TYPE Original Research
PUBLISHED 14 July 2023
DOI 10.3389/fnbot.2023.1219170

Active fault-tolerant anti-input saturation control of a cross-domain robot based on a human decision search algorithm and RBFNN

Ke Wang, Yong Liu* and Chengwei Huang

School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China

Check for updates

OPEN ACCESS

EDITED BY
Long Jin,
Lanzhou University, China

REVIEWED BY
Ruoxi Qin,
Henan Key Laboratory of Imaging and Intelligent Processing, China
Hao Xu,
Anhui University of Technology, China

*CORRESPONDENCE
Yong Liu
✉ liuy1602@njust.edu.cn

RECEIVED 08 May 2023
ACCEPTED 26 June 2023
PUBLISHED 14 July 2023

CITATION
Wang K, Liu Y and Huang C (2023) Active fault-tolerant anti-input saturation control of a cross-domain robot based on a human decision search algorithm and RBFNN. *Front. Neurorobot.* 17:1219170. doi: 10.3389/fnbot.2023.1219170

This article presents a cross-domain robot (CDR) that experiences drive efficiency degradation when operating on water surfaces, similar to drive faults. Moreover, the CDR mathematical model has uncertain parameters and non-negligible water resistance. To solve these problems, a radial basis function neural network (RBFNN)-based active fault-tolerant control (AFTC) algorithm is proposed for the robot both on land and water surfaces. The proposed algorithm consists of a fast non-singular terminal sliding mode controller (NTSMC) and an RBFNN. The RBFNN is used to estimate the impact of drive faults, water resistance, and model parameter uncertainty on the robot and the output value compensates the controller. Additionally, an anti-input saturation control algorithm is designed to prevent driver saturation. To optimize the controller parameters, a human decision search algorithm (HDSA) is proposed, which mimics the decision-making process of a crowd. Simulation results demonstrate the effectiveness of the proposed control methods.

KEYWORDS

cross-domain robot (CDR), radial basis function neural network (RBFNN), active fault-tolerant control (AFTC), anti-input saturation, human decision search algorithm (HDSA)

1. Introduction

In recent years, there has been a growing interest in multi-environment robots as single-environment robots are no longer sufficient to meet various practical needs (Cohen and Zarrouk, 2020). Researchers have proposed different designs to achieve this, such as bionic robots (Chen et al., 2021) and the legged amphibious robot (Xing et al., 2021). Furthermore, with the advancements in rotorcraft unmanned aerial vehicle (UAV) technology, researchers have started exploring the potential of integrating rotorcraft UAVs with wheeled mobile robots (WMRs) (Wang et al., 2019a). To enhance the capabilities of robots, cross-domain robots (CDRs) have been designed, which are capable of operating in multiple environments, including water, land, and air (Guo et al., 2019; Zhong et al., 2021). The robot presented in this paper is a CDR that combines a quadrotor UAV with a WMR equipped with webbed plates. These webbed plates on the wheels enable the robot to generate power at the water surface through their interaction with the water (Wang et al., 2022a,b).

The CDR presented in this study employs the same drive motors for ground and water surface operations. Assuming proper functionality during ground motion, a driver fault is considered to have occurred during the robot's operation on the water surface.

Fault-tolerant controls (FTCs) are control algorithms that effectively deal with system faults (Najafi et al., 2022; Nan et al., 2022). Sliding-mode controllers (SMCs) are commonly employed in passive fault-tolerant algorithms due to their robustness in maintaining control performance when the maximum system fault is known. However, the use of non-singular terminal sliding mode control (NTSMC) and SMC results in jitter problems, and this robust control approach is considered too conservative (Ali et al., 2020; Hou and Ding, 2021; Guo et al., 2022). To address these issues, FTCs frequently employ adaptive sliding mode control (Wu et al., 2020) and integral sliding mode control (Yu et al., 2022). Additionally, observers are commonly used to detect drive faults. In Wang F. et al. (2022), a disturbance observer (DO) is used to quickly compensate and correct unknown actuator faults of unmanned surface vehicles (USVs). In the context of autonomous underwater vehicles (AUVs), a sliding mode observer-based fault-tolerant control algorithm has been proposed in the literature (Liu et al., 2018). However, the design of higher-order observers requires complex mathematical proofs and the adjustment of many parameters. Neural networks (NNs) are often used to estimate system model parameters and uncertainty terms due to their ability to approximate arbitrary non-linear functions. In Zhang et al. (2022), NNs are used to rectify the model parameters of a USV, and an NN-based adaptive observer is developed to estimate errors caused by drive faults. As demonstrated in Gao et al. (2022), NNs can directly estimate system faults by approximating the uncertainty terms in the system. Event-triggered fault-tolerant control is a type of AFTC algorithm that has the potential to reduce system hardware requirements. However, it requires the development of trigger thresholds and corresponding fault control algorithms, which increase the difficulty and complexity of controller design (Huang et al., 2019; Wu et al., 2021; Zhang et al., 2021). Another important consideration in the FTC algorithm is the control of input saturation. One efficient approach for solving this issue is to introduce virtual states in the controller. These virtual states regulate the input error of the controller, thereby suppressing control input saturation (Wang and Deng, 2019). Additionally, designing adaptive laws is an effective way to address control input saturation. In this approach, the adaptive control input decreases as the actual control input approaches the maximum physical constraint (Shen et al., 2018).

The controller design presented above does not involve any optimization of the controller parameters. To address this limitation, reinforcement learning techniques have been developed to optimize control parameters. In Gheisarnejad and Khooban (2020), a reinforcement learning algorithm is employed to optimize the PID controller parameters. Another study (Zhao et al., 2020) trains the optimal trajectory following controller using deep reinforcement learning. However, reinforcement learning algorithms typically require a significant amount of data and multiple iterations to achieve optimal results. Swarm intelligence (SI) optimization algorithms are a promising approach in practical applications, including data classification, path planning, and controller optimization (Xue and Shen, 2020, 2022). Among the various SI optimization algorithms, particle swarm optimization (PSO) is a classical algorithm known for fast convergence and few parameters (Song and Gu, 2004). However, traditional PSO algorithms tend to fall into local optima. Ant colony optimization (ACO) is another common SI optimization algorithm. ACO can jump out of local optima but has slower convergence (Dorigo et al., 1996). In addition, the gray wolf optimizer (GWO) simulates the predation process of wolves (Mirjalili et al., 2014) and the Harris hawk optimizer (HHO) simulates the predation process of hawks (Heidari et al., 2019). These algorithms have shown improvements in convergence speed and accuracy compared with other animal predation simulation algorithms. Other popular SI optimization algorithms include the firefly algorithm (Fister et al., 2013) and the sine/cosine search algorithm (Mirjalili, 2016). Each SI optimization algorithm has its own strengths and weaknesses and no single algorithm can effectively handle all optimization problems. The goal is to achieve satisfactory results in terms of convergence speed, accuracy, and robustness for a specific optimization problem.

