Hindawi
Complexity
Volume 2019, Article ID 7456031, 10 pages
https://doi.org/10.1155/2019/7456031

Research Article

The Action Control Model for Robotic Fish Using Improved Extreme Learning Machine

XueXi Zhang, ShuiBiao Chen, ShuTing Cai, XiaoMing Xiong, and Zefeng Hu

School of Automation, Guangdong University of Technology, Guangzhou 510006, China

Correspondence should be addressed to ShuTing Cai; shutingcai@gdut.edu.cn

Received 17 August 2018; Revised 11 December 2018; Accepted 12 February 2019; Published 25 February 2019

Guest Editor: Zhaojie Ju

To achieve fast and accurate adjustment of robotic fish, this paper proposes state prediction model based on the extreme learning machine optimized by particle swarm algorithm. The proposed model can select desirable actions for robotic fish according to precisely predicted states, "adjusting position" or "pushing ball" defined herein. Specifically, the extreme learning machine (ELM) is leveraged to predict the state of robotic fish, from the observations of current surrounding environment. As the outputs in ELM are varying with the randomly initialized parameters, particle swarm optimization (PSO) algorithm further improves the accuracy and robustness of the ELM by optimizing initial parameters. The empirical results on URWPGSim2D simulation platform indicate that the robotic fish tends to carry out appropriate actions using the state prediction model so that we can complete the game efficiently. It proves that the proposed model can make best use of the real-time information of robotic fish and water polo and derive fulfilling action strategy in various scenarios, which meet the requirements of motion control for robotic fish.

1. Introduction

With the rapid development of marine science and technology, underwater robot is applied widely and prevalently in various occasions. The simulation study of underwater robot is becoming one of hot issues in its research fields [1]. In recent years, robot contests are burgeoning in many countries around the world so that the new ideas and progress of robot research can be sufficiently propagated [2]. Against this background, Peking University along with several universities and research institutes established the URWPGSim2D platform, exclusively focusing on simulation research for underwater robots. The platform takes the fish as the simulation object and bionic water as the environment. It builds a real-time simulation system for robotic water polo match under water [3, 4].

Based on the simulation with URWPGSim2D platform, the essential operation on the robotic fish is accurately shifting it to specified location so as to complete more complex tasks in the dynamic environment. However, underwater robots differ greatly from traditional mobile robots in terms of dynamic characteristics, basic motion control methods, and the broadness of work. Specifically,

(1) the disturbance in water environment is generally existed, as the fish whirls while swimming and water generate resistance on the fish, leading to the difficulty for robotic fish to keep a straight line when swimming;

(2) robotic fish cannot be swerved like a mobile robot and its emergency braking performance and fast steering performance are obviously weaker; i.e., the control delay is severe [5];

(3) the real-time requirement is exceedingly high, because the robotic fish must perform the action at a certain speed and has no chance waiting for the decision result for a long time.

Since the resistance in the water is small, the robotic fish still shift along the travel direction, even if the fish itself has stopped swinging. This situation is constantly disturbing the decision-making process. Therefore, it is very difficult for the robotic fish and water polo to stabilize at a certain point in the water, and it is very disturbing for the robotic fish to accurately push the water polo to the target point.

Yu et al. proposed the point-to-point control algorithm for robotic, the purpose of which is to eliminate the initial direction error and distance error between the current and target position [6]. However, because of the uncertain conditions in the environment and the water interference

exerting on robotic fish, the control effect of robotic fish is not ideal. So far most of the research results focus on the path planning of single fish considering obstacle avoidance or heading ball target. Zhao et al. propose an offensive strategy based on a virtual tangent circle for robotic fish competition [7]; Gao et al. propose a method for path planning of robotic fish balls based on fuzzy logic and geometry [8]; Xie et al. propose fuzzy control based steering control algorithm [9]. Guo et al. propose several path finding and path planning methods for smart agents, which consider the criterion of 'faster' or 'arriving on tome' in the route optimization [10–14]. In terms of robustness research, ke haokang also proposed the corresponding stability path planning strategy [15]. These traditional strategies are complex, and a slight carelessness will make the control conditions affect each other and lead to control errors. Therefore these algorithms are not entirely suitable for robotic fish and need further improvement. Chai proposes a method for robotic fish path planning based on genetic algorithm [16]. He exploits grid method to divide the working space of robotic fish and fully considered the influence of underwater environment on robotic fish. However, the algorithm also has the following defects: if the grid is coarse, the precision is low; if it is finely divided, the amount of grid data is too large so that the limitations of the action decision are caused.

To solve the above shortages, this paper present a motion control algorithm based on the extreme learning machine (ELM) optimized by particle swarm optimization (PSO) algorithm [1, 3, 4]. Learning from the environmental information around the robotic fish, the proposed algorithm determines the current position of robotic fish and then chooses the suitable action strategy according to the position. Extreme learning machine [17] belongs to neural network learning algorithm, which has been used in many fields such as identification [18], prediction [19], and medical diagnosis [20]. The advantage of the ELM is "once for all", meaning that the output weights of the model can be directly calculated by the random input weights and the bias. However, as the network is randomly initialized every time, the final learned weights are not exactly the same when the training is terminated, causing that the errors of the training are also not exactly the same (roughly the same). Therefore, the PSO algorithm is introduced to optimize the extreme learning machine, so that the predicted result is unchangeable. The empirical results show that the motion control algorithm based on the extreme learning machine optimized by particle swarm optimization algorithm can achieve desirable swimming path for robotic fish and improve the control effect of robotic fish.

