applied sciences

MDPI

Article

Online Learning Based Underwater Robotic Thruster Fault Detection

Gaofei Xu [1], Wei Guo [1,*], Yang Zhao [2], Yue Zhou [3], Yinlong Zhang [2], Xinyu Liu [1], Gaopeng Xu [2] and Guangwei Li [1]

[1] Institute of Deep-Sea Science and Engineering, Chinese Academy of Sciences, Sanya 572000, China; xugf@idsse.ac.cn (G.X.); liuxy@idsse.ac.cn (X.L.); ligw@idsse.ac.cn (G.L.)
[2] State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China; zhaoyang@sia.cn (Y.Z.); zhangyinlong@sia.cn (Y.Z.); xugaopeng@sia.cn (G.X.)
[3] College of Engineering Science and Technology, Shanghai Ocean University, Shanghai 201306, China; y-zhou@shou.edu.cn
* Correspondence: guow@idsse.ac.cn

Abstract: This paper presents a novel online learning-based fault detection designed for underwater robotic thruster health monitoring. In the fault detection algorithm, we build a mathematical model between the control variable and the propeller speed by fitting collected online work status data to the model. To improve the accuracy of online modeling, a multi-center PSO algorithm with memory ability is utilized to optimize the modeling parameters. Additionally, a model online update mechanism is designed to accommodate the model to the change of thruster work status and sea environment. During the operation, propeller speed of the underwater robot is predicted through the online learning-based model, and the model residuals are used for thruster health monitoring. To avoid false alarm, an adaptive fault detection strategy is established based on model online update mechanism. The proposed method has been extensively evaluated using different underwater robotics, through a sea trial data simulation, a pool test fault detection experiment and a sea trial fault detection experiment. Compared with fixed model-based method, speed prediction MAE of the online learning model is at least 37.9% lower than that of the fixed model. The online learning-based method show no misdiagnosis in experiments, while the fixed model-based method is misdiagnosed. Experimental results show that the proposed method is competitive in terms of accuracy, adaptability, and robustness.

Keywords: underwater robotic; thruster system; time delay estimation; particle swarm optimization; online learning; adaptive fault detection

Citation: Xu, G.; Guo, W.; Zhao, Y.; Zhou, Y.; Zhang, Y.; Liu, X.; Xu, G.; Li, G. Online Learning Based Underwater Robotic Thruster Fault Detection. *Appl. Sci.* **2021**, *11*, 3586. https://doi.org/10.3390/app11083586

Academic Editor: Silvio Cocuzza

Received: 8 March 2021
Accepted: 14 April 2021
Published: 16 April 2021

1. Introduction

Underwater robotics has come to play an increasingly important role in sea exploration. A large number of studies on robotics safety have been performed, due to the complexity of underwater operating conditions [1]. The thruster system is one of the most crucial components of an underwater robotic vehicle. When an unexpected failure occurs, the thruster breaks down and the underwater vehicle becomes incapable of completing its job, causing huge losses for sea exploration. Thus, it is necessary to perform online monitoring of the thruster system and to implement diagnostic operations in the presence of systematic failures [2–4].

At present, a propeller thruster driven by a motor is still predominantly used in underwater vehicles [5], compared with jet- or bionic-based propulsion thrusters. Thus, motor-driven thrusters have become the main object of underwater vehicle fault detection. The sensors deployed in underwater robotics to monitor system states are limited, due to space and power limitations. Commonly the sensors include Doppler velocimeters (DVLs), inertial measurement units (IMUs), digital compasses, barometers, satellite and underwater acoustic positioning devices, current sensors, voltage sensors, pressure sensors, and rotation

speed sensors [6–8]. Through the use of such sensing modalities, the underwater robot is able to collect important measurements, such as body velocity and heading attitude. The parameters related to robotics faults can be estimated, in an online manner, by analyzing the thruster control variables. Unfortunately, the underwater robot moving states generally deviate from the expected values in the presence of thruster faults. Some researchers have found that thruster faults can be detected by comparing the residuals between the expected values and the real measurements (i.e., a fault is determined to occur if the residuals exceed specific thresholds). These are the basic principles of designing underwater robotics fault detection systems. Representative methods can be categorized into three types: analytical model-based [9,10], filtering-based [7,11,12], and data-driven approach-based [13–15]. These methods have been applied to several underwater fault detection scenarios, thus promoting the development of underwater robotics.

Limited by the underwater conditions and applications, underwater robotics fault detection still suffers from the following issues:

(1) Most fault detection methods are based on robotic moving state abnormality; however, other factors, such as sea current, also affect the robotic state. Thus, it is challenging to distinguish faults from unexpected sea current disturbances, which makes these methods prone to resulting in unsatisfactory estimates.

(2) The majority of fault detection methods require specific robotics models, which need a complete set of data for model fitting. For instance, in the analytical modeling approach an exact system model is needed for the process, involving data such as observer design, Kalman filter gain update, and particle filter state transition probability distribution. However, it is impractical to obtain a precise mathematical model of the underwater circumstance. In data-driven approach-based fault detection methods, a large amount of data is required for model training, in order to satisfy underwater application requirements. Unfortunately, it is challenging to obtain data in relation to robotics faults, especially in the initial stage of prototype design.

(3) The model parameters for state-of-the-art fault detection methods remain constant and lack online adjustment. However, the model parameters can change due to complicated underwater conditions and long-term operations [6,16–18], which may result in unpredicted faults [19].

In order to address these issues, researchers have proposed a variety of solutions from several perspectives. For example, Zhang et al. [20] developed an independent component analysis method, together with wavelet packet decomposition and empirical mode decomposition, to analyze the effects of sea currents and vehicle thruster faults upon autonomous underwater vehicle (AUV) movement speed, which further determines whether the sources of faults are from the thruster itself or the sea current.

A hidden Dirichlet distribution analysis model has been designed to detect the abnormality of the AUV vertical surface [19]. The AUV movement information and operation commands are coded into a string. The latent Dirichlet allocation (LDA) model is trained using the AUV data. Several topics are extracted from the historical data independently, in which there exists a topic with dominant probability at each moment. Afterwards, the operation performed by the AUV corresponds to the instantaneous topic (e.g., floating on the water, constant depth navigation, pitch navigation, and floating up). In practical applications, the state and operation instruction information of the AUV are first encoded into a state byte. Then, the topic most related to the current state in the LDA topic model generated by training is searched for, as the topic of the current moment, according to the nearest neighbor principle. This method no longer requires a systematic mathematical model, but the learning model is unsatisfactory when the training data are of a relatively small volume. For states that have never appeared in the training data, the LDA model recognizes them as the unknown states or as a combination of several known states that have appeared in the training data.

It has been found that the ocean current shows more obvious impacts on the motion state of underwater vehicles in practical applications, while the impact on the state of the

propeller itself is relatively smaller. As far as the fault detection of propulsion systems is concerned, methods based on propeller models might overcome the current influence, to a certain extent. In terms of modeling methods, propeller models include analytical and data-driven models. In analytical models, specific parameter information relating to the propulsion motor, transmission device, propeller, and so on, is required, which is generally impractical to obtain [6]. In addition, it is difficult to adjust the information online once the parameters of the analytical model have been determined. Therefore, it would be an ideal choice to build a propeller model based on data-driven methods.

In [6], a propeller fault detection method based on an energy consumption model is proposed, using a data-driven strategy. First, the control variable (composed of a special excitation signal) is applied to the propeller in order to obtain the corresponding propeller current data. Then, a relationship model between the control voltage and the current of the thruster is built by using the LWPR. In some practical applications, the short-term energy consumption is calculated using both the current calculated by the model and the measured current. Then, the residual index of energy consumption is calculated, and a fault is detected when this exceeds the corresponding threshold. This method is able to adjust the model online by introducing regularization in the model; however, it is unable to optimize the amount of data used in online modeling. In practice, the change of an environmental or system state is periodic, and the optimal amount of modeling data often changes in different situations. Inappropriate modeling can introduce degraded system information, which is related to other stages in the model and, thus, may affect the accuracy of the model.

Similar to underwater robotics, the fault detection of unmanned aerial vehicle (UAV) is susceptible to the influences of gusts and turbulence and model complexity. Considerable researches have been conducted for UAV fault detection, which also can be divided into model-based methods and data-driven methods.

In model-based approaches, relations between measurements and estimated states are exploited to detect possible faults. The most common ways to implement a model-based approach is to estimate the states, to estimate the model parameters, or parity-space. In general, the model-based methods mainly integrate with Kalman filter, particle filter or their variants by establishing a physical model as the state transition equation [21]. A fault detection method based on the suboptimal fading unscented Kalman filter (SFUKF) is designed in [22]. Liu et al. [23] introduced an adaptive estimation method based on a bank of unscented Kalman filters (UKF) to monitor the actuator health in a UAV. In the literature [24], Guo et al. utilized Extended Kalman Filter (EKF) and the UAV kinematics model to realize analytical redundancy. Abbaspour et al. [25] developed a detection strategy, in which the weighting parameters are updated by EKF, to find faults in sensors and actuators. In [26], Yi and Zhang presented a model-based fault detection method based on particle filter.

In data-driven approaches, historical flight data are utilized for training the models. Once the new flight data are inconsistent with the pattern learned by the trained model, a fault alarm is given. Amidst data driven approaches for the fault detection of UAV, machine learning methods such as artificial neural networks [27] and support vector machines (SVM) appear more recently in the literature [28–30]. Baskaya et al. [30] proposed a classification method based on principal component analysis (PCA) and SVM to detect faults of UAV control surfaces. Guo et al. [31] proposed an optimized one-class SVM method regulated by local density to realize fault detection for UAV. A fault detection method based on a stacked long short-term memory (LSTM) neural network is designed in literature [32], where the LSTM model is used to realize accurate prediction of the monitored parameter, and fault detection is achieved via a statistical threshold of the smoothed prediction residuals. In [33], a data-driven multivariate regression approach based on long short-term memory with residual filtering (LSTM-RF) is proposed to fulfill UAV fault detection and recovery.

In order to perform fault detection in the presence of ocean current interference and real-time state changes, we propose an online learning-based fault detection method for

underwater robotic thruster systems. First, a control-variable-speed model of the thruster is built online, from which the propeller speed is predicted. Then, when the residual of the model exceeds a threshold value due to sea current interference, the operation state will change and the model will be changed by an online update mechanism. Eventually, the fault detection model will be triggered.

In order to improve the accuracy of online modeling, the time delay between control variables and speed is estimated using correlation analysis, such that the modeling data are realigned in accordance with the time delay estimation results. Secondly, a multi-center particle swarm optimization algorithm with short-term memory merit is designed, which combines the prior knowledge and swarm intelligence to analyze the model in the modeling process. Our fault detection method is based on the thruster control-variable-speed model. Therefore, it has the ability to adapt to interference factors, such as the influence of the ocean current and the change of robot operation states. Neither an accurate mathematical model of the underwater vehicle nor complete training data are required in our method.

The rest of the paper is organized as follows. In Section 3, the online estimation of time delay in relation to the thruster system is described. In Section 4, the online modeling between control variables and rotation speed is introduced, and multi-center particle swarm optimization is utilized to optimize the model parameters. In Section 5, the online model update mechanism is introduced. In Section 6, the experimental platform is described, and the sea trials and pool tests are analyzed. Section 7 concludes the work.

2. General Framework of Online Learning-Based Fault Detection

The general framework of the proposed underwater robotics thruster fault detection method is illustrated in Figure 1.

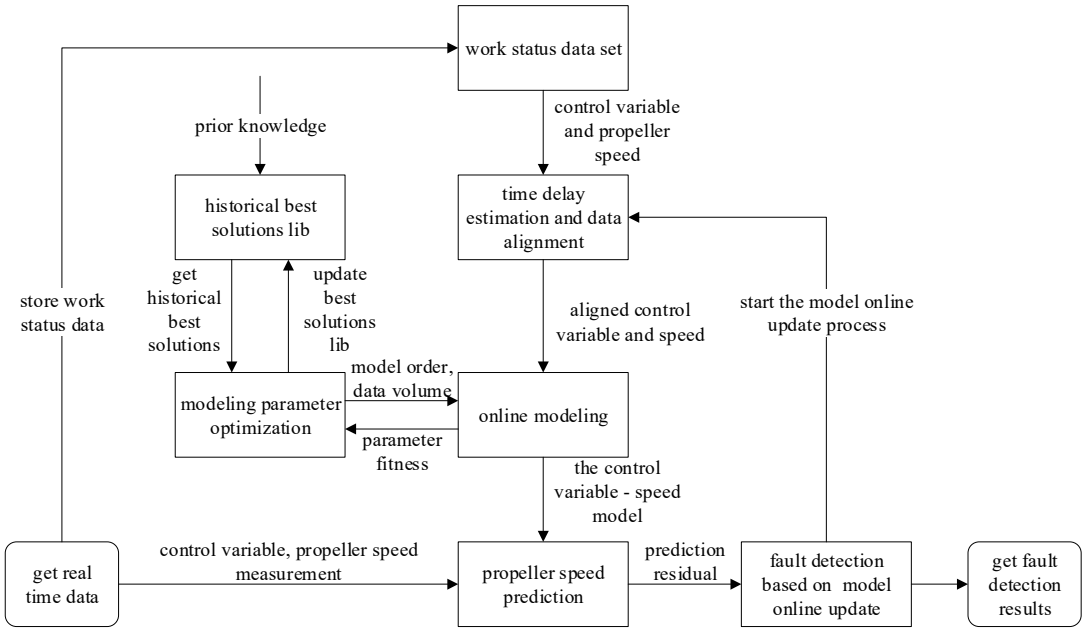


Figure 1. Structure of the online learning-based fault detection algorithm.

In the online learning-based fault detection algorithm, propeller speed of the underwater robot thruster is predicted through an online learning model. Additionally, an adaptive fault detection strategy is established based on model online update mechanism. In order to achieve data alignment, the time delay between the control variable and the measured propeller speed is estimated by analyzing the trend of the correlation changes. The polynomial fitting is introduced to build a control variable–speed model based on the aligned data. To improve the accuracy of online modeling, a multi-center particle swarm optimization (PSO) algorithm with memory ability is utilized to optimize the model

order and modeling data volume. In the proposed PSO algorithm, we designed an initial optimization strategy based on prior knowledge, historical optimal solution and improved tent mapping. Eventually, a multi-center collaborative search strategy is introduced to reduce the odds of local minima.