Based on the previous discussion, an AFTC is proposed for the CDR on the ground and on the water surface. This control algorithm consists of three main parts:

a. To enhance the robustness of the robot control system, a fast NTSMC is designed based on the concept of passive FTC. Compared with traditional NTSMC and SMC, the proposed NTSMC has reduced control input chatter. Additionally, to reduce controller conservatism, an RBFNN is designed to detect and compensate for drive faults. The adaptive weight control law of the RBFNN is based on the Lyapunov function.
b. To prevent drive saturation, an anti-input saturation control algorithm based on the hyperbolic tangent (tanh) function is employed. An adaptive rate is designed to prevent singularities in this algorithm. This method does not require complex mathematical proofs and requires fewer tuning parameters.
c. A new SI optimization algorithm named HDSA is proposed for the optimization of the weight update rate parameter of RBFNNs. The proposed algorithm is compared with other SI optimization algorithms, and the test results demonstrate its faster convergence rate and higher accuracy.

2. Related work and mathematical models

2.1. HDSA's related work

To demonstrate the advantages of the proposed HDSA optimization algorithm, the results of the HDSA tests are shown in this section. The theory of HDSA is discussed in detail in the section entitled "RBFNN-Based Active Fault-Tolerant Control Algorithm". The effectiveness of the proposed optimization algorithm was evaluated by comparing the test results of HDSA with other popular optimization algorithms, such as particle swarm optimization (PSO) (Song and Gu, 2004), the sine/cosine algorithm (SCA) (Mirjalili, 2016), the gray wolf optimizer (GWO) (Mirjalili et al., 2014), the firefly algorithm (FA) (Fister et al., 2013), and the Harris hawk optimizer (HHO) (Heidari et al., 2019). Twenty standard test functions were used for evaluation, which are presented in Tables 5–7 (included in the Simulation Results section).

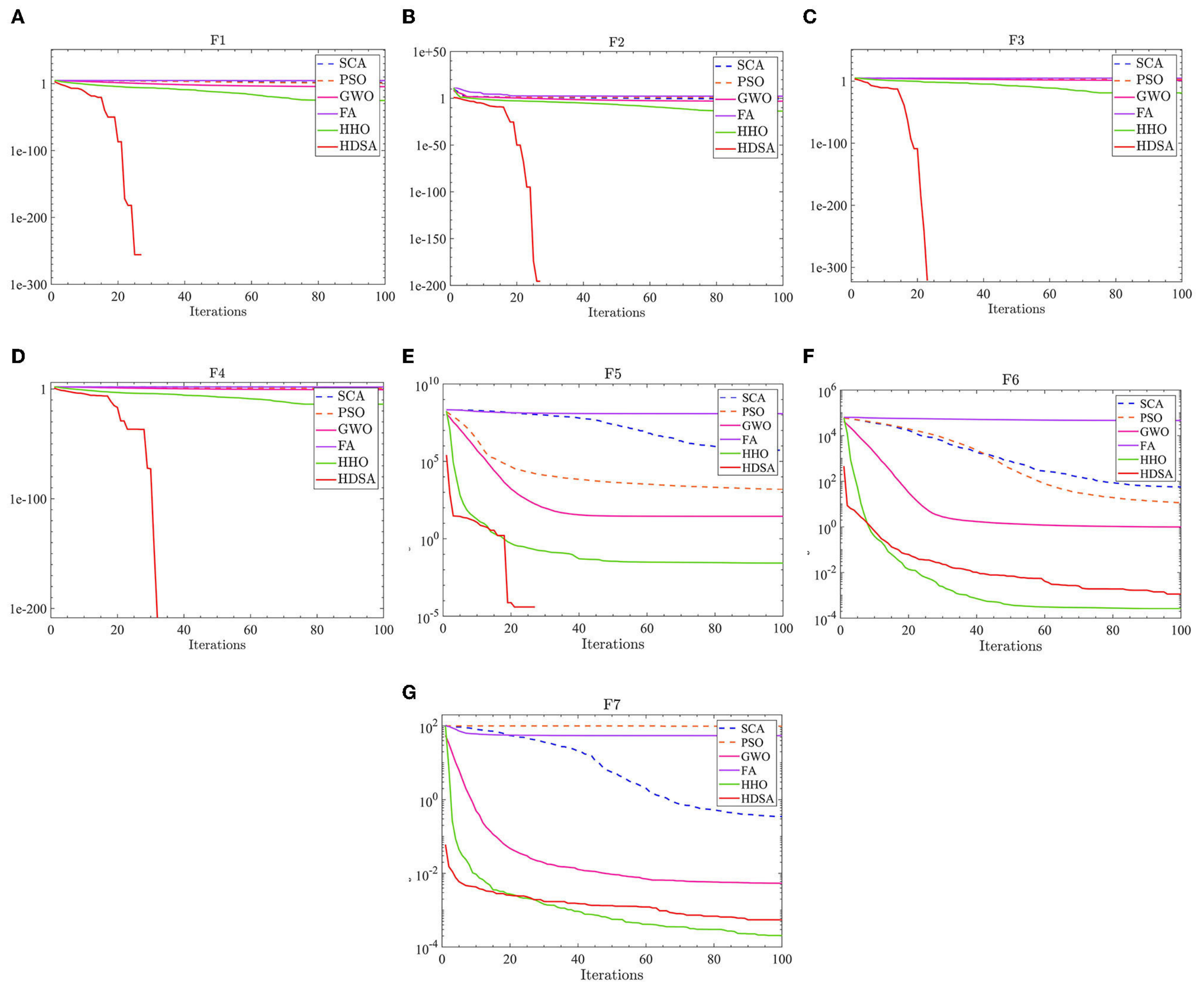

FIGURE 1
Single peak function test results. **(A–G)** represent the test results of the six algorithms in functions F1 to F7.

The number of populations was $pop = 100$ and the maximum number of iterations was $M = 100$. The average fitness over 30 independent runs was considered as the optimization result. The convergence characteristics of the six algorithms in the single-peak function test are depicted in Figure 1, while Figure 2 illustrates the convergence characteristics in the multi-peak function test. Furthermore, Figure 3 demonstrates the convergence characteristics of the six algorithms on fixed-dimensional multi-peak functions. The test results of the six algorithms, based on 30 independent runs, are summarized in Tables 1, 2. In Tables 1, 2, purple indicates the optimal value of the test functions, pink indicates the mean value of the test functions, and white indicates the mean squared deviation of the test functions.

The results of the single-peak functions F1–F7 test results are presented in Tables 1, 2. In these tests, the mean and optimal values obtained by HDSA in F1–F5 are both 0, indicating that HDSA achieves the highest accuracy among the six algorithms. Although the accuracy of HDSA is slightly inferior to HHO in the F6–F7 test functions, it still outshines SCA, PSO, GWO, and FA. HDSA has a standard deviation of 0 in tests F1–F5, suggesting that HDSA is the most stable algorithm. Although its stability is slightly lower than HHO in tests F6–F7, it still outperforms the other four methods. Convergence speed is depicted in Figure 2. HDSA has a significantly faster convergence speed compared with the other five algorithms, but its convergence accuracy in the F6–F7 tests is lower than that of HHO.

