applied system innovation

MDPI

Article

Dynamic Path Planning for Unmanned Surface Vehicles with a Modified Neuronal Genetic Algorithm

Nur Hamid [1,*], Willy Dharmawan [1,2] and Hidetaka Nambo [1]

[1] Department of Electrical Engineering and Computer Science, Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa 920-8641, Japan; willy.dharmawan12@gmail.com (W.D.); nambo@blitz.ec.t.kanazawa-u.ac.jp (H.N.)
[2] Centre of Electronics, BRIN, Puspiptek, Serpong, South Tangerang 15314, Indonesia
* Correspondence: nur.hamid@stu.kanazawa-u.ac.jp or nur.hamid@ui.ac.id

Abstract: Unmanned surface vehicles (USVs) are experiencing significant development across various fields due to extensive research, enabling these devices to offer substantial benefits. One kind of research that has been developed to produce better USVs is path planning. Despite numerous research efforts employing conventional algorithms, deep reinforcement learning, and evolutionary algorithms, USV path planning research consistently faces the challenge of effectively addressing issues within dynamic surface environments where USVs navigate. This study aims to solve USV dynamic environmental problems, as well as convergence problems in evolutionary algorithms. This research proposes a neuronal genetic algorithm that utilizes neural network input for processing with a genetic operator. The modifications in this research were implemented by incorporating a partially exponential-based fitness function into the neuronal genetic algorithm. We also implemented an inverse time variable to the fitness function. These two modifications produce faster convergence. Based on the experimental results, which were compared to those of the basic neural-network-based genetic algorithms, the proposed method can produce faster convergent solutions for USV path planning with competitive performance for total distance and time traveled in both static and dynamic environments.

Keywords: dynamic environment; path planning; unmanned surface vehicles; modified genetic algorithms

Citation: Hamid, N.; Dharmawan, W.; Nambo, H. Dynamic Path Planning for Unmanned Surface Vehicles with a Modified Neuronal Genetic Algorithm. *Appl. Syst. Innov.* **2023**, *6*, 109. https://doi.org/10.3390/asi6060109

Academic Editor: Abdelkader Sbihi

Received: 29 September 2023
Revised: 30 October 2023
Accepted: 6 November 2023
Published: 14 November 2023

1. Introduction

An unmanned surface vehicle (USV) is a type of waterborne vehicle that operates on the surface of a body of water with autonomous navigation and self-reliant planning capability [1]. USVs can be remotely controlled or can operate autonomously, relying on various sensors, navigation systems, and communication technologies. These vehicles are widely used for a variety of purposes, ranging from scientific research [2], environmental monitoring [3], and disaster robotics [4] to maritime security and military applications [5,6]. USVs have played an essential role in oceanographic research, security and surveillance, defense and military applications, network monitoring cooperating with aerial and ground vehicles, search and rescue, and autonomous transportation [2]. More USV implementations are in the pipeline, with an increasing amount of automation research being conducted on these devices.

Automation plays a crucial role in the development and operation of USVs. There are various critical aspects of automation in unmanned surface vehicles including navigation and path planning, obstacle avoidance and detection, remote operation and supervision, data collection, mission execution, communication and data transmission, energy management, redundancy, and fail-safe mechanisms. These aspects of automation are crucial for USVs, especially when operated in dynamic and uncertain environments, such as open water, rivers, or busy harbors. Handling dynamic obstacles, changing currents, and

unpredictable weather conditions can be challenging. From these many aspects of USV automation, path planning stands out as a critical technology guaranteeing safe navigation by creating trajectories that avoid collisions [7].

The subject of this research is a USV that performs a path planning task in a dynamic environment. We also conducted experiments in a static environment as a comparison. Figure 1 illustrates the simple USV path planning process in a dynamic environment. USV dynamic path planning becomes essential because this process can bridge the automation stage at the previous level that has been achieved (manual control and static environment) with more advanced research (autonomous navigation and cognitive planning in a complex environment) [2]. This is what motivated us to research USV path planning in both dynamic and static environments using the proposed genetic algorithm modification method.

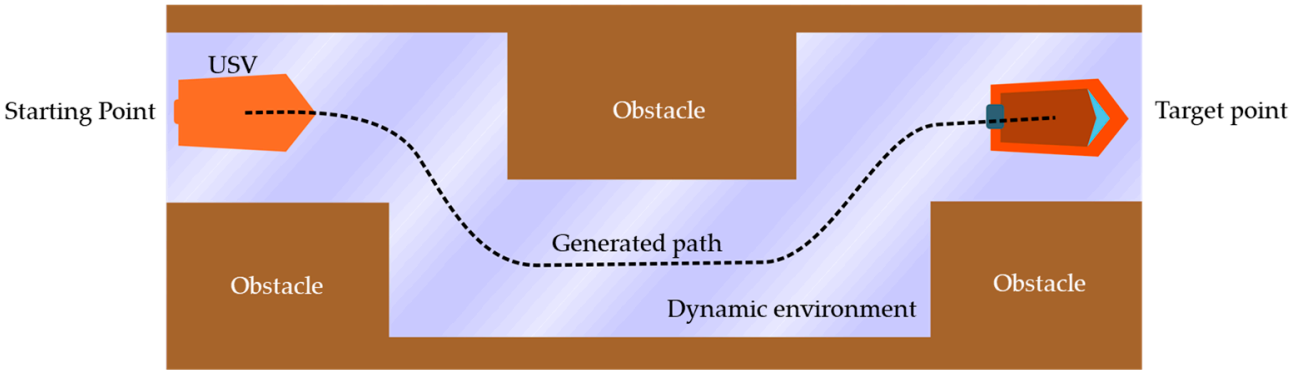

Figure 1. Illustration of how a USV performs path planning in a dynamic environment.

2. Related Work

Given the wide range of applications for USVs across different domains, numerous studies have explored path planning using various approaches. In this literature review, we classify the related algorithms into three types: conventional algorithms, deep reinforcement learning, and evolutionary algorithms. The conventional algorithm is used because it is relatively simple and produces quite good results. For the conventional or simple algorithm, some researchers modify A* algorithms. The constrained A* method [8] is an approach used in optimal path planning for USVs in a maritime environment. Another A* modification is the smoothed A* algorithm [9], applied with a new path-smoothing process. The improved D* Lite algorithm [10] was implemented for the path planning of unmanned surface vessels in an unknown environment, and a greedy algorithm [11] with an adaptively rotatable distance sensor was developed to equip an autonomous evacuation boat designed for dynamic flood situations.

Apart from researching modified simple algorithms, researchers also conduct studies using deep reinforcement learning (RL) and its modifications. In contrast to traditional algorithms, deep RL-based path planning algorithms offer a novel approach, incorporating advanced artificial intelligence at a high level [7]. RL demonstrates the capability to acquire high performance when operating in unfamiliar environments by learning through interactions and experiences gained from training environments, all without prior information or knowledge [12]. RL research into complete-coverage path planning [13] was conducted to solve the slow convergence speed problem. Other RL studies on cooperative path planning [7] aimed to solve compliance with vehicle motion constraints implemented in computer-based simulations and real-world maritime settings. Then, RL for an intelligent controller [14] was proposed, involving the utilization of an intelligent adaptive PID controller, which was improved through the integration of proximal policy optimization (PPO). This enhancement was made with the aim of attaining a high level of automation for USVs. Then, distributional RL was used in robust USV navigation [12] and learned to capture the uncertainty of action outcomes. A RL method with ANOA was also implemented for autonomous navigation and obstacle avoidance [15]. While RL has demonstrated promising

results in USV path planning tasks, it is not without its limitations. RL typically demands substantial volumes of training data to acquire effective policies, and as the dataset grows, it necessitates more significant computational resources to process and train effectively [16].

To overcome the shortcomings of the RL approach, nature-based algorithms offer some advantages. Nature-based algorithms can work with limited data and may not require extensive training. These algorithms are easier to implement and require fewer computational resources. In addition, nature-based algorithms can also provide better performance than conventional approaches because nature algorithms involve the learning and optimization scenario. Some studies have been conducted that use nature-based algorithms for USV path planning, such as the improved biological-inspired neural network [17], trajectory-cell-based algorithm [18], bacterial foraging optimization algorithm [19], improved shark-inspired algorithm [20], and plant growth algorithms [21]. Physics-based algorithms also provide methods like the artificial potential field approach [22]. Among the numerous nature-based algorithms available, our choice is to adapt the genetic algorithm (GA). This algorithm is versatile and capable of handling a wide array of optimization scenarios whether the fitness function exhibits continuity or discontinuity, linearity or nonlinearity, or stationarity or nonstationary (changing over time), even when influenced by random noise [23]. The GA also exhibits superior robustness and inherently maintains a diverse population when compared to those of other evolutionary algorithms [24].

