next step, regardless of the policy they follow. To assess cooperation within and between teams, training and simulations will be conducted with two agents per team for Scenarios A and B, and wit three agents per team for Scenario C. This is intended to study the scalability of the model to larger fleets and environments with minimal impact on the computational cost, since the agents of the same team share experiences and use a common neural network. All simulations and trainings have been conducted on a workstation running Ubuntu 22.04, equipped with a GPU Nvidia RTX 4090 25 GB. PyTorch libraries are used to define DNNs, and the code is available on Github https://github.com/amendb/HeterogeneousTrashCleanup for reproducing the results.

Due to the off-policy nature of DDDQL, it may benefit from using simpler algorithms to learn from certainly successful actions, instead of basing learning entirely on random decisions. For this reason, two types of training will be tested. One uses *epsilon*-greedy policies, where actions are completely randomly selected. In the other, half of the actions are chosen randomly, while the other half are selected by another algorithm to store these transitions in the memory replay. These trainings will be conducted with A-Star Greedy (DDDQL+Greedy) and PSO (DDDQL+PSO) in Table I, which have demonstrated notably good performance, and can help to learn more effectively from successful actions. Random actions are still required, as it is important to learn which choices lead to bad rewards.

The training process comprises a total of 60,000 episodes. In each episode, as described in the Section III-B section, agents randomly depart from a set of predefined initial positions. Since safety regulations in aquatic environments require that vehicles have a sufficient energy margin, it is assumed that this requirement is met at a higher level and that each agent is capable of carrying out a mission of 150 movement steps. Each episode generates a unique trash distribution. A pollution hotspot is randomly selected from the entire set of visitable locations in the scenario, and around this random hotspot, trash items are randomly scattered according to a multivariate normal distribution, ensuring a natural clustering effect similar to real-world trash accumulation patterns. The number of trash items present in each episode is also not fixed. Instead, it is drawn from a normal distribution, which introduces more variability between episodes. The environmental currents, represented as wind direction, are also random in each episode. The wind vector is sampled from a uniform distribution for the X and Y axes. Beyond the primary influence of the wind, each trash item experiences an independent random motion component at each

TABLE I
METRICS COMPARISON BETWEEN DDDQL TRAININGS AND OTHER ALGORITHMS AT THE END OF AN EVALUATION SET OF 100 EPISODES

Algorithm	Scenario A				Scenario B				Scenario C				Average scenario time of computation (ms)		
	PTC		MSE		PTC		MSE		PTC		MSE				
	Mean	CI 95%	Mean	CI 95%	Mean	CI 95%	Mean	CI 95%	Mean	CI 95%	Mean	CI 95%	A	B	C
DDDQL	98.81	0.43	0.0011	0.0003	96.14	1.08	0.0019	0.0005	76.09	6.49	0.0032	0.0008	5.88	5.82	9.79
DDDQL + Greedy	**99.02**	0.32	**0.0008**	0.0002	96.02	1.95	0.0029	0.0018	**91.81**	2.68	**0.0021**	0.0005	5.90	5.88	9.72
DDDQL + PSO	**99.02**	0.28	0.0010	0.0003	**97.63**	0.65	**0.0016**	0.0005	84.33	5.26	0.0027	0.0012	5.92	5.73	9.62
Random Walker	47.52	6.15	0.0263	0.0043	50.98	6.43	0.0273	0.0053	29.17	7.62	0.0445	0.0063	**0.42**	**0.47**	0.69
Lawn Mower	43.49	6.63	0.0307	0.0045	36.70	7.24	0.0310	0.0054	26.38	7.45	0.0439	0.0064	0.44	0.49	**0.67**
PSO	93.97	1.82	0.0045	0.0010	75.66	6.63	0.0173	0.0052	53.75	9.03	0.0280	0.0060	0.51	0.53	0.76
Euclidean Greedy	94.22	3.68	0.0035	0.0019	73.22	7.19	0.0124	0.0037	48.29	9.17	0.0344	0.0070	1.79	1.61	2.52
A-Star Greedy	97.35	2.08	0.0027	0.0024	90.19	4.22	0.0088	0.0041	51.34	9.18	0.0331	0.0071	45.13	71.33	93.14

Highlighted values correspond to the best performance value. Average computation time per scenario for all agents is also included.

step of the environment. This randomness introduces motion patterns that increase the unpredictability of the trash dynamics, making the environment more challenging and realistic. In the context of Scenario C, the obstacles are also randomly generated in each episode.

