UDC 004.8 : 004.94 https://doi.org/10.20535.2786-8729.5.2024/315700

NEURAL NETWORK MODEL FOR AUTONOMOUS NAVIGATION OF A WATER DRONE

Hlib Chekmezov*
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
https://orcid.org/0009-0003-0404-7834

Oleksii Molchanov
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
https://orcid.org/0000-0001-8384-0918

*Corresponding author: chekmezov.hlib@lll.kpi.ua

Water drones have significant potential for use in environmental monitoring, search and rescue operations, and marine infrastructure inspection, but the specific conditions of the water environment make it difficult to implement stable autonomous navigation.

The object of research presented in this paper is the machine learning process for autonomous navigation of a water drone model in a simulated water environment. The purpose of the study is to implement a neural network model for autonomous navigation of a water drone using a reinforcement learning method that provides improved obstacle avoidance and adaptation to water currents.

To achieve this purpose, a new neural network model for autonomous drone navigation in the water environment based on the reinforcement learning method is proposed, which differs from the existing ones in that it uses an improved drone control algorithm that takes into account the speed and direction of the water current, which makes it possible to stabilize the process of generating neural network coefficients.

To ensure an effective learning process and optimization of the model, a simulation training environment was developed using the USVSim simulator, which contains various factors that interfere with the drone's movement, such as water current and the presence of other objects. The water drone, acting as an agent, gradually learns to choose the most effective actions to maximize positive rewards through trial and error, interacting with the environment and adapting to changing conditions. This process takes place through the use of a Deep Q-Network: the drone provides the value of its current state to a deep neural network; the neural network processes the data, predicts the value of the most effective action, and gives it to the agent. The current state of the drone is information in the form of a set of sensor readings measuring the distance to the nearest obstacles, drone's heading and current distance to goal. The value of the effective action received from the neural network is converted into a command for the rudder that the drone can understand. The value of the drone's thruster power is calculated by separate formulas using trigonometric functions.

The results of the study showed that the use of the proposed model allows the drone to make decisions in a dynamic water environment when rapid adaptation to changes is required. The model successfully adapted its strategy based on feedback from the environment, so it can be concluded that the implemented model shows significant potential for further research and applications in the field of autonomous water drones, especially in changing and unpredictable environments.

Key words: water drone, machine learning, DQN, autonomous navigation, algorithm.

1. Introduction

Autonomous water drones, or unmanned surface vehicles (USVs), have attracted considerable interest in recent years due to their potential to revolutionize industries such as environmental monitoring [1], maritime search and rescue [2] and infrastructure inspection. These drones are increasingly valued for their ability to access complex environments, collect real-time data, and perform tasks that may be too dangerous or inaccessible for human operators. However, effective

water drone deployment faces unique challenges because aquatic environments present navigational complexities unlike those found on land or in the air. Various water factors, such as water currents, for example, can vary significantly in strength and direction, requiring constant adaptation. In addition, water drones must be prepared for unpredictable obstacles, such as floating debris and other vessels, which require immediate, context-sensitive responses.

Static control methods used in autonomous vehicles are generally effective in predictable, controlled environments. However, in real-world aquatic environments where rapid environmental changes require real-time decision making, they tend to underperform. Reinforcement Learning (RL) is a method that allows aquatic drones to iteratively learn and optimize their responses to environmental challenges through a process of trial and error. By constantly interacting with the environment, RL-trained drones can gain a deeper "understanding" of complex environments, allowing them to navigate more effectively. The development of new and improvement of existing machine learning methods for drones can accelerate the process of model training and increase its adaptability in a dynamic aquatic environment. These advances will not only increase the autonomy and efficiency of water drones, but also expand the range of their potential applications, making them more adaptable to work in complex real-world environments. Therefore, the autonomous navigation of water drones using machine learning is an urgent task to solve.

2. Literature review and problem statement

While analyzing approaches to autonomous navigation, it was found that most of them are focused on air [3] and land [4] environments. Such approaches, however, cannot be directly applied in aquatic environments due to specific conditions.

Modern methods for solving the problem of autonomous navigation are divided into two main types: static and dynamic. Static methods are typically based on the use of a pre-created map or fixed route, which limits their ability to adapt to changes in the environment. These methods are well suited for stable and predictable environments (static environments). Dynamic methods, on the other hand, are designed to adapt in real time, using data from sensors to make navigation decisions even in the face of uncertainty and unpredictable changes (dynamic environments) [5].

One of the most prominent examples of static methods is the use of the Dijkstra algorithm, a classic algorithm for finding an optimal path developed by Dutch scientist Edsger Dijkstra in 1959. This algorithm is used to find the shortest path in a weighted graph from one vertex to all others [6]. By using a pre-created map, Dijkstra's algorithm can determine the optimal route between given points, taking into account all known obstacles on the way and building the shortest path in a static environment.

Dynamic methods to autonomous navigation, as opposed to static methods, use real-time data from sensors and other sources, enabling systems to respond quickly to changes in the environment. They continuously take into account changing navigation conditions and unpredictable obstacles, providing greater flexibility and the ability to make independent decisions in complex situations.

The Rapidly Exploring Random Tree (RRT) algorithm is an example of a dynamic methods to autonomous navigation. It builds a tree of possible trajectories by randomly selecting points in space and connecting them to the nearest vertices of the tree [7]. If the connection passes through free space and meets the constraints, the new state is added to the tree. The tree grows in the direction of large unexplored areas, which helps to cover the entire space. The length of the connections is limited by the growth factor: if a random point is too far away, a state at the maximum allowed distance is added. Thanks to this, the RRT algorithm works efficiently in complex environments with unpredictable obstacles, quickly finding a passable path, although without a guarantee of optimality.

Another interesting dynamic method for solving the problem of autonomous navigation is reinforcement learning, a machine learning method where an agent learns to make decisions through interaction with its environment [8]. An autonomous agent is any entity (or system) that can make decisions and act according to its environment. This method is based on the state-action-reward sequence, where the action chosen by the agent brings a reward or punishment. Over time, the agent, learning from rewards and punishments, masters actions that help achieve its goal in the environment.