Some GA research tries to modify the population to achieve better performance. An improved genetic algorithm [25] tried to solve the population prematurity and slow convergence speed problem in USV path planning by managing the number of offspring. Other research focusing on an improved genetic algorithm [26] was carried out for cooperative collision avoidance. Through building fitness and iteratively optimizing the adjustment of velocity and heading, the researchers were able to use the algorithm to safely operate multiple USVs. Then, the improved genetic algorithm [1] was used to try to solve the convergence speed problem by maximizing the cumulative detection probability (CDP). Combining the GA with other algorithms provides better performance. Thus, the genetic algorithm was combined with the simulated annealing algorithm (improved hybrid GA) for path planning [27] in order to solve its lack of searching ability and the large amount of calculation required. Other research combined the GA with the artificial potential field (APF) algorithm [28] for multi-objective, multi-robot path planning in a continuous environment. A modified genetic algorithm [29] for a USV under environmental loads was proposed by integrating the three objective functions (minimizing travel time, reaching a target point, and avoiding obstacles). All GA modification research was carried out to produce better algorithm performance.

In this research, we examine a neuronal GA—a combination of a genetic algorithm and a neural network—using neural network variable values as input to the genetic operator of the GA. The modification is carried out by implementing a partially exponential-based fitness function to produce a faster convergent solution (details of the proposed methodology are presented in Section 4). The proposed method aims to solve the same convergence problem as does other GA research, as well as other unresolved problems in the GA in the form of implementation in a dynamic environment.

3. Problem Formulation and Contribution

Unmanned surface vehicle (USV) path planning refers to the procedure of establishing the most advantageous or efficient course for an autonomous surface vehicle to traverse a water body, which could be a lake, river, or ocean. This process aims to guide a USV from its current position to a predetermined destination. The central objective of path planning is to facilitate the USV's successful arrival at its target by steering clear of obstacles, accommodating dynamic environmental changes, and adhering to specific constraints. When the USV is in the environment, its task is to analyze the environment and determine where it is moving to and at what acceleration. Figure 2 shows the types of forces acting on the USV and the determination of movement decisions based on environmental detection [11].

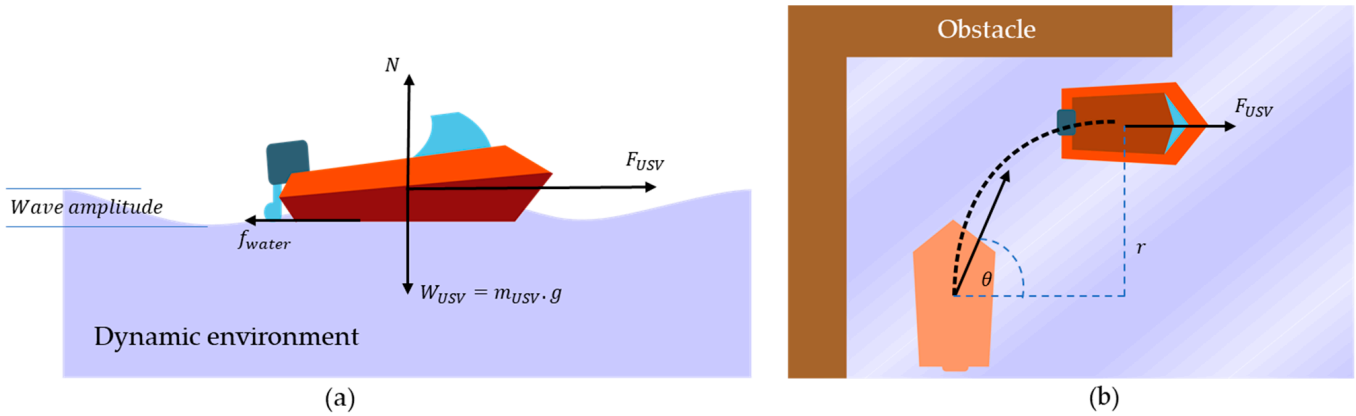

Figure 2. Types of forces acting on the USV (**a**) and the determination of movement decisions based on environmental detection (**b**).

In this research, the target or goal of a USV (or an evolutionary algorithm, hereinafter referred to as an agent) is to achieve the expected fitness value. The fitness value can be achieved if the agent can move across a predetermined route and avoid obstacles or boundary walls. To carry out these tasks, the agent performs two maneuvers: moving straight and turning (right or left). Figure 3 shows a simple map of the experimental environment and the expected path (the black dotted line). The experiment is initiated from the bottom-left corner, as opposed to other starting locations. This choice is made because, from this starting point, the agent encounters an equal number of right and left turns in the first four movements. Starting from a different point with an uneven distribution of turns would make it easier for the agent to detect a specific turn, but it would lead to failure when the agent must navigate in a different direction.

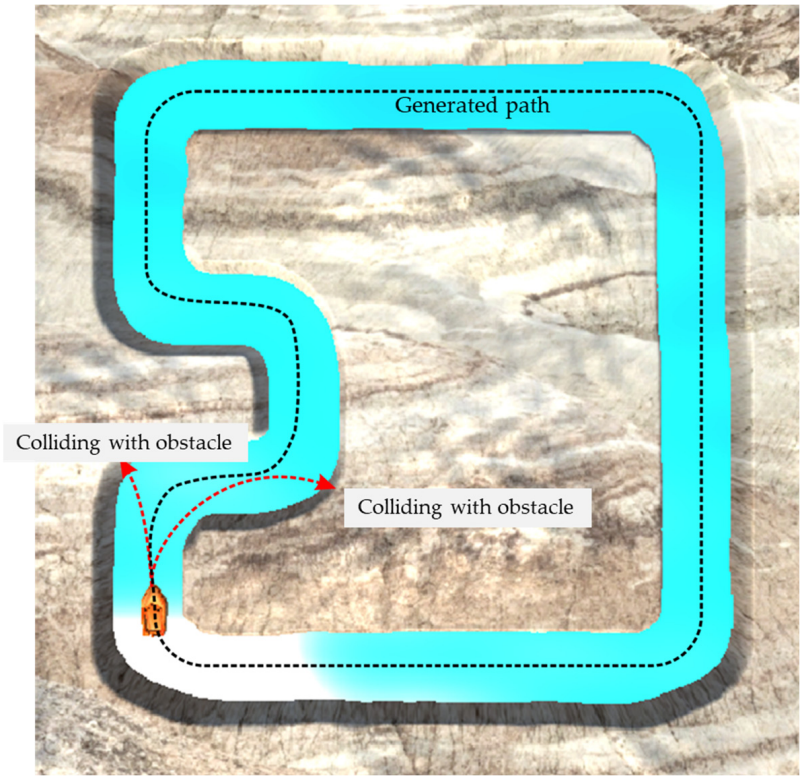

Figure 3. Simple map of the experimental environment and the expected path (the black dotted line).

The best and most concise path does not collide with obstacles or boundaries and has a convergent path (i.e., it can be reused for the next iteration or round). We consider it to be used in the next round or iteration because in the case of a dynamic environment, the USV may not necessarily use the same path in the next round. These different paths are the result of adapting the USV to dynamic environments such as water surface waves. For the USV to generate the optimal path, it is essential to start from the initial position on x_0 , y_0, and z_0, and then transition to the trajectory using two maneuvers involving forward motion with an acceleration value represented as α, and a turning action (either to the right or left) with an angle value denoted as θ. These two maneuvers are produced by the neural network using the weight and bias parameters. The optimal weight and bias values obtained from the network play a pivotal role in determining the most effective maneuver for the USV. These values are instrumental in evaluating the USV's acceleration and the angle at which it turns during its movement. We optimize the weight and bias parameters using modified genetic algorithms.

Through this research, we contribute to implementing the neuronal genetic algorithm (combining neural network parameters—weights and biases—as the input for the GA). For GA modification, we introduce a partially exponential-based fitness function to the neuronal genetic algorithm. We also add an inverse time variable to the fitness function. These two modifications aim to produce faster convergence. We implement the proposed methodology for both static and dynamic environments.

4. Method

4.1. Experimental Setup

Our experiments take place within the mapped environment displayed in Figure 3. We customized this environment using insights from prior research [11]. We implemented a dynamic water environment capable of being adjusted to accommodate various wave conditions, encompassing parameters such as height, speed, and scale. The dynamic water environment is described as a floating surface containing multiple waves, and its attributes are established based on Equation (1):

$$y_{x,z} = A\cos\left(\omega t + \left(\frac{x_t,\ z_t}{K}\ 2\pi\right)\right), \tag{1}$$

where $y_{x,z}$ denotes the magnitude of the height value for each point in a Cartesian coordinate system, A is the amplitude or maximum of height, and ω is the wave speed value. Additionally, t is the time, x_t and z_t are the x and z positions, and K is the wave unit. This formula enables the agent to remain buoyant on the water surface. To create a dynamic environment, we input specific values into the equation, allowing us to configure the environment to have varying wave effects. Conversely, we rendered the environment static by inputting a value of 0 for 'A,' thereby eliminating the influence of waves. The experiments in this study used the input values v_x 2, v_y 3, scale x 4, scale y 6, and height 1.2. Using a larger value makes the dynamic environment more challenging to solve.