The training parameters and reward function ponderation mentioned below have been obtained using a hyperparameter optimization algorithm based on Bayesian optimization called Tree-structured Parzen Estimator (TPE),[3] which provides a systematic search for optimal values. Transitions from each team are stored in a prioritized experience replay memory of 1×10^6 experiences. Memory batches of 128 are sampled to adjust the network weights with a learning rate of 4.9×10^{-4}. The final policy chosen is the one that obtains the best performance on average. The reward function ponderations defined in Section III-C3 were fixed to: $c_\alpha = 2.63$, $c_\delta = 14.33$, $c_\beta = 5.83$, $c_\gamma = 1.73$. With the same parameter settings DRL algorithms have been able to adapt to different scenarios with different characteristics without the need for prior knowledge thanks to its model-free nature. For the evaluation process, the average results of the same 100 random episodes have been recorded for each algorithm. These episodes have never been seen during the DRL training process. This diversity, along with the different scenarios, makes it possible to generalize the evaluation of the effectiveness of the algorithms.

Two metrics will be used to quantitatively measure the performance of them:

- **PTC**: The percentage of trash cleaned (PTC) over time, that quantifies the effectiveness of the cleaning algorithm. It is calculated as the ratio of the final amount of trash removed by the subset of cleaners agents from $\mathcal{P}$ to the total amount of trash present in the environment K at the beginning of the episode:

$$PTC(t) = \frac{\sum_{\tau=0}^{t} C(\mathcal{P}, Y^\tau)}{K} \cdot 100$$

- **MSE**: Mean Squared Error over time between the model $\hat{Y}$ and the trash ground truth Y. As $Y, \hat{Y}$ are sparse matrices and the elementwise error will produce high error variations, a Gaussian Filter G with $\sigma = 1$ is applied to compare the density of the trash distribution instead:

$$MSE(t) = \frac{1}{H \times W} \sum_{\tau=0}^{t} \left(G \circledast Y^\tau - G \circledast \hat{Y}^\tau \right)^2$$

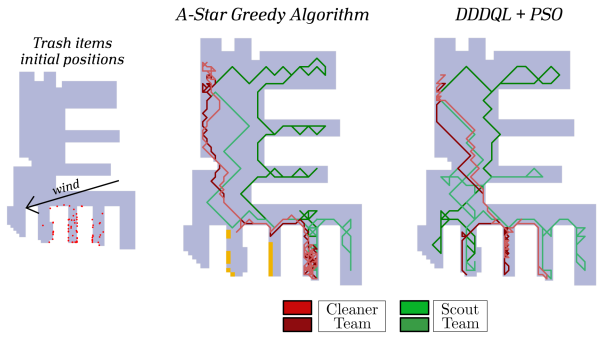

Fig. 5. Comparison paths in a random episode at Scenario B between A-Star Greedy and DDDQL+PSO algorithms. Nodes with trash remaining at the end of the episode are marked in yellow. The starting positions of trash items are shown in red on the map on the left, along with the wind direction.

Evaluation results can be observed in Table I, where the final average metrics are summarized and compared. The results show, as expected, that the algorithms that follow fixed patterns (lawn mower and random walker) perform significantly worse than the other approaches, both in terms of average MSE and PCT. The PSO and Euclidean Greedy algorithms demonstrate strong performance, occasionally outperforming each other depending on the scenario. They are competitive options in terms of efficiency and challenging DRL algorithms in Scenario A, where the open layout allows simpler algorithms to achieve remarkable results, but declining in Scenario B,[4] and even more noticeably in Scenario C. However, PSO is the best performing baseline in Scenario C. The improvement of the Greedy algorithm with the A-Star path planner is substantial, especially in Scenario B. Due to the large number of corridors in this scenario, calculating distances using the A-Star planner allows for more effective navigation compared to using Euclidean distances when facing the edges of the map. Fig. 5 shows the performance of the A-Star Greedy algorithm versus the best DRL training in a

[3] https://optuna.org/

[4] Algorithms behaviors can be dynamically visualized in the attached video.