In this paper, we research the competition strategy for the "Underwater Transport" project on URWPGSim2D platform.

2. Algorithm Introduction

2.1. Single Hidden Layer Feedforward Neural Network (SLFN). Single hidden layer feedforward neural network (SLFN) has strong learning ability [21]; it can approximate the complex nonlinear function and solve the problem that the traditional parameters cannot solve. The configuration of SLFN literally includes input layer, hidden layer (also known as intermediate layer), and output layer. Each layer consists of a number of simple neurons in parallel operations, which are fully connected to those of the neighbour layer, and the neurons in the same layer are not connected. An illustration of the SLFN configuration is shown in Figure 1.

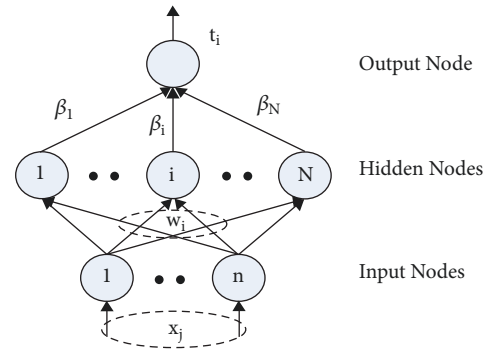

Figure 1: An illustration of single hidden layer neural network model.

From the perspective of mathematics, the standard SLFNs model is expressed as

$$\sum_{i=1}^{\widetilde{N}} g\left(w_i \cdot x_j + b_i\right)\beta_i = o_j, \quad j = 1, 2, \ldots, N \tag{1}$$

where w_i denotes the weight matrix between the input layer and the hidden layer; b_i is the bias vector; β_i is the weight matrix between hidden layer and output layer; o_j is the actual output vector; $g(\cdot)$ is the activation function; N is the total number of training samples; $\widetilde{N}$ is the number of hidden layer units; $w_i \cdot x_j$ means the inner product of w_i and x_j.

Generally, standard SLFNS model can approximate N samples with zero error such that

$$\sum_{i=1}^{\widetilde{N}} \left\| o_j - t_j \right\| = 0 \tag{2}$$

which implies the existence of w and b to satisfy

$$\sum_{i=1}^{\widetilde{N}} g\left(w_i \cdot x_j + b_i\right)\beta_i = t_j, \quad j = 1, 2, \ldots, N \tag{3}$$

With the representation by matrix, (3) can be written in a compact fashion as

$$H\beta = T \tag{4}$$

where $T \in R^{N\times m}$, $\beta \in R^{\widetilde{N}\times m}$, and $H = H(W, b) = (h_{ij})_{N\times\widetilde{N}}$, where $h_{ij} = g(w_i \cdot x_j + b_i)$

At present, BP algorithm is the most mature and most popular algorithms for feedforward neural networks [22]. The main idea under the algorithm is reducing the expected error between predicted output and the actual results (i.e., gradient descent method) to train the network, and the connection weights of the network are adjusted to reduce the

error [23]. BP network has strong fitting ability for nonlinear relation, fault-tolerant ability and precise searching ability. However, at the heart of it is gradient descent method, so it has the following flaws:

(1) If the learning rate of algorithm is too low, the convergence speed decreases; otherwise learning rate is too high to result in divergence.

(2) The BP algorithm may cause the feedforward neural network to be over trained, weakening the generalization ability, and finally get a poor trained network. Therefore, the results should typically verified.

(3) If the cost function is nonconvex, the BP algorithm may generally converge to a local minimum solution.

(4) In most practical applications, the gradient based learning algorithms need to consume a large amount of computation time.

2.2. Extreme Learning Machine (ELM). Single hidden layer feedforward neural network (SLFN) has two prominent abilities:

(1) It can fit complex mapping function directly from the training samples.

(2) It easily provides models for natural or artificial phenomena, which is intricate for traditional classification parameter technology to deal with.

The single hidden layer feedforward neural network is lack of fast learning method. To solve this problem, Huang et al. conducted an in-depth study of the single hidden layer feedforward neural network, and then he put forward and proved two theories [24].

Theory 1. Given a standard SLFN with n-L-m structure and a set of training samples $\{x_j, t_j\}_{j=1}^{N}$, $x_j \in R^n$, $t_j \in R^m$, if the excitation function $g : R \longrightarrow R$ is infinitely differentiable in any region, then for arbitrary interval in the R^n and R space, random generation of w_i and b_i from arbitrary continuous probability distribution has the following:

(1) The reversible probability of hidden layer response matrix H is one.

(2) The probability of $\|T - H\beta\|_F = 0$ is one.

Theory 2. Given arbitrary small positive number $\varepsilon > 0$, a standard SLFN with n-L-m structure, and a set of training samples $\{x_j, t_j\}_{j-1}^{N}$, $x_j \in R^n$, $t_j \in R^m$, $t_j \in R^m$, if the activation function $g : R \longrightarrow R$ is infinitely differentiable in any region, then for random generated w_i and b_i from arbitrary continuous probability distribution within arbitrary interval in R^n and R space, there is $L \leqslant N$ making the probability of $\|T - H\beta\| \leq \varepsilon$ equal to one.

As a matter of fact, a plenty of experimental results validate that adjusting the input weights and the bias vector w_i and b_i cannot yield any benefits. In 2006, Huang et al. put forward the extreme learning concepts of feedforward neural networks and introduce the basic principle in detail. Extreme Learning Machine (ELM) [25] is a special type of single hidden layer feedforward neural network (SLFN) with only one hidden node layer [26]. It was later extended to general SLFN, and its hidden node is similar to neurons. The basic components of the ELM are shown in Figure 2.