4.2. Proposed Methodology

In this research, we combine basic neural network algorithm methods and genetic algorithms. The neural network is the basis for generating USV motion parameter values, acceleration, and turning angle. The input for the network is the value obtained from the number of n sensors. The sensor used in this research platform is a ray cast. This sensor's real-world implementation is the same as that of a ray sensor that detects distance values (radar or lidar). Figure 4 shows the configuration of the sensor embedded in the agent. The three-sensor configuration has direction angles of 45°, 0°, and −45°. The five-sensor configuration has direction angles of 45°, 22.5°, 0°, −22.5°, and −45°. The seven-sensor configuration has direction angles of 67.5°, 45°, 22.5°, 0°, −22.5°, −45°, and −67.5°. The angle 0° is in the same direction as the USV's rectilinear motion direction.

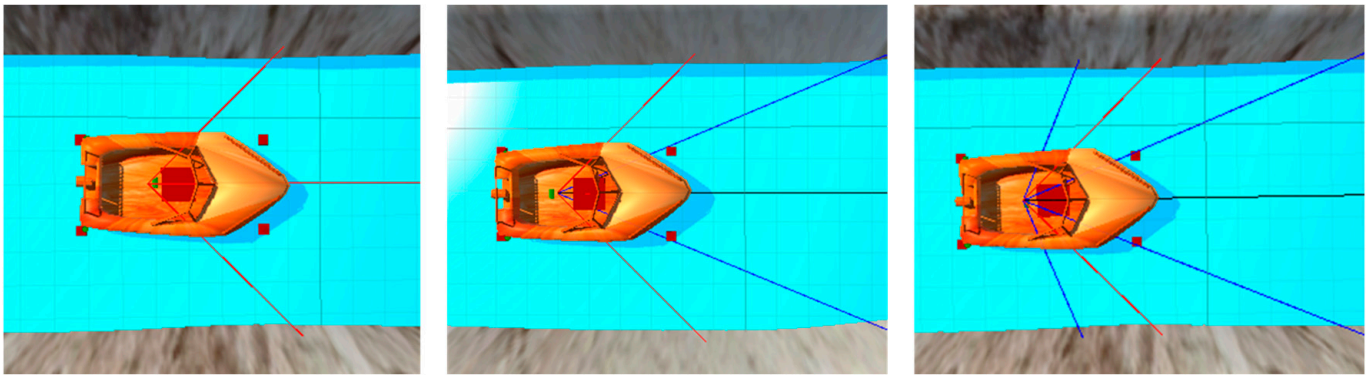

Figure 4. Ray cast sensor configuration on the USV (agent) implemented in a 3D simulation environment: (**a**) three sensors, (**b**) five sensors, and (**c**) seven sensors.

A neural network consists of three main layers, including the input layer, hidden layer (single or multiple), and output layer [1]. The inputs from the sensor embedded in USVs are fed to the network with n neurons and m number of layers. During the learning process, the network is optimized based on the parameter of weights and biases. The initial values for weight and bias are randomly generated. In this configuration, the activation function for acceleration is sigmoid, while the steering angle uses the TanH function. The output values range from 0 to 1 for acceleration and from −1 to 1 for the steering angle. Figure 5 provides a visual representation of the neural network utilized in this study.

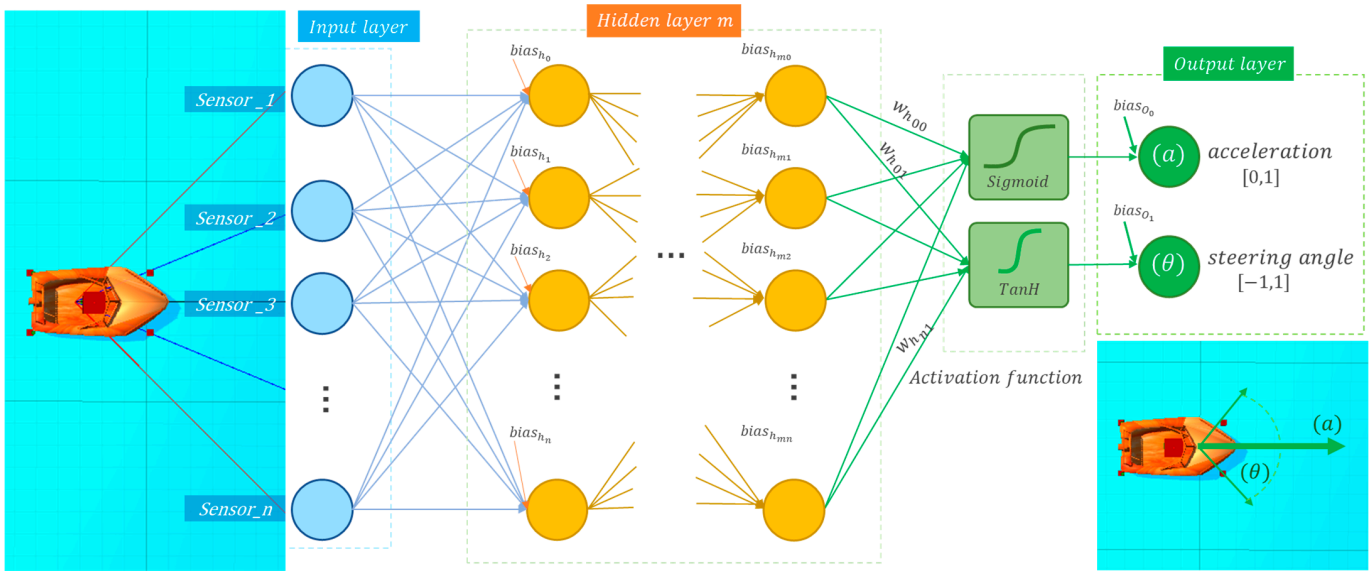

Figure 5. Fundamental framework for neural network with n input (from the distance-based ray cast sensor) and two outputs (acceleration, α [0,1], and the steering angle, θ [−1,1]).

In each individual process, the neural network generates a weight value represented as w and a bias value represented as b. The agent, such as an unmanned surface vehicle (USV), associated with a particular neural network model constitutes an individual genome within the population of the genetic input algorithm. In a broader view, the genetic algorithm process is illustrated in Figure 6 [27].

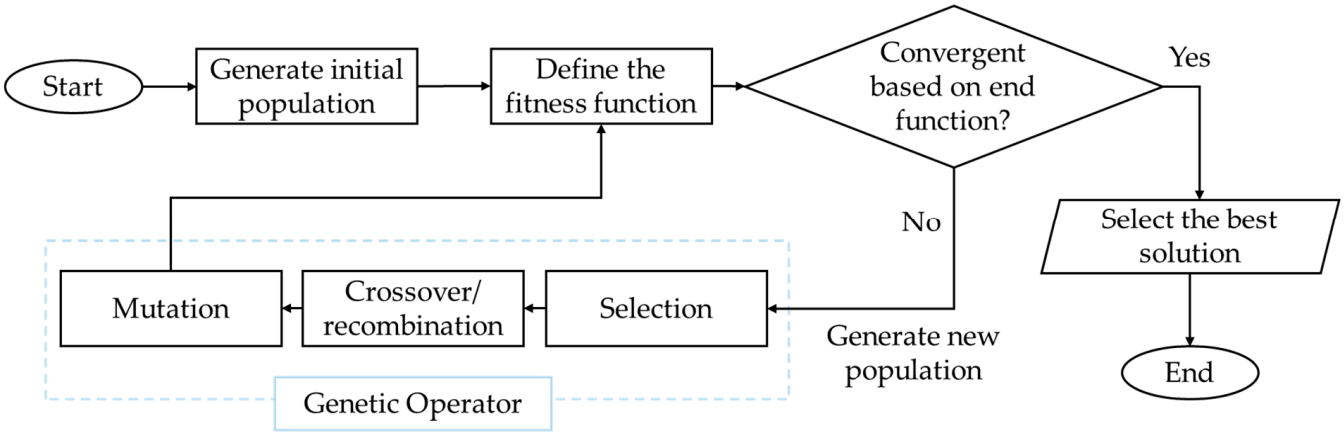

Figure 6. General framework for genetic algorithm in the path finding task.

a. Generate Initial Population

As shown in Figure 6, the genetic algorithm begins with the process of generating an initial population. In this context, "population" refers to neural networks characterized by random weight and bias parameters. Each genome, representing either an agent or a USV, is depicted using its dedicated neural network model. Each model will be expected as our solution for the path planning task. In our experiment, we continued to create the initial population until we reached the maximum number of genomes within a single generation, which was set at 40 agents. Once this maximum value was reached for a generation, we either reset the process to the current genome or assigned the first genome from the existing population to the new generation's USV.