To solve the problem of autonomous navigation of a water drone, the agent must receive a large positive reward when reaching the target point, and a large negative reward when encountering an obstacle.

According to the study [9], unmanned surface vessels (or water drones) face a number of challenges when trying to achieve autonomy. One of the key challenges is environmental perception, as safe autonomous navigation requires accurate recognition and assessment of sea conditions. An efficient decision-making process is also important: water drones need to process information in real time and autonomously select appropriate actions for maneuvering. The control algorithms must be advanced enough to manage the dynamic movements of the vessel, adapting to changes in maneuvering conditions. In addition, the level of autonomy must vary depending on the situation, which requires flexibility and additional customization of the systems. Finally, for reliability, comprehensive testing under different conditions is required to verify the capabilities of the autonomous mooring system and ensure its safety and stability.

One of the possible solutions to the problem of autonomous navigation of water drones is proposed in the research [10]. This research is based on the use of machine learning, namely the use of various Deep Q-Network (DQN) optimization algorithms. The results show that the proposed implementation is capable of accurately avoiding obstacles, recognizes buoys, and autonomously makes collision-avoidance decisions in complex environments with static obstacles and buoys. The authors note the problem of exposure to external factors like wind and waves. To solve the problem of autonomous navigation in such conditions, they propose an autonomous port collision avoidance algorithm based on deep reinforcement learning: that is, a dynamic method instead of a static one. However, the consideration of the influence of external factors in their study is used to select the method itself to solve this problem because they note the influence of these factors specifically on the buoys. These factors are not taken into account in the training process of the drone itself.

In the research [11], a method is proposed to improve the autopilot capabilities of surface unmanned vessels by combining a path planning strategy with a collision avoidance function under uncertainty. The solution is based on DQN, which allows the agent to learn in a visually modeled environment. To improve the results, an Artificial Potential Field (APF) [12] method is used to optimize the action space and the DQN reward function. Despite the fact that this method emphasizes the solution of the collision avoidance problem without taking into account external water factors, it effectively copes with the task of path planning and collision avoidance.

Research [13] proposes a smoothed path planning algorithm A* for an autonomous system. The authors note that their proposed algorithm is designed to improve the feasibility of collision avoidance path planning in a real environment. Although A* is more commonly used for static environments where the state of objects is known in advance, the authors were able to adapt it for path planning with both static and dynamic obstacles.

The issues of ensuring the autonomy of water drones, taking into account such features as difficult water conditions (like water currents) and water obstacles remain insufficiently researched. Although some research in this area has already been conducted and is ongoing, the overall knowledge base on navigation algorithms and methods for autonomous water drones is still limited. The aquatic environment requires the development of adapted solutions that can take into account water dynamics, including water current directions and velocities, as well as effective interaction with obstacles. This creates a need for new research and development aimed at creating and optimizing autonomous navigation and decision-making solutions for water drones that can operate effectively in the water environment.

3. The aim and objectives of the study

The object of study presented in this article is the machine learning process for autonomous navigation of a water drone model in a simulated water environment.

The purpose of this study is to implement a neural network model for autonomous navigation of a water drone using the reinforcement learning method, which provides increased efficiency of

obstacle avoidance and adaptation to water currents. This study seeks to enhance the reliability and flexibility of navigation in unpredictable water conditions through reinforcement learning techniques.

To achieve this aim, the following tasks are set:

– to develop and configure a neural network model for autonomous navigation and create a simulation for training and testing the implemented model that can adapt to varying water currents and navigate around obstacles effectively.

– to make a comparative analysis of training results using proposed model for cases with and without thruster power control.

4. The study materials and methods for neural network model for autonomous navigation of a water drone

The research methods of the article are the search and analysis of theoretical material on possible solutions to the autonomous navigation problem of water drones, external factors of the water environment and their impact. The formalization method is used for an algorithm that takes into account the direction and speed of the water current in the learning process. Methods of modeling, experiment, observation, measurement, analogy and testing are also used to verify the performance of the proposed model under simulation conditions.

4.1. Reinforcement learning for autonomous water drones

To solve the problem of autonomous drone navigation, a model based on reinforcement learning [8] with deep neural network [14], that takes into account the specifics of the water environment, including the current and the presence of obstacles is proposed. This allows the USV to adapt to the dynamic conditions of the water environment. The drone, acting as an agent, learns through trial and error, gradually mastering the optimal actions to maximize rewards. Interacting with the environment, it constantly adapts to changes, improving its navigation skills in the process.

This can be achieved by following these steps:

1. Analyze the elements of the learning environment: define the task for the drone and the factors that will affect the learning process, such as the direction and speed of the currents, obstacles, etc. This will help to understand which variables are important for effective training.

2. Setting up the environment parameters: after the analysis, an environment that reflects the conditions of the task should be created. For a water drone, this may include adding obstacles, modeling water current, etc.

3. Defining the reward function: creating a system of rewards and penalties that guides the model to complete the task. For example, a drone receives a reward for reaching a goal and a penalty for colliding with obstacles.

4. Hyperparameter tuning: identify and set optimal values for hyperparameters, such as the learning factor, to help ensure the stability of the learning process.

5. Configuring additional parameters related to the model training process: adjust certain settings required for training. For the water drone in this study, this includes setting the logic that adjusts the drone's thruster power depending on the speed and direction of the water current.

6. Starting the learning process: after the settings, the learning process starts. Initially, the agent explores the environment with random actions, but over time, it adapts, determining the best strategies.

7. Monitoring and correction of the training process: It is important to monitor the training results regularly. If the model is not achieving the desired results, learning process should be stopped, parameters should be updated, and training should be restarted.

8. Evaluation of results: if the model completes the task, you need to evaluate the quality of its work. If the results do not meet expectations, you should return to the parameter settings and restart the training process.