The test results for the multi-peak functions F8–F13 are presented in Tables 1, 2. In the tests from F9 to F13, HDSA exhibits significantly better stability and convergence accuracy compared with the other five algorithms. It achieves higher accuracy and the smallest standard deviation. As depicted in Figure 3, except for the F8 test function, HDSA showcases the fastest convergence speed and highest convergence accuracy among the algorithms.

The results of the fixed dimensional multi-peak functions F14–F20 test results are shown in Tables 1, 2. In the F14 test, SCA has the best optimal and average accuracy, while HDSA exhibits slightly lower average accuracy and stability compared with SCA, PSO, and HHO. However, HDSA still manages to find the optimal

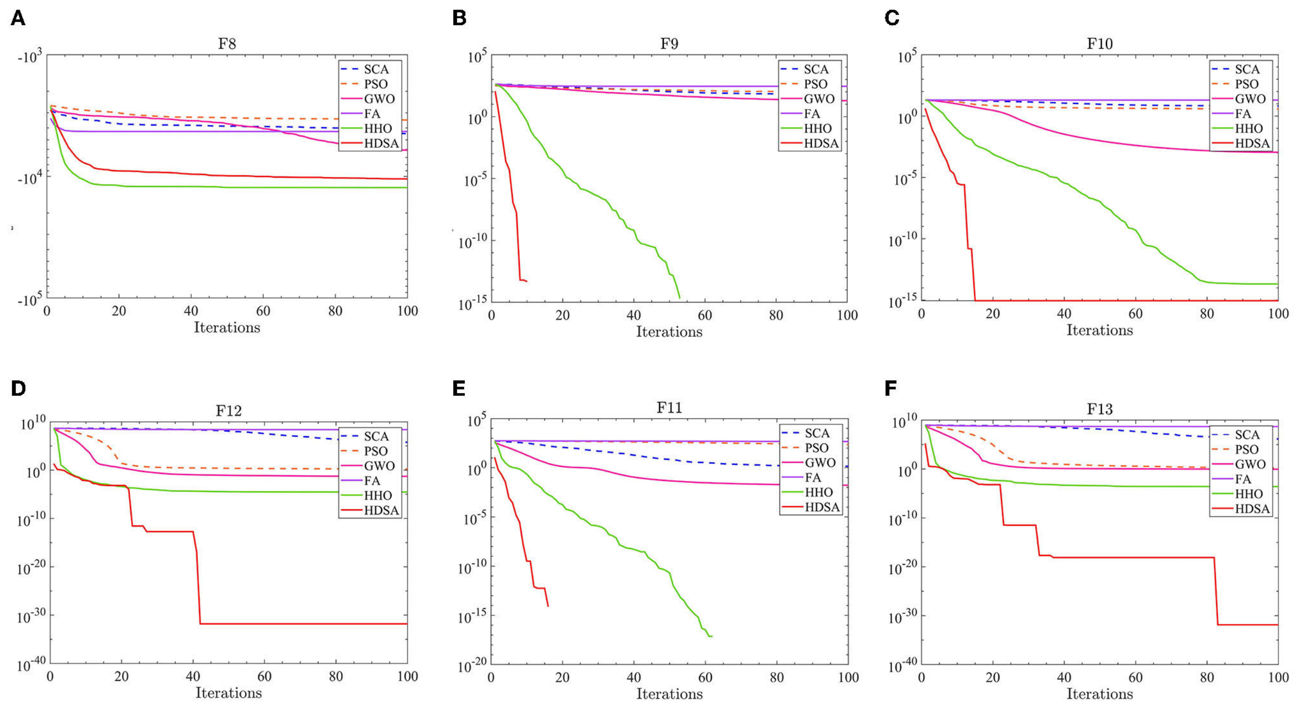

FIGURE 2
Multi-peak function test results. **(A–F)** represent the test results of the six algorithms in functions F8 to F13.

solution in 30 runs. In the F15–F18 test results, HDSA, SCA, GWO, and HHO perform closely, with good stability and accuracy. In the F19–F20 tests, HDSA outperforms the other five algorithms significantly in terms of accuracy and stability. As shown in Figure 3, HDSA exhibits the fastest convergence speed among the other test functions, except for F15, F17, and F18. In the F15 test, HDSA is only slightly slower than HHO, while in the F17 and F18 tests, HDSA converges slightly slower than FA.

2.2. Mathematical model of the CDR

Before discussing the mathematical model of the CDR, the following assumptions are made: **Assumption 1:** The center of gravity and the geometric center of the robot body coincide. **Assumption 2:** The motor output torque meets the actual performance requirements of the robot during ground and water motion. **Assumption 3:** The robot's vertical swing, horizontal rocking, and longitudinal rocking during its movement on the water surface are ignored. **Assumption 4:** The motion of the robot on the ground is purely rolling, without any sliding motion.

The CDR designed in this study can be seen as a combination of a quadrotor UAV and a WMR. Figure 4A shows the robot moves on the ground. Figure 4B shows the robot moves on the water surface by webbed plates. Figure 4C shows the robot moves on the water surface by propllers. The robot moves in the air in a similar way to the quadrotor UAV as shown in Figures 4D, E. Figure 4F shows the structure of the robot, where webbed plates are mounted on the wheels. These webbed plates generate traction and rotational torque on the water surface by interacting with the water. However, as this paper focuses primarily on the FTC algorithm of the robot on the ground and on the water surface, the discussion does not explore the robot's aerial motion in detail.

The robot in the inertial frame and in the body frame is shown in Figure 5.

In Figure 5, d is the distance from the geometric center of the robot O_b to the mass center of the robot. b is the axis radius and r is the wheel radius. ω_l, ω_r are the angular velocities of the left and right wheels. ψ is the angle between the robot body coordinate system b and the inertial coordinate system A, and ψ is the yaw angle of the robot. The kinematic model of the robot on the ground and water surface can be represented as (Liu et al., 2020):

$$\dot{q} = R\eta \tag{1}$$

where $q = \begin{bmatrix} x & y & \psi \end{bmatrix}$ represents the position and orientation of the robot in the inertial frame, while $\eta = \begin{bmatrix} u & v & r \end{bmatrix}$ is used to denote the longitudinal velocity, lateral velocity, and yaw angular velocity in the body frame. The coordinate conversion matrix is denoted by R, where $R = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$. The dynamics model of the robot's motion on the ground can be expressed as

$$M(q)\ddot{q} + C_m(q,\dot{q})q + F(\dot{q}) + \tau_d = B(q)\tau \tag{2}$$

The matrices M are symmetric positive definite inertia matrices, while C_m represents the centripetal and Coriolis matrix. The term $F(\dot{q})$ denotes mechanical friction, while τ_d is used to represent external disturbances. The input transformation matrices are