The main novelty in this research is related to the fitness function modification. To assess the performance of the pathfinding model generated by the network, we employed a fitness function. A higher fitness value for a generated path or solution indicates a more accurate representation of the path, while a lower fitness value suggests the opposite [30]. The fitness function is related to the variables of speed, distance, and sensors. The common fitness function for a USV with three input sensors follows Equation (2):

$$Fitness = (\overline{v} \times m_{\overline{v}}) + (D_T \times m_{D_T}) + \left(\frac{a_{sens} + b_{sens} + c_{sens}}{3} \times m_s\right), \quad (2)$$

where $(\overline{v}) = D_T / (\sum t)$ denotes the average speed; $m_{\overline{v}}$ is the average speed multiplier; D_T denotes the total distance traveled; m_{D_T} is the distance multiplier; a_{sens}, b_{sens}, and c_{sens} are the values of sensor a, sensor b, and sensor c; and m_s is the sensor multiplier.

If the combination of these three variables shows different values, then the difference will be very small if it is in linear space (illustrated in Figure 7). If we apply the fitness function in exponential space, the slightly different output fitness values will be more visible because there is the influence of the power values p, q, and r. We also add an inverse of the time parameter so that agents that move in a short time obtain a greater fitness value. Therefore, the form of the proposed fitness function formulation will be as in Equation (3):

$$Modified\ Fitness = \left[(\overline{v} \times m_{\overline{v}})^p + (D_T \times m_{D_T})^q + \left(\frac{\sum_{i=1}^{n} Sensor_i}{n} \times m_s\right)^r\right] \times \frac{1}{(T \times m_T)}, \quad (3)$$

where n denotes the number of sensor inputs, T is the execution time, and m_T is the time multiplier with a constant number. This proposed fitness function modification is expected to produce a GA that can reach a convergent solution more quickly with a shorter USV travel time.

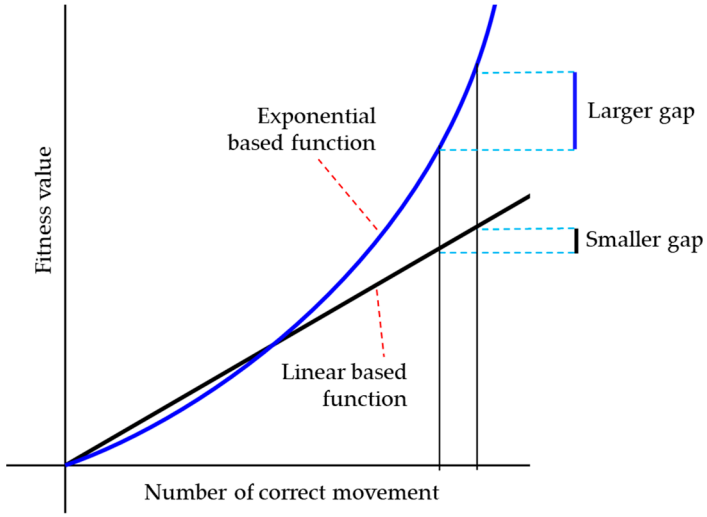

Figure 7. Illustration of the difference in fitness values in linear and exponential space.

b. Repopulation (Generate New Population)

As explained in the GA flowchart in Figure 6, if one of the networks in the initial population produces a solution that converges or fits the end function, then the learning process will stop. However, very rarely does the initial population produce convergent solutions. If this happens, then the next process is repopulation or generating a new population. The randomly generated population is evaluated or optimized with genetic operators including selection, crossover, and mutation. Following the application of these three operators, the updated population is assessed to determine whether or not the system has generated a convergent solution. If a convergent solution has not yet been achieved, the population will undergo further processing using the genetic operator to produce a convergent solution.

1. Selection

The selection operator operates by choosing individual agents from the population, and this selection can be random or based on their fitness values [28]. In this stage, we organize and rank the population according to their modified fitness values using Equation (2). The top-performing individuals are chosen as parents to undergo further processing with the next genetic operator. In this study, we identify the best ten agents and the worst four agents for this purpose. Figure 8 illustrates the selection process of a USV as an agent based on the fitness value.

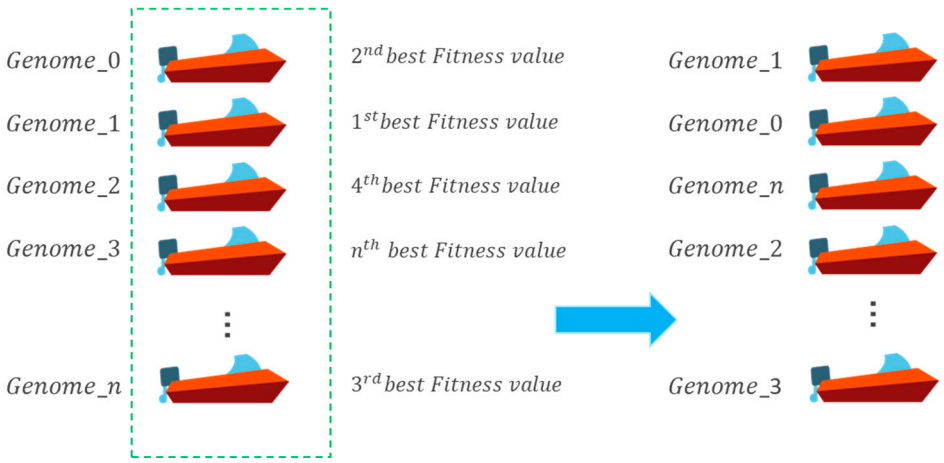

Figure 8. USV selection process based on the fitness value (ranked from the best to the worst fitness value).

2. Crossover

In the crossover operator, a higher probability denoted as p_c is implemented compared to that of other genetic operators. This operator involves exchanging one array or segment of one chromosome with the corresponding segment from another chromosome, and this exchange occurs at a random location. The main purpose of this operator is to facilitate the merging or combination of convergent characteristics in a subspace and to generate expected solutions [23]. In the context of the USVs, this operator combines all of the network arrays and swaps the entire set of weights and biases between two parent chromosomes to create two offspring chromosomes. This step can be referred to as an exploration process within genetic algorithms [31]. Figure 9 shows how the USV performs the crossover process.

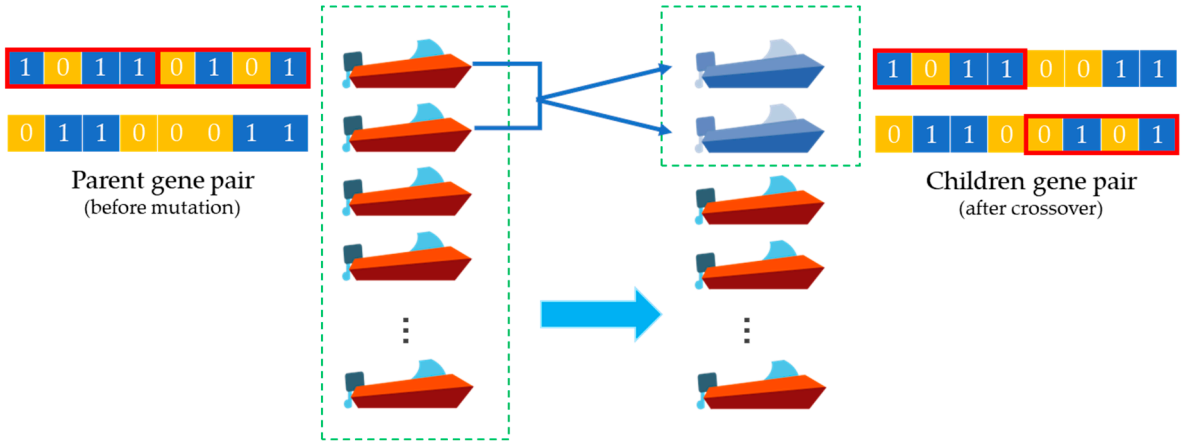

Figure 9. USV performing the crossover process to produce new children. USVs with red color are the original ones, and the blue color indicates the USV children after crossover process.

3. Mutation

The mutation operator operates by randomly flipping selected bits, and the mutation probability, p_m, is typically set to be smaller than that of other operators. This operator serves to enrich the diversity within the population [29] and offers a means to break free from local optima [23]. In the context of the USVs, this operator introduces randomization by altering both the row and column of the weight matrix. It then adjusts the values by adding a random rank value ranging from -1 to 1. This step can be thought of as an exploitation task within the genetic algorithm. Figure 10 shows the mutation process of the USV.