Such steps allow the drone to analyze the current state of the environment in real time without the need for static maps or fixed routes. Because the drone is adaptable to changing conditions, the learning process becomes more stable, which is key to successfully achieving the goal.

To implement this model, it was decided to use DQN, an algorithm in deep reinforcement learning that combines Q-learning with a deep neural network (Convolutional Neural Network) to teach it to approximate the Q function. This function establishes a connection between state-action pairs and their expected rewards [15]. The advantages of DQN over classical Q-learning are scalability to high-dimensional spaces (neural networks allow to approximate the Q-function in environments with large or continuous state spaces), stability and efficiency of training. This is achieved by using a replay memory and two neural networks.

However, DQN also has its limitations. One of the main disadvantages is a strong dependence on hyperparameters and random values. Hyperparameters are settings that are not optimized by the model itself but are set before training and affect the algorithm's performance. These initial parameters determine the model training process and significantly affect its efficiency.

Randomness also plays an important role in DQN: it appears during the initialization of neural network parameters, in the research policy, in the sampling from the playback memory, and can also be inherent in the environment in which the agent is trained. For example, if the environment contains random elements (such as unpredictable rewards or punishments), this creates an additional level of uncertainty. In such conditions, the agent needs to learn how to make optimal decisions in situations of uncertainty, which greatly complicates the learning process. Due to the presence of such characteristics in DQN, there is a consequence that each run can produce a different result: randomness in initialization and training means that repeated runs of the same algorithm can lead to different results.

Updating network values in a DQN is based on the Bellman equation, which is presented in formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a';\theta^{-}) - Q(s,a;\theta)], \quad (1)$$

where, θ – the weights of the main network; θ^{-} – the weights of the target network; s – the current state; a – the selected action; r – the calculated reward; s' – the next state; $\max_{a'} Q(s',a';\theta^{-})$ – the maximum value of Q for the next state.

4.2. Integration of water current speed and direction into the training process

As mentioned above, the water environment has its own peculiarities that need to be taken into account when training the model. The water current is an important factor affecting this process. If the drone's thruster is turned off and the rudder remains in the same position, the drone is at rest, and its direction and speed change only under the influence of external forces, such as the water current.

For a more stable learning process, it is advisable to provide as many stable values as possible during the training process. This reduces fluctuations in the neural network results and forecast errors, which makes training more uniform. Such conditions contribute to better memorization and generalization of information by the neural network, as well as reduce the risk of excessive changes in model weights during training.

The speed of the drone during training is a key parameter. In an environment without external factors, the drone will move at a constant speed at a fixed thruster power. However, in the case of an environment with external influences, such as a water current, its speed can change even with a constant thruster power. For example, moving in the direction of the current, the drone will gain additional acceleration, while moving against the current will slow it down. Thus, if you leave the thruster power fixed, moving along the water current will increase the speed, and moving against the water current will decrease it.

Such changes in speed can negatively affect the learning process. For example, when moving along the current, the drone may not have time to make the right decision in the current situation, and in a random environment, the likelihood of such situations is quite high. If such conditions are repeated many times, this can lead to the drone learning the wrong actions that do not correspond to the desired result. Therefore, to ensure the most stable drone speed in the environment, it is important to take into account the state of the environment and constantly adjust key parameters to ensure optimal learning conditions.

The proposed model uses an additional algorithm for adjusting the thruster power according to the direction and speed of the water current. This algorithm is based on the use of trigonometric formulas to calculate the current angle between the drone's direction vector and the water current vector, as well as monotonic cubic interpolation to determine the required power corresponding to the calculated target drone speed. Figure 1 shows the possible drone and water current direction vectors and the corresponding angles: α – the angle between the unit vector of the X-axis and the water current direction vector; ψ – the angle between the unit vector of the X-axis and the drone direction vector; δ – the resulting angle between the current direction vector and the drone direction vector; $\bar{v}_{cur\_xy}$ – the current direction vector; $\bar{b}$ is the drone direction vector. If the value of the angle δ is greater than 180 degrees, it is adjusted by formula to be in the range [0; π]:

$$\delta = 2\pi - \delta. \tag{2}$$

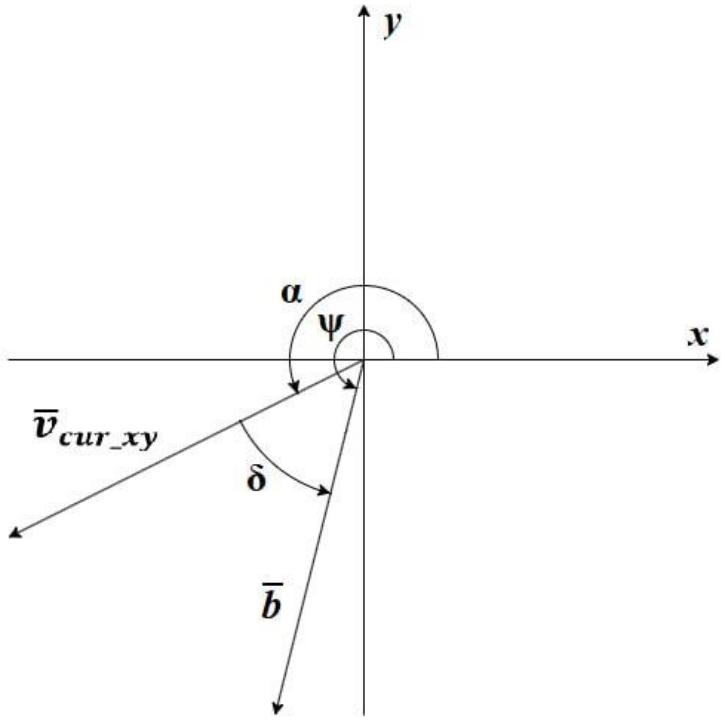

Fig. 1. Possible location of the drone and water current direction vectors and corresponding angles