3. Propulsion System Time Delay Online Estimation

Time delay is a widespread phenomenon in electromechanical systems. For systems with complex structures such as underwater robots, multiple sampling cycles from the controller output control to the propeller generating the corresponding actions are generally required. When modeling and fault detection are based on data, the existence of a time delay can have a serious impact. Therefore, it is necessary to estimate the time delay between the control variable and the rotational speed measurement value, and to realign the data accordingly.

Current commonly used delay estimation methods include the online measurement, step response, relay feedback, and optimization-based parameter identification methods [34–37], among which the online measurement method requires special hardware to measure the time delay. The step response and relay feedback methods need to apply additional excitation signals to the system, while the optimization-based parameter identification method needs to determine the model structure of the system in advance. In practice, the system delay is not always fixed [38], and may change with the operating environment or the state of the system itself. Therefore, it is necessary to estimate the system delay online. The above method was applied.

There are certain limitations to the online estimation of the time delay of an underwater vehicle propulsion system. For any two variables, the correlation refers to the follow-up relationship between variables. The more consistent the follow-up changes of the variables, the stronger the correlation. If there is no time delay between two correlated variables, their changes are completely synchronized and the correlation is strongest at this time. If there is a time delay, the follow-up change relationship will be destroyed and the correlation will be weakened. Generally speaking, the greater the delay, the more obvious the decrease in correlation. Therefore, by maximizing the correlation coefficient between two related variables, an approximate estimation of the time delay can be achieved [39].

This paper analyzes the change trend of the correlation between the control variable and the measured value of the thruster speed, in order to realize the online time delay estimation. This method is completely based on data and does not require a systematic model. In order to reduce the amount of calculation and facilitate online application, the Pearson correlation coefficient is used to measure the correlation.

Let the control variable in a certain period of time be $U = (u_1, u_2, \cdots, u_N)$ and the measured value of the propeller speed in the same period of time be $V = (v_1, v_2, \cdots, v_N)$. Then, the Pearson correlation coefficient between the control variable and the speed in this period of time is:

$$r(U,V) = \frac{\sum_{i=1}^{N}(u_i - \overline{u})(v_i - \overline{v})}{\sqrt{\sum_{i=1}^{N}(u_i - \overline{u})^2 \sum_{i=1}^{N}(v_i - \overline{v})^2}} \tag{1}$$

where $\overline{u}, \overline{v}$ are the average values of the control variable and speed, respectively, in the time period.

Suppose that, in this time period, the time delay between the control variable and the rotational speed measurement value has τ sampling periods, $\tau = 0, 1, \cdots, \tau_{\max}$. Then, we realign the control variable and the measured value of the rotational speed, according to the assumed time delay. From the aligned data, take $U_\tau = (u_1, u_2, \cdots, u_{N-\tau_{\max}})$ and $V_\tau = (v_{\tau+1}, v_{\tau+2}, \cdots, v_{\tau+N-\tau_{\max}})$ to calculate the corresponding correlation coefficient:

$$r_\tau = r(U_\tau, V_\tau), \tau = 0, 1, \cdots, \tau_{\max} \tag{2}$$

According to the above steps, the correlation coefficients of the realigned data under different hypothetical time delays are sequentially calculated. When the correlation coefficient has a maximum value, the corresponding time delay is the estimated result, $\hat{\tau}$, of the time delay between the control variable and the rotational speed measurement value, which is:

$$\hat{\tau} = \underset{\tau=0,1,\cdots,\tau_{\max}}{\operatorname{argmax}} \; r_\tau \tag{3}$$

4. Propulsion System Online Modeling

4.1. Propeller Control Variable and Speed Model

The rotation speed is an important parameter that characterizes the operating status of a propeller. For a properly designed propeller, interference from environmental factors (e.g., ocean currents) will not have a serious impact on the speed of the propeller. Therefore, the detection of thruster faults based on abnormal changes in rotational speed can effectively eliminate the interference of environmental factors, such as ocean currents. We first establish a model of the relationship between the propeller's "control variable-rotation speed" and then predict the residual, based on the model's speed, to achieve fault detection.

Limited by many factors, most current propellers operate in open-loop speed regulation mode. Even those propellers working in closed-loop speed regulation mode have a certain dynamic in the speed regulation process. Therefore, in practical applications, the relationship between the control amount of the propeller and the rotational speed is obviously non-linear and fluctuates with the change of the working state.

When detecting a fault of the thruster, the residual of the speed prediction will be normal (and, thus, cause false detection) only when the "control variable–speed" model can accurately describe the non-linear change of the speed. As the actual control variable–speed model is non-linear and changes with the working state, it is difficult to accurately describe the control variable–speed relationship through a fixed model established in advance. Therefore, in this article, we establish a control variable–speed model through online learning, and update the speed prediction model used in the online fault detection process as needed.

Polynomial fitting [40] is a non-linear relationship learning method based on data, which involves simple operations and is easy to realize online. In this paper, polynomial fitting is used to conduct online learning of the non-linear relationship between the control variable and speed.

For the realigned control variable sequence $U_{\hat{\tau}}$ and speed sequence $V_{\hat{\tau}}$, the control variable–speed model form obtained by polynomial fitting is:

$$v_N = a_0 + a_1 u_{N-\hat{\tau}} + \cdots + a_q u_{N-\hat{\tau}}^q \tag{4}$$

where v_N is the current measured value of the propeller speed, $u_{N-\hat{\tau}}$ is the corresponding control variable after the data is aligned, and q is the order of the "control variable–speed" model, which is a modeling parameter that needs to be determined before polynomial fitting. The parameter $A = \left[a_0, a_1, \cdots, a_q\right]^T$ in the control variable–speed model can be obtained according to the following formula:

$$A = \left(X^T X\right)^{-1} X^T Y \tag{5}$$

in which

$$X = \begin{bmatrix} 1 & 1 & 1 & 1 \\ u_{N-w-\hat{\tau}+1} & u_{N-w-\hat{\tau}+2} & \cdots & u_{N-\hat{\tau}} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N-w-\hat{\tau}+1}^q & u_{N-w-\hat{\tau}+2}^q & \cdots & u_{N-\hat{\tau}}^q \end{bmatrix} \tag{6}$$

$$Y = \left[v_{N-w+1}, v_{N-w+2}, \cdots, v_N\right]^T \tag{7}$$

where w is the length of the control variable and speed sequence used when calculating the control variable–speed model parameters; that is, the amount of data used for modeling.

When modeling the control variable–speed relationship by polynomial fitting, the model order q and the modeling data volume w are modeling parameters that need to be determined in advance. If the order of the model is too low, it will have difficulty in reflecting the dynamic change information of the system; meanwhile, if the order is too high, overfitting can easily occur. During the operation of the underwater robot, the environment and its own state change over time, which will cause changes in the system model. These changes are generally gradual, and the model can be approximated as unchanged in a short time period. Therefore, during the entire operation process, the "control variable–rotation speed" model has a phased nature, where different time periods of data correspond to different models. Therefore, in order to improve the modeling accuracy, in addition to selecting the appropriate model order, it is also necessary to select the appropriate amount of modeling data.

In order to ensure the accuracy of the online modeling of the control variable–speed model, we propose the use of an improved PSO algorithm, which is suitable for online applications, for online optimization to achieve the appropriate selection of model order and modeling data volume.

4.2. Multi-Center Particle Swarm Optimization Algorithm with Memory Ability

For optimization problems that cannot obtain a global model, the swarm intelligence algorithm has obvious advantages over other methods. The PSO algorithm is one of the most recognized achievements in the field of swarm intelligence. It has a simple structure, is easy to implement, and does not rely on the characteristic information of the problem [41]. It has been widely studied and applied. In this paper, the model order and modeling data volume optimization problem cannot be described with a global model and the optimization process needs to be performed online; therefore, the PSO algorithm is used to solve the problem.

The standard PSO algorithm has the disadvantage of easily falling into local optima. To solve this problem, researchers have proposed a variety of improved algorithms. The strategies adopted include optimizing the initial population, changing parameter values, changing the topology, introducing mutation operations, second search optimization, joint optimization with other swarm intelligence algorithms, and so on. Among these, optimizing the initial value can increase the possibility that the initial value falls into the global optimal interval. Changing the parameter value can speed up the convergence speed and optimize the search behavior. Both of these strategies can improve the performance of the algorithm without significantly increasing its complexity. Changing the topological structure can limit the particle's perception of the optimal value of the current group. Although it can avoid the particles falling into local optima, to a certain extent, it may also seriously affect the search efficiency. Introducing a mutation operation is equivalent to replacing some of the particles trapped in local optima with new particles, and then searching again. At this time, a large number of searches are generally required and the results of the new search are uncertain. Second search optimization assumes that the particles have a known evaluation function—that is, the global model—which violates the original intention of the algorithm. The joint optimization algorithm requires multiple searches, and the algorithm complexity is significantly higher than the original algorithm.

The online optimization problem in this article requires the optimization algorithm to be as simple and easy to implement as possible, as well as to obtain better results in a shorter computing time. Based on the existing research results, we propose a multi-center PSO algorithm with memory capabilities. Its main strategies include initial population optimization based on historical optimal solutions and improved chaotic mapping, and the use of a multi-center collaborative search.

4.2.1. Initial Population Optimization Based on Historical Optimal Solution and Improved Chaotic Map

In the PSO algorithm, the initial population is usually generated by randomly selecting values in the search space. For the online optimization problem in this article, it is necessary to optimize the model order and the amount of modeling data many times during the operations of the underwater robot. After each optimization, an optimal solution is obtained. In the next optimization, the previously obtained optimal solution may be close to the optimal solution of this optimization problem. If these optimal solutions are stored as part of the initial population in the next optimization, it can help to optimize the initial population, thus improving the performance of the algorithm. In addition, by introducing the prior knowledge of the control variable–speed model into the historical optimal solution library, the optimization of the joint drive of knowledge and data can be realized.

The PSO algorithm attempts to distribute the initial population as evenly as possible in the search space, in order to improve the probability of finding the optimal solution. Therefore, we first take a small amount of the initial population as the historical optimal solution, and then randomly select a historical optimal solution to map to the entire search space, as the remaining initial population.

Tent mapping is a commonly used chaotic map with good ergodicity [42]. Thus, we use tent mapping to map the randomly selected historical optimal solution to the search space. The mathematical expression of tent mapping is:

$$t_{k+1} = \begin{cases} 2t_k & 0 \le t_k \le 0.5 \\ 2(1-t_k) & 0.5 \le t_k \le 1 \end{cases} \tag{8}$$

Tent mapping can be problematic as the value can fall into a fixed point or a small period. In response to this problem, Nie et al. [43] proposed a method to multiply the mapping value by a number in [0,1] when the mapping value falls into a fixed point or a small period. An improved method of random numbers between.

When optimizing the initial population through tent mapping, the initial value is first mapped from the search space to the [0,1] space. In practical applications, it has been found that the initial value of the mapping into [0,1] may be an approximation of an infinite loop decimal. In this case, although the tent mapping does not fall into a fixed point or a small period, there will be a large number of values within a very close distance, thus affecting the mapping effect. We address this problem by limiting the minimum distance between two adjacent value points. If $|t_{k+1} - t_k| \le 0.02$, then we take:

$$t_{k+1} = \begin{cases} t_{k+1} + 0.4rand & t_{k+1} \le 0.5 \\ t_{k+1} - 0.4rand & t_{k+1} > 0.5 \end{cases} \tag{9}$$

Figure 2 shows the distribution of value points obtained by 60 iterations in two-dimensional space, using the method in this paper and the method of [43], when the initial value is (0.833333, 0.666667). It can be seen that the too-close value points became more evenly distributed in space. A large number of value points obtained by the method of [43] can be seen to be gathered in a small area.

4.2.2. Multi-Center Collaborative Search

During the search process of the PSO algorithm, all particles learn from the particle with the best current fitness, which has a larger weight. If the particles with the best fitness currently fall into a local optimum, this may cause the entire population to gather in the local optimum area. At this time, even if strategies such as mutation and second search are used in order to try to jump out of the local optimal area, many search iterations are wasted. In addition, judging whether the population has fallen into a local optimum is difficult.

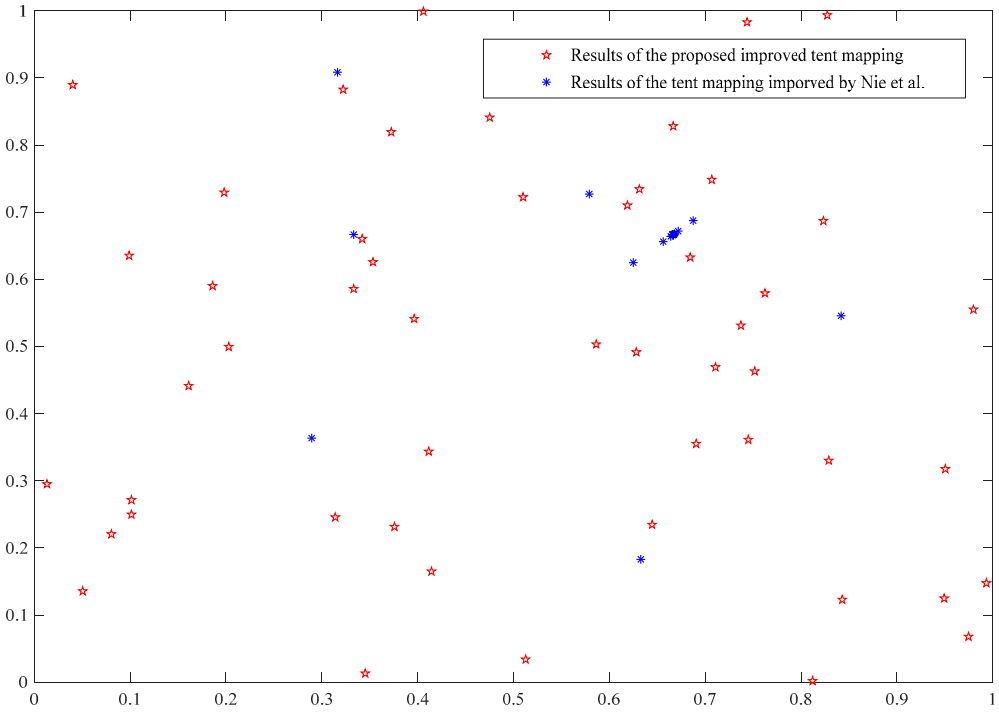


Figure 2. Comparison of data point distribution.