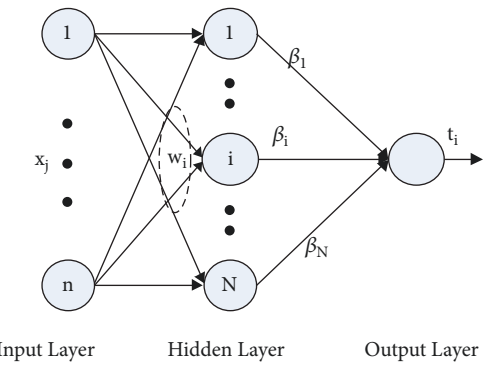

Figure 2: An illustrative network structure of extreme learning machine.

Given the input data X, the output of the network is $\sum_{i=1}^{\widetilde{N}} g(w_i \cdot x_j + b_i)\beta_i$. When the number of hidden units is the same as that of training samples and matrix H is reversible, then (4) has a unique solution. However, in most practical situations, the number of hidden units is far less than training samples. According to Bartlett theory, extreme learning machine can obtain the minimum error solution and good generalization, with least square method to calculate the output weights. Specifically, when w and b are fixed, it is equivalent to calculate the least square solution of linear system in (4), such that

$$\left|H \cdot \widehat{\beta} - T\right| = \min_{\beta} \left|H \cdot \beta - T\right| \tag{5}$$

$$\widehat{\beta} = H^{+}T \tag{6}$$

where H^+ is the Moore-Penrose generalized inverse of H. The minimum norm least square solution is unique, so that the training error reaches the minimum. That is to say, for the randomly assigned weight and bias vectors, the weights of the output layer can be obtained by solving the least square solution of the linear equation, as long as the number of hidden layer neurons is appropriately set up. The ELM algorithm is detailedly described as follows.

Step 1. It specifies the training sample set $\{x_j, t_j\}_{j-1}^{N}$, number of hidden nodes L, and the excitation function $g(\cdot)$.

Step 2. Input weight and bias vectors w_i, b_i, $i = 1, \ldots, L$ are randomly generated.

Step 3. The response matrix H of the characteristic of the training sample in the hidden layer is calculated.

Step 4. It is calculated according to (4).

When β is derived, a single hidden layer feedback neural network is completed. For an unknown test sample X, we can use a single hidden layer feedback neural network to predict its label, following the formula as follows:

$$f_L(x) = h(x)\beta \tag{7}$$

where h(x) is the response of the hidden layer of neural network for X.

The ELM algorithm ignores adjustment of the input weight and bias vectors and the choice of parameters is simple. Therefore the iteration is not required during the whole training process so as to significantly improve the training speed. The most prominent advantage of ELM algorithm is its high efficiency. At the same time, the ELM algorithm overcomes the limitations of local optimization and over fitting typically existing in gradient algorithm (such as BP algorithm), so that the better results are well guaranteed [27].

2.3. Particle Swarm Optimization Algorithm (PSO). Particle swarm optimization (PSO) algorithm was proposed by the American scientist Kennedy and Eberhart in 1995. The initial idea is originated from the simulation of a biological and social system [28]. After repeated theoretical and experimental validation, the researchers found that the PSO algorithm can be used as a new and efficient global optimization method. The main optimization strategy can be described as food search by a flock of birds. Assuming there is only one piece of food in this area, the birds are randomly searching for food. In the very beginning, all the birds did not know where the food was, but they are becoming aware of which bird is nearest to the food and the location at which they were once closest to the food. According the above two information, each bird is trying to determine the flight direction for food search.

Inspired from the procedure above, PSO algorithm is put forward to solve various optimization problem. We can take the birds foraging process as an optimization problem, in which the position of each bird is considered as a potential solution to the problem, corresponding to the position of the particles in the n-dimensional search space [29]. Regarding the food as the optimal solution to the problem, when the food is discovered, it is equivalent to search for the optimal solution. In the iterative process, all particles are estimated by a function to measure their fitness values. Each particle modifies the subsequent direction and distance of its flight according to the following information:

(1) Its current position.

(2) Its current speed.

(3) The distance between its current position and its historical optimal position.

(4) The distance between its current position and the historical optimal position of the bird flock.

In particular, the procedure of PSO algorithm is listed as follows.

Step 1. It initializes a group of random particles (Population size M), with randomly initialized position X and the velocity V in the range allowed, and specifies the inertia weight w, learning factors c_1, c_2, etc.

Step 2. It calculates the initial fitness value of each particle Ifitness, with the best fitness value of the Ifitness set as the global initial fitness Gfitness.

Step 3. The fitness value of each particle is compared with that of its best historical position Ibest. If it is better, it will be the optimal value in the particle's history. Accordingly the best position of the individual history is updated by the current position. Otherwise, it stays the same.

Step 4. The fitness value of each particle is compared with the fitness value of the historical optimal position of the bird flock Gbest. If it is better, then it will be the global optimal value of the particles' history. Otherwise, it stays the same.

Step 5. The velocity and position of the particle are updated according to formula equations (8) and (9)

$$v_i = w \times v_{i-1} + c_1 \times (Pbest - x_{i-1}) + c_2 \times (Gbest - x_{i-1}) \tag{8}$$

$$x_i = x_{i-1} + v_i \tag{9}$$

Step 6. If the fitness value is good enough or the maximum number of iterations is reached, then stop; otherwise, return to Step 2.