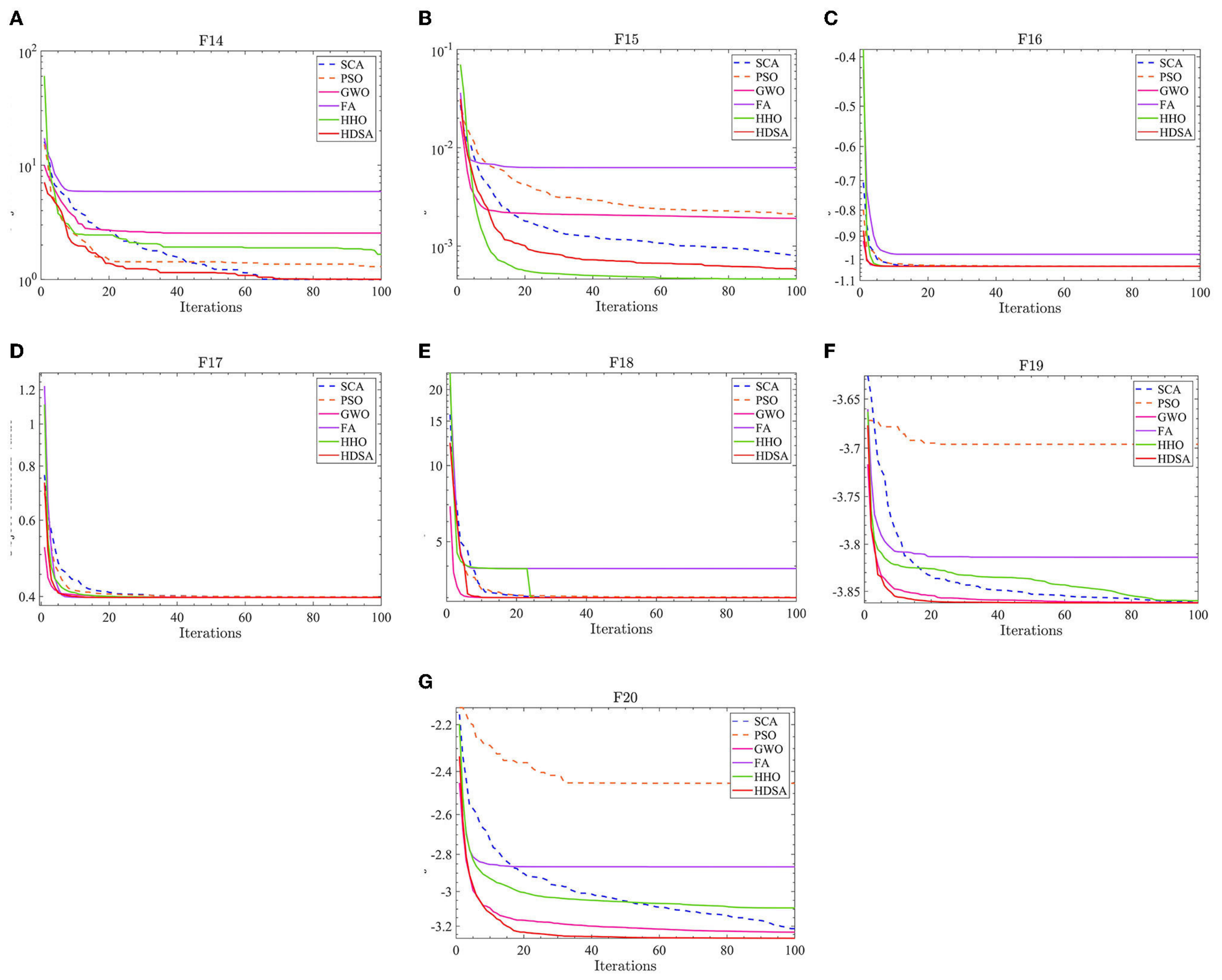

FIGURE 3
Fixed dimensional multi-peak function results. **(A–G)** represent the test results of the six algorithms in functions F14 to F20.

denoted as $B(q)$. Furthermore, the robot drive motors in the left and right wheel output torque are represented by $\tau = \begin{bmatrix} \tau_l & \tau_r \end{bmatrix}^T$.

$$M(q) = \begin{bmatrix} m & 0 & md\sin\psi \\ 0 & m & -md\cos\psi \\ md\sin\psi & -md\cos\psi & I \end{bmatrix},$$

$$B(q) = \frac{1}{r}\begin{bmatrix} \cos\psi & \cos\psi \\ \sin\psi & \sin\psi \\ L & -L \end{bmatrix},$$

$$C_m(q,\dot{q}) = \begin{bmatrix} md\dot{\psi}^2\cos\psi & md\dot{\psi}^2\sin\psi & 0 \end{bmatrix}^T$$

The mass of the robot is represented by m. The I is a scalar quantity and represents the rotational inertia of the robot as it rotates in the X-Y plane. The angular velocity of the robot is assumed to vary smoothly, so that $\dot{\psi} \approx 0$. According to **assumption 1**, the Coriolis matrix can be assumed to be negligible, resulting in $C_{\rm m} \approx 0$. According to **assumption 1**, $d = 0$, so the matrix $M(q) = diag\begin{bmatrix} m & m & I \end{bmatrix}$. Based on these assumptions, the dynamics model of the robot on the ground can be rewritten as follows:

$$\bar{M}\ddot{q} + \bar{C}q + +\bar{F}(\dot{q}) + \bar{\tau}_d = \bar{B}\tau \tag{3}$$

where $\bar{C} = R^{-1}C_m\dot{R}$, $\bar{M} = R^{-1}MR$, $\bar{B} = R^{-1}B$. $\bar{F}(\dot{q}) = \begin{bmatrix} f_u & f_v & f_r \end{bmatrix}^T$ is the mechanical friction and $\bar{\tau}_d = \begin{bmatrix} d_u & d_v & d_r \end{bmatrix}^T$ is the external disturbance. Rewriting 3 into algebraic form can be expressed as:

$$\begin{cases} \dot{u} = \left(F_u - f_u - d_u\right)/m + v\omega \\ \dot{v} = -u\omega - \left(f_v + d_v\right)/m \\ \dot{r} = \left(T_r - f_r - d_r\right)/I \end{cases} \tag{4}$$

The traction force is represented by F_u, while T_r represents the torque. To model the dynamics of the robot on the water surface, we can refer to the USV dynamics model (Chen et al., 2019), which can be expressed as follows

$$M_w(q)\dot{\eta} + C_w\left(q,\eta\right) + D_w\left(\eta\right)\eta + F_w(\eta) + \tau_{dw} = \tau_w \tag{5}$$

TABLE 1 Test results of HDSA SCA and PSO algorithms run independently 30 times.