To improve this problem, we propose a multi-center collaborative search strategy. The basic idea is to allow the independent exploration of particles, which are far away from the current optimal particle with better fitness during the search process, guiding some particles to search collaboratively in the outer area of the current optimal particle. In other words, there are multiple search centers in the entire population. Among them, there is a main center with the best fitness, which guides most particles to search around it. In addition, there are multiple sub-centers with good fitness, thus guiding some particles to search in other areas.

The multi-center collaborative search strategy mainly affects the speed update process of the PSO algorithm. After determining the optimal particle fitness, before the particle position update, the algorithm mainly includes three processes: calculating the weighted fitness, selecting the learning center, and speed update. It is agreed, in this article, that the closer the particle is to the optimal position, the smaller its fitness. Note that during the kth iteration of the PSO algorithm, the current position of particle i, the individual's optimal position, and the individual's optimal fitness are $\boldsymbol{x}_i^k$, $\boldsymbol{p}_{ibest}^k$ and f_{ibest}^k, respectively. Assuming that the particle with the optimal fitness of the optimal individual in the population at this time is particle c, then c is called the central particle in the iteration process. At this time, the optimal position and optimal fitness of the entire population are $\boldsymbol{p}_{gbest}^k = \boldsymbol{p}_{cbest}^k$ and $f_{gbest}^k = f_{cbest}^k$, respectively. Based on the above agreement, the specific implementation process of the multi-center collaborative search strategy is as follows.

Step 1: Sort each particle i in the population in ascending order, according to the optimal fitness of the individual, and mark the set of particles with the top 50% of optimal fitness f_{ibest}^k of the individual as X_{top}^k.

Step 2: For particle i in X_{top}^k, calculate the Euclidean distance between its individual optimal position $\boldsymbol{p}_{ibest}^k$ and the population optimal position $\boldsymbol{p}_{gbest}^k$:

$$d_{ipg}^k = d(\boldsymbol{p}_{ibest}^k, \boldsymbol{p}_{gbest}^k) = \sqrt{\sum_{j=1}^{D} (p_{ibest,j}^k - p_{gbest,j}^k)^2} \tag{10}$$

Step 3: For particle i in X_{top}^k, calculate its weighted fitness as follows:

$$f_{iw}^k = \frac{d_{ipg}^k}{f_{ibest}^k} \tag{11}$$

The larger the weighted fitness, the more suitable the particle is as the auxiliary center.

Step 4: Calculate the weighted fitness of all particles in X_{top}^k, and sort the particles in descending order of weighted fitness. According to the number of auxiliary centers set in advance, the sorted particles are sequentially selected as auxiliary center particles, where the individual optimal position of the auxiliary center particles is used as the auxiliary center. In this paper, the number of auxiliary centers is set to 2. Assuming that the auxiliary center particles are particle m and particle n, the corresponding auxiliary centers are $\boldsymbol{p}_{mbest}^k$ and $\boldsymbol{p}_{nbest}^k$, respectively.

Step 5: For the central particle c and auxiliary central particles m and n, set their learning centers to their individual optimal positions. For other particles in the population (e.g., particle i), calculate the distance between its current position $\boldsymbol{x}_i^k$ and the population center and auxiliary center:

$$\begin{cases} d_{ixg}^k = d(\boldsymbol{x}_i^k, \boldsymbol{p}_{gbest}^k) \\ d_{ixm}^k = d(\boldsymbol{x}_i^k, \boldsymbol{p}_{mbest}^k) \\ d_{ixn}^k = d(\boldsymbol{x}_i^k, \boldsymbol{p}_{nbest}^k) \end{cases} \tag{12}$$

If $d_{ixg}^k \leq \min(d_{ixm}^k, d_{ixn}^k)$—that is, the distance between particle i and the center of the population is not greater than the distance from the auxiliary center—set the learning center of particle i to $\boldsymbol{L}_i^k = \boldsymbol{p}_{gbest}^k$. Otherwise, determine which auxiliary center particle i is closest to (here, it is assumed to be closest to $\boldsymbol{p}_{mbest}^k$), and determine the learning center of particle i, according to the following rules:

If $rand \leq \frac{3}{4}\frac{f_{gbest}^k}{f_{mbest}^k}$, then $\boldsymbol{L}_i^k = \boldsymbol{p}_{mbest}^k$, otherwise $\boldsymbol{L}_i^k = \boldsymbol{p}_{gbest}^k$.

Step 6: Select the learning center for all particles in the population, according to the above method. Then, update the speed of each particle as follows:

$$v_{i,j}^{k+1} = \omega v_{i,j}^k + c_1 r_1 (p_{ibest,j}^k - x_{i,j}^k) + c_1 r_1 (L_{i,j}^k - x_{i,j}^k) \tag{13}$$

After the speed update is completed, the position update and subsequent operations can be performed, according to the standard PSO algorithm. The complete process of the multi-center PSO algorithm with memory capability proposed in this paper is shown in Figure 3.

Figure 4 shows the position of each particle and the distribution of learning centers in the mid-, late-, and end-stage populations during a certain optimization process. There was a total of 20 particles in the population, with 1 center and 2 auxiliary centers. The left side of the arrow in the figure shows the number of the particle, while the right side is the learning center of the particle during this iteration. Figure 4a shows the mid-search stage. At this time, particle 7 is the global center of the population, and particles 13 and 14 are auxiliary centers. There are 13, 3, and 4 particles in the population learning from particles 7, 13, and 14, respectively. Under the effect of the collaborative search strategy, the particles search extensively in the better area, in order to avoid falling into local optima too quickly. In the late search stage, shown in Figure 4b, the population has roughly converged to a better area and, under the guidance of the population center and the auxiliary center, they search within a smaller range. At the end of the search, shown in Figure 4c, the population is close to convergence and the particles are gathered in the population and auxiliary centers for fine searching. Thanks to the rules for determining the learning center in the collaborative search strategy presented in this article, there may still be particles located in the auxiliary center that leave the auxiliary center to learn from the population center, searching the

area between the auxiliary center and the population center, and further reducing the final population center. It is possible for these to become stuck in local optimal positions.

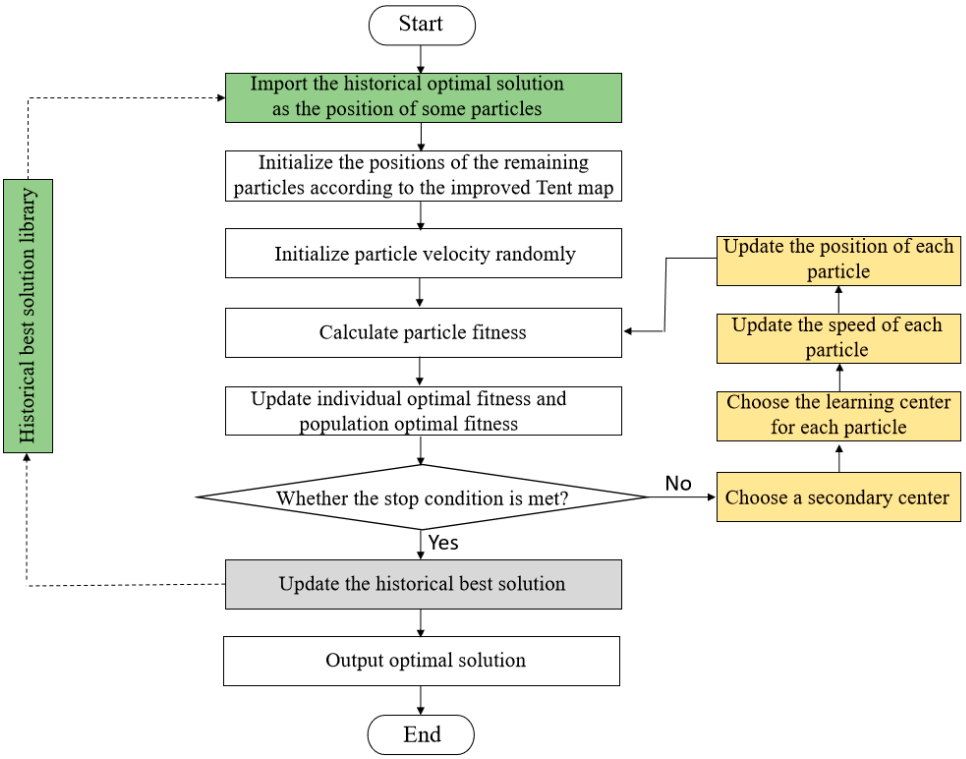
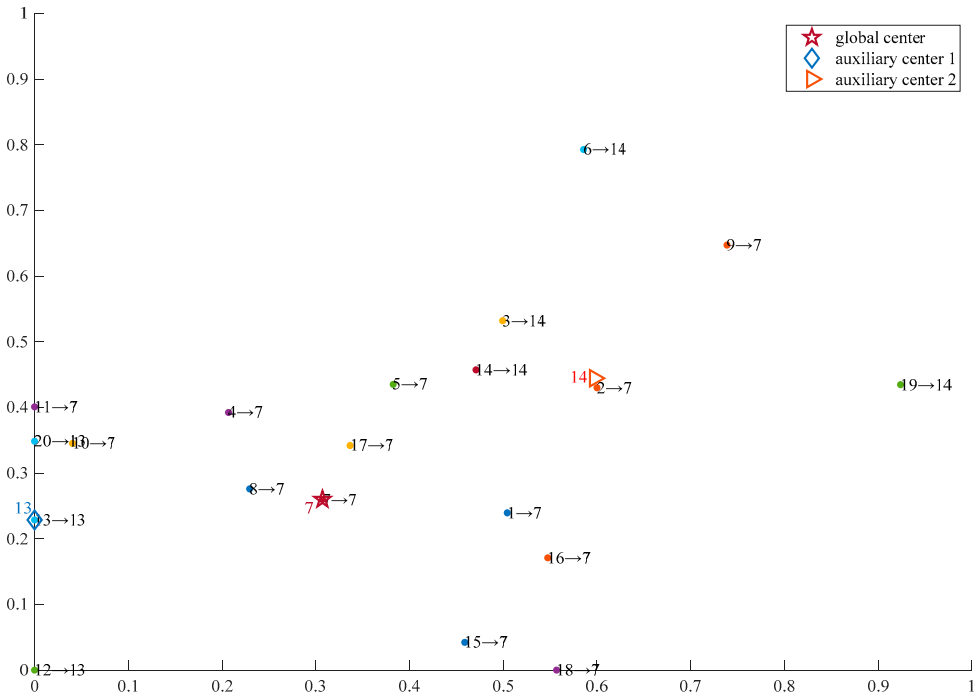
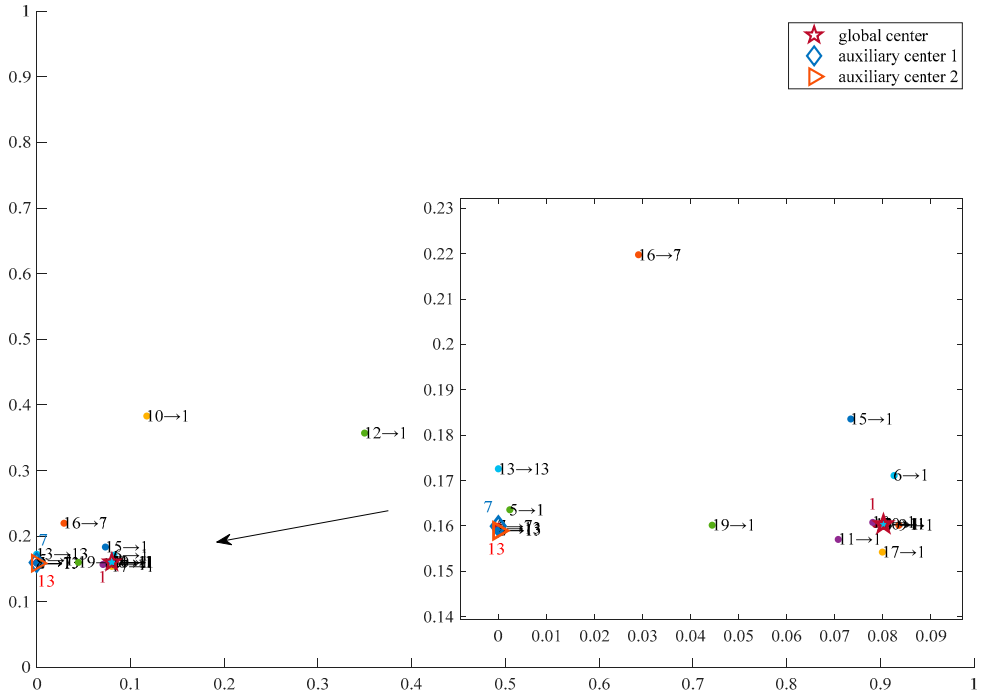


Figure 3. The schematic flowchart of the modified PSO algorithm.