3. ELM for Decision-Making of Robotic Fish

In the simulation competition of robotic fish, the main research falls on how to make the robot fish complete given tasks in a dynamic environment, where path planning and action are essential to complete the task. The motion control is the key module to control the movement of the robotic fish in the water according to the predetermined trajectory. It ensures accurate implementation of the game strategy. In other words, the quality of the action control will directly affect the task completion of robotic fish.

3.1. Action Decision

3.1.1. Determination of Hitting Point. When moving the water polo, the robotic fish should select an action strategy based on surrounding environmental information. If the water polo is between the fish and landmark, the fish should take the water polo to the landmark. We define this process as "pushing ball". If the fish is between the water polo and landmark, or if the landmark is between water polo and fish, the robotic fish need to adjust its position at first until it reaches the "pushing ball" state and then it begins to push the water polo. We define this process as "adjusting position". In the classic action decision strategy for robotic fish, if the robotic fish is located between the water polo and the landmark, especially when the three are in the same line, the robotic fish will push the water polo away from the landmark. This will increase the time of target completion and may even lead to the failure of the game, which is shown in Figure 3.

From the above analysis, this paper chooses the hitting point and action strategy based on the current environmental information surrounding the robotic fish. The main idea of the strategy is shown in Figure 4. The centers of landmark and water polo are connected and the line intersects with water polo at the distant point P. Then the perpendicular of the line

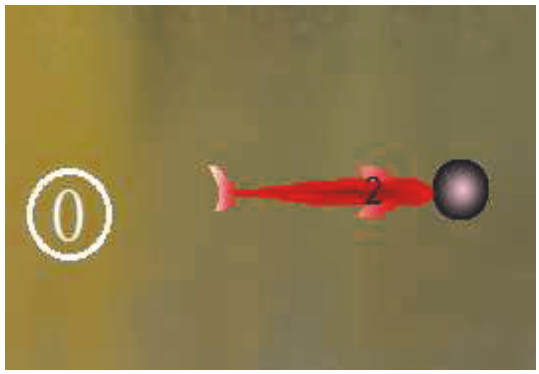

Figure 3: An illustration of the classic action decision strategy.

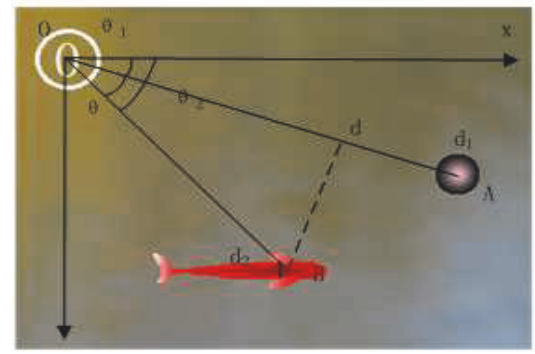

Figure 5: Determine the robotic fish position.

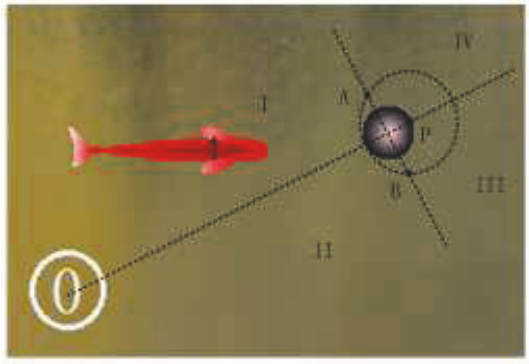

Figure 4: Determination of hitting point.

is drawn through the center of the water polo, so the field is divided into I, II, III, and IV four regions. Taking P as the center and the diameter of water polo as the radius, a circle is drawn. The circle intersects with the previous perpendicular at Point A and B. On the top of these definitions, if the robotic fish is in the I area, A is the hitting point; if the robotic fish is in the II area, B is the hitting point; if the robotic fish is in the III or IV area, P is the hitting point.

3.1.2. Determination of the State. In order to determine the correct action for robotic fish, it is necessary to determine the state of the robotic fish, i.e., "pushing ball" or "adjusting position", according to the surrounding environmental information. This can be expressed as a classification process in which the categories include the four regions I, II, III, and IV as defined above. In order to make the classification more accurate, firstly we need to abstract the biggest factors affecting the robotic fish position. Obviously, we can determine the location of the robotic fish according to the coordinates of robotic fish, water polo, and landmarks. For the computer, the information expressed by these three coordinates is not enough to determine the fish's location accurately. Additionally, three coordinates mean that there are six model parameters, and it will definitely increase the complexity of algorithm and consumes more computation time.

In this paper, to accurately describe the location information of the robotic fish, we use slope and distance to determine the position of the robotic fish as shown in the Figure 5, θ_1 and θ_2 are the slope of the water polo and the robotic fish, respectively, and d_1 and d_2 are the distance from the water polo and the robotic fish to the landmark. Let $\theta = \theta 1 - \theta 2$, and d is defined as the projection of OB on OA, i.e., $d = d_2 \cos\theta$. If D=1(-1), $d > d_1$ ($d < d_1$). The location of the robotic fish can be determined by the relation between θ and D, such that one has the following:

(1) When $\theta > 0$, the robotic fish locates in area I or IV area. Obviously, if $\theta > 90^\circ$, the robotic fish locates in the area I; if $\theta < 90^\circ$ and $D < 0$, it locates in the area I; if $\theta < 90^\circ$ and $D > 0$, it locates in the area IV.