F	HDSA			SCA			PSO		
	Best	Ave	Std	Best	Ave	Std	Best	Ave	Std
F1	0	0	0	0.043621199	55.2091773	36.42876978	4.456913956	8.976645307	46.32264983
F2	0	0	0	0.010156561	0.288456338	0.399117642	6.86827201	10.3267055	10.28163277
F3	0	0	0	9.53E+03	2.23E+04	5.86E+03	2.70E+02	7.57E+02	2.15E+04
F4	0	0	0	37.35838437	59.20677056	8.010506582	1.92884739	3.830868295	55.38478266
F5	0	0	0	37.92815456	6.61E+05	8.02E+05	5.46E+02	1.57E+03	6.59E+05
F6	3.14E−05	0.001383925	0.001405319	4.67425676	1.15E+02	1.15E+02	6.428054849	10.03538963	57.17359472
F7	5.90E−05	5.11E−04	4.07E−04	0.024658139	0.341143612	0.269446788	45.07882375	96.08387995	98.09394285
F8	−1.26E+04	−1.07E+04	1.97E+03	−4.81E+03	−4.37E+03	2.21E+02	−4.12E+03	−3.49E+03	9.36E+02
F9	0	0	0	0.860127299	78.49328591	70.22545167	30.55768733	94.82695388	34.53524407
F10	8.88E−16	8.88E−16	0	0.187682845	10.65598272	8.935131413	2.867362804	3.884021036	6.796254632
F11	0	0	0	0.513962142	1.962336683	2.819722656	1.72E+02	2.25E+02	2.24E+02
F12	1.57E−32	1.57E−32	5.47E−48	1.043279428	3.39E+05	8.56E+05	0.650894524	1.738034681	3.39E+05
F13	1.35E−32	1.84E−23	9.89E−23	10.16366581	2.10E+06	2.56E+06	0.62640203	1.796606655	2.10E+06
F14	0.998003838	4.801561855	4.696216357	0.998003841	0.998323781	9.29E−04	0.998003838	1.163740602	0.405679435
F15	3.08E−04	4.82E−04	2.60E−04	4.25E−04	8.05E−04	1.88E−04	5.35E−04	0.003654847	0.007160687
F16	−1.031628435	−1.03162038	8.23E−06	−1.031628443	−1.031626913	1.82E−06	−1.031615014	−1.031069614	6.31E−04
F17	0.397888187	0.397903308	1.88E−05	0.397889317	0.397918592	3.12E−05	0.397935785	0.399283994	0.001760397
F18	3.000000032	3.000013391	1.62E−07	3.000000177	3.000055929	8.99E−05	3.000002051	3.007764558	0.014062845
F19	−3.862751312	−3.862443601	2.78E−04	−3.86268097	−3.861957813	0.00102005	−3.849759489	−3.653030339	0.272064362
F20	−3.320685667	−3.277232199	0.056398698	−3.314075954	−3.22298511	0.922404083	−2.942883457	−2.387193028	0.041529991

M_w is the inertia matrix. The traction force and torque of the robot at the water surface are $\tau_w = \begin{bmatrix} F_u & 0 & T_r \end{bmatrix}^T$. $\tau_{dw} = \begin{bmatrix} d_{uw} & d_{vw} & d_{rw} \end{bmatrix}^T$ is the lumped disturbance and $F_w(\eta) = \begin{bmatrix} f_{uw} & f_{vw} & f_{rw} \end{bmatrix}$ is the water resistance.

$$M_w = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{32} & m_{33} \end{bmatrix},$$

$$C_w\left(q,\eta\right) = \begin{bmatrix} 0 & 0 & C_{13}\left(\eta\right) \\ 0 & 0 & C_{23}\left(\eta\right) \\ -C_{13}\left(\eta\right) & -C_{23}\left(\eta\right) & 0 \end{bmatrix},$$

$$D_w\left(\eta\right) = \begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & d_{23} \\ 0 & d_{32} & d_{33} \end{bmatrix}.$$

The disturbances are represented by τ_{dw}. On the other hand, $D_w\left(\eta\right)$ represents the water resistance. The Coriolis force matrix can also be neglected according to **Assumption 1** and **Assumption 3**, so $C_w\left(q,\eta\right) \approx 0$. The elements of the non-diagonal matrix in matrix $D_w\left(\eta\right)$ and matrix M_w are small and can be neglected. This model simplification approach is also more common (Liao et al., 2016; Wang et al., 2019b; Deng et al., 2020), where $m_{11} = m - X_{\dot{u}}$, $m_{22} = m - Y_{\dot{v}}$, and $m_{33} = I_z - N_{\dot{r}}$ are the inertia parameters of the three axes and $X_{\dot{u}}$, $Y_{\dot{v}}$, and $N_{\dot{r}}$ are the additional inertia parameters due to the wet water of the robot shell and the viscosity of the water. The dynamics model of the robot on the water surface can be expressed as:

$$\begin{cases} \dot{u} = \frac{m_{22}}{m_{11}} v\omega - \frac{X_u}{m_{11}} u - \frac{X_{|u|u}}{m_{11}} \left|u\right| u + \frac{F_u}{m_{11}} + \frac{d_u}{m_{11}} \\ \dot{v} = -\frac{m_{11}}{m_{22}} u\omega - \frac{Y_u}{m_{22}} v - \frac{Y_{|v|v}}{m_{22}} \left|v\right| v + \frac{d_v}{m_{11}} \\ \dot{\omega} = \frac{m_{11}-m_{22}}{m_{33}} u\upsilon - \frac{N_\omega}{m_{33}} \omega - \frac{N_{|\omega|\omega}}{m_{33}} \left|\omega\right| \omega + \frac{T_r}{m_{33}} + \frac{d_r}{m_{33}} \end{cases} \tag{6}$$

X_u, $X_{|u|u}$, Y_u, $Y_{|v|v}$, and N_ω, $N_{|\omega|\omega}$ are the resistance coefficients. The resistance of the robot moving on the water surface can be approximated as a quadratic function of the velocity and angular velocity.

The mathematical model should be rewritten into a form that better suits the needs of the subsequent controller design. The dynamics model of the robot's motion on the ground is rewritten according to 4 as

$$\begin{cases} \dot{u} = F_u/m - \underbrace{\left(f_u + d_u\right)/m + v\omega}_{d_{ug}} \\ \dot{r} = T_r/I - \underbrace{\left(f_r + d_r\right)/I}_{d_{rg}} \end{cases} \tag{7}$$

TABLE 2 Test results of GWO FA and HHO algorithms run independently 30 times.