The next step is to implement the concept of using the obtained δ value to adjust the motor power. First, we need to determine the desired speed for training the model. After that, we need to develop logic that will adjust this speed according to the value of the angle δ. For example, the abstract value of v_{tar_start} can be taken as the initial desired value, and v_{tar} can be taken as the resulting velocity value. If the angle δ is less than 90 degrees, then the value of v_{tar} is calculated using the formula:

$$v_{tar} = v_{tar\_start} - \frac{\left|\bar{v}_{cur\_xy}\right| * (\frac{\pi}{2} - \delta)}{\frac{\pi}{2}}. \tag{3}$$

If the angle δ is greater than 90 degrees, the value of v_{tar} is formed by formula:

$$v_{tar} = v_{tar\_start} + \frac{\left|\bar{v}_{cur\_xy}\right| * (\delta - \frac{\pi}{2})}{\frac{\pi}{2}}. \tag{4}$$

So, if the angle between the drone's heading vector and the water current vector is acute, it means that the current is favoring the desired drone speed. In this case, we need to subtract a portion of the water current speed from the expected drone speed. If the angle is obtuse, the current is resisting the drone's movement — in this case we need to add part of the water current speed to the desired drone speed to compensate for its influence. The value of the corrected drone speed can vary within the range $[v_{tar\_start} - v_{cur}; v_{tar\_start} + v_{cur}]$, where, v_{tar_start} is the initial desired drone speed and v_{cur} is the water current speed.

Figure 1 shows the main elements that need to be found and considered in the proposed model.

4.3. Training environment

Training water drones directly in real-world conditions has practical limitations. Operating in vivo carries risks, including possible hardware damage from collisions, environmental impact, and high operational costs. Direct training of hardware is often inefficient and can lead to costly failures. To mitigate these issues, a robust simulation environment must be used. A simulated environment allows for controlled experiments without risk, enabling drones to safely learn and test navigation strategies.

The USVSim simulator [16] is used to create a prototype of a neural network model proposed in this study. It was chosen because it is a powerful simulation product that contains many aspects that must be taken into account for a correct and convenient simulation process. It provides several pre-created boats and the ability to customize various external factors inherent in the water environment. A view of one of the prepared environments for training the drone is shown in Figure 2.

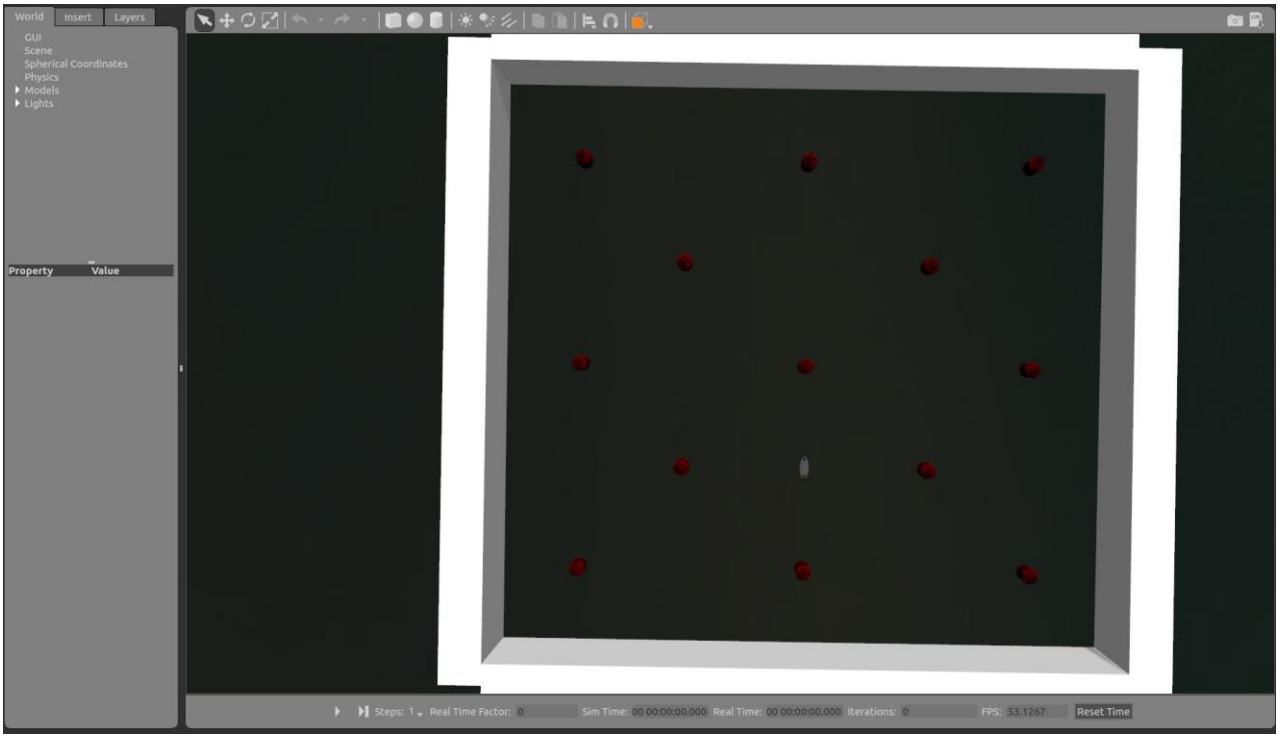

Fig. 2. The environment in which the drone training took place

A computer with an Intel Core i5-7300HQ processor, 16 GB of RAM, NVIDIA GeForce GTX 1050 GPU and Ubuntu 16.04.7 LTS (Xenial Xerus) operating system is used to simulate the aquatic environment and conduct the experiments.

The range of possible thruster power values is set to [0; 12]. For each integer in this range, the corresponding speed in meters per second is calculated. The interpolation method was chosen to determine the required power value from the corrected velocity v_{tar}. Interpolation is a method of

finding intermediate values between known points in a discrete data set [17]. It allows us to estimate the value of a function at points that are not specified in the original set, "filling in the blanks" to get a more complete picture of the function change. There are several types of interpolation:

– linear interpolation is the simplest method that determines a new value based on a straight line connecting the two closest known points;

– polynomial interpolation uses polynomials to calculate intermediate values, which allows to more accurately reflect complex changes in a function;

– spline interpolation applies splines to smoothly transition between points.