(2) When $\theta < 0$, the robotic fish locates in area II or III area. Therefore, if $\theta < -90^\circ$, the robotic fish locates in the area II; if $\theta > -90^\circ$ and $D < 0$, it locates in the area II; if $\theta > -90^\circ$ and $D > 0$, it locates in the area III.

By introducing θ and D, the location of robot fish can be more accurately described. Furthermore, the number of characteristic parameters is decreased yielding the reduced computation time.

3.2. The ELM Optimized by PSO Algorithm

3.2.1. The Basic Ideas of the Algorithm. According to the previous analysis, the robot fish has four position states, and each position state corresponds to one category. So we can simply define the label of these four categories as 1, 2, 3, and 4. Because of θ and D we can accurately describe the environmental information around the robotic fish; in the ELM based action decision model, parameters consist of labels θ and D. Therefore, the purpose of action decision model is to determine the decision function which reproduces the relationship among labels θ and D. Nevertheless, the network is randomly initialized every time, so the error of each training is not exactly the same, causing that the trained weights are not exactly the same (roughly the same). That means the results after each training are slightly different. To solve this problem, the network will be saved every time we find a better result, so that the predicted results will not change. For the above shortages, this paper proposes PSO algorithm to optimize ELM and to search for best initial network to make the predicted results unchanged and optimal [29].

3.2.2. Algorithm Implementation. Neural network and PSO algorithm are two different optimization algorithms. They show different optimization characteristics and are suitable for different optimization problems. However, these two kinds of optimization methods are both developed by simulating or revealing some natural phenomena or processes, so there must be some similarities between them [30]. Thus it is possible to combine their strengths to build a more effective optimization method.

When PSO is adopted to optimize the ELM, the position of each particle in the particle swarm corresponds to the input weight and bias vectors of the ELM [31]. After the output weights of the ELM calculated by a given training set, the output error of a given test set is calculated based on the output weight. The output error is used as the fitness value, and the smaller error indicates the particles have better performance in the search. The error of the output layer of the network is minimized by moving the particle in the weight space, namely, updating the weight of the network. In this way, PSO algorithm optimizes the input weight and bias vectors of the ELM to obtain a smaller error. The particle with the smallest error in each iteration is the global optimum particle so far. The training process is repeated until the error is small enough to meet the requirement or the number of iterations is reached [32]. When the algorithm terminates, the set of weights is the final results. The proposed algorithm with PSO algorithm optimizing ELM is implemented as follows.

Step 1 (initialization of ELM). We set the number of neurons for the input layer, hidden layer and output layer in the network.

Step 2 (initialization of the particle swarm). By setting the maximum and minimum velocity, Vmax and Vmin, of the particle, respectively, the velocity of each particle is randomly generated within the interval [Vmax, Vmin]. The parameters like the inertia weight w, learning factor, and iteration number are also initialized.

Step 3 (fitness calculation for each particle). The output value of a network is calculated based on the ELM to derive the error. In the same way, errors of all particles are calculated. These errors are regarded as the fitness of particles. When using the extreme learning machine to calculate the fitness, the activation function of each neuron is hardlim.

Step 4 (termination test). If the algorithm reaches the maximum number of iterations or the particle fitness value is less than a specified value, algorithm proceeds to Step 7, or it goes to Step 5.

Step 5 (updating the individual and global extremum). For each particle, its current fitness value Ifitness is compared with its optimal value Ibest. If Ifitness < Ibest, then Ibest = Ifitness, and the best position of individual history is replaced by the current position. Similarly, individual fitness value Ifitness is compared with the global optimal value Gbest. If Ifitness < Gbest, then Gbest = Ifitness, and the best position of global history is replaced by the current position.

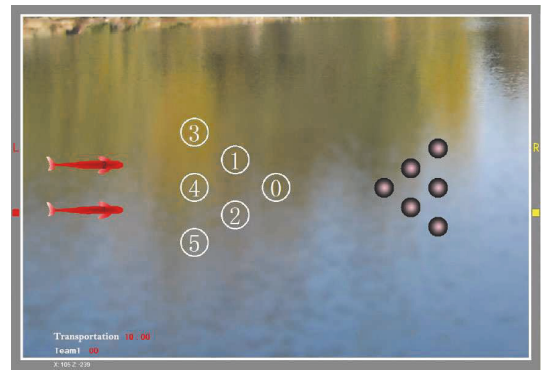

Figure 6: The field of Underwater Transport project.

Step 6 (updating the speed and position of each particle). The velocity and position of the particle are updated according to formula equations (8) and (9) and then judge whether the speed and position of the particles are within the preset range.

Step 7. When the iteration stops, the optimal solution of the problem is the learned weights and bias of ELM which corresponds to the global extremum.

4. Empirical Results

4.1. Introduction to Game Robotic Fish Contest

4.1.1. Introduction to Platform. The underwater robot contest held in China adopts the robot water ball 2D version software (URWPGSim2D) as the platform for 2D simulation competition. URWPGSim2D software provides "Local" and "Remote" operation mode. The Local pattern used in official matches, and this pattern only needs to start a server process (URWPGSim2DServer) [33]. Strategy component DLL files can load directly on the server side, and meanwhile all strategy is calculated on the server side. The simulated field is roughly identical with entity pool in the aspects of 2D model definition, structure, and size ratio. The full size of the field is 3000mm∗2000mm, as shown in the Figure 6. In this paper, we take the field geometry center as the origin of coordinate. The right direction is defined as the X axis positive direction, and the Z axis positive direction points down. Based on X axis positive direction, clockwise 180 degrees is 0 to $-\pi$, and anticlockwise 0 to $-\pi$ [34].