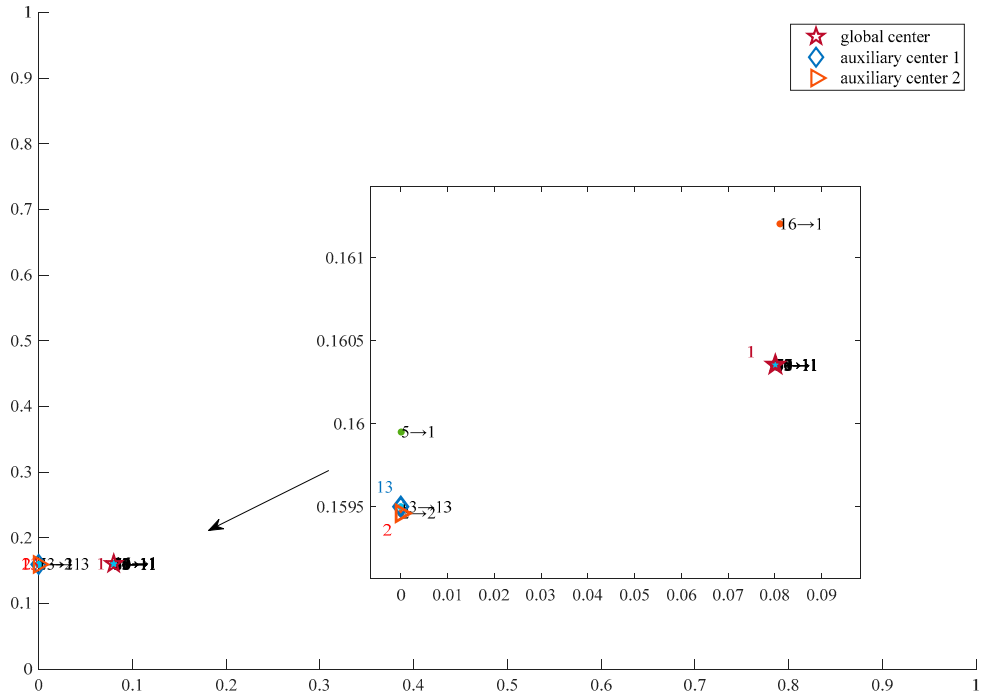


(a) Middle search stage

Figure 4. *Cont.*



(**b**) Late search stage



(**c**) Search end stage

Figure 4. Particle position and learning center distribution.

4.3. Propulsion System Online Modeling Process

The implementation process of online modeling of the underwater robot propulsion system proposed in this paper is as follows.

Step 1: During the operation of the underwater robot, obtain online propeller control and speed data. In order to improve the accuracy of modeling, the data to be acquired include a certain amount of change in thruster control.

Step 2: Estimate the time delay of the propulsion system in the current state, according to the acquired control variable and speed data.

Step 3: According to the estimated time delay $\hat{\tau}$, realign the propeller control variable and speed data:

$$\begin{cases} U_{\hat{\tau}} = (u_1, u_2, \cdots, u_{N-\hat{\tau}}) \\ V_{\hat{\tau}} = (v_{\hat{\tau}+1}, v_{\hat{\tau}+2}, \cdots, v_N) \end{cases} \tag{14}$$

Step 4: Use the multi-center PSO algorithm with memory capability to optimize the thruster model order q and data volume w used for modeling, where $1 \le q \le q_{\max}$, $w_{\min} \le w \le w_{\max}$. When evaluating the fitness of particles, use the following data sets:

$$\begin{cases} U_{\hat{\tau},w} = (u_{N-\hat{\tau}-w+1}, u_{N-\hat{\tau}-w+2}, \cdots, u_{N-\hat{\tau}}) \\ V_{\hat{\tau},w} = (v_{N-w+1}, v_{N-w+2}, \cdots, v_N) \end{cases} \tag{15}$$

Establish the rotational speed model $\hat{v} = F_{q,w}(u)$ through polynomial fitting and evaluate the model using the mean absolute percentage error (MAPE):

$$f_{q,w}^{\mathrm{MAPE}}(U_{\hat{\tau},w}, V_{\hat{\tau},w}) = \frac{1}{N_{\hat{\tau},w}} \sum_{i=1}^{N_{\hat{\tau},w}} \left| \frac{v_i - \hat{v}_i}{v_i} \right| \tag{16}$$

Then, the fitness value of the particle whose current position is in (q, w) is:

$$f_{q,w} = \max(f_{q,w}^{\mathrm{MAPE}}(U_{\hat{\tau},w}, V_{\hat{\tau},w}), f_{q,w}^{\mathrm{MAPE}}(U_{test}, V_{test})) \tag{17}$$

Among them, (U_{test}, V_{test}) are the public test data taken from $(U_{\hat{\tau}}, V_{\hat{\tau}})$.

Step 5: According to the optimal modeling parameter (q^*, w^*) obtained under the current data set through polynomial fitting, a control variable—speed model $\hat{v} = F_{q^*,w^*}(u)$ is established.

Through the above-mentioned steps, a control variable–rotation speed model, which can describe the propeller in the current operating state, can be established, in an online manner, for subsequent speed prediction and fault detection.

5. Adaptive Fault Detection Based on Online Model Update

5.1. Propulsion System Model Online Update

After building the propeller "control variable–rotation speed" model online, in the next operation process, the propeller control variable is input into the model to obtain the speed prediction value. Comparing the measured value of the propeller speed at time t with the corresponding predicted speed value, the model residual of the propulsion system can be obtained as follows:

$$e_t = v_t - F_{q^*,w^*}(u_{t-\hat{\tau}}) \tag{18}$$

During the operation of the underwater robot, the operating environment and the system itself will change over time, which causes changes in the control variable–speed model. This can cause discrepancies between the previously established model and the actual model which, in turn, will cause the model prediction residuals to gradually increase.

In order to avoid false detection caused by an increase of model prediction residuals, we designed an online model update mechanism. When the model residual exceeds the threshold, the online modeling process is restarted, and the latest data are used to establish a new control variable–speed model.

5.2. Propulsion System Failure Detection

During the operation of the underwater robot, it is assumed that the control variable–speed model obtained by the k^{th} online modeling is currently being used to monitor the state of the propeller. If, at time t, the model predicts the residual:

$$e_t^k = v_t - F_{q_k^*,w_k^*}^k(u_{t-\hat{\tau}_k}) \tag{19}$$

to exceed the threshold, at this point, the online model update starts and a new model $F^{k+1}_{q^*_{k+1},w^*_{k+1}}(u_{t-\hat{\tau}_{k+1}})$ is obtained. Then, the residual of the new model at time t is calculated:

$$e_t^{k+1} = v_t - F^{k+1}_{q^*_{k+1},w^*_{k+1}}(u_{t-\hat{\tau}_{k+1}}) \tag{20}$$

In the online update process of the model, due to the optimization of the modeling parameters, the new model obtained can better fit the modeling data than the original model.

In the absence of failure, it can be considered that the actual model of the propulsion system is approximately unchanged in the short-term; in other words, it can be considered that the data $(u_{t-\hat{\tau}_{k+1}}, v_t)$ corresponding to time t and other modeling data acquired recently are from the same actual system, where both can be described by the new model $F^{k+1}_{q^*_{k+1},w^*_{k+1}}(u_{t-\hat{\tau}_{k+1}})$. Therefore, for the data at time t, the prediction residual of the new model should not exceed the threshold.

Based on the above analysis, if it is detected that the residual of the model exceeds the threshold at time t and, after updating the model online, the residual of the new model obtained at time t does not exceed the threshold—indicating that the residual of the original model exceeded the threshold due to gradual system change or noise—the system is determined not to have malfunctioned.

If the propulsion system fails at time t, the actual system model will change significantly, causing the data generated in the fault state at time t and the data obtained in the recent normal state to come from different systems. As shown in Figure 5, when the model is updated, as the algorithm sets the minimum amount of modeling data, $w_{\min}$, in the k + 1th online modeling, most of the modeling data used are the data generated in the normal state; only an extremely small part comes from the data generated in the fault state. Therefore, the new model obtained has a better fitting effect on the data generated in the normal state, and has a poor fitting effect on the data generated in the fault state. In this case, the residual of the new model at time t will still exceed the threshold. The algorithm will continuously update the model multiple times until the data generated in the fault state in the modeling data account for the majority, and the residual of the model becomes lower than the threshold again.

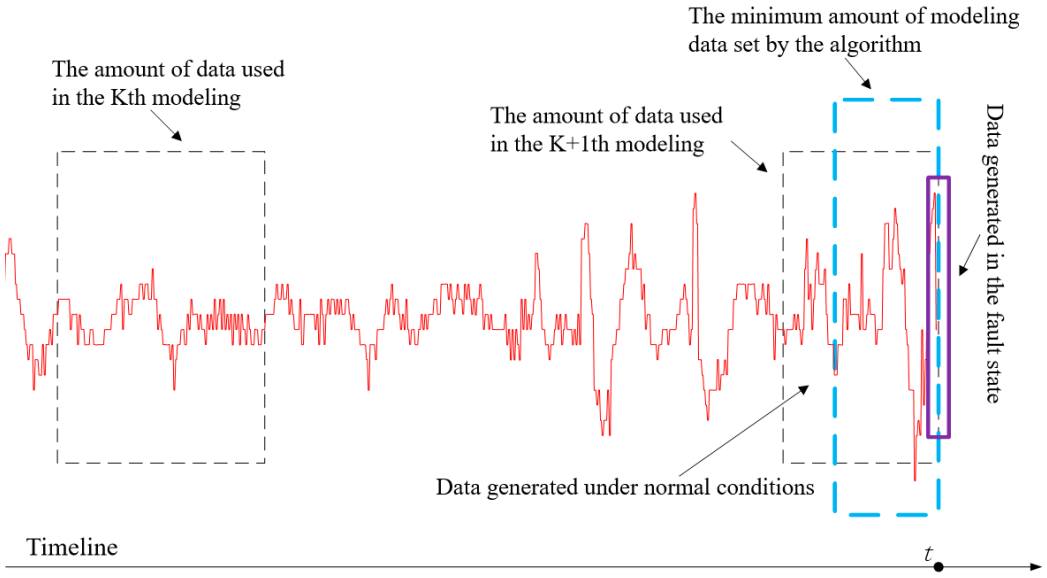


Figure 5. Modeling data in fault conditions.

Combining the above analysis, the thruster fault detection rules are as follows:
(1) If the model residual is less than the threshold, there is no fault;

(2) If the residual of the model at time t exceeds the threshold, the online model update mechanism is activated; if the residual of the new model at time t is less than the threshold, no fault occurs;

(3) If the residual of the model at time t exceeds the threshold and, after the model is updated online the residual of the new model at time t also exceeds the threshold, the model will continue to be updated online after the data at time $t+1$ is collected. If the residual still exceeds the threshold, it is judged that the thruster has failed since time t.

6. Experimental Results and Analysis

6.1. Implementation of the Fault Detection Algorithm

The pipeline of the underwater robotics thruster fault detection implementation is illustrated in Figure 6. Before fault detection, the historical best solution library was initialized based on prior knowledge of the thruster model. During operation of the underwater robot, the system monitoring information was collected, which was fed into the online learning-based fault detection algorithm.

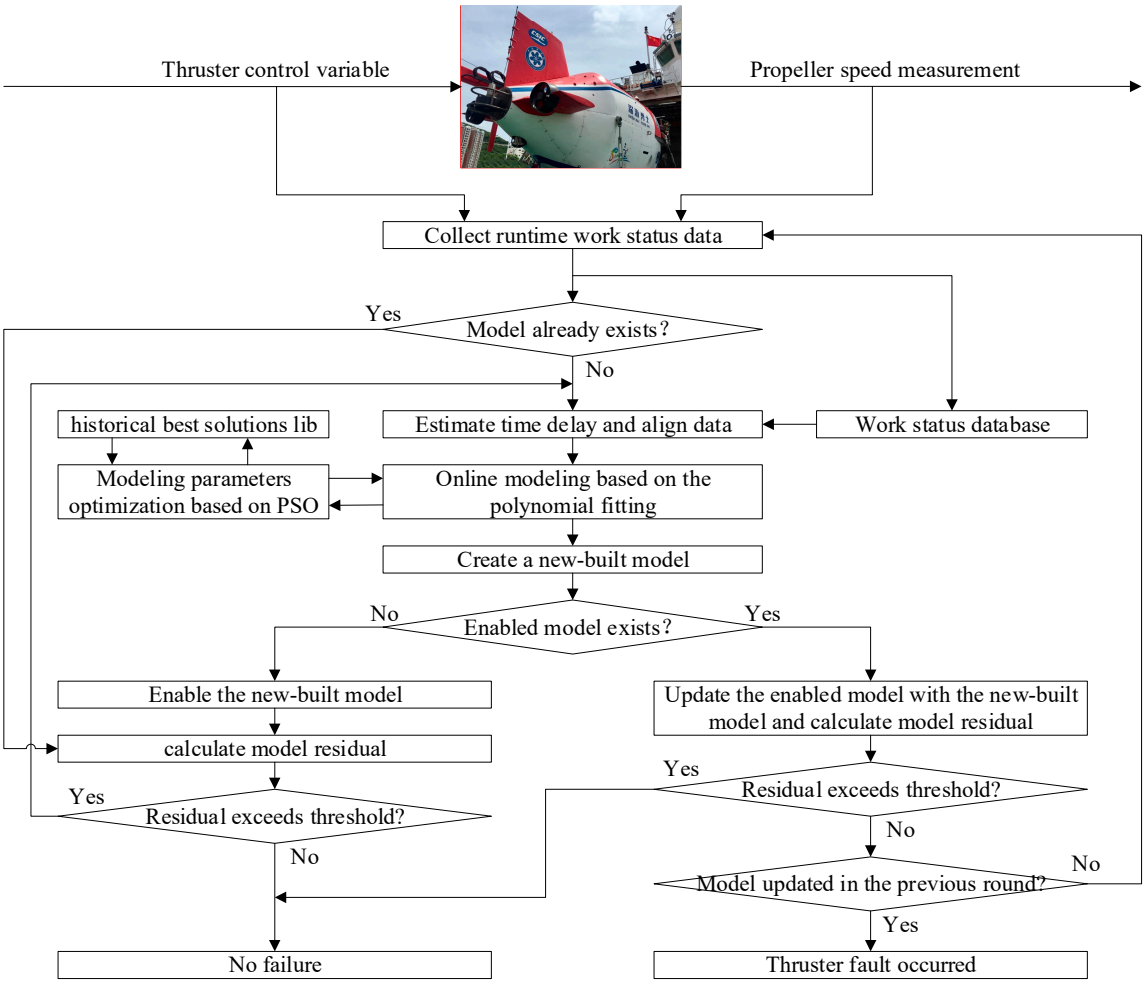


Figure 6. The schematic flowchart of the thruster fault detection algorithm.

Whenever a set of runtime work status data is collected, the fault detection algorithm first checks whether there is currently a control variable–speed model available. If there already exists a control variable–speed model, the predicted propeller speed is calculated and the model prediction residual will be evaluated by the residual threshold. If the model prediction residual is within the threshold, it means that the thruster is not malfunctioning. If the model prediction residual exceeds the threshold, or there is no control variable–speed

model available, the online modeling procedure will start. It consists of online time delay estimation, data alignment and online optimization based polynomial fitting modeling.