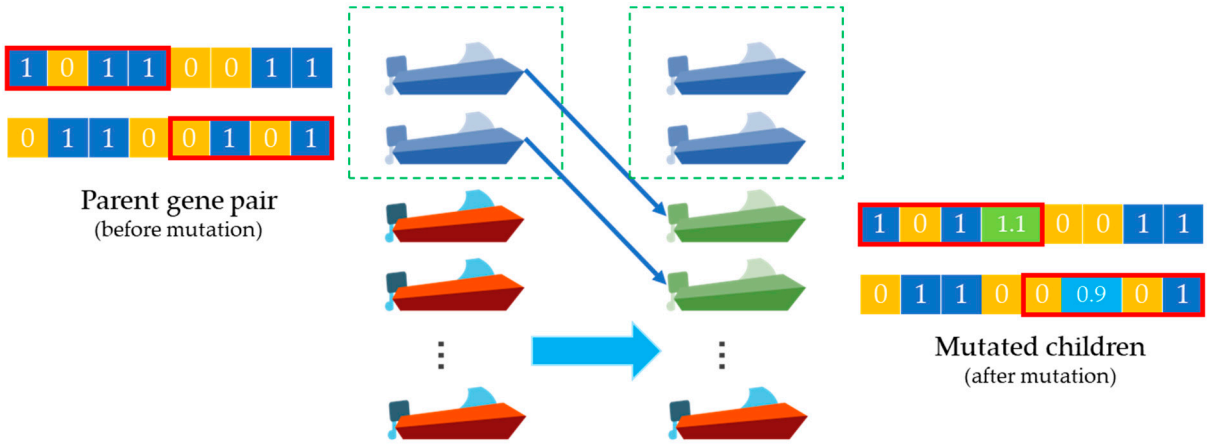

Figure 10. USV performing the mutation process to reproduce mutated children. The red USVs are the original ones, the blue USVs are the children after crossover process, and the green color indicates USV children after mutation (mutated children).

c. End function

The "end" function serves to conclude the iteration process or initiate a new iteration when the USV encounters difficulties in executing the pathfinding task. Specifically, we halt and restart the learning process under the following conditions:

1. If the USV collides with obstacles or barriers in the environment;
2. When the agent produces a suboptimal or too-weak solution (determined by both the achieved fitness value and the time iteration);
3. If the USV submerges below the water's surface.

Ultimately, the process concludes when the USV successfully reaches the target, completing more than one full round without colliding with any obstacles.

4.3. Performance Evaluation

In this study, we use three parameters to evaluate the performance of the proposed method (the modified neuronal GA) compared to that of the baseline method (the neural network-based GA). These three parameters include convergence speed, travel distance, and travel time. The speed of convergence is determined according to the total number of genomes required by the model to be able to produce. The fewer total genomes needed, the faster a model will produce a convergent solution or path (the USV does not hit an obstacle). In our research, we used 40 genomes for one generation, so the total number of genomes follows Equation (4):

$$Total\ Genome = 40\ * \sum generation + \sum current\ genome. \tag{4}$$

The distance evaluation parameter is determined by the total distance traveled by the USV to make one full rotation on the track (from one point back to the starting point). The total displacement path is described as follows (Equation (5)) [11]:

$$D = \sum_{i=0}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}, \tag{5}$$

where x_i, y_i, and z_i denote the X, Y, and Z coordinate axis values of node i. For the travel time parameter, this is determined based on the time required for the USV to complete one full circle from one place back to the same place. Travel time values are obtained from the simulator platform that we use.

To measure the significance of the performance of the proposed algorithm compared with that of the baseline method, we used the t-test. This test is commonly employed when we have two sets of data and want to assess whether or not any observed differences between them are statistically significant. Therefore, the t statistic for a paired t-test is as follows [32]:

$$t = \frac{\overline{X_1} - \overline{X_2}}{s_d / \sqrt{n}}, \tag{6}$$

where $\overline{X_1} - \overline{X_2}$ denotes the mean of the differences between paired observations, s_d is the standard deviation of the differences, and n is the number of pairs (or the sample size). This t Stat value will be compared with the t Critical value to determine whether the performance of the proposed method produces a significant increase or not. If t Stat > t Critical, or p-Value < alpha, then we can conclude that the increase in the performance of the proposed method is significant, and vice versa.

5. Results and Analysis

We tested our proposed methodology using a three-dimensional simulator platform. This simulation was useful to determine the performance of the proposed method before being applied directly in the real world. We used 40 genomes for one generation, and we selected the ten best agents and four worst agents. The term genome represents the agent or USV unit that is generated to produce a solution. We conducted the experiment in

two different environments: static and dynamic. For the experiments in each environment, we varied the number of sensor inputs and the mutation rate value. We used three different types of sensors input according to the configuration in Figure 4. Variations in the mutation rate values started from 0.025, 0.055, 0.105, 0.155, and 0.205 to 0.225. We conducted experiments with these variations using a baseline method (the neural network-based genetic algorithm or NNGA) and compared it to our proposed methodology (the modified neuronal GA).

5.1. Sample Experiment

Based on the experimental results, we successfully simulated a USV in static and dynamic environments using both the baseline method and the proposed method. Figures 11 and 12 show examples of the experimental results in the static and dynamic environments for the baseline method and the proposed method, respectively.

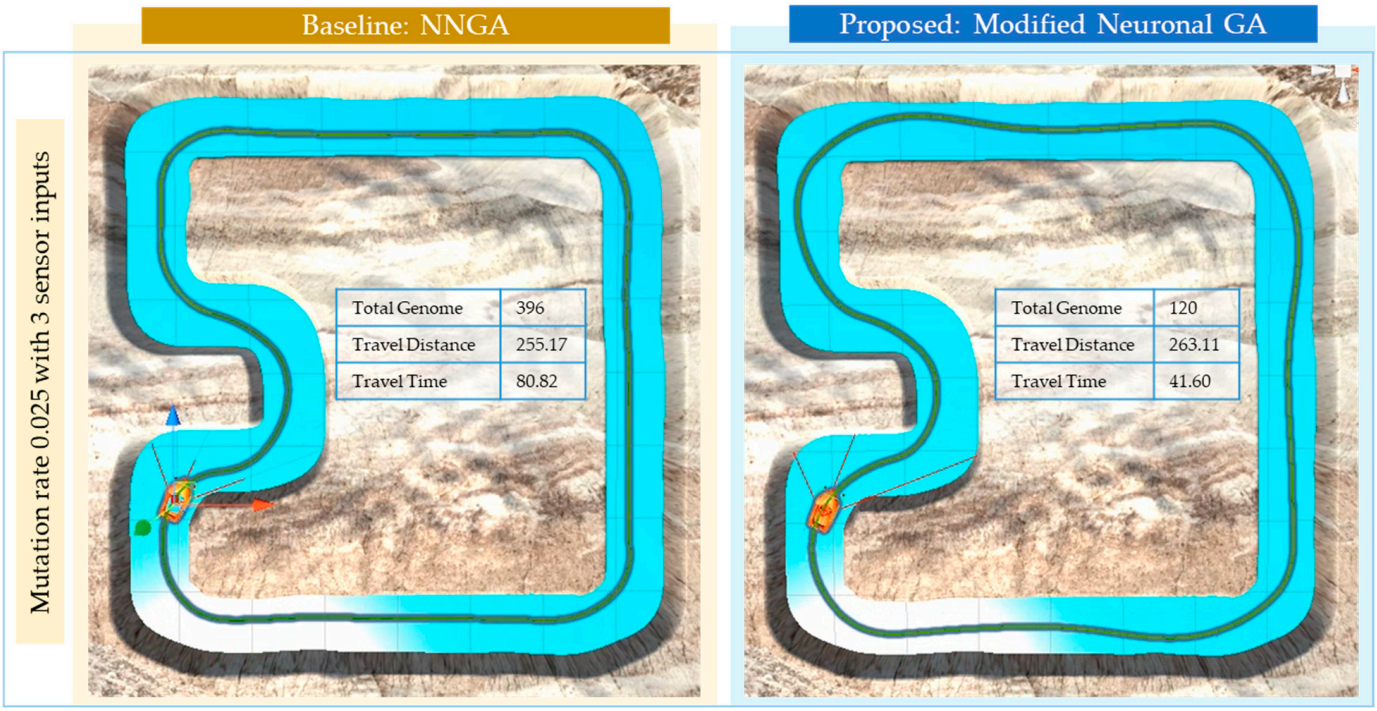

Figure 11. Examples of experimental results in static environments for the baseline method (**left**) and the proposed method (**right**) with three sensor inputs and a mutation rate value of 0.025 (this figure is based on Tables A1 and A2 in Appendix A).

Figure 11 shows how paths in a static environment are formed. The experimental image was produced using a USV with three sensor inputs and a mutation rate value of 0.025. The experimental results show how the proposed method can produce paths that converge faster than those of the baseline method. The proposed method only requires a total of 120 genomes to be able to converge, but the baseline method requires a total of up to 369 genomes. Although, for one full round, the proposed method produces a longer path, the proposed method produces a faster travel time with 41.60-time units (around two times faster) compared to that of the baseline method, which is 80.82-time units.