F	GWO			FA			HHO		
	Best	Ave	Std	Best	Ave	Std	Best	Ave	Std
F1	2.69E−06	2.59E−05	1.52E−05	2.29E+04	4.77E+04	9.85E+03	1.08E−33	1.95E−26	7.53E−26
F2	4.61E−04	8.55E−04	2.86E−04	53.06138425	1.06E+02	18.47075013	1.08E−17	2.26E−14	6.46E−14
F3	2.333236967	16.15496494	16.17029995	3.37E+04	6.69E+04	1.69E+04	6.30E−32	4.03E−18	2.17E−17
F4	0.076161242	0.190101216	0.067944463	46.02316618	63.05292108	7.574286394	1.03E−17	4.60E−14	1.18E−13
F5	26.18035457	28.03756308	0.956956464	5.84E+07	1.29E+08	3.79E+07	1.17E−04	0.043168201	0.061375953
F6	3.62E−04	0.996486127	0.498192351	3.06E+04	4.64E+04	7.31E+03	2.08E−06	2.68E−04	3.23E−04
F7	0.001822731	0.004742428	0.001594265	10.11397979	48.34775676	17.25308781	9.10E−06	1.82E−04	1.50E−04
F8	−8.30E+03	−6.30E+03	1.05E+03	−5.73E+03	−4.35E+03	6.34E+02	−1.26E+04	−1.25E+04	2.43E+02
F9	10.61773399	21.78763545	6.854472334	2.15E+02	0.860127299	34.00946589	0	0	0
F10	6.79E−04	0.001208914	4.59E−04	19.41517193	19.96298677	0.1314286	8.88E−16	1.33E−14	2.00E−14
F11	2.16E−05	0.020113329	0.01797284	4.03E+02	4.93E+02	48.66049354	0	0	0
F12	0.01735841	2.091212922	0.039891925	5.55E+07	2.26E+08	1.13E+08	2.47E−07	2.25E−05	2.25E−05
F13	0.39714087	0.90669375	0.26593137	1.29E+08	4.74E+08	1.87E+08	5.84E−11	3.13E−04	4.99E−04
F14	0.998003838	2.149370759	1.977247758	0.998003838	9.85228046	7.376397236	0.998003838	1.592846754	1.007706592
F15	3.33E−04	0.002524088	0.005948479	5.95E−04	0.009720032	0.008406408	3.09E−04	4.19E−04	2.61E−04
F16	−1.031628453	−1.031628406	8.86E−04	−1.031621754	−1.030900759	0.002366781	−1.031628453	−1.031628451	1.05E−08
F17	0.397887459	0.397888965	1.47E−06	0.397894813	0.398122914	3.37E−04	0.397887358	0.397893418	2.30E−05
F18	3.000000021	3.000091041	9.87E−05	3.000120892	3.027874998	0.065760186	3	3.000000968	4.43E−06
F19	−3.86278078	−3.861772215	0.00183244	−3.861890169	−3.830959144	0.086620017	−3.862769505	−3.861362289	0.001672668
F20	−3.321992055	−3.265460239	0.071106357	−3.201236207	−2.894366935	0.195231925	−3.263585483	−3.123254299	0.085277304

Where d_{ug} is the lumped disturbance and $d_{ug} \leq \bar{d}_{ug}$, $\bar{d}_{ug}$ is the upper limit of the total disturbances. d_{rg} is the lumped disturbance and $d_{rg} \leq \bar{d}_{rg}$, $\bar{d}_{rg}$ is the upper limit of the total disturbances. The dynamics model of the robot on the water surface is

$$\begin{cases} \dot{u} = \frac{F_{uc}}{m} \underbrace{- \frac{\xi_u F_{uc}}{m_{11}} - F_{ua} - \frac{X_u}{m_{11}} u - \frac{X_{|u|u}}{m_{11}} |u|\, u + \Delta_F}_{-D_{uw}} \\ \quad \underbrace{+ \frac{m_{22}}{m_{11}} v\omega + \frac{d_u}{m_{11}}}_{d_{uw}} \\ \dot{r} = \frac{T_{rc}}{I} \underbrace{- \frac{\xi_r T_{rc}}{m_{33}} - T_{ra} - \frac{N_\omega}{m_{33}} \omega - \frac{N_{|\omega|\omega}}{m_{33}} |\omega|\, \omega + \Delta_T}_{-D_{rw}} \\ \quad \underbrace{+ \frac{m_{11} - m_{22}}{m_{33}} uv + \frac{d_r}{m_{33}}}_{drw} \end{cases} \tag{8}$$

where F_{uc} is the desired tractive force and $F_{uc} = F_u$ represents no force loss. $\xi_u \in \left[0 \ 1 \right)$ is the force loss parameter. Δ_F is the force disturbance due to mass change. d_{uw} is a lumped disturbance, $d_{uw} \leq \bar{d}_{uw}$. $\bar{d}_{uw}$ is the upper bound of d_{uw}. D_{uw} is the uncertainty term when the robot moves on the water surface due to changes in system parameters, water resistance, and driver faults. T_{rc} is the desired torque and $T_{rc} = T_r$ represents no force loss. $\xi_r \in \left[0 \ 1 \right)$ is the power loss parameter. Δ_T is the torque disturbance due to the change of inertia parameter. d_{rw} is a lumped disturbance, $d_{rw} \leq \bar{d}_{rw}$. $\bar{d}_{rw}$ is the upper bound of d_{rw}. D_{rw} is the uncertainty term due to changes in system parameters, water resistance, and driver faults during robot rotation on the water surface.

3. Active fault tolerance control algorithm and human decision search algorithm

3.1. RBFNN-based active fault-tolerant control algorithm

Both the yaw control and the linear velocity control of the robot are essentially single-input single-output (SISO) second-order non-linear affine systems. Without loss of generality, a second-order non-linear affine SISO system with drive faults can be expressed as:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f(x) + g(x) u_c + D \ + d \\ y = x_1 \end{cases} \tag{9}$$

u_c is unconstrained control input, u_a is the drive bias, ξ is the power loss parameter, $\xi \in \left[0 \ 1 \right)$, 0 represents no power loss, and 1 represents a complete loss of efficiency. $D = -g(x) \xi u_c + u_a$ is the uncertainty term due to the driver fault. The disturbance d has a

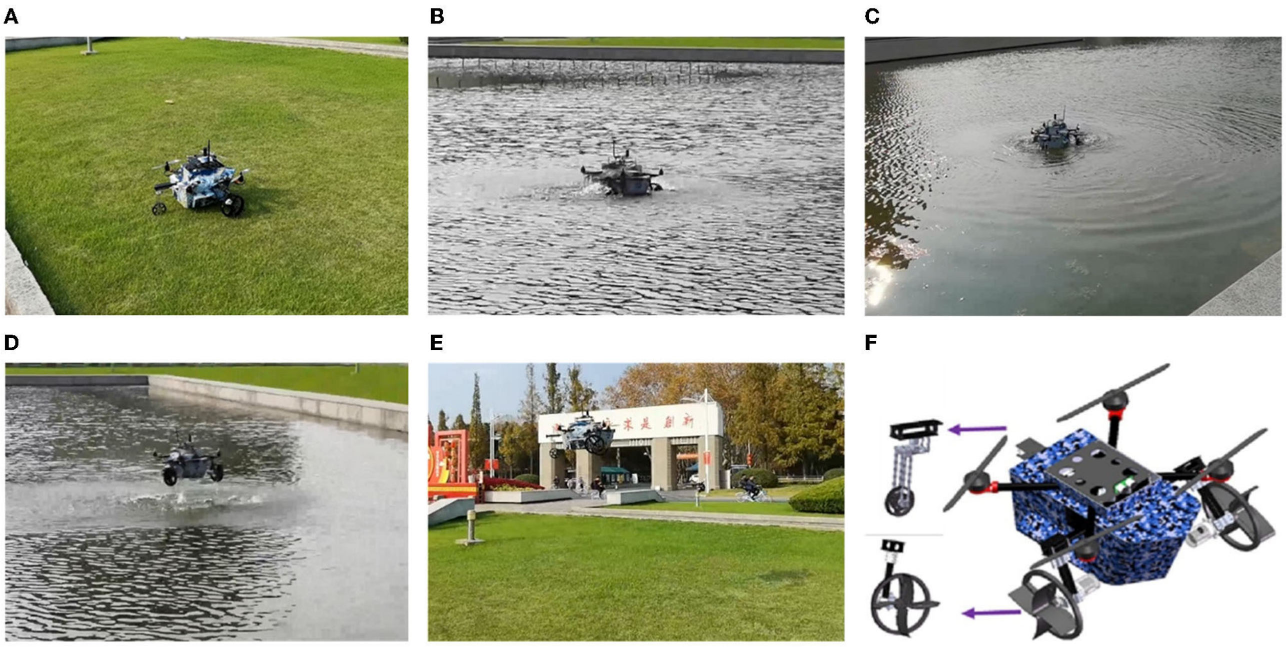

FIGURE 4
(A) The robot moves on the ground. **(B)** The robot moves on the water surface by webbed plates. **(C)** The robot moves on the water surface by propllers. **(D)** The robot takes off from water surface. **(E)** The robot flying in the air. **(F)** The structure of robot.