When a new control variable–speed model is built through online modeling, the fault detection algorithm will check whether there is an existing enabled model. If no enabled model exists before, then enable the new-built model and calculate the model residual, the followed process is the same as previously described. If there already exists an enabled model before the new model is established, the new-built model will replace the enabled model. Afterwards, propeller speed prediction residual of the new-built model will be calculated. If the residual is within the predefined range, the thruster system is determined to work smoothly. Otherwise, the fault detection algorithm will check if the model has been updated in the previous round. When the work status data of the next sampling period is collected, the model online update process will be started, and the following process is the same as previously described. If the model has been updated in the previous step, a thruster fault occurs.

Compared with existing methodologies, main feature of the online learning-based fault detection algorithm is the dynamic model based on online modeling and the adaptive fault detection strategy based on online model update. To better demonstrate the performance of the proposed method, we designed a comparison method that uses a fixed model and performs fault detection based only on fixed thresholds. In the fixed-model based method, when a set of runtime work status data is collected, a control variable–speed model will be created in the same way as the proposed online modeling method. After that, the control variable–speed model will be used for propeller speed prediction in the whole fault detection process. If the speed prediction residual of the fixed model exceeds the predefined threshold, a thruster fault was determined to have occurred. For comparison, the predefined threshold and range of the modeling data is the same as that uses in the proposed online learning model-based method.

6.2. Sea trial Data Simulation and Analysis

Firstly, the proposed online learning-based fault detection method was tested using data collected from a sea trial of a human occupied vehicle (HOV). The HOV was named 'SHEN HAI YONG SHI', as shown in Figure 7, specifications of the HOV and its thruster studied in the sea trial are described in Table 1.

Figure 7. The HOV 'SHEN HAI YONG SHI'.

Table 1. Specifications of the HOV.

No.	Specifications	Value
1	Main dimensions of the HOV	$9.3 * 3 * 4$ m
2	Weight of the HOV	20 t
3	Thruster control variable type	Voltage
4	Thruster control variable range	−5 V~+5 V
5	Maximum propeller speed	700 r/min
6	Propeller speed sensor	Integrated Hall sensor
7	Feedback signal of propeller speed sensor	Voltage pulse

Thruster control variable and propeller speed during sea trial of the HOV is shown in Figure 8, in which the model residual over a certain period of time in the process of fault detection was obtained, as shown in Figure 9. As there were several interfering factors in the sea trial, the model residual fluctuated violently. In order to reduce the number of online model updates and computational burden, the model residual threshold was set to $|e_t| \leq 80$. The range of the modeling data was set as $w_{\min} = 50$, $w_{\max} = 500$. After 500 groups of collection, the thruster fault detection started.

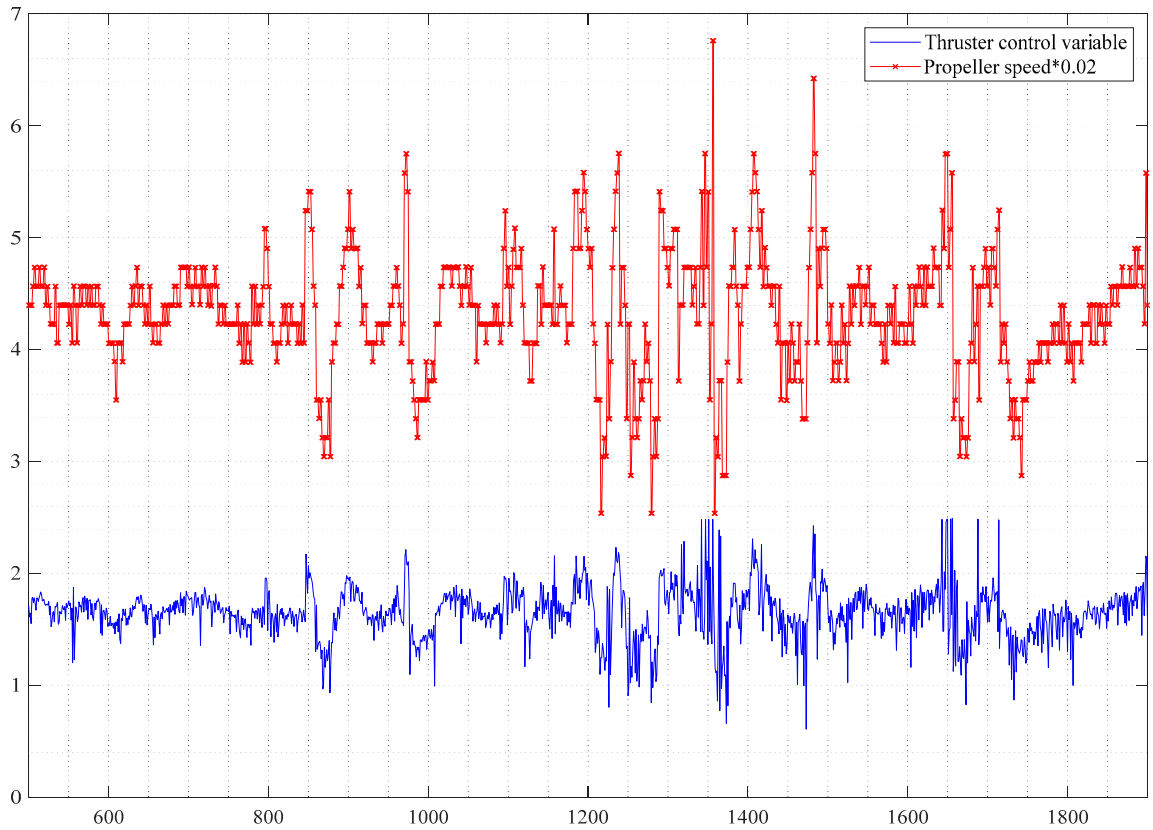


Figure 8. Thruster control variable and propeller speed during sea trial of the HOV.

As can be seen in Figure 9, the model residual exceeded the threshold by a large margin in the 847th and 868th sampling periods. However, false alarms were avoided by using the adaptive fault detection strategy based on online model update mechanism. Similarly, the detected model residual exceeded the threshold in the 1357th sample. Comparably, the residual was below the threshold after online model update in the 1358th sample. Successive residuals of the model updated in the 1358th sample keep below the threshold for a longer period of time, which could verify the superiority of the proposed method.

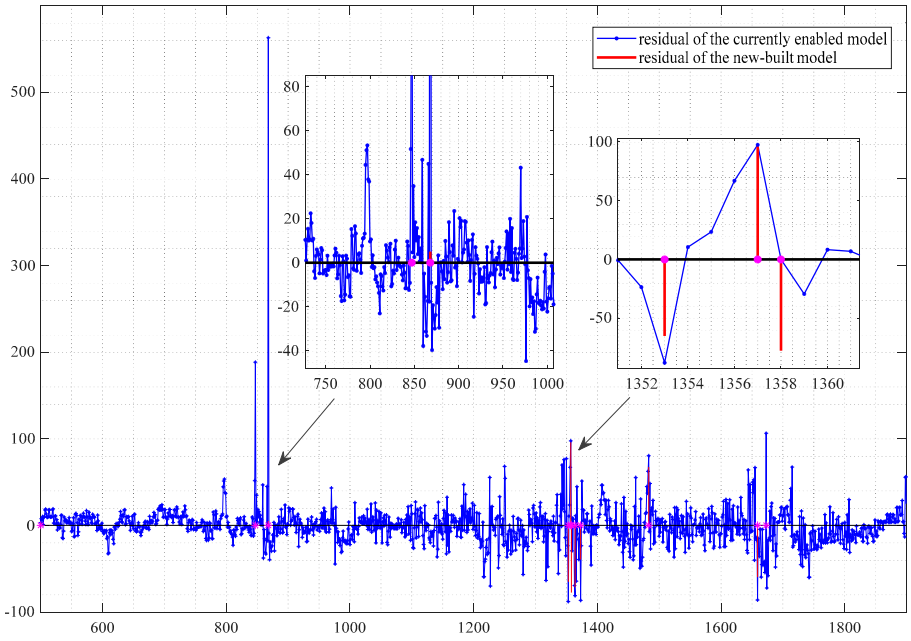


Figure 9. Model residuals of the online learning-based fault detection method during sea trial of the HOV.

It was found that, in the process of fault detection using sea trial data of the HOV, although the model residual fluctuated violently due to various factors, there was no false detection alarm when using the proposed online learning-based method.

Model residuals of the fixed model-based fault detection method during sea trial of the HOV is shown in Figure 10. As mentioned before, in the fixed-model based method, when a set of runtime work status data is collected, a control variable–speed model will be created in the same way as the proposed online modeling method. After that, the model will be used for propeller speed prediction in the whole fault detection process. If the speed prediction residual of the fixed model exceeds the predefined threshold, a thruster fault was determined to have occurred. For comparison, the predefined threshold and range of the modeling data is the same as that uses in the proposed online learning model-based method.

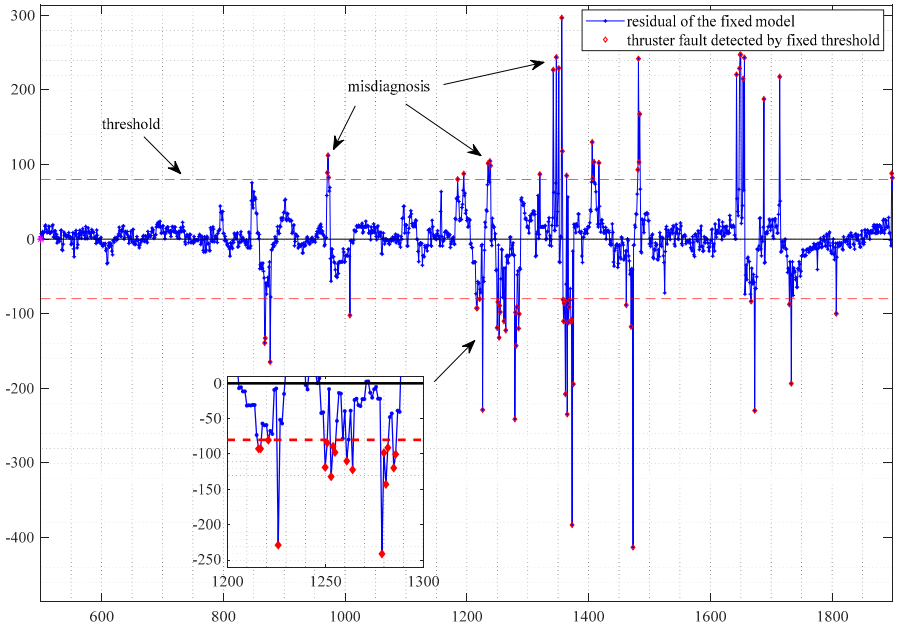


Figure 10. Model residuals of the fixed model-based fault detection method during sea trial of the HOV.

As can be seen in Figure 10, residuals of the fixed model are similar to the online learning-based model after the fixed model has just been established, this is mainly because that the two models are built with the same method and the same data. After the 850th sampling period, due to the change of thruster work status and environmental factors, residuals of the fixed model become large and exceeds the threshold multiple times, resulting in a series of misdiagnosis.

6.3. Pool Test Fault Detection Results and Analysis

After the Sea trial data simulation, the proposed online learning-based thruster fault detection algorithm was verified in a pool test, considering the fact that a manual thruster fault is risky and sea trial faults are challenging to capture. A saucer-shape AUV is used for the pool test, as shown in Figure 11, specifications of the saucer-shape AUV and its thruster studied in the pool test is described in Table 2.

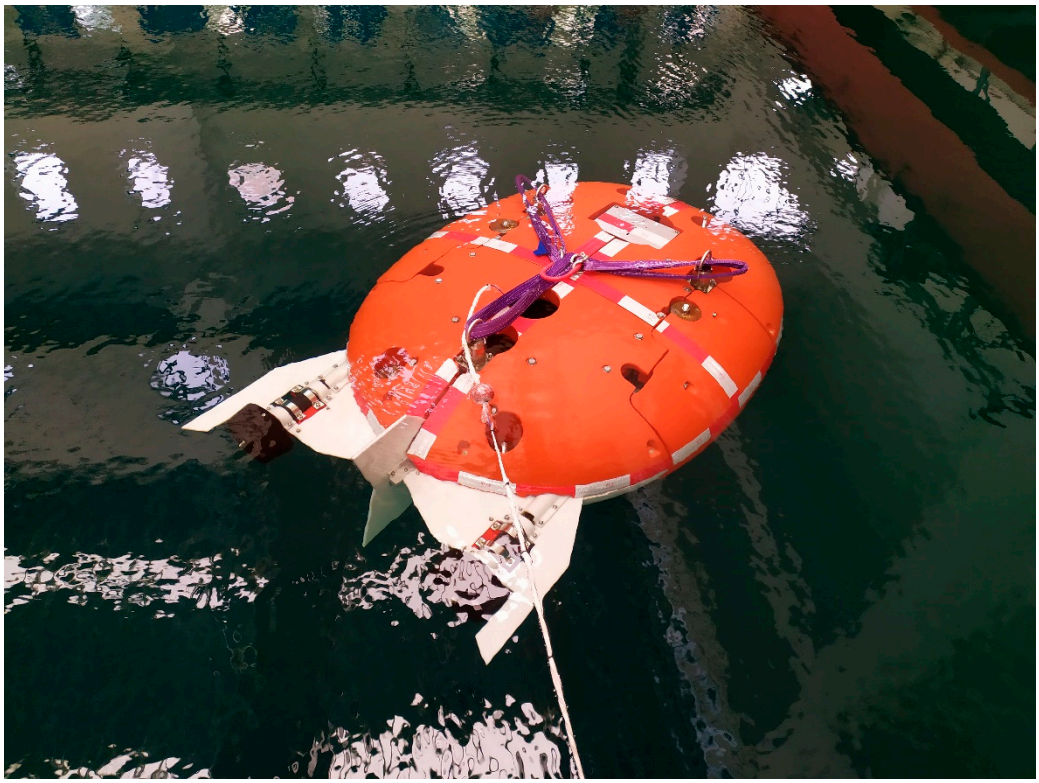


Figure 11. The saucer-shape AUV in pool test.

Table 2. Specifications of the saucer-shape AUV.