Figure 12 shows the results of path planning in a dynamic environment. The experimental image was produced using a USV with five sensor inputs and a mutation rate value of 0.255. These experiments show how the proposed method can produce paths that converge more quickly with 162 genomes (around 3.2 times faster) when compared to those of the baseline method with 522 total genomes. In this figure, it can also be seen that for experiments conducted in a dynamic environment, it is possible for a USV to create or generate different paths between the first round and the next round. For the baseline

method, the USV produces different paths when turning to the left and right twice in a row (in the area to the left of the center). However, the experimental results with the proposed method produce a significantly different path with every right turn. This does not happen in a static environment (where there is no water wave factor), where the path is almost the same for the first and subsequent rounds.

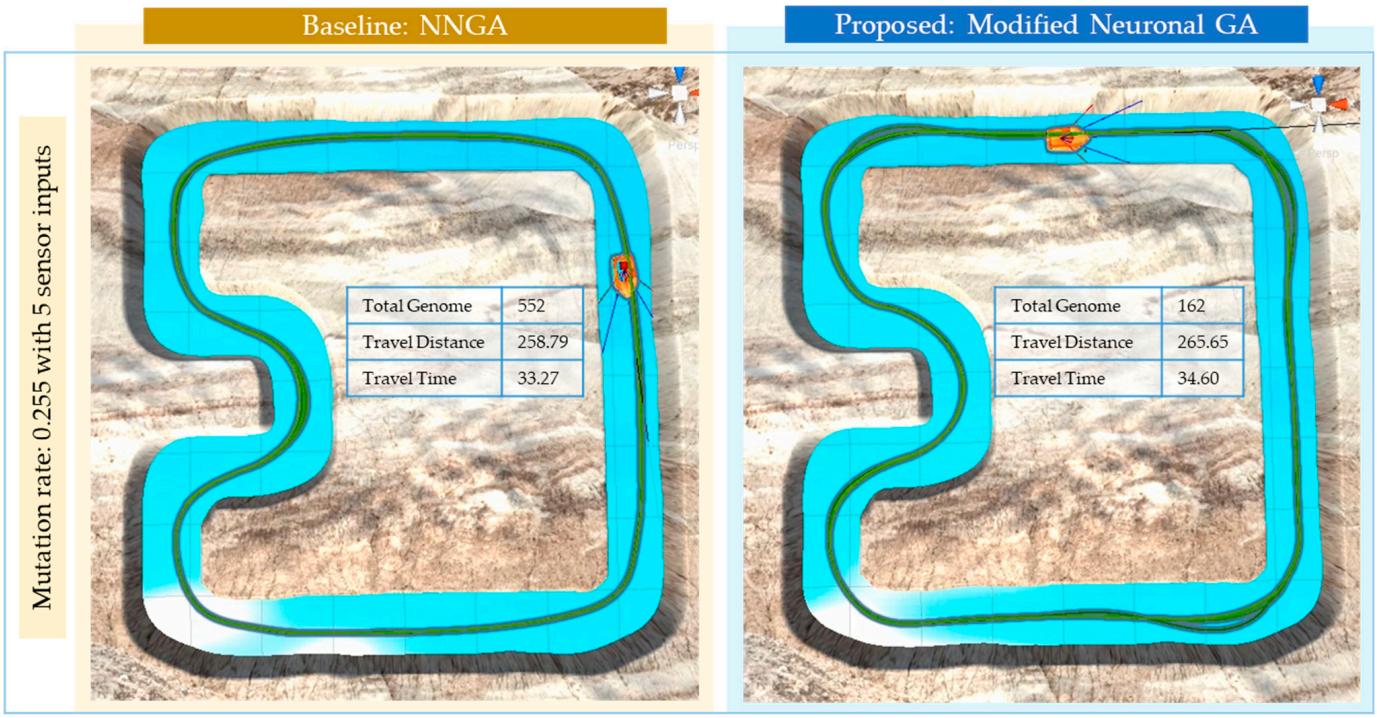

Figure 12. Examples of experimental results in dynamic environments for the baseline method (**left**) and the proposed method (**right**) with five sensor inputs and a mutation rate value of 0.255 (this figure is based on Tables A3 and A4 in Appendix A).

5.2. Overall Analysis

Table 1 shows a summary of the experimental results from a static environment for the baseline and proposed methods. Table 2 shows the result for the dynamic environment. The total genome, distance traveled, and time traveled are average values for all experiments with varying mutation rate values. Distance is expressed in distance units and time is expressed in time units in the simulator. Detailed experimental results for each mutation rate, total generation, and current genome value are provided in Appendix A.

Table 1. Summary of experimental results from a static environment for the baseline and proposed methods.

Input Number	Baseline (NNGA)			Proposed (Modified Neuronal GA)		
	Total Genome	Distance	Time	Total Genome	Distance	Time
3 Input	354.83	259.68	64.29	239.33	262.77	54.11
5 Input	301.67	255.98	46.68	291.83	257.53	50.86
7 Input	307.50	271.50	55.79	187.50	266.74	69.27
Mean	321.33	262.39	55.58	239.56	262.35	58.08

From Table 1, we can see that, in general, the proposed method can produce convergent solutions more quickly (requiring fewer total genomes) than the baseline method can. These positive results were achieved for experiments with three, five, and seven inputs. The USV with seven sensor inputs shows the most significant difference (1.64 times faster). For

the travel distance parameters, overall, the proposed method is slightly better (produces shorter paths) compared to those of the baseline method. Because seven sensor inputs have a significant increase compared to three and five inputs, the average travel distance is better using the proposed method. The detailed results in the table in Appendix A also show that the proposed method is superior for travel distance in many experiments. Nevertheless, for travel time parameters, the proposed method does not show better results. Only at input sensor three does the proposed method show a faster travel time. However, the decline in performance was declared not significant, as proven by the significance test in Section 5.3. Therefore, we can indicate that in this static case, the proposed method can still compete because it is superior in terms of travel distance parameters. Moreover, this study has limitations for not considering other USV performance parameters such as stability and smoothness. USVs with advantages in the distance aspect are suitable for implementing victim evacuation schemes, such as those in flood disasters, accidents at sea, and many more.

Table 2. Summary of experimental results from a dynamic environment for the baseline and proposed methods.

Input Number	Baseline (NNGA)			Proposed (Modified Neuronal GA)		
	Total Genome	Distance	Time	Total Genome	Distance	Time
3 Input	416.00	260.86	59.82	358.67	264.84	47.58
5 Input	558.50	258.28	53.06	440.67	263.36	42.36
7 Input	765.00	268.30	58.61	459.67	271.46	45.86
Mean	579.83	262.48	57.16	419.67	266.55	45.27

When compared with experiments conducted in a static environment, those conducted with the proposed method show better performance in a dynamic environment. For all input number sensor values, the proposed method can produce a convergent solution around 1.38 times faster (and requires less total genome) compared to that with the baseline method. Moreover, for all input number values, the proposed method is also around 1.26 times faster in terms of travel time compared to that under the baseline method. However, for all input numbers, the proposed method produces a longer travel distance. This longer distance is covered more quickly, indicating that in the proposed method, the USV moves with a higher acceleration value compared to that under the baseline method. Therefore, the proposed method produces faster solutions than does the baseline method.

5.3. Significance Test

In this study, we used the *t*-test to test the significance of the results of the experiment. We used a paired-samples *t*-test by assuming equal variances. Each pair of data points (result comparison between the baseline and proposed method in Appendix A) is treated as a single observation, and the degrees of freedom are calculated based on the number of pairs. Therefore, the degrees of freedom (df) = (n – 1) = (18 – 1) = 17. *t* Critical is obtained using Equation (6). This equation was used to calculate the value of *t* Critical, *t* Statistics, and *p*-Value for the data point. In this *t*-test, we used the alpha or significance level (α) of 0.05. The Critical *t* value, *t* Statistics, and *p*-Value for the experimental results in a static environment are presented in Table 3 below.

Table 3. *t* Critical, *t* Statistics, and *p*-Value for experimental results in a static environment.

Result Comparison of Baseline and Proposed Method	*t* Critical	*t* Stat	*p*-Value
Total genome (convergence speed)	1.69	1.74	0.045
Travel distance	1.69	0.015	0.493
Travel time	1.69	−0.410	0.342

Table 3 shows that the total genome (convergence speed) in a static environment produces a *t* Critical value of 1.69, a *t* Stat value of 1.74, and a *p*-Value of 0.045. Because *t* Stat > *t* Critical, and *p*-Value < alpha, we can conclude that the increase in convergence speed in static environment experiments with the proposed method is significant. From this table, even though the overall distance traveled under the proposed method is slightly better than that under the baseline method, the increase cannot be said to be significant. This is because, for travel distance, *t* Statistics < *t* Critical, and *p*-Value > alpha. Then, as seen in Table 1, the proposed method cannot produce better performance for the travel time parameter, although based on Table 3, the decrease in performance is not significant because *t* Statistics < *t* Critical, and *p*-Value > alpha, so the performance of the proposed method can still compete with the performance of the baseline method for this parameter.