Monotone cubic interpolation, a variant of cubic interpolation that preserves the monotonicity of the interpolated data set [18], was used to calculate the required power. Its use provides smooth transitions between points while maintaining the monotonicity of the original set, which is especially important to avoid unwanted fluctuations in values between points. PchipInterpolator function from the *scipy* (Python programming language package) was used to implement monotone cubic interpolation while learning process.

Figure 2 shows a test environment for training a model for autonomous navigation with external factors such as water current and obstacles (buoys and walls).

5. Results of the research on the neural network model for autonomous navigation of a water drone

5.1. Analysis of the proposed model for the autonomous navigation of water drone

The result of this study is a neural network model for autonomous navigation of a water drone. The proposed model uses an approach that reduces the uncertainty of the learning environment by considering the direction and speed of the water current to control the power of the drone's thruster during training. As a result, the model is able to reach the required goal faster, as fewer uncertainties contribute to a faster learning process.

Unlike studies [10 – 11] in paragraph 2, which use DQN for planning drone trajectories, the proposed model stabilizes learning by accounting for environmental factors, such as water current. Solutions [10 – 11] use a dynamic approach, relying on the model, which learns through trial and error and gradually begins to make correct decisions based on the current state of the environment. In these studies, optimization of the reward function is used to improve the results, which can help the drone perform the desired task faster. However, stabilizing the learning process by considering environmental factors, as it is done in the proposed model, also has a positive effect on the results. It reduces uncertainty and helps the model to focus on reaching the target point.

To consider the speed and direction of the water current in the learning process, certain information is required. This includes the speed and direction of the water current, the drone's direction, and its capability to achieve specific speed. To obtain such values for real conditions, it is necessary to use a variety of sensors. For example, compasses and GPS are used to obtain the direction of the drone. The maximum possible speed and corresponding thruster power ratings are usually provided by the boat manufacturer or tested by trial runs in calm water conditions. The speed and direction of the water current can be found, for example, with the help of "floating buoys". These devices are installed on the surface of the water and drift under the influence of the water current. They are equipped with GPS, which allows tracking of their movements in real time, which provides data on the direction and speed of water current on the surface. If training takes place in a simulator, then such data can be obtained much more easily — usually, they are stored in the form of certain variables that can be accessed.

The results of using the proposed model in practice are presented in 5.2. Figure 3 shows an example of considering the water current speed and direction during training.

```
[INFO] [1730195228.213035, 29.803000]: yaw: 0.546481497073
[INFO] [1730195228.213380, 29.803000]: Target speed: 0.58, Speed of water: 0.180277563773, Computed speed: 0.529925204282
[INFO] [1730195228.214757, 29.805000]: Current angle: 65.0011508903
[INFO] [1730195228.215280, 29.805000]: Computed thruster_power: 4.9980910745
[INFO] [1730195228.563637, 30.007000]: yaw: 0.482502236472
[INFO] [1730195228.564088, 30.007000]: Target speed: 0.58, Speed of water: 0.180277563773, Computed speed: 0.522582415763
[INFO] [1730195228.566356, 30.008000]: Current angle: 61.3354092815
[INFO] [1730195228.568721, 30.009000]: Computed thruster_power: 4.81782175412
[INFO] [1730195228.986515, 30.309000]: yaw: 0.384528148843
[INFO] [1730195228.987196, 30.311000]: Target speed: 0.58, Speed of water: 0.180277563773, Computed speed: 0.511338100043
[INFO] [1730195228.987571, 30.311000]: Current angle: 55.7219075586
[INFO] [1730195228.989052, 30.312000]: Computed thruster_power: 4.56727257268
[INFO] [1730195229.542752, 30.607000]: yaw: 0.273304247996
[INFO] [1730195229.543129, 30.607000]: Target speed: 0.58, Speed of water: 0.180277563773, Computed speed: 0.498573126293
[INFO] [1730195229.543473, 30.607000]: Current angle: 49.3492474592
[INFO] [1730195229.543854, 30.607000]: Computed thruster_power: 4.3156485812
[INFO] [1730195229.906184, 30.803000]: yaw: 0.222607892264
[INFO] [1730195229.906546, 30.803000]: Target speed: 0.58, Speed of water: 0.180277563773, Computed speed: 0.492754793676
[INFO] [1730195229.906808, 30.803000]: Current angle: 46.444560239
[INFO] [1730195229.907159, 30.803000]: Computed thruster_power: 4.21099103226
[INFO] [1730195230.559119, 31.103000]: yaw: 0.162473921766
[INFO] [1730195230.559607, 31.103000]: Target speed: 0.58, Speed of water: 0.180277563773, Computed speed: 0.485853322418
[INFO] [1730195230.559928, 31.103000]: Current angle: 42.9991375241
[INFO] [1730195230.560313, 31.103000]: Computed thruster_power: 4.09392789104
[INFO] [1730195230.964266, 31.342000]: yaw: 0.152781142888
[INFO] [1730195230.964598, 31.342000]: Target speed: 0.58, Speed of water: 0.180277563773, Computed speed: 0.484740899036
[INFO] [1730195230.965736, 31.343000]: Current angle: 42.4437822026
[INFO] [1730195230.968950, 31.344000]: Computed thruster_power: 4.07571871287
[INFO] [1730195231.396373, 31.607000]: yaw: 0.0774401385445
[INFO] [1730195231.397005, 31.607000]: Target speed: 0.58, Speed of water: 0.180277563773, Computed speed: 0.476094142939
[INFO] [1730195231.397372, 31.607000]: Current angle: 38.1270606295
[INFO] [1730195231.398013, 31.608000]: Computed thruster_power: 3.94045164099
```

Fig. 3. Logging of calculated data to maintain specific power and velocity

Figure 3 shows that the drone's thruster power is constantly changing depending on the direction and speed of the water current. Adaptation of the drone's thruster power to the water current occurs throughout the training process.