No.	Specifications	Value
1	Main dimensions of the AUV	$1.6 * 1.2 * 0.6$ m
2	Weight of the AUV	438 kg
3	Thruster control variable type	Voltage
4	Thruster control variable range	−5 V~+5 V
5	Maximum propeller speed	1200 r/min
6	Propeller speed sensor	Integrated Hall sensor
7	Feedback signal of propeller speed sensor	Voltage pulse

In the pool test, the motion state change was simulated by randomly dragging the saucer-shape AUV. The influence of environmental factors such as ocean currents was simulated by water injection and the formation of a turbulent flow by reflecting the pro-

peller wake on the pool wall. Furthermore, we threw cloth strips into the pool, in order to simulate the thruster intaking unexpected marine debris such as fishing nets or seaweed.

Thruster control variable and propeller speed during pool test of the saucer-shape AUV is shown in Figure 12, the model residual data obtained by the proposed fault detection method is shown in Figure 13. In the tests, the model residual threshold was set to $|e_t| \leq 8$ and the modeling range was set as $w_{\min} = 50$, $w_{\max} = 500$. Initially, the thruster system worked in a fault-free state. Near the 1280th sampling period, the thruster inhaled the first cloth strips and the thruster is malfunctioning. Near the 1380th sampling period, the thruster inhaled a second cloth strips, and another failure state occurred. Thruster of the saucer-shape AUV entangled with cloth strips is shown in Figure 14.

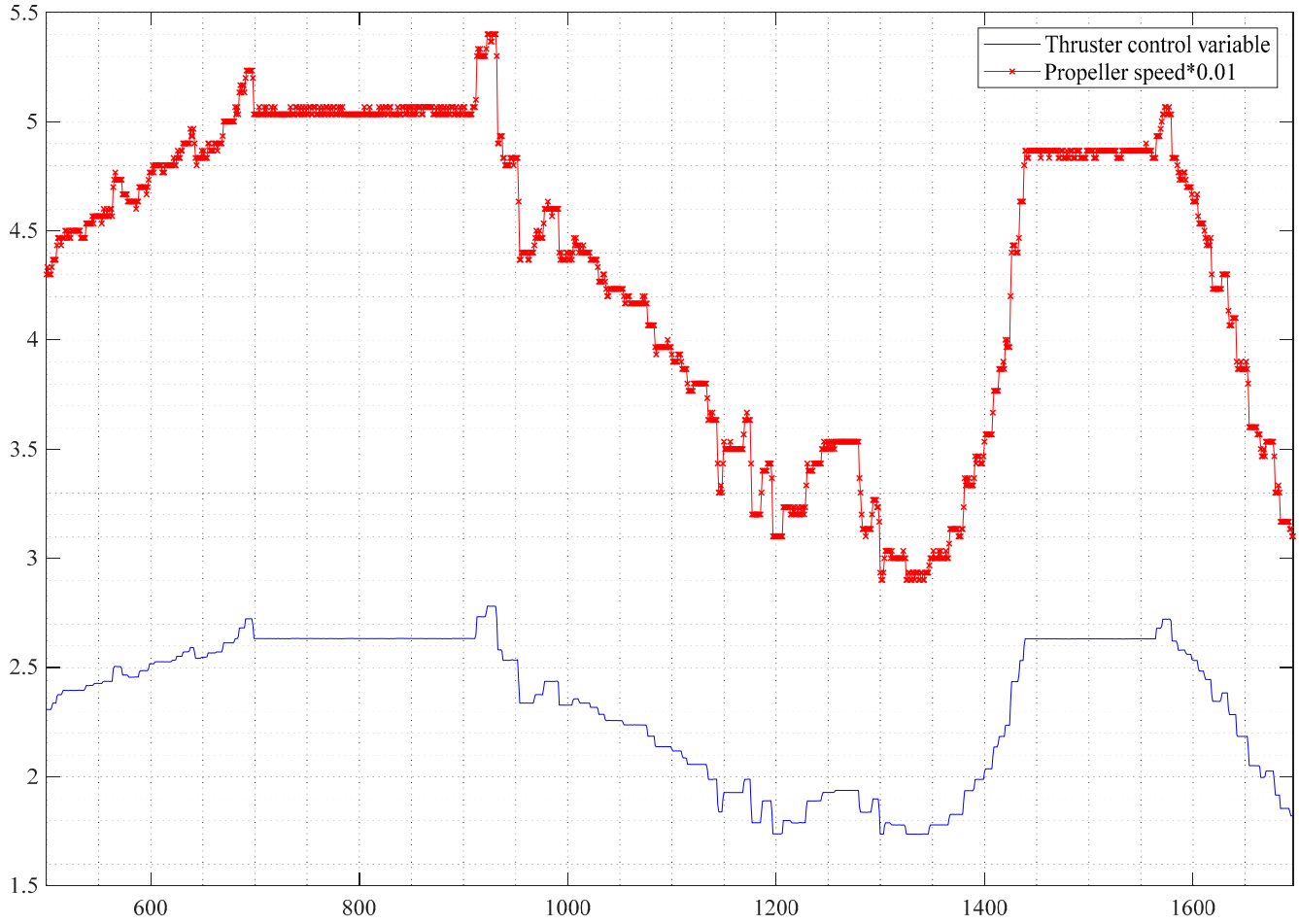


Figure 12. Thruster control variable and propeller speed during pool test of the saucer-shape AUV.

As can be seen in Figure 13, the residual exceeded the threshold in the 698th and 1096th sampling periods. After model updating online, the residual of the new-built model had fallen within the threshold, and the system triggered no false alarm for fault detection. There were obvious trends in the variation of model residuals in the 500–700 sampling periods and 700–900 sampling periods. Combined with the change of control variable and propeller speed in the test process, the model residual change was mainly caused by a change of thruster operation status. In the previous analysis, we mentioned that the gradual changes in the working environment and the system status will lead to a change in the system model. The phenomenon was observed in the experiment, consistent with the analysis, which demonstrates that it is reasonable to use a data-driven approach to build and update the system model online.

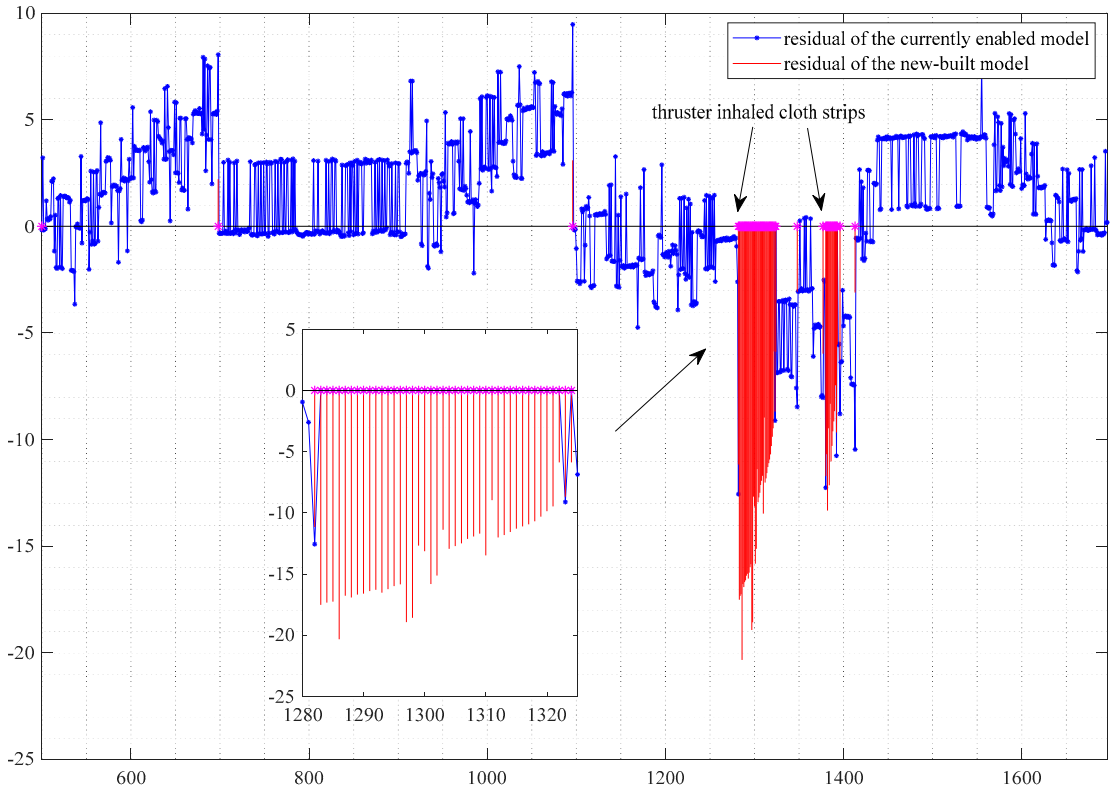


Figure 13. Model residuals of the online learning-based fault detection method during pool test of the saucer-shape AUV.

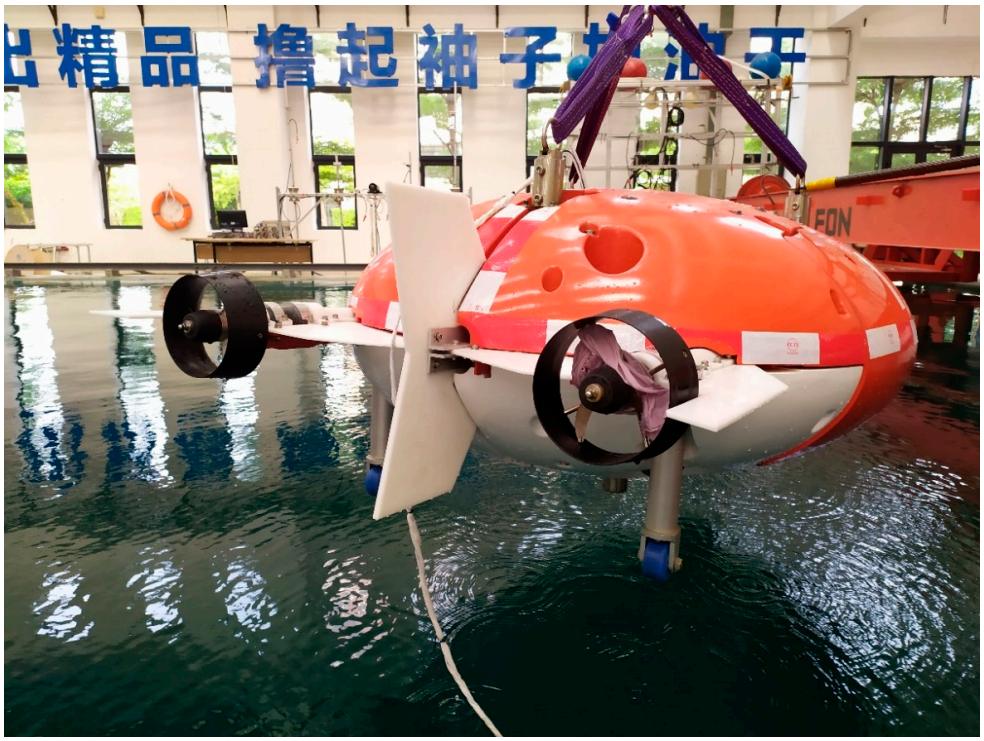


Figure 14. Thruster of the saucer-shape AUV entangled with cloth strips in the pool test.

As mentioned before, near the 1280th sampling period, the thruster inhaled the first cloth strips and became entangled. It can be seen from Figure 8 that the model residual exceeded the threshold in the 1282nd sampling period. At that time, the fault detection algorithm started the online model update mechanism, however, the residual of the new-

built model still exceeded the threshold. Therefore, the algorithm updated the model online for the second time after collecting a new set of work status data in the 1283rd sampling period. It can be seen from Figure 13 that after the second model update, the model residual still exceeds the threshold. Therefore, the algorithm detected that a thruster fault occurred in the 1282nd sampling period.

After the thruster fault was detected, the algorithm continuously updated the model online, as the residual of the model continued to exceed the threshold. When the model was updated after the failure, more and more data were generated in the fault state in the modelling data, and the new-built model came closer to the real fault state model. Then, the model residual gradually reduced. By observing the new-built model residuals, corresponding to the 1283–1324 sampling periods in Figure 13, it was found that the model residual trend was consistent with the above analysis. The above-mentioned online modelling process ended at the 1324th sampling period, and so lasted about 43 sampling periods, which was close to the minimum modelling data set size $w_{\min} = 50$ in the proposed algorithm. At this point, the algorithm had built a fault state model through online learning.

In the 1380th sample, the thruster inhaled the second cloth strips and the fault state of the thruster changed, which meant the occurrence of a new fault. The proposed method also successfully detected the new fault by the adaptive fault detection strategy based on online model update, which partly proves the online learning ability of the online learning-based modeling algorithm in the presence of work status changes.

Model residuals of the fixed model-based fault detection method during pool test of the saucer-shape AUV is shown in Figure 15. As mentioned before, parameters in the fixed model-based fault detection are the same as that uses in the proposed online learning model-based method.