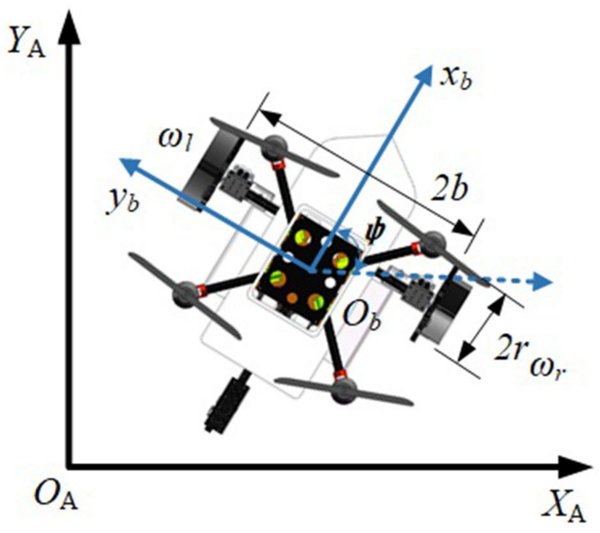

FIGURE 5
Robot in the inertial frame and the body frame.

well-defined upper limit and $|d| \le \bar{d}$. x_1, x_2 are system states. $f(x)$ is the system function and $g(x)$ is the input function. Owing to the physical constraints of the controlled object, the control input is subject to saturation:

$$u_{con} = \begin{cases} u_{\max},\ |u_c| > u_{\max} \\ u_c,\ u_c \le u_{\max} \end{cases} \tag{10}$$

$u_{\max}$ is the physical constraint. To make the control input smoother, the cutoff function is usually replaced by a saturation function, such as *tanh*.

$$u_{con} = u_{\max} \tanh(u_f/u_{\max}) \tag{11}$$

where u_{con} is the constrained control input and u_f is a function of u_c. Thus, the control objective is to design the constrained control law u_{con} so that it satisfies the control requirements even in the presence of drive faults and external disturbances in the controlled object. The steps for designing an AFT controller are the following:

Step 1: Define the state error $e_1 = x_{1d} - x_1$. Establish the Lyapunov function $V_1 = \frac{1}{2}e_1^2$. Taking the derivative of V_1 with respect to the time t gives

$$\dot{V}_1 = e_1\dot{e}_1 = e_1(\dot{x}_{1d} - x_2) \tag{12}$$

Define the virtual state $\alpha_x = k_1e_1 + \dot{x}_{1d}$ as the desired input of the next step. If x_2 can follow α_x, $\dot{V}_1 = -k_1e_1^2$. So, the next step of the control law must ensure that $\alpha_x - x_2 = 0$. α_x is the next desired state x_{2d}.

Step 2: Define the state error $e_2 = x_{2d} - x_2$, and the fast NTSMC is designed as

$$S = e_2 + \alpha e_1 + \beta e_1^{\lambda} \tag{13}$$

where α and β are positive adjustable parameters and λ is a positive odd number. The sliding mode convergence law is

$$\dot{S} = -k_2S - k_3|S|^{\gamma_1}\operatorname{sgn}(S) \tag{14}$$

where k_1, k_2, and γ_1 are positive adjustable parameters. *sgn* is the symbolic function. The derivation of 13 yields:

$$\dot{S} = \dot{e}_2 + \alpha\dot{e}_1 + \lambda\beta {e_1}^{\lambda-1}\dot{e}_1 = -k_2S - k_3|S|^{\gamma_1}\operatorname{sgn}(S) \tag{15}$$

where

$$\begin{aligned}\dot{e}_2 &= \dot{x}_{2\mathrm{d}} - \dot{x}_2\\ &= \dot{\alpha}_x - f(x) - g(x)u_c - d - D\\ &= -k_2S - k_3|S|^{\gamma_1}\operatorname{sgn}(S)\end{aligned} \tag{16}$$

The controller law can be designed as follows:

$$u_c = \frac{1}{g(x)}\left(\dot{\alpha}_x - f(x) - D + k_2S + k_3|S|^{\gamma_1}\operatorname{sgn}(S) + \alpha\dot{e}_1 + \lambda\beta {e_1}^{\lambda-1}\dot{e}_1\right) \tag{17}$$

In 17, the uncertain term due to drive faults D is known. Establishing the Lyapunov function $V_2 = \frac{1}{2}S^2$, the derivative of V_2 yields

$$\begin{aligned}\dot{V}_2 &= S\dot{S}\\ &= S\left(\dot{e}_2 + \alpha\dot{e}_1 + \lambda\beta {e_1}^{\lambda-1}\dot{e}_1\right)\\ &= S\left(\dot{\alpha}_x - f(x) - g(x)u_c - d - D + \alpha\dot{e}_1 + \lambda_1\beta {e_1}^{\lambda_1-1}\dot{e}_1\right)\end{aligned} \tag{18}$$

Bringing 17 into 18 yields

$$\begin{aligned}\dot{V}_2 &= S\dot{S}\\ &= S\left(-d - k_2S - k_3|S|^{\gamma_1}\operatorname{sgn}(S)\right)\\ &= -k_2S^2 - k_3|S|^{\gamma_1+1} - Sd\\ &\le -k_2S^2 - k_3|S|^{\gamma_1+1} + |S|\,\bar{d}\\ &= -k_2S^2 - k_3|S|^{\gamma_1+1} + |S|\,\bar{d}\\ &= -k_2S^2 - |S|\left(k_3|S|^{\gamma_1} - \bar{d}\right)\end{aligned} \tag{19}$$

When $k_3 > \bar{d}/|S|^{\gamma_1}$, $k_3|S|^{\gamma_1} - \bar{d} = \varepsilon$, $\varepsilon > 0$, thus:

$$\dot{V}_2 \le -2k_2V_2 - \varepsilon\,|S| \le -2k_2V_2 - \sqrt{2}\varepsilon V_2^{1/2} < -\alpha_1V_2^{1/2} - \beta_1V_2 \tag{20}$$

where $\alpha_1 = 2k_2$, $0 < \beta_1 < \sqrt{2}\varepsilon$.