4.1.2. Game Rules. The Underwater Transport is participated by a team, each team has two robotic fish, six types of water polo, and six circular landmarks. The Underwater Transport competition project adopts standard venue, the robotic fish, and other venue elements, as shown in Figure 6.

In the initial state, two robotic fish are located in the left half of the game venue; six types of water polo are numbered from 0 to 5, following the order from left to right, from top to bottom. The left half of the venue has white landmarks, with corresponding number.

When the game starts, the robotic fish push the polo to the corresponding landmarks. When the ball is shifted into the

Table 1: The performance comparison of three models.

Algorithm	Number of Iteration	Number of neurons	Average time[s]	Accuracy
BP	1000	160	32.502347	0.9364
ELM	1	250	0.411309	0.9894
ELM optimized by PSO	6	250	2.027528	1

corresponding landmark (the radius of landmark is 80mm), the team obtains scores. The total game time is 10 minutes.

Every time polo is successfully pushed to the corresponding landmark, one score is counted and the current spent time is recorded. The repeated push to the same position has no scores. Until all 6 balls are pushed to the landmarks, the game is finished and the remaining time is recorded. When all teams finish their games, the team with the highest recorded score wins. If the same scores exist, the team with less time wins. This project involves many entities and objectives, so the strategy is complex and flexible [35].

4.2. The Determination of Robotic Fish State. In this paper, for the "Underwater Transport" project with URWPGSim2D platform, we establish the motion control model for robot fish separately based on the BP neural network, the ELM, and the proposed model, i.e., ELM optimized by PSO. Then we discuss the advantages and disadvantages of these three models.

The learning accuracy of BP neural network is affected by the number of hidden layer, the number of neurons in each layer, and the number of iterations. In contrast, the learning precision of ELM depends on the randomly initialized parameters and the number of neurons in the hidden layer; the exact ELM model contains hundreds of hidden layer neurons. Considering the BP neural network and ELM belong to the network category, these two algorithms will face a common problem that is how the number of neurons in the hidden layer should be determined. At present, there is no scientific model or formula, so the experience is more or less drawn to solve practical problems.

In this paper, we need to determine the optimal hidden layer neurons of the ELM and the single hidden layer BP neural network for robot fish motion control. And they are determined by trial and error. In our experiment, they are obtained by the MATLAB simulation test. 500 training sample data and 100 test data were used, and the number of hidden nodes tested is 100, 150, 200, 250, 300, 350, and 400. The results are shown in Figure 7. It is shown that when the number of hidden layer neurons in the ELM is 250 and the number in the single hidden layer BP neural network is 160, the learning accuracy of these two models is relatively high. They can accurately locate the robotic fish and provide a good basis to decide actions for robotic fish. It is observed that above results are valid for our model.

Based on the above result, we further build the models by BP neural network, ELM, and ELM optimized by PSO algorithm. The three models are all three-layer networks. The number of neurons in the input and output layer are 2 and 1 for each network. The number of neurons in the hidden layer is 160, 250, and 250, respectively. The particle swarm

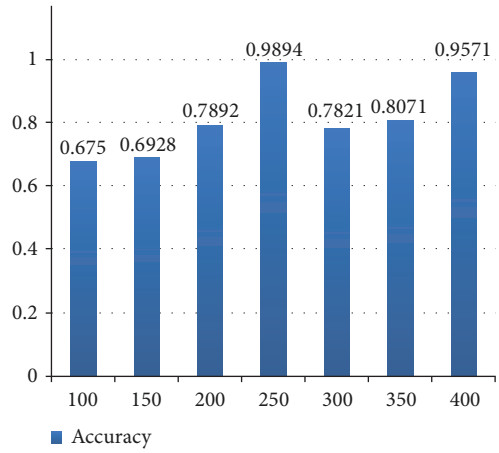

Figure 7: The results with different numbers of hidden neurons in ELM.

size is 10; the learning factor is c1=c2=2.0; the inertia weight is w=0.6; the maximum iteration number is 6. The experimental results for them are shown in Table 1. The highest accuracy value of BP and ELM is recorded in Table 1 from 600 repeated experiments and the accuracy value of ELM optimized by PSO is 1 all the time.

From Table 1, it can be found that the deviation of the BP neural network is too large to search the optimal solution, while the ELM can search the global optimal solution with a higher accuracy and shorter time. ELM takes 0.4s because it requires only one calculation after initializing the input weights and biases. Therefore, the ELM is better than the BP algorithm in terms of time and accuracy. However, considering that the weights of the ELM, which is initialized randomly, have a certain degree of influence on the learning accuracy, the output weights and the training errors are not exactly the same when the training is terminated. That is to say, the output weight of the ELM is not always the optimal solution, but it does not mean they get very bad results. Their accuracy is also more than 90%. What we need to do is to fine-tune its input weight and bias. In this paper, PSO is introduced to find an optimal set of input weights and biases. Its greatest function is to guide the direction in which input weights and biases change. After using the PSO algorithm to optimize the ELM, we can determine a set of input weights and bias, which makes the ELM learning accuracy to the highest. From Table 1 it can be found that, as the ELM optimized by PSO algorithm needs to carry on 6 iterations, it takes a little more time. However, the time consumption is acceptable, considering the accuracy has improved to 1 and there is zero error in the experiment of robotic fish. The maximum number of iterations is 6. Before reaching 6

Table 2: The data of the race.