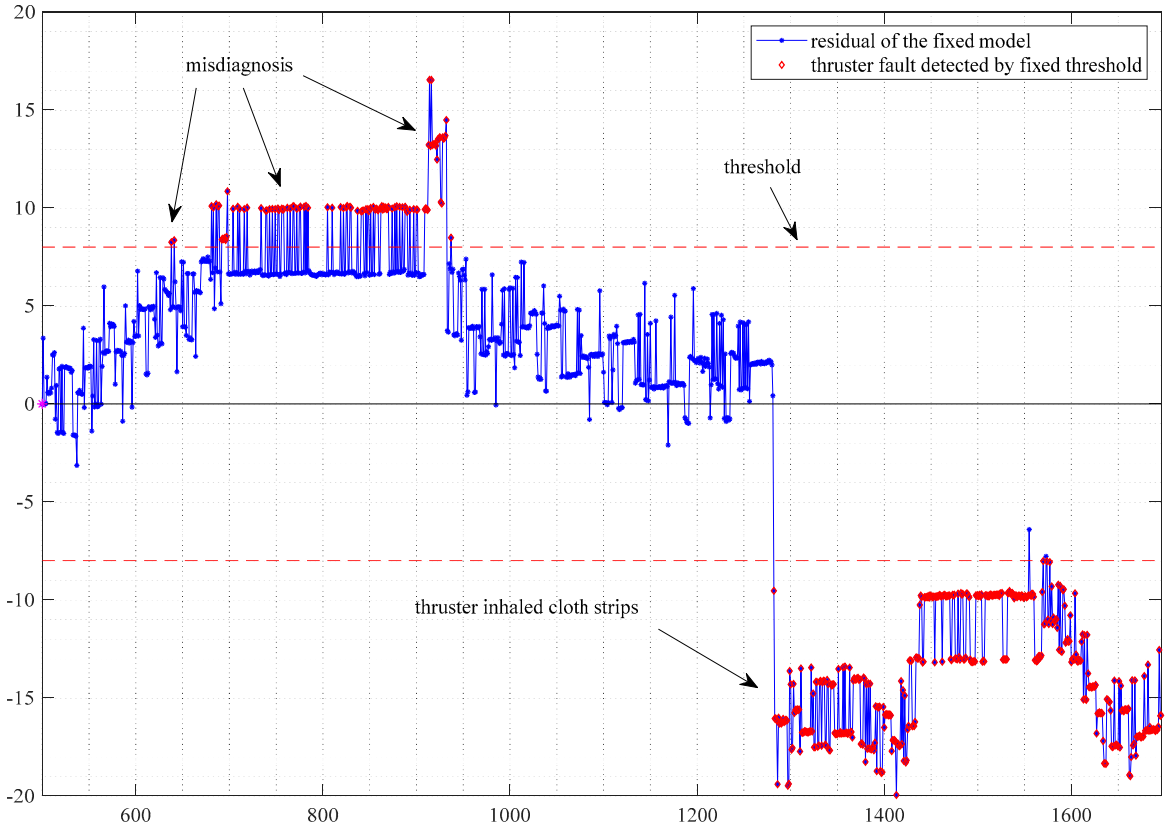


Figure 15. Model residuals of the fixed model-based fault detection method during pool test of the saucer-shape AUV.

As can be seen in Figure 15, after the 700th sampling period, due to the change of thruster work status, residuals of the fixed model become large and exceeds the threshold multiple times, resulting in a series of misdiagnosis. In contrast, as shown in Figure 13, when residual exceeds the threshold at the 700th sampling period, the model online update process starts and thus avoid misdiagnosis.

6.4. Sea Trial Fault Detection Results and Analysis

To better demonstrate performance of the proposed method, a sea trial fault detection experiment was conducted. A portable AUV is used for sea trial, as shown in Figure 16, specifications of the portable AUV and its thruster studied in the sea trial is described in Table 3.

Figure 16. The portable AUV in sea trial.

Table 3. Specifications of the portable AUV.

No.	Specifications	Value
1	Main dimensions of the AUV	$\Phi 0.2 * 1.6$ m
2	Weight of the AUV	46 kg
3	Thruster control variable type	PWM signal
4	Thruster control variable range	0~100
5	Maximum propeller speed	500 r/min
6	Propeller speed sensor	Integrated Hall sensor
7	Feedback signal of propeller speed sensor	CAN bus data

Thruster control variable and propeller speed during sea trial of the portable AUV is shown in Figure 17. In the sea trial, the model residual threshold was also set to $|e_t| \leq 8$ and the modeling range was set as $w_{\min} = 50$, $w_{\max} = 500$.

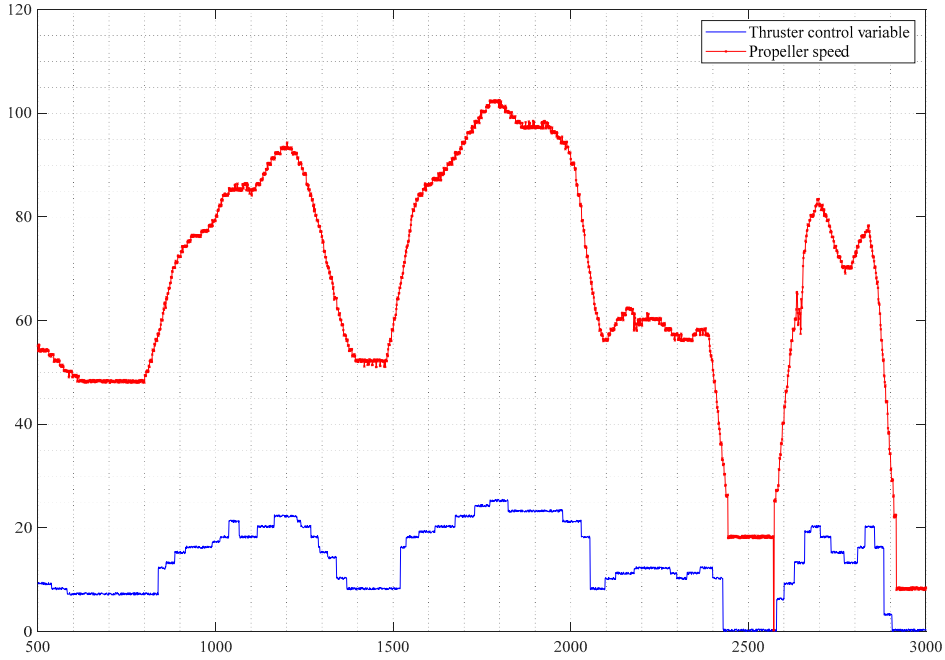


Figure 17. Thruster control variable and propeller speed of the portable AUV.

During the sea trial, the thruster system worked in a fault-free state initially, the portable AUV sailing in shallow sea water. Near the 2400th sampling period, the AUV encounters weeds, some weeds are caught in the propeller, causing minor failures of the thruster. Then, control variable of the portable AUV reduced to zero to protect the propulsion system. When the thruster restarts, more weeds are entangled in the propeller. The thruster was blocked and caused a serious malfunction, as shown in Figure 18.

Figure 18. Thruster of the portable AUV blocked by weeds during the sea trial.

Model residuals obtained by the proposed fault detection method during sea trial of the portable AUV is shown in Figure 19. As described above, thruster of the portable AUV worked in a fault-free state before the 2400th sampling period, and the online learning-based fault detection algorithm did not make any false alarms. When the portable AUV encounters weeds near the 2400th sampling period and causes a minor failure of the thruster, the online learning-based fault detection algorithm detected the occurrence of fault in the 2427th sampling period. When the thruster was blocked near the 2870th sampling period, the proposed algorithm detected a severe fault in the thruster.

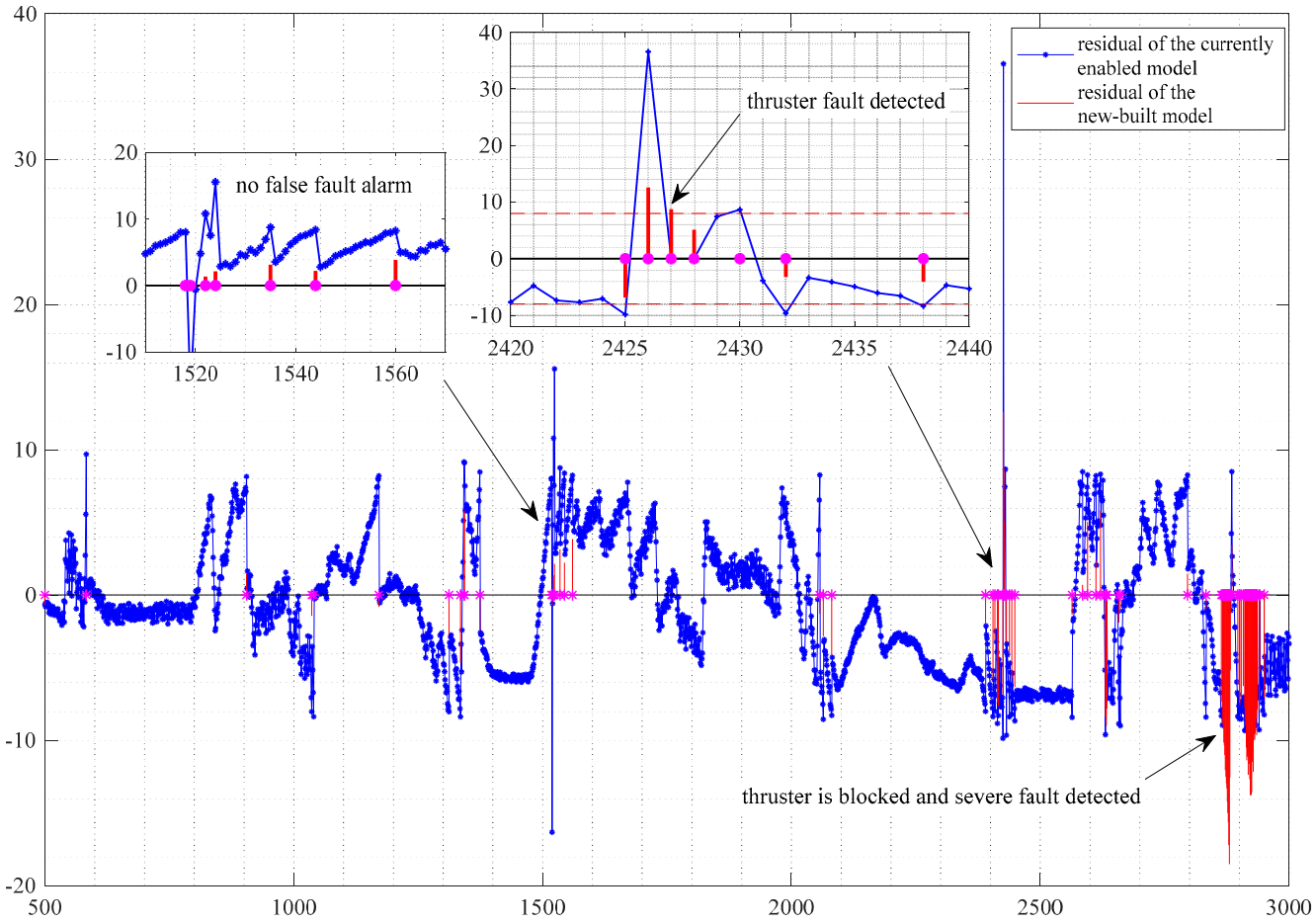


Figure 19. Model residuals of the online learning-based fault detection method during sea trial of the portable AUV.

Model residuals of the fixed model-based fault detection method during sea trial of the portable AUV is shown in Figure 20. As mentioned before, parameters in the fixed model-based fault detection are the same as that uses in the proposed online learning model-based method. As can be seen in Figure 20, residuals of the fixed model are similar to the online learning-based model after the fixed model has just been established. After the 580th sampling period, due to the change of thruster work status and the influence of environment, residuals of the fixed model exceed the threshold, resulting in a series of misdiagnosis.

6.5. Quantitative Analysis of Experimental Results

In order to quantitatively analyze performance of the proposed method, Mean Absolute Error (MAE) of the propeller speed prediction model was calculated. As shown in Table 4, MAE of the proposed online learning-based model is 10.54, 1.70 and 2.91 in the sea trial data simulation, pool test fault detection experiment and sea trial fault detection experiment, respectively. By contrast, MAE of the fixed model is 16.98, 5.53 and 258.23

respectively. MAE of the proposed online learning-based model is at least 37.9% lower than the fixed model.

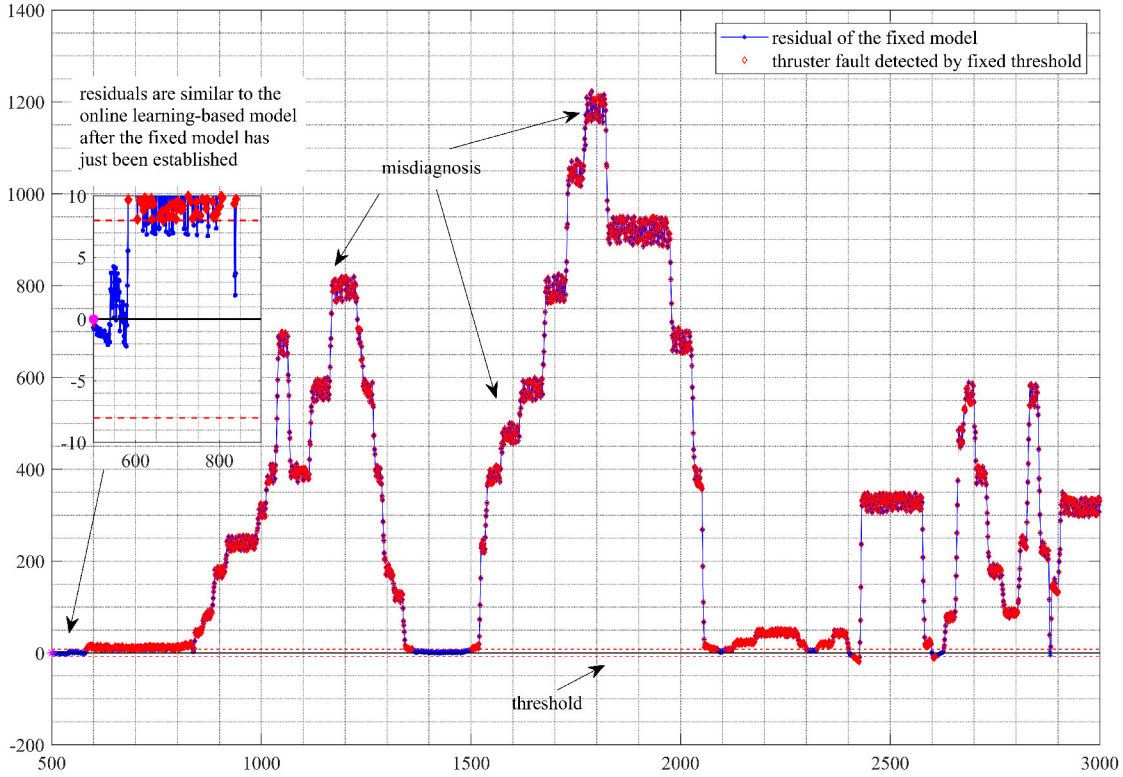


Figure 20. Model residuals of the fixed model-based fault detection method during sea trial of the portable AUV.

Table 4. MAE of the propeller speed prediction model.