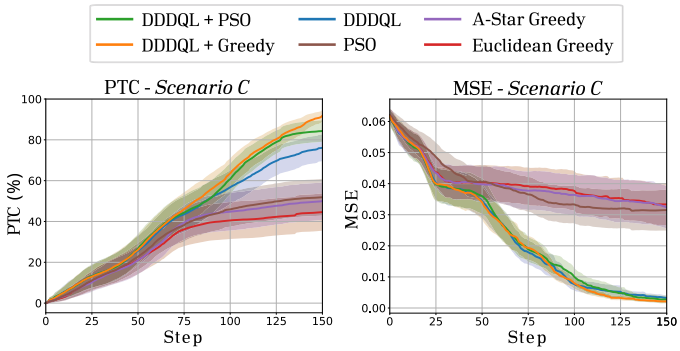

Fig. 6. Graphical comparison over time of the percentage of trash cleaned (left) and mean squared error (right) for each of the best algorithms in Scenario C, with confidence intervals represented as shaded regions.

specific episode. In this it can be appreciated how the DRL algorithm is able to collect all the trash elements, while the A-Star Greedy leaves several nodes with trash due to the lack of exploration in several of the corridors. In Scenario A, the easiest scenario, it comes very close on the PTC metric to DRL trainings, but obtains an MSE three times worse, and larger confidence intervals. Despite the good results of the baselines, none of them surpass the results of DRL trainings, whose best trainings obtain outstanding results in all scenarios, with a relatively moderate inference time. In particular, in Scenario C, the best DRL approach outperforms the best baselines (A-Star Greedy and PSO algorithms) by almost 40% in PTC metric, and more than 15 times better MSE. Even the help of the A-Star path planner is not enough to get Greedy decent results. In this environment, characterized by complex structure, larger fleet and map size, and random obstacles, the drop in baselines is very significant, as seen in Fig. 6. This proves to be a more challenging scenario than the previous ones, and DRL demonstrate superior adaptability to demanding environments and robustness to changes.

In terms of MSE, it is important to keep in mind that as more trash is cleaned up, the MSE tends to decrease naturally, since the model to be estimated becomes simpler. The DRL-based algorithms consistently deliver clearly superior results compared to other methods, as seen in Table I. The values achieved by the DRL-based approaches are significantly lower, highlighting their ability to generate more accurate representations of the environment. This trend is further reinforced by the smaller confidence intervals (CI) observed in the DRL results, indicating that the DRL-based algorithms not only perform better on average but also exhibit less variability between different episodes, making them more robust across diverse scenarios. Table I also includes the inference times for each algorithm measured in milliseconds. Since A-Star Greedy relies on explicit calculations for distances to trash items, agent positions and path planning, its computational effort is substantially higher than DRL, which maintains better scalability. DRL-based approaches have slightly higher computational times compared to simpler baselines such as PSO, due to the decision making process employing DNNs. However, this computational cost of DRL methods is compensated by their higher adaptability and generalized performance. Furthermore, assuming an average ASV speed of between 1 and 2 meters per second, depending on the type of agent, inference times on the order of milliseconds can be negligible in a real implementation. Additionally, the reported computation time corresponds to obtaining actions for the entire fleet. Since each agent can compute its action independently, this time can be significantly reduced through parallelization.

Among the DRL algorithms, the best-performing models are consistently those trained using examples of heuristic actions, whether from A-Star Greedy or PSO strategies. At worst, these models perform at least as well as those trained with entirely random actions, indicating that this approach does not introduce any negative effects. However, its true advantage becomes evident in that the top-performing DRL models always follow this training strategy. This effect is particularly noticeable in Scenario C, where DRL models trained with heuristic actions show a significant improvement over those trained without them, observable in Fig. 6 and Table I. This suggests that as the exploration space becomes more complex and extensive, taking advantage of heuristic actions during training helps the model to focus on more effective strategies, reducing the need for inefficient exploration. In contrast, when the task is simpler with less restrictive boundaries, as in Scenario A, the full training proves sufficient to achieve similar results without the need for samples of other heuristics.