Strategy	Race	First half	Second half
Original strategy	First race	3	1
	Second race	2	2
	Third race	3	2
Improved strategy	First race	0	9
	Second race	0	9
	Third race	2	7

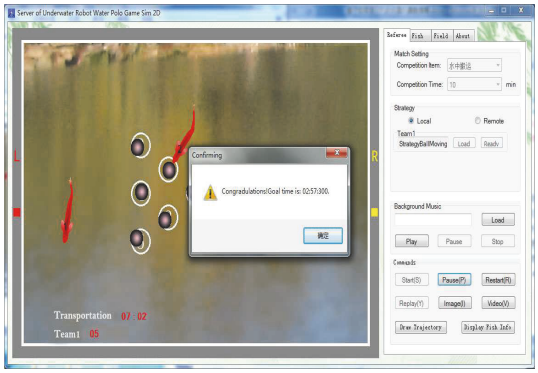

Figure 8: The completion of the game by the proposed model.

iterations, the model has found a set of input weights and biases that are good enough for fitness. Consequently, the ELM has better performance when it is optimized by PSO algorithm.

4.3. Competition Experiment. To verify the performance of the proposed model, we further implement the experiment in the competition. We use our model, i.e., ELM optimized by PSO, to select action for robotic fish in the game. The model is run to complete the game 20 times on the URWPGSim2D platform; finally, the range of time for completing the game is from 167s to 235s as shown in Figure 8. In order to verify the optimized effect, we compare competition scores of the proposed method with the original ELM. We record their scores as experimental data, as shown in Table 2. It is clear that the number of goals for our improved ELM is significantly improved.

The robotic fish action decision strategy based on ELM optimized by PSO algorithm is a kind of dynamic self-organizing strategy; it makes a decision in real time according to the current data of the dynamic variables in the platform. It has a short execution cycle and runs the adjustment each 0.1s. The robotic fish can be regarded as staying in a dynamic environment; thus it is more efficient. The proposed method can not only realize the autonomous decision-making of the hitting point selection of the robotic fish, but also refine the angle range to improve the flexibility of the robotic fish. Compared with the classical action control strategy, our method makes the robotic fish move fast and stable at a predetermined location.

5. Conclusions and Future Work

The proposed method for motion control of robotic fish, namely, the extreme learning machine optimized by particle swarm optimization algorithm, concurrently considers the complexity of the underwater environment and the movement characteristics of the simulated robotic fish. It is the first time that the landmark is used as the coordinate center, and the relative position among the water polo, robotic fish, and landmark are calculated by the slope and distance D, so that the state of the robotic fish can be correctly determined. Meanwhile, according to the consistency of the robot fish movement, the extreme learning machine is exploited to automatically select the hitting point for robotic fish. Then particle swarm optimization algorithm further improves the accuracy and robustness of the ELM by optimizing initial parameters. After implementing our method on the URWPGSim2D platform, the empirical results indicate that it can complete the game with better performance, not only improving the stability of the strategy, but also being able to meet the requirements of action decision for robotic fish. As for the real environment, we are building a physical robot fish. After finishing, we will verify the algorithm validity. In future, we will also further study the action control and path planning for multiple cooperative robot fishes, by investigating and exploring the cooperative routing solutions from the transportation filed [36–38].

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] R. Bogue, "Underwater robots: a review of technologies and applications," *Industrial Robot: An International Journal*, vol. 42, no. 3, pp. 186–191, 2015.

[2] W. Chen, T. Liao, Z. Li et al., "Using FTOC to track shuttlecock for the badminton robot," *Neurocomputing*, vol. 334, pp. 182–196, 2019.

[3] W. Hu and L. Liu, "Cooperative output regulation of heterogeneous linear multi-agent systems by event-triggered control," *IEEE Transactions on Systems Man and Cybernetics*, 2016.

[4] R. Nair, L. Behera, and S. Kumar, "Event-triggered finite-time integral sliding mode controller for consensus-based formation of multirobot systems with disturbances," *IEEE Transactions on Control Systems Technology*, 2017.

[5] M. Yufeng, P. Yongjie, and L. Wei, "Dynamic obstacle avoidance method for underwater robots in velocity vector coordinate system," *Journal of Harbin Engineering University*, vol. vo. 31, no. 2, pp. 159–164, 2010.

[6] J. Liu, I. Dukes, and H. Hu, "Novel mechatronics design for a robotic fish," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[7] Z. Shengchang, J. Zhijian, X. Guangming et al., "Offensive strategy of robotic fish competition based on virtual tangent circle," *Ordnance Industry Automation*, vol. 29, no. 11, pp. 89–91, 2010.

[8] G. Wei, T. Xiuhua, and L. Zonggang, "Path planning of robot fish top ball based on fuzzy logic and geometry," *Ordnance Industry Automation*, vol. 29, no. 11, pp. 85–88, 2010.

[9] K. Haokang and W. Chao, "Robot fish stability and route planning based on URWPGSim2D platform," *Ordnance Industry Automation*, vol. 37, no. 04, pp. 93-86, 2018.

[10] J. Guo, Y. Wu, X. Zhang et al., "Finding the 'faster' path in vehicle routing," *IET Intelligent Transport Systems*, vol. 11, no. 10, pp. 685–694, 2017.