5.2. Results of practical use of the proposed model

As mentioned in 4.3, the USVSim simulator is used to implement and test the proposed model. For testing, a special training environment was created in which a water current was implemented; there were various obstacles for movement and the drone itself was located. The use of monotone cubic interpolation made it possible to obtain a "smooth" monotone function of the dependence of speed on thruster power, shown in Figure 4.

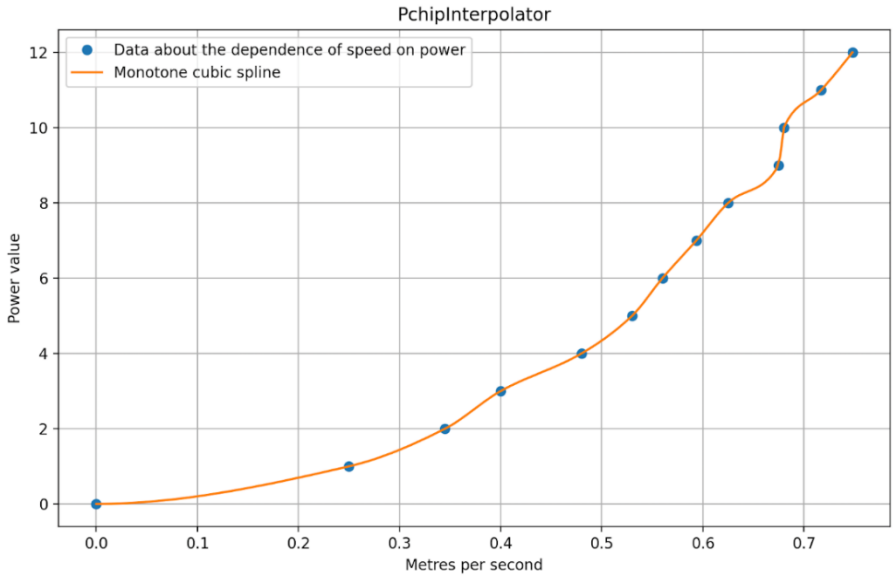

Fig. 4. Using monotone cubic interpolation for the calculated power range

To verify the effectiveness of the proposed model, several processes of training for autonomous navigation of a water drone with the same hyperparameters and settings of the simulation training environment were conducted. Each training consisted of 1500 episodes. The test was to compare the average number of points reached per episode with DQN and with the proposed model.

Since the results of each new training run are not like each other, several training processes were carried out for both: with DQN and with the proposed model. From the obtained results, the best ones were selected and the average number of points reached per episode was calculated.

One of the best results for the case of training the model without considering the speed and direction of the water current (DQN) is shown in Figure 5. With a constant maximum thruster power, in the finest attempt at learning, the drone begins to perform the task of navigating between the set points from approximately the 1200th episode. Different restarts gave a difference of about 100–200 episodes from this value. The average number of points reached is approximately ~2–3.

The result of one of the best learning processes considering the speed and direction of the water current (proposed model) is shown in Figure 6. In this case, the drone starts to perform the task from 800–900 episodes with a difference of 150–200 episodes for different restarts. The average number of points reached per episode is approximately ~4.