Table 4 shows the *t* Critical, *t* Statistics, and *p*-Value for the experimental results in a static environment. Table 4 shows a *t* Critical value of 1.69, a *t* Stat value of 1.71, and a *p*-Value of 0.049. Because *t* Stat > *t* Critical, or *p*-Value < alpha, we can conclude that the increase in the convergence speed in dynamic environment experiments with the proposed method is significant. Then, for the travel time parameter, we obtain a *t* Critical value of 1.69, *t* Stat value of 2.14, and *p*-Value of 0.019. Because *t* Stat > *t* Critical, or *p*-Value < alpha, we can conclude that the increase in the USV travel speed for one rotation in the dynamic environment experiment using the proposed method is significant. For the travel distance parameter, although the proposed method does not produce better performance, based on Table 4, we know that the decrease is not significant. This is because even though *p*-Value < alpha, *t* Stat < *t* Critical.

Table 4. *t* Critical, *t* Statistics, and *p*-Value for experimental results in a dynamic environment.

Result Comparison of Baseline and Proposed Method	*t* Critical	*t* Stat	*p*-Value
Total genome (convergence speed)	1.69	1.71	0.049
Travel distance	1.69	−1.95	0.029
Travel time	1.69	2.14	0.019

Based on the experimental results, the proposed method produces better performance for convergence speed in both static and dynamic environments. This performance increase is significant based on the statistical tests. For the travel distance parameter, the proposed method is superior in static environments, but not in dynamic environments. However, for both environments, the increase and decrease in the performance of the proposed method is not significant based on statistical tests. For the travel time parameter, the proposed method does not produce better performance for static environments, but it manages to show significantly better performance in dynamic environments (also proven via statistical tests). Based on the experimental results, compared to that on under the baseline method, the proposed method can produce a faster convergent solution for USV path planning with competitive performance for total distance and time traveled in both static and dynamic environments. This is because we applied a partially exponential fitness function, as explained in the Section 4.2. This proves that if we apply the fitness function in exponential space, the slightly different output fitness values will be more visible because there is the influence of the power values p, q, and r. In addition to this, an inverse of the time parameter allows the agent to move in a shorter time to obtain a greater fitness value. Thus, this form of modification can be applied in other genetic algorithm research to produce higher convergence speeds.

6. Conclusions

This research was conducted to solve unmanned surface vehicle (USV) dynamic environmental problems as well as convergence problems in evolutionary algorithms. This research proposes a neuronal genetic algorithm that utilizes neural network inputs for processing with a genetic operator. The modifications in this research were carried out by applying a partially exponential-based fitness function to the neuronal genetic algorithm.

We also implemented an inverse time variable to the fitness function. Both modifications aim to produce faster convergence. Based on the experimental results and compared to basic neural-network-based genetic algorithms, the proposed method can produce faster convergent solutions and requires a smaller total genome. This improvement was declared significant using statistical test methods. Then, for the travel distance parameter, the proposed method provides better performance in static environments. For travel time, the proposed method provides significantly better performance in dynamic environments. This proves that, in exponential space, slightly different output fitness values will be more visible because there is the influence of power values. In addition, an inverse of the time parameter allows the USV to move in a shorter time to obtain a greater fitness value. Therefore, this form of modification can be applied in other genetic algorithm research to produce better convergence speeds.

There are many kinds of improvements that can be implemented in the future. The challenge associated with travel distance may be resolved by considering the radius value between the center of the map and the position of the USV when moving, although this is not necessarily relevant to real-world cases. Other research on unaddressed USV problems, such as autonomous navigation and cognitive planning in a complex environment, can be carried out using other GA modifications.

Author Contributions: Conceptualization, N.H., W.D. and H.N.; methodology, N.H.; software, N.H.; validation, N.H., W.D. and H.N.; formal analysis, N.H.; investigation, N.H.; resources, N.H.; data curation, N.H.; writing—original draft preparation, N.H.; writing—review and editing, N.H. and W.D.; visualization, N.H.; supervision, H.N.; project administration, H.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://github.com/nurhamid26/ModifiedNeuronalGA (accessed on 5 November 2023).

Acknowledgments: The first author would like to express the gratitude to Ministry of Education, Culture, Sports, Science and Technology (MEXT) Japan and Japan International Cooperation Agency (JICA) for providing the scholarship during the study at Kanazawa University, Japan.

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

Appendix A

This appendix displays detailed results for all experiments carried out in static and dynamic environments, using the baseline and proposed methods. The experimental results displayed are detailed based on each mutation rate value, the summary or average of which is shown in the Section 5.

Table A1. Detailed results from static environment experiment using baseline method (NNGA).

Input Number	Mutation Rate	Generation	Genome	Total Genome	Distance	Time
3 inputs	0.025	9	36	396	255.17	80.82
	0.055	7	5	285	263.95	50.09
	0.105	10	4	404	256.08	39.24
	0.155	4	1	161	260.21	82.62
	0.205	12	29	509	265.74	49.63
	0.255	9	14	374	256.92	83.34
		Average		354.83	259.68	64.29

Table A1. *Cont.*

Input Number	Mutation Rate	Generation	Genome	Total Genome	Distance	Time
5 inputs	0.025	3	6	126	253.98	81.64
	0.055	16	0	640	255.72	41.22
	0.105	11	0	440	255.29	31.30
	0.155	4	2	162	261.14	31.91
	0.205	4	2	162	255.64	53.60
	0.255	7	0	280	254.09	40.38
		Average		301.67	255.98	46.68
7 inputs	0.025	3	1	121	278.39	35.50
	0.055	11	2	442	268.97	85.07
	0.105	7	2	282	259.50	43.01
	0.155	9	0	360	272.56	34.60
	0.205	4	0	160	270.07	61.81
	0.255	12	0	480	279.52	74.74
		Average		307.50	271.50	55.79

Table A2. Detailed results from static environment experiment using proposed method (modified neuronal GA).

Input Number	Mutation Rate	Generation	Genome	Total Genome	Distance	Time
3 inputs	0.025	3	0	120	263.11	41.60
	0.055	5	25	225	262.46	53.25
	0.105	6	0	240	259.23	54.12
	0.155	5	2	202	259.90	82.82
	0.205	5	7	207	268.57	33.03
	0.255	11	2	442	263.36	59.85
		Average		239.33	262.77	54.11
5 inputs	0.025	4	3	163	255.79	70.86
	0.055	13	24	544	256.00	47.68
	0.105	9	0	360	266.94	32.38
	0.155	2	1	81	258.46	64.35
	0.205	7	2	282	254.00	54.00
	0.255	8	1	321	254.00	35.91
		Average		291.83	257.53	50.86
7 inputs	0.025	3	1	121	262.41	70.95
	0.055	5	27	227	262.43	71.06
	0.105	1	0	40	269.13	75.92
	0.155	5	19	219	271.37	49.21
	0.205	3	32	152	268.11	70.25
	0.255	9	6	366	267.00	78.23
		Average		187.50	266.74	69.27

Table A3. Detailed results from dynamic environment experiment using baseline method (NNGA).

Input Number	Mutation Rate	Generation	Genome	Total Genome	Distance	Time
3 inputs	0.025	13	37	557	267.42	80.52
	0.055	11	5	445	255.96	44.06
	0.105	18	2	722	256.62	66.25
	0.155	5	0	200	266.07	35.67
	0.205	9	8	368	258.20	55.44
	0.255	5	4	204	260.91	77.00
		Average		416.00	260.86	59.82
5 inputs	0.025	15	6	606	261.62	38.18
	0.055	19	0	760	256.11	73.35
	0.105	6	0	240	263.27	60.87
	0.155	17	19	699	255.57	81.94
	0.205	13	4	524	254.34	30.76
	0.255	13	2	522	258.79	33.27
		Average		558.50	258.28	53.06
7 inputs	0.025	17	36	716	270.44	78.50
	0.055	1	32	72	271.06	48.56
	0.105	24	1	961	269.28	43.37
	0.155	26	1	1041	268.33	64.45
	0.205	24	0	960	269.12	76.15
	0.255	21	0	840	261.55	40.60
		Average		765.00	268.30	58.61

Table A4. Detailed results from dynamic environment experiment using proposed method (modified neuronal GA).