Greedy algorithms obtain competitive results in problems where the objective function is submodular. In simple terms, submodularity can be defined as an intuitive notion of diminishing returns. As detailed in work [25], it indicates that the benefits of adding resources (agents or efforts) diminishes as the set grows. In submodular problems with constraints (such as a limited number of agents, trash items, or moves), greedy algorithm can achieve a solution that approaches the optimum by maximizing marginal gains at each step [26]. However, its short-term decision-making limits its effectiveness compared to DRL and often leads to the policy getting stuck in local minima, particularly evident in Scenario C. In contrast, DRL optimizes for long-term reward, allowing agents to plan more strategically. Furthermore, DRL's adaptability is evidenced from the better results despite the variability between episodes and scenarios. Finally, algorithms like greedy require preprocessing and careful selection of the most critical information, tailored to the specific logic of the algorithm. In this work, greedy needs the distances to all trash items to be calculated to know the closest one. In contrast, DRL assimilates this information directly from the raw state representations and extracts the relevant features needed for decision making without manual intervention. This superiority of DRL is especially pronounced as the difficulty of environments increases, as can be seen in Scenarios B and C.

An ablation study shown in Fig. 7 has been conducted in Scenario B to analyze the impact of removing specific channels from the state observation function. The first experiment, Independent Models, examines the effect of the teams not sharing the contamination model. Now, cleaners rely solely on their limited field of view without access to the enhanced contamination model created by scouting teams. As a result, trash collection is affected. The second experiment (orange line) removes the channel for the positions of other agents. This lack of positional knowledge reduces coordination, which worsens the metrics, leading to inefficient coverage. These studies highlight the

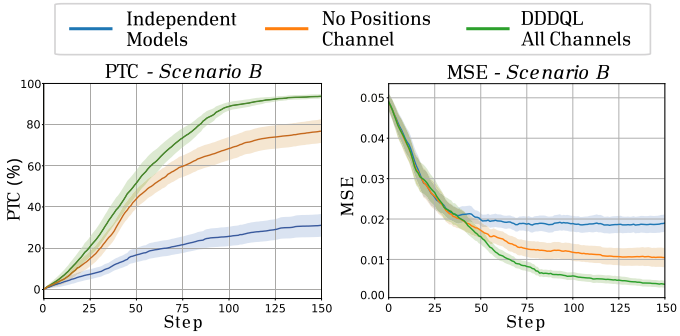

Fig. 7. Results of the ablation study on the observation function channels of the proposed algorithm, with confidence intervals represented as shaded regions.

crucial role that information exchange plays in enabling effective cooperation between fleets.

V. Conclusion

This letter presents a model-free DRL-based framework for IPP with heterogeneous fleets of ASVs for floating waste collection. The proposed system uses two teams of ASVs: scouts, responsible for mapping the trash distribution, and cleaners, in charge of waste collection. Coordination between these teams is achieved by information sharing through the observation function of a DRL algorithm. This allows agents to learn strategies to maximize cleaning efficiency. Simulations demonstrate the effectiveness of the approach, outperforming baselines in three scenarios, with different complexities. The DRL-based algorithms exhibit superior adaptability to complex environments, with better performance metrics. Training with other heuristic actions further enhances the performance of DRL, especially in scenarios where the action exploration space is more complex and extensive. Algorithms such as Greedy and PSO algorithms require prior tuning for each environment, while DRL, being model-free, adapts without prior knowledge and generalizes across situations. Moreover, the best baseline (A-Star Greedy) has an inference time over 10 times longer on average, highlighting DRL's efficiency advantage. Due to the high economic and engineering challenges, real experiments have not been feasible. However, a mandatory next step for this work is to translate the proposed framework to a real experimental setting.

References

[1] World Economic Forum, Ellen MacArthur Foundation, and McKinsey & Company, "The new plastics economy: Rethinking the future of plastics," 2016, [Online]. Available: https://www.ellenmacarthurfoundation.org/the-new-plastics-economyrethinking-the-future-of-plastics.

[2] Y. Cai and S. X. Yang, "A survey on multi-robot systems," in *Proc. World Automat. Congr.*, 2012, pp. 1–6.