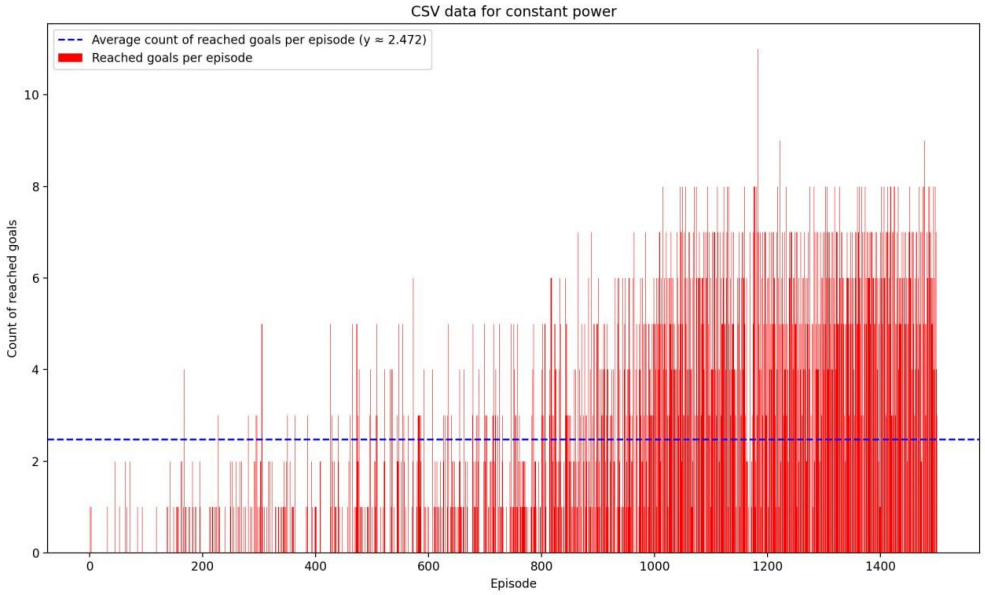

Fig. 5. Results for one of the runs with a constant maximum drone thruster power

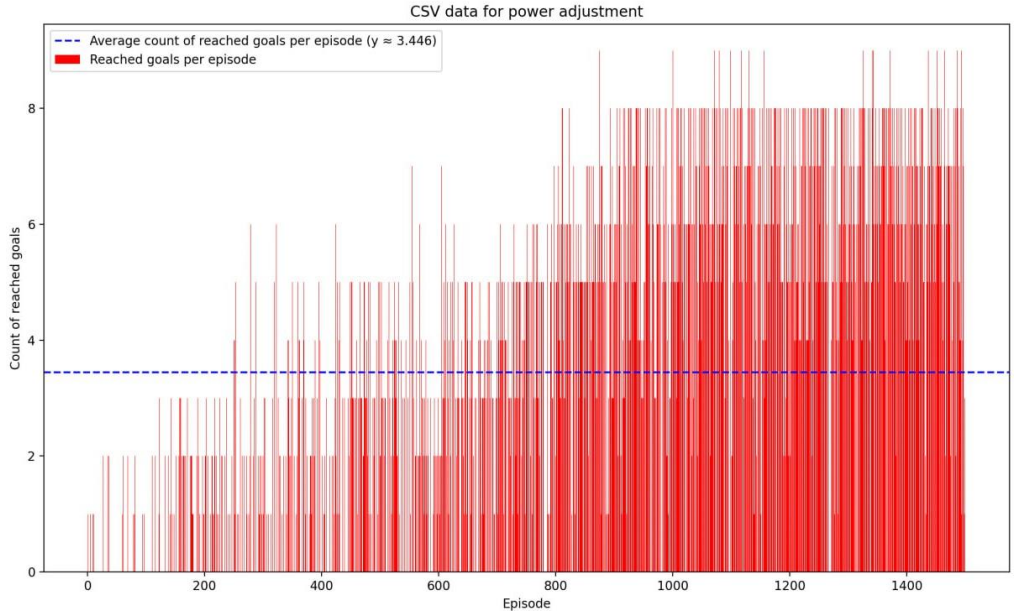

Fig. 6. Results for one of the runs with a thruster power adjustment

The results presented in Figures 5 and 6 show that the model adapted to the dynamic conditions of the aquatic environment. It started to move more or less stably between the target points even using the standard DQN. However, the implementation of the proposed model allowed us to obtain the desired expected result earlier than with standard DQN.

Thus, usage of the proposed model makes it possible to reduce the amount of time and computing resources required. It can also significantly improve the learning process and results, which is very important for such tasks.

6. Analyzing of the obtained results of the proposed neural network model for autonomous navigation of a water drone

Experimental results confirm the feasibility of using the proposed model for autonomous water drone navigation. The model effectively avoids obstacles and adapts to water currents in a simulated environment, highlighting the importance of a structured training dataset and well-tuned reward mechanisms for optimal performance.

It should be noted that although the proposed model performed well under simulated conditions, further refinement may be required to improve the reliability of the model in more diverse aquatic environments. Moreover, alternative reinforcement learning methods, such as, for example, proximal policy optimization (PPO) [19], could be explored to potentially improve the drone's decision making under more stringent environmental conditions.

This analysis highlights that the adaptability of the model is not only influenced by its internal parameters, but also by the complexity of the environment within the simulation. The ability to dynamically adjust to currents and obstacles in real time offers great potential for the application of such autonomous navigation systems in real-world scenarios.

The knowledge gained from this study can serve as a basis for further improving water drone autonomy, supporting the integration of autonomous systems into broader maritime applications. Future research could investigate additional factors, such as waves, coordination of multiple drones, and advanced environmental sensing. These factors could enhance the performance and applicability of autonomous water drones in both controlled and unpredictable aquatic environments.

7. Conclusions

After the completion of the study, all defined tasks were accomplished. The neural network model for achieving autonomous navigation for water drones using reinforcement learning is proposed and described. It utilizes DQN to enable adaptive obstacle avoidance and thruster power adjustment in dynamic aquatic environments. Aspects, which should be taken into account for successful training of a water drone model for autonomous navigation, were analyzed.

Implementing such model in a simulated environment provides an opportunity to improve the autonomous navigation capabilities of water drones prior to real-world deployment. The results, obtained in the USVSim, show that neural networks can significantly improve the ability of a water drone to operate independently, adjusting to environmental changes without manual intervention. These results can be applied to a variety of water drone applications, from environmental monitoring to search and rescue operations. Future implementations of this model can be incorporated into various systems designed to control and analyze autonomous navigation in different aquatic environments, providing a basis for further development and operational use of water drones in challenging environments.

References

[1] G. Katsouras et al., “Use of Unmanned Surface Vehicles (USVs) in Water Chemistry Studies,” *Sensors*, vol. 24, no. 9, p. 2809, Apr. 2024. https://doi.org/10.3390/s24092809.

[2] Z. Liu, Y. Zhang, X. Yu, and C. Yuan, “Unmanned surface vehicles: An overview of developments and challenges,” *Annual Reviews in Control*, vol. 41, pp. 71–93, 2016. https://doi.org/10.1016/j.arcontrol.2016.04.018.