	MAE of the Online Learning-Based Model	MAE of the New-Built Online Learning-Based Model	MAE of the Fixed Model
Sea trial data simulation	10.54	0.25	16.98
Pool test fault detection	1.70	0.44	5.53
Sea trial fault detection	2.91	0.24	258.23

It is worth noting that MAE of the new-built online learning-based model is 0.25, 0.44 and 0.24 in the sea trial data simulation, pool test fault detection experiment and sea trial fault detection experiment respectively, which illustrates the effectiveness of the proposed online modeling algorithm.

7. Conclusions

In this paper, we presented an online learning based underwater robotics thruster fault detection method, which is able to robustly detect faults in the presence of sea current disturbances and time-varying conditions during the operation of the underwater robot. The main contributions include the following.

(1) A fault detection framework based on online learning was proposed, which mainly includes online modeling using a data-driven strategy and modified PSO optimization, with fault detection based on an online updating mechanism. This fault detection frame-

work has impressive generalizability and is suitable for fault detection based on other state parameters, such as the current.

(2) A multi-center particle swarm optimization algorithm with memory ability was proposed, in order to solve the online optimization problem in the process of modeling. We designed an initial optimization strategy based on prior knowledge, historical optimal solution and improved tent mapping, as well as a multi-center collaborative search strategy, which can effectively reduce the odds of becoming stuck in local minima.

The possible future improvements of the proposed online learning-based fault detection method are as follows.

(1) To facilitate implementation, the proposed method uses polynomial fitting as the basis for establishing the control variable–speed model. However, polynomial fitting is not necessarily the most suitable method in this application. Some other mathematical modeling [44] or machine learning-based [45] methods that can achieve small sample learning [46] or adaptive online learning [47] can be tried. The literature [48] proposes a promising optimal learning method based on deterministic artificial intelligence, which is worthy of further study.

(2) In the implementation of the proposed method, residual threshold needs to be determined in advance based on prior knowledge. If the residual error threshold can be automatically generated online during the fault detection process, the adaptiveness of the fault detection algorithm could be further improved. In previous research [49], we proposed an interval prediction method of oscillating time series based on grey system modelling, which may provide some ideas for the estimation of residual threshold.

Author Contributions: Data curation, Y.Z. (Yinlong Zhang); Investigation, X.L. and G.X. (Gaopeng Xu); Methodology, Y.Z. (Yue Zhou); Project administration, G.X. (Gaofei Xu) and W.G.; Resources, G.L.; Software, Y.Z. (Yang Zhao). All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Hainan Provincial Natural Science Foundation of China, grant number 520QN298; National Natural Science Foundation of China, grant number 61903357; Liaoning Provincial Natural Science Foundation of China, grant number 2021JH6/10500114, 2020-MS-032; China Postdoctoral Science Foundation, grant number 2020M672600; The National Key R&D Program of China, grant number 2020YFC1521704.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are available from the corresponding author, Wei Guo, upon request.

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

References

1. Afande, N.; Saad, S.; Ridao, P.; Cieslak, P.; Al-Saggaf, U.M. Implementation of nonlinear adaptive u-model control synthesis using a robot operating system for an unmanned underwater vehicle. *IEEE Access* **2020**, *8*, 205685–205695.
2. Zhang, F.; Sun, X.; Li, Z.; Mohsin, I.; He, K. Influence of processing parameters on coating removal for high pressure water jet technology based on wall-climbing robot. *Appl. Sci.* **2020**, *10*, 1862. [CrossRef]
3. Urkovi, P.; Ehuli, L. Diversity maintenance for efficient robot path planning. *Appl. Sci.* **2020**, *10*, 1721.
4. Cruz, C.M. Soft underwater robot actuated by shape-memory alloys "JellyRobcib" for path tracking through fuzzy visual control. *Appl. Sci.* **2020**, *10*, 7160. [CrossRef]
5. He, J.L.; Dong, M.L.; Sun, G.K. Research on Underwater Motion Control of Compound Driving Small Amphibious Robot. *Chin. J. Sci. Instrum.* **2019**, *40*, 222–229.
6. Byun, S.H.; Kim, S.M.; Park, C. Cyclostationary analysis of underwater noise for vehicle propeller monitoring. In Proceedings of the IEEE Oceans 2016 MTS/IEEE 2016, Monterey, CA, USA, 19–23 September 2016; pp. 1–4.
7. Carolis, V.D.; Maurelli, F.; Brown, K.E. Energy-aware fault-mitigation architecture for underwater vehicles. *Auton. Robot.* **2016**, *41*, 1–23.
8. Zhao, B.; Blanke, M.; Skjetne, R. Particle filter for fault diagnosis and robust navigation of underwater robot. *IEEE Trans. Control Syst. Technol.* **2014**, *22*, 2399–2407. [CrossRef]

9. Yi, S.; Yang, M.M.; Wei, B. Research on Energy Capture Method and Device of Spherical Underwater Robot Based on Suspended Pendulum. *Chin. J. Sci. Instrum.* **2020**, *41*, 214–221.
10. Zeng, J.B.; Li, S.; Li, Y.P. Research and application of the control system for a portable autonomous underwater vehicle. *Robot* **2016**, *38*, 91–97.
11. Dai, Y.; Zhu, X.; Chen, L.S. A mechanical-hydraulic virtual prototype co-simulation model for a seabed remotely operated vehicle. *Int. J. Simul. Model.* **2016**, *15*, 532–541. [CrossRef]
12. Huang, S.; Tan, K.K. Design and analysis of fault diagnosis and fault-tolerant control for a class of MIMO nonlinear state systems. *Instrumentation* **2019**, *4*, 45–52.
13. Li, X.; Huang, G.; Zhang, P.; Zhang, Q. Reliable indoor pseudo lite positioning based on a robust estimation and partial ambiguity resolution method. *Sensors* **2019**, *19*, 3692. [CrossRef] [PubMed]
14. Sun, Y.S.; Ran, X.R.; Li, Y.M. Thruster fault diagnosis method based on Gaussian particle filter for autonomous underwater vehicles. *Int. J. Nav. Archit. Ocean Eng.* **2016**, *8*, 243–251. [CrossRef]
15. Yu, S.S.; Yue, M.L.; Guo, C.Z. Actuator fault diagnosis of autonomous underwater vehicle based on improved Elman neural network. *J. Cent. South Univ.* **2016**, *23*, 808–816.
16. Yuan, J.; Wan, J.; Zhang, W.; Liu, H.; Zhang, H. An underwater thruster fault diagnosis simulator and thrust calculation method based on fault clustering. *J. Robot.* **2021**, *10*, 139–146.
17. Wang, L.; Liu, L.; Qi, J.; Peng, W. Improved quantum particle swarm optimization algorithm for offline path planning in AUVs. *IEEE Access* **2020**, *8*, 143397–143411. [CrossRef]
18. Lv, T.; Zhou, J.; Wang, Y.; Gong, W.; Zhang, M. Sliding mode-based fault tolerant control for autonomous underwater vehicle. *Ocean Eng.* **2020**, *216*, 1–9. [CrossRef]
19. Raanan, B.Y.; Bellingham, J.; Zhang, Y. Detection of unanticipated faults for autonomous underwater vehicles using online topic models. *J. Field Robot.* **2018**, *35*, 705–716. [CrossRef]
20. Zhang, M.; Yin, B.; Liu, W. Thruster fault feature extraction for autonomous underwater vehicle in time-varying ocean currents based on single-channel blind source separation. *Proc. Inst. Mech. Eng. Part I J. Syst. Control Eng.* **2016**, *230*, 46–57. [CrossRef]
21. D'Amato, E.; Mattei, M.; Notaro, I.; Scordamaglia, V. UAV Sensor FDI in duplex attitude estimation architectures using a set-based approach. *IEEE Trans. Instrum. Meas.* **2018**, *67*, 2465–2475. [CrossRef]
22. Zhong, W.; Xin, C. Fault detection of UAV fault based on a SFUKF. *IOP Conf. Ser. Mater. Sci. Eng.* **2019**, *563*, 52–99. [CrossRef]
23. Liu, L.; Ma, Y.; Xu, B.; Xiang, C.; Yang, X. Fault detection and isolation based on UKFs for a novel ducted fan UAV. In Proceedings of the IEEE/CSAA International Conference on Aircraft Utility Systems, Beijing, China, 10–12 October 2016; pp. 212–218.
24. Guo, D.; Zhong, M.; Zhou, D. Multi-sensor data-fusion-based approach to airspeed measurement fault detection for unmanned aerial vehicles. *IEEE Trans. Instrum. Meas.* **2018**, *67*, 317–327. [CrossRef]
25. Abbaspour, A.; Aboutalebi, P.; Yen, K.K.; Sargolzaei, A. Neural adaptive observer-based sensor and actuator fault detection in nonlinear systems: Application in UAV. *ISA Trans.* **2017**, *67*, 317–329. [CrossRef]
26. Yi, Y.; Zhang, Y. Fault diagnosis of an unmanned quadrotor helicopter based on particle filter. In Proceedings of the 2017 International Conference on Unmanned Aircraft Systems (ICUAS 2017), Miami, FL, USA, 13–16 June 2017; pp. 1432–1437.
27. Ouadine, A.Y.; Mjahed, M.; Ayad, H.; Kari, A. UAV Quadrotor fault detection and isolation using artificial neural network and Hammerstein-Wiener model. *Stud. Inform. Control* **2020**, *29*, 317–328. [CrossRef]
28. He, Y.; Peng, Y.; Wang, S.; Liu, D. ADMOST: UAV flight data anomaly detection and mitigation via online subspace tracking. *IEEE Trans. Instrum. Meas.* **2019**, *68*, 1035–1044. [CrossRef]
29. He, Y.; Peng, Y.; Wang, S.; Liu, D.; Leong, P.H.W. A structured sparse subspace learning algorithm for anomaly detection in UAV flight data. *IEEE Trans. Instrum. Meas.* **2018**, *67*, 90–100. [CrossRef]
30. Baskaya, E.; Bronz, M.; Delahaye, D. Fault detection & diagnosis for small UAVs via machine learning. In Proceedings of the IEEE/AIAA 36th Digital Avionics Systems Conference (DASC), St. Petersburg, FL, USA, 17–21 September 2017; pp. 1–6.
31. Guo, K.; Liu, L.S.; Shi, S.H.; Liu, D.T.; Peng, X.Y. UAV sensor fault detection using a classifier without negative samples: A local density regulated optimization algorithm. *Sensors* **2019**, *19*, 771. [CrossRef]
32. Wang, B.; Peng, X.; Jiang, M.; Liu, D. Real-time fault detection for UAV based on model acceleration engine. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 9505–9516. [CrossRef]
33. Wang, B.; Liu, D.; Peng, Y.; Peng, X. Multivariate regression-based fault detection and recovery of UAV flight data. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 3527–3537. [CrossRef]
34. Truong, D.Q.; Ahn, K.K.; Trung, N.T. Design of an advanced time delay measurement and a smart adaptive unequal interval grey predictor for real-time nonlinear control systems. *IEEE Trans. Ind. Electron.* **2018**, *60*, 4574–4589. [CrossRef]
35. Liu, T.; Wang, Q.G.; Huang, H.P. A tutorial review on process identification from step or relay feedback test. *J. Process Control* **2016**, *23*, 1597–1623. [CrossRef]
36. Jin, C.Y.; Ryu, K.H.; Su, W.S. PID auto-tuning using new model reduction method and explicit PID tuning rule for a fractional order plus time delay model. *J. Process Control* **2014**, *24*, 113–128. [CrossRef]
37. Lin, Q.; Loxton, R.; Chao, X. Parameter estimation for nonlinear time-delay systems with noisy output measurements. *Automatica* **2015**, *60*, 48–56. [CrossRef]
38. Wei, Q.; Cui, L. Predictive display for telerobot based on time-delay prediction. *Robot* **2017**, *39*, 298–306.

39. Li, L.J.; Dong, T.T.; Zhang, S. Time-delay identification in dynamic processes with disturbance via correlation analysis. *Control Eng. Pract.* **2017**, *62*, 92–101. [CrossRef]
40. Garcia-Tejeda, Y.V.; Barrera-Figueroa, V. Least squares fitting-polynomials for determining inflection points in adsorption isotherms of spray-dried acai juice (*Euterpe Oleracea* Mart.) and soy sauce powders. *Powder Technol.* **2019**, *342*, 829–839. [CrossRef]
41. Zhou, Y.; Yue, L.B.; Zhang, L.X. Structural damage detection and classification based on clone selection algorithm of particle swarm mutation. *Inf. Control* **2015**, *44*, 436–441.
42. Reinert, G.; Yang, C. A bound on the rate of convergence in the central limit theorem for renewal processes under second moment conditions. *J. Appl. Probab.* **2020**, *57*, 343–360. [CrossRef]
43. Nie, R.; Zhang, W.G.; Li, G.W. Adaptive chaos hybrid multi-objective genetic algorithm based on the Tent map. *J. Beijing Univ. Aeronaut. Astronaut.* **2012**, *38*, 1010–1016.
44. Grešová, E.; Svetlík, J. Mathematical Modeling of the Manufacturing Sector's Dominant Part as a Base for Automation. *Appl. Sci.* **2021**, *11*, 3295. [CrossRef]
45. Choi, H.; Park, S. A Survey of Machine Learning-Based System Performance Optimization Techniques. *Appl. Sci.* **2021**, *11*, 3235. [CrossRef]
46. Hsiao, Y.-D.; Kang, J.-L.; Wong, D.S.-H. Development of Robust and Physically Interpretable Soft Sensor for Industrial Distillation Column Using Transfer Learning with Small Datasets. *Processes* **2021**, *9*, 667. [CrossRef]
47. Beyer, K.; Beckmann, R.; Geißendörfer, S.; von Maydell, K.; Agert, C. Adaptive Online-Learning Volt-Var Control for Smart Inverters Using Deep Reinforcement Learning. *Energies* **2021**, *14*, 1991. [CrossRef]
48. Sands, T. Development of Deterministic Artificial Intelligence for Unmanned Underwater Vehicles (UUV). *J. Mar. Sci. Eng.* **2020**, *8*, 578. [CrossRef]
49. Xu, G.F.; Wang, X.H.; Li, Z.G.; Zhao, Y. Interval prediction of oscillating time series based on grey system modelling. *Int. J. Model. Identif. Control* **2019**, *33*, 138–151. [CrossRef]