[3] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: A survey," *ACM Comput. Surv.*, vol. 52, no. 2, pp. 1–31, 2019.

[4] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Auton. Agents Multi-Agent Syst.*, vol. 33, no. 6, pp. 750–797, 2019.

[5] M. Popović, J. Ott, J. Rückin, and M. J. Kochenderfer, "Learning-based methods for adaptive informative path planning," *Robot. Auton. Syst.*, vol. 179, 2024, Art. no. 104727.

[6] M. J. T. Kathen, F. Peralta, P. Johnson, I. J. Flores, and D. G. Reina, "AquaFel-PSO: An informative path planning for water resources monitoring using autonomous surface vehicles based on multi-modal PSO and federated learning," *Ocean Eng.*, vol. 311, 2024, Art. no. 118787.

[7] F. Peralta, D. G. Reina, and S. L. Toral, "Water quality online modeling using multi-objective and multi-agent Bayesian optimization with region partitioning," *Mechatronics*, vol. 91, 2023, Art. no. 102953.

[8] M. Grzelczak and P. Duch, "Deep reinforcement learning algorithms for path planning domain in grid-like environment," *Appl. Sci.*, vol. 11, no. 23, 2021, Art. no. 11335.

[9] A. Vashisth, J. Rückin, F. Magistri, C. Stachniss, and M. Popović, "Deep reinforcement learning with dynamic graphs for adaptive informative path planning," *IEEE Robot. Automat. Lett.*, vol. 9, no. 9, pp. 7747–7754, Sep. 2024.

[10] X. Lei, Z. Zhang, and P. Dong, "Dynamic path planning of unknown environment based on deep reinforcement learning," *J. Robot.*, vol. 2018, no. 1, 2018, Art. no. 5781591.

[11] S. Y. Luis, D. Shutin, J. M. Gómez, D. G. Reina, and S. T. Marín, "Deep reinforcement multiagent learning framework for information gathering with local Gaussian processes for water monitoring," *Adv. Intell. Syst.*, vol. 6, no. 8, Aug. 2024, Art. no. 2300850.

[12] Y. Yang, L. Juntao, and P. Lingling, "Multi-robot path planning based on a deep reinforcement learning DQN algorithm," *CAAI Trans. Intell. Technol.*, vol. 5, no. 3, pp. 177–183, 2020.

[13] J. Woo and N. Kim, "Collision avoidance for an unmanned surface vehicle using deep reinforcement learning," *Ocean Eng.*, vol. 199, 2020, Art. no. 107001.

[14] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3389–3396.

[15] A. M. Barrionuevo, S. Y. Luis, D. G. Reina, and S. L. Toral Marín, "Informative deep reinforcement path planning for heterogeneous autonomous surface vehicles in large water resources," *IEEE Access*, vol. 12, pp. 71835–71852, 2024.

[16] Y. Gao et al., "Asymmetric self-play-enabled intelligent heterogeneous multirobot catching system using deep multiagent reinforcement learning," *IEEE Trans. Robot.*, vol. 39, no. 4, pp. 2603–2622, Aug. 2023.

[17] A. Akib et al., "Unmanned floating waste collecting robot," in *Proc. TENCON 2019 - IEEE Region 10 Conf.*, 2019, pp. 2645–2650.

[18] T. Deng, X. Xu, Z. Ding, X. Xiao, M. Zhu, and K. Peng, "Automatic collaborative water surface coverage and cleaning strategy of UAV and USVs," *Digit. Commun. Networks*, 2022, doi: 10.1016/j.dcan.2022.12.014, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864822002826.

[19] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, vol. 30, pp. 2094–2100.

[20] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton Univ. Press, 1957.

[21] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.

[22] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.

[23] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone, "Reward (MIS) design for autonomous driving," *Artif. Intell.*, vol. 316, 2023, Art. no. 103829.

[24] H. Sowerby, Z. Zhou, and M. L. Littman, "Designing rewards for fast learning," 2022, *arXiv:2205.15400*.

[25] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *J. Artif. Intell. Res.*, vol. 42, pp. 427–486, 2017.

[26] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Math. Program.*, vol. 14, pp. 265–294, 1978.