[3] A. Bhar and M. Sayadi, “On designing a configurable UAV autopilot for unmanned quadrotors,” *Frontiers in Neurorobotics*, vol. 18, May. 2024. https://doi.org/10.3389/fnbot.2024.1363366.
[4] A. Kumar et al., “Enhanced Route navigation control system for turtlebot using human-assisted mobility and 3-D SLAM optimization”, *Heliyon*, vol. 10, p. e26828, Mar. 2024. https://doi.org/10.1016/j.heliyon.2024.e26828.
[5] A. Marashian and A. Razminia, “Mobile robot’s path-planning and path-tracking in static and dynamic environments: Dynamic programming approach,”, *Robotics and Autonomous Systems,* vol. 172, p. 104592, Feb. 2024. https://doi.org/10.1016/j.robot.2023.104592.
[6] A. Bhargava, *Grokking Algorithms*, 2nd ed., New York, NY, US: Manning Publications Co. LLC, 2024.
[7] T. Xu, “Recent advances in Rapidly-exploring random tree: A review,” *Heliyon*, vol. 10, p. e32451, Jun. 2024. https://doi.org/10.1016/j.heliyon.2024.e32451.
[8] R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction,” *IEEE Transactions on Neural Networks*, vol. 9, no. 5, p. 1054, Sep. 1998. https://doi.org/10.1109/tnn.1998.712192.
[9] G. Wu, D. Li, H. Ding, D. Shi, and B. Han, “An overview of developments and challenges for unmanned surface vehicle autonomous berthing,” *Complex & Intelligent Systems*, vol. 10, pp. 981–1003, Aug. 2023. https://doi.org/10.1007/s40747-023-01196-z.
[10] Y. Zhao, F. Han, D. Han, X. Peng, and W. Zhao, “Decision-making for the autonomous navigation of USVs based on deep reinforcement learning under IALA maritime buoyage system,” *Ocean Engineering*, vol. 266, p. 112557, Dec. 2022. https://doi.org/10.1016/j.oceaneng.2022.112557.
[11] L. Li, D. Wu, Y. Huang, and Z. Yuan, “A path planning strategy unified with a COLREGS collision avoidance function based on deep reinforcement learning and artificial potential field,” *Applied Ocean Research*, vol. 113, p. 102759, Aug. 2021. https://doi.org/10.1016/j.apor.2021.102759.
[12] K. Messaoudi, O. S. Oubbati, A. Rachedi, A. Lakas, T. Bendouma, and N. Chaib, “A survey of UAV-based data collection: Challenges, solutions and future perspectives,” *Journal of Network and Computer Applications*, vol. 216, p. 103670, Jul. 2023. https://doi.org/10.1016/j.jnca.2023.103670.
[13] R. Song, Y. Liu, and R. Bucknall, “Smoothed A* algorithm for practical unmanned surface vehicle path planning,” *Applied Ocean Research*, vol. 83, pp. 9–20, Feb. 2019. https://doi.org/10.1016/j.apor.2018.12.001.
[14] Hugging Face. “The Deep Q-Learning Algorithm”. Accessed: Nov. 15, 2024. [Online]. Available: https://huggingface.co/learn/deep-rl-course/unit3/deep-q-algorithm.
[15] M. Roderick, J. MacGlashan, and S. Tellex, “Implementing the Deep Q-Network,” Nov. 2017, arXiv:1711.07478. https://doi.org/10.48550/arXiv.1711.07478.
[16] M. Paravisi, D. H. Santos, V. Jorge, G. Heck, L. M. Gonçalves, and A. Amory, “Unmanned Surface Vehicle Simulator with Realistic Environmental Disturbances,” *Sensors*, vol. 19, no. 5, p. 1068, Mar. 2019. https://doi.org/10.3390/s19051068.
[17] J. F. Steffensen, *Interpolation*, 2nd ed., Mineola, NY, US: Dover Publications, 2006.
[18] F. N. Fritsch and R. E. Carlson, “Monotone Piecewise Cubic Interpolation,” *SIAM Journal on Numerical Analysis*, vol. 17, no. 2, pp. 238–246, Apr. 1980. https://doi.org/10.1137/0717021.
[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” Jul. 2017, arXiv:1707.06347. https://doi.org/10.48550/arXiv.1707.06347.

УДК 004.8 : 004.94

НЕЙРОМЕРЕЖЕВА МОДЕЛЬ ДЛЯ АВТОНОМНОЇ НАВІГАЦІЇ ВОДНОГО ДРОНА

Гліб Чекмезов
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна
https://orcid.org/0009-0003-0404-7834

Олексій Молчанов
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна
https://orcid.org/0000-0001-8384-0918.

Водні дрони мають значний потенціал для застосування в екологічному моніторингу, пошуково-рятувальних операціях та інспекції морської інфраструктури, однак специфічні умови водного середовища ускладнюють реалізацію стабільної автономної навігації.

Об’єктом дослідження, представленого в цій статті, є процес машинного навчання для автономної навігації моделі водного дрона у симуляційному водному середовищі. Метою дослідження є реалізація моделі нейронної мережі для автономної навігації водного дрона з використанням методу навчання з підкріпленням, що забезпечує підвищення ефективності уникнення перешкод та адаптацію до водних течій.

Для досягнення даної мети запропоновано нову нейромережеву модель для автономної навігації дрона у водному середовищі на базі методу навчання з підкріпленням, яка відрізняється від наявних тим, що використовує удосконалений алгоритм управління дроном, який враховує швидкість та напрям течії, що дає змогу стабілізувати процес генерування коефіцієнтів нейронної мережі.

Задля забезпечення ефективного процесу навчання та оптимізації моделі розроблено симуляційне навчальне середовище за допомогою симулятора *USVSim*, яке містить різноманітні фактори, що заважають руху дрона, такі як течія води та наявність інших об’єктів. Водний дрон, виступаючи як агент, поступово навчається обирати найбільш ефективні дії для досягнення максимальної позитивної винагороди методом спроб і помилок, взаємодіючи із середовищем та пристосовуючись до змінних умов. Цей процес відбувається за рахунок використання *Deep Q-Network*: дрон надає значення свого поточного стану до глибокої нейронної мережі; нейронна мережа обробляє отримані дані, прогнозує значення найбільш ефективної дії та віддає його агенту. Поточним станом дрона є інформація у вигляді набору показників датчиків вимірювання відстані до найближчих перешкод, напрям дрона та поточна відстань до цільової точки. Значення ефективної дії, отриманої від нейронної мережі, перетворюється на команду для керма, зрозумілу для дрона. Значення потужності мотора дрона розраховується окремими формулами з використанням тригонометричних функцій.

Результати дослідження показали, що використання запропонованої моделі дає дрону змогу приймати рішення в умовах динамічної водної обстановки, коли необхідна швидка адаптація до змін. Модель успішно адаптувала свою стратегію на основі зворотного зв’язку з середовищем, тому можна зробити висновок, що реалізована нейромережева модель демонструє значний потенціал для подальших досліджень та застосувань у сфері автономних водних дронів, особливо у мінливих і непередбачуваних середовищах.

Ключові слова: водний дрон, машинне навчання, DQN, автономна навігація, алгоритм.