Input Number	Mutation Rate	Generation	Genome	Total Genome	Distance	Time
3 inputs	0.025	11	29	469	268.65	34.80
	0.055	4	1	161	260.77	41.11
	0.105	2	16	96	262.57	35.16
	0.155	4	3	163	254.76	40.28
	0.205	17	32	712	273.19	87.53
	0.255	13	31	551	269.12	46.58
		Average		358.67	264.84	47.58
5 inputs	0.025	4	16	176	263.50	56.00
	0.055	15	2	602	261.65	36.50
	0.105	2	18	98	257.56	63.49
	0.155	26	2	1042	275.83	32.56
	0.205	14	4	564	255.98	30.98
	0.255	4	2	162	265.65	34.60
		Average		440.67	263.36	42.36

Table A4. *Cont.*

Input Number	Mutation Rate	Generation	Genome	Total Genome	Distance	Time
7 inputs	0.025	6	0	240	272.13	64.22
	0.055	16	14	654	273.00	48.98
	0.105	12	15	495	273.45	32.91
	0.155	5	9	209	267.76	39.68
	0.205	5	0	200	275.31	34.29
	0.255	24	0	960	267.10	55.10
		Average		459.67	271.46	45.86

References

1. Guo, H.; Mao, Z.; Ding, W.; Liu, P. Optimal search path planning for unmanned surface vehicle based on an improved genetic algorithm. *Comput. Electr. Eng.* **2019**, *79*, 106467. [CrossRef]
2. Zhou, C.; Gu, S.; Wen, Y.; Du, Z.; Xiao, C.; Huang, L.; Zhu, M. The review unmanned surface vehicle path planning: Based on multi-modality constraint. *Ocean Eng.* **2020**, *200*, 107043. [CrossRef]
3. Chang, H.-C.; Hsu, Y.-L.; Hung, S.-S.; Ou, G.-R.; Wu, J.-R.; Hsu, C. Autonomous water quality monitoring and water surface cleaning for unmanned surface vehicle. *Sensors* **2021**, *21*, 1102. [CrossRef]
4. Jorge, V.A.M.; Granada, R.; Maidana, R.G.; Jurak, D.A.; Heck, G.; Negreiros, A.P.F.; dos Santos, D.H.; Gonçalves, L.M.G.; Amory, A.M. A survey on unmanned surface vehicles for disaster robotics: Main challenges and directions. *Sensors* **2018**, *19*, 702. [CrossRef]
5. Akdağ, M.; Solnør, P.; Johansen, T.A. Collaborative collision avoidance for Maritime Autonomous Surface Ships: A review. *Ocean Eng.* **2022**, *250*, 110920. [CrossRef]
6. Shao, Z.; Lyu, H.; Yin, Y.; Cheng, T.; Gao, X.; Zhang, W.; Jing, Q.; Zhao, Y.; Zhang, L. Multi-Scale Object Detection Model for Autonomous Ship Navigation in Maritime Environment. *J. Mar. Sci. Eng.* **2022**, *10*, 1783. [CrossRef]
7. Zhou, X.; Wu, P.; Zhang, H.; Guo, W.; Liu, Y. Learn to Navigate: Cooperative Path Planning for Unmanned Surface Vehicles Using Deep Reinforcement Learning. *IEEE Access* **2019**, *7*, 165262–165278. [CrossRef]
8. Singh, Y.; Sharma, S.; Sutton, R.; Hatton, D.; Khan, A. A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents. *Ocean Eng.* **2018**, *169*, 187–201. [CrossRef]
9. Song, R.; Liu, Y.; Bucknall, R. Smoothed A* algorithm for practical unmanned surface vehicle path planning. *Appl. Ocean Res.* **2019**, *83*, 9–20. [CrossRef]
10. Yu, J.; Yang, M.; Zhao, Z.; Wang, X.; Bai, Y.; Wu, J.; Xu, J. Path planning of unmanned surface vessel in an unknown environment based on improved D*Lite algorithm. *Ocean Eng.* **2022**, *266*, 112873. [CrossRef]
11. Hamid, N.; Dharmawan, W.; Nambo, H. Autonomous Evacuation Boat in Dynamic Flood Disaster Environment. In Proceedings of the ICACSIS 2022: 14th International Conference on Advanced Computer Science and Information Systems, Depok, Indonesia, 1–3 October 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 117–122. [CrossRef]
12. Lin, X.; McConnell, J.; Englot, B. Robust Unmanned Surface Vehicle Navigation with Distributional Reinforcement Learning. *arXiv* **2023**. Available online: http://arxiv.org/abs/2307.16240 (accessed on 30 July 2023).
13. Xing, B.; Wang, X.; Yang, L.; Liu, Z.; Wu, Q. An Algorithm of Complete Coverage Path Planning for Unmanned Surface Vehicle Based on Reinforcement Learning. *J. Mar. Sci. Eng.* **2023**, *11*, 645. [CrossRef]
14. Lai, P.; Liu, Y.; Zhang, W.; Xu, H. Intelligent controller for unmanned surface vehicles by deep reinforcement learning. *Phys. Fluids* **2023**, *35*, 037111. [CrossRef]
15. Wu, X.; Chen, H.; Chen, C.; Zhong, M.; Xie, S.; Guo, Y.; Fujita, H. The autonomous navigation and obstacle avoidance for USVs with ANOA deep reinforcement learning method. *Knowl.-Based Syst.* **2020**, *196*, 105201. [CrossRef]
16. Prudencio, R.F.; Maximo, M.R.O.A.; Colombini, E.L. A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *99*, 1. [CrossRef]
17. Tang, F. Coverage path planning of unmanned surface vehicle based on improved biological inspired neural network. *Ocean Eng.* **2023**, *278*, 114354. [CrossRef]
18. Du, Z.; Wen, Y.; Xiao, C.; Huang, L.; Zhou, C.; Zhang, F. Trajectory-cell based method for the unmanned surface vehicle motion planning. *Appl. Ocean Res.* **2019**, *86*, 207–221. [CrossRef]
19. Long, Y.; Liu, S.; Qiu, D.; Li, C.; Guo, X.; Shi, B.; AbouOmar, M.S. Local Path Planning with Multiple Constraints for USV Based on Improved Bacterial Foraging Optimization Algorithm. *J. Mar. Sci. Eng.* **2023**, *11*, 489. [CrossRef]
20. Liang, J.; Liu, L. Optimal Path Planning Method for Unmanned Surface Vehicles Based on Improved Shark-Inspired Algorithm. *J. Mar. Sci. Eng.* **2023**, *11*, 1386. [CrossRef]
21. Bai, X.; Li, B.; Xu, X.; Xiao, Y. USV path planning algorithm based on plant growth. *Ocean Eng.* **2023**, *273*, 113965. [CrossRef]

22. Luan, T.; Tan, Z.; You, B.; Sun, M.; Yao, H. Path planning of unmanned surface vehicle based on artificial potential field approach considering virtual target points. *Trans. Inst. Meas. Control* 2023. [CrossRef]
23. Yang, X.-S. *Nature-Inspired Optimization Algorithms*; Elsevier: London, UK, 2018; Volume 118. [CrossRef]
24. Zitouni, F.; Harous, S. Integrating the Opposition Nelder–Mead Algorithm into the Selection Phase of the Genetic Algorithm for Enhanced Optimization. *Appl. Syst. Innov.* **2023**, *6*, 80. [CrossRef]
25. Xin, J.; Zhong, J.; Yang, F.; Cui, Y.; Sheng, J. An improved genetic algorithm for path-planning of unmanned surface vehicle. *Sensors* **2019**, *19*, 2640. [CrossRef]
26. Wang, H.; Fu, Z.; Zhou, J.; Fu, M.; Ruan, L. Cooperative collision avoidance for unmanned surface vehicles based on improved genetic algorithm. *Ocean Eng.* **2021**, *222*, 108612. [CrossRef]
27. Zhang, W.; Xu, Y.; Xie, J. Path Planning of USV Based on Improved Hybrid Genetic Algorithm. In Proceedings of the 2019 European Navigation Conference (ENC), Warsaw, Poland, 9–12 April 2019; pp. 1–7. [CrossRef]
28. Nazarahari, M.; Khanmirza, E.; Doostie, S. Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Syst. Appl.* **2019**, *115*, 106–120. [CrossRef]
29. Kim, H.; Kim, S.-H.; Jeon, M.; Kim, J.; Song, S.; Paik, K.-J. A study on path optimization method of an unmanned surface vehicle under environmental loads using genetic algorithm. *Ocean Eng.* **2017**, *142*, 616–624. [CrossRef]
30. Hao, K.; Zhao, J.; Li, Z.; Liu, Y.; Zhao, L. Dynamic path planning of a three-dimensional underwater AUV based on an adaptive genetic algorithm. *Ocean Eng.* **2022**, *263*, 112421. [CrossRef]
31. Pehlivanoglu, Y.V.; Pehlivanoglu, P. An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems. *Appl. Soft Comput.* **2021**, *112*, 107796. [CrossRef]
32. Potochnik, A.; Colombo, M.; Wright, C. T test as a parametric statistic. *Korean J. Anesthesiol.* **2015**, *68*, 540–546. [CrossRef]