[11] Z. Cao, H. Guo, J. Zhang, F. Oliehoek, and U. Fastenrath, "Maximizing the probability of arriving on time: a practical q-learning method," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pp. 4481–4487, 2017.

[12] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "Finding the shortest path in stochastic vehicle routing: a cardinality minimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1688–1702, 2016.

[13] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "Improving the efficiency of stochastic vehicle routing: a partial lagrange multiplier method," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3993–4005, 2016.

[14] Z. Cao, Y. Wu, A. Rao et al., "An accurate solution to the cardinality-based punctuality problem," *IEEE Intelligent Transportation Systems Magazine*, 2018.

[15] X. Chaoping, K. Feng, and T. Jin, "Research on steering control of bionic robot fish based on fuzzy control," *Robotics Technology and Application*, no. 4, pp. 26–28, 2009.

[16] C. Zhongming, Y. Mei, and L. Shu, "Path planning of robot fish based on genetic algorithm," *Ordnance Industry Automation*, vol. 29, no. 44, pp. 92–96, 2010.

[17] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[18] J. Yang, Y. Jiao, N. Xiong, and D. Park, "Fast face gender recognition by using local ternary pattern and extreme learning machine," *KSII Transactions on Internet and Information Systems*, vol. 7, no. 7, pp. 1705–1720, 2013.

[19] S. Sun, Y. Wei, K.-L. Tsui, and S. Wang, "Forecasting tourist arrivals with machine learning and internet search index," *Tourism Management*, vol. 70, 2019.

[20] W. Zhu, W. Huang, Z. Lin, Y. Yang, S. Huang, and J. Zhou, "Data and feature mixed ensemble based extreme learning machine for medical object detection and segmentation," *Multimedia Tools and Applications*, vol. 75, no. 5, pp. 2815–2837, 2016.

[21] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, 1998.

[22] C. Cortes, X. Gonzalvo, V. Kuznetsov et al., "Adanet: Adaptive structural learning of artificial neural networks," 2016, https://arxiv.org/abs/1607.01097.

[23] Q. Wei and D. Liu, "Neural-network-based adaptive optimal tracking control scheme for discrete-time nonlinear systems with approximation errors," *Neurocomputing*, vol. 149, pp. 106–115, 2015.

[24] S. Ding, H. Zhao, Y. Zhang, X. Xu, and R. Nie, "Extreme learning machine: algorithm, theory and applications," *Artificial Intelligence Review*, vol. 44, no. 1, pp. 103–115, 2015.

[25] J. Tang, C. Deng, and G. B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Transactions on Neural Networks & Learning Systems*, vol. 27, no. 4, pp. 809–821, 2016.

[26] G. G. Wang, M. Lu, Y. Q. Dong, and X. J. Zhao, "Self-adaptive extreme learning machine," *Neural Computing and Applications*, vol. 27, pp. 291–303, 2016.

[27] K. Du and M. N. Swamy, "Particle swarm optimization," in *Search and Optimization by Metaheuristics*, pp. 153–173, Birkhauser, Cham, Switzerland, 2016.

[28] K. Mohammadi, S. Shamshirband, P. L. Yee, D. Petković, M. Zamani, and S. Ch, "Predicting the wind power density based upon extreme learning machine," *Energy*, vol. 86, pp. 232–239, 2015.

[29] P. Ghamisi and J. A. Benediktsson, "Feature selection based on hybridization of genetic algorithm and particle swarm optimization," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 2, pp. 309–313, 2015.

[30] D. Yang and F. Han, "An improved ensemble of extreme learning machine based on attractive and repulsive particle swarm optimization," in *Proceedings of the International Conference on Intelligent Computing*, pp. 213–220, Springer, Cham, Switzerland, 2014.

[31] Ş. Gülcü and H. Kodaz, "A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 45, pp. 33–45, 2015.

[32] L. Bin, L. Yibin, and L. Meng, "A parameter adaptive particle swarm optimization algorithm for extreme learning machine," in *Proceedings of the 27th Chinese Control and Decision Conference (2015 CCDC)*, pp. 2448–2453, Qingdao, China, 2015.

[33] S. Huang and S. Li, "Studying and implementing for water transportation of 2D simulation water robot," *DEStech Transactions on Engineering and Technology Research*, 2017.

[34] Y. Ze, S. Xuhe, and L. Li, "Study on strategy based on region partition for robotic fish's shooting," in *Proceedings of the 2015 Sixth International Conference on Intelligent Systems Design and Engineering Applications (ISDEA)*, pp. 756–759, Guiyang, Guizhou, China, August 2015.

[35] H. Bao, S. Li, and Q. Guo, "Design and realization of synchronised swimming of URWPGSim 2 D," *Journal of Beijing Information Science & Technology University*, pp. 84–88, 2011.

[36] Z. Cao, H. Guo, J. Zhang, and U. Fastenrath, "Multiagent-based route guidance for increasing the chance of arrival on time," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pp. 3814–3820, Guiyang, China, February 2016.

[37] Z. Cao, S. Jiang, J. Zhang, and H. Guo, "A Unified Framework for Vehicle Rerouting and Traffic Light Control to Reduce Traffic Congestion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1958–1973, 2017.

[38] Z. Cao, H. Guo, and J. Zhang, "A multiagent-based approach for vehicle routing by considering both arriving on time and total travel time," *ACM Transactions on Intelligent Systems and Technology*, vol. 18, no. 3, article 25, 2018.