LEMMA 1 [44] (Jiang and Lin, 2020): Consider a smooth positive definite $V(x)$, $x \in R_n$. Suppose that real numbers $p_1 \in (0,1)$, $\alpha > 0$, and $\beta > 0$ exist such that $V(x) < -\alpha V(x)^{p_1} - \beta V(x)$. Then, an area $U_0 \in R_n$ exists, such that any $V(x)$ starting from U_0 can reach $V(x) = 0$ in finite time T_v, which is expressed as $T_v \le \frac{1}{\beta(1-p_1)}\ln\left(\frac{V^{1-p_1}(x_0)+\alpha}{\alpha}\right)$.

According to **lemma 1**, V_2 can converge to 0 in finite time. In the above discussion, the uncertainty term D is assumed to be known, but the actual uncertain term D is unknown. As RBFNN can approximate arbitrary uncertain non-linear functions and does not depend on a mathematical model, it is more suitable for estimating stochastic uncertain terms. Therefore, optimal neural network weights w^* must exist such that $D = \varepsilon_0 + {w^*}^Th$, ε_0 is the estimated residual and h is the neuron. $\tilde{w} = \hat{w} - w^*$, $\hat{w}$ is an estimate of w^* and w^* is a constant, so $\dot{\tilde{w}} = \dot{\hat{w}}$. Rewrite 9 as:

$$\begin{cases}\dot{x}_1 = x_2\\ \dot{x}_2 = f(x) + g(x)u_c + d + \varepsilon_0 + {w^*}^Th\\ y = x_1\end{cases} \tag{21}$$

Step 3: Establish the Lyapunov function V_3 as

$$V_3 = \frac{1}{2}S^2 + \frac{1}{2}tr(\tilde{w}^T\Gamma^{-1}\tilde{w}) \tag{22}$$

The derivation of formula 22 yields

$$\begin{aligned}V_3 &= S\dot{S} + \tilde{w}^T\Gamma^{-1}\dot{\hat{w}}\\ &= S\left(\dot{\alpha}_x - f(x) - g(x)u_c - d - \varepsilon_0 - {w^*}^Th + \alpha\dot{e}_1 + \lambda_1\beta e_1^{\lambda_1-1}\dot{e}_1\right)\\ &\quad + \tilde{w}^T\Gamma^{-1}\dot{\hat{w}}\end{aligned} \tag{23}$$

The control law is designed to

$$u_c = \frac{1}{g(x)}\left(\dot{\alpha}_x - f(x) - \hat{w}^Th + k_2S + k_3|S|^{\gamma_1}\operatorname{sgn}(S)\right) \tag{24}$$

Bringing formula 24 into 23 yields

$$\dot{V}_3 = -k_2S^2 - k_3|S|^{\gamma_1+1} - S\varepsilon_1 + \tilde{w}^T(Sh + \Gamma^{-1}\dot{\tilde{w}}) \tag{25}$$

where $\varepsilon_1 = d + \varepsilon_0$, the upper limit of the estimation error of the neural network is $\bar{\varepsilon}_0$. $\bar{\varepsilon}_0 \ge \varepsilon_0$, $\bar{d} \ge d$, so that $\varepsilon_1 \le \bar{d} + \bar{\varepsilon}_0 = \bar{\varepsilon}_1$. The update law of the RBFNN weights is designed as

$$\dot{\hat{w}} = -\Gamma Sh \tag{26}$$

Bringing 26 into 25 yields

$$\begin{aligned}\dot{V}_3 &= -k_2S^2 - k_3|S|^{\gamma_1+1} - S\varepsilon_1\\ &\le -k_2S^2 - k_3|S|^{\gamma_1+1} + |S|\,\bar{\varepsilon}_1\\ &= -k_2S^2 - |S|\left(k_3|S|^{\gamma_1} - \bar{\varepsilon}_1\right)\end{aligned} \tag{27}$$

when $k_3 > \bar{\varepsilon}/|S|^{\gamma_1}$, $k_3|S|^{\gamma_1} - \bar{\varepsilon} = \varepsilon_2$, where $\varepsilon_2 > 0$, thus:

$$\begin{aligned}\dot{V}_3 &\le -2k_2V_2 - \varepsilon_2\,|S| \le -2k_2V_2 - \sqrt{2}\varepsilon_2V_2^{1/2}\\ &< -\alpha_1V_2^{1/2} - \beta_1V_2 < 0\end{aligned} \tag{28}$$

According to **lemma 1**, V_2 can converge to 0 in finite time.

The control input u_c in formula 24 is the unconstrained, to prevent the control input saturation, define $u_d = u_c$, where u_d is the desired value in the next step, and the state error $e_3 = u_d - u_{con}$. u_{con} satisfies the constrained control input of the saturation function *tanh*; therefore, parameter u_f must exist, such that $u_{con} = u_{\max}\tanh(u_f/u_{\max})$, where u_{max} is the maximum input.

$$\dot{u}_{con} = \left(1 - \tanh^2(u_f/u_{\max})\right)\dot{u}_f \tag{29}$$

Step 4: Establish the Lyapunov function $V_4 = \frac{1}{2}e_3^2$ and derive V_3 and bring it into 29 to obtain:

$$\begin{aligned}\dot{V}_4 &= e_3\dot{e}_3\\ &= e_3(\dot{u}_{\mathrm{d}} - \dot{u}_{con})\\ &= e_3\left(\dot{u}_d - \left(1 - \tanh^2(u_f/u_{\max})\right)\dot{u}_f\right)\end{aligned} \tag{30}$$

$\dot{u}_f$ is designed as

$$\dot{u}_f = \begin{cases}\left(k_4e_3 + |e_3|^{\gamma_2}sgn(e_3) + \dot{u}_d\right)/\left(1 - \tanh^2(u_f/u_{\max})\right), \delta \ge \Delta\\ |\delta e_3|^{\gamma_2}sgn(e_3) + \dot{u}_d/\left(1 - \tanh^2(u_f/u_{\max})\right), \delta < \Delta\end{cases} \tag{31}$$

where $\delta = |u_f| - 2u_{\max}$, Δ is a smaller normal value. $\gamma_2 \in (0,1)$. The convergence of the controller is discussed in the following cases. When $\delta \ge \Delta$, substituting 31 into 30 yields

$$\begin{aligned}\dot{V}_4 &= -k_4e_3^2 - |e_3|^{\gamma_2+1} = -2k_4V_3 - 2^{(\gamma_2+1)/2}V_3^{(\gamma_2+1)/2}\\ &< -\alpha_2V_4^{(\gamma_2+1)/2} - \beta_2V_4\end{aligned} \tag{32}$$

where $0 < \alpha_2 < 2^{(\gamma_2+1)/2}$, $2k_3 = \beta_2$. According to **Lemma 1**, V_4 can converge to 0 in finite time. When $\delta < \Delta$, substituting 31 into 30 yields

$$\begin{aligned}\dot{V}_4 &= -\left(|\delta|^{\gamma_2}|e_3|^{\gamma_2+1}\right)/\left(1-\tanh^2(u_c/u_{\max})\right)\\ &= -\left(|\delta|^{\gamma_2}2^{(\gamma_2+1)/2}/\left(1-\tanh^2(u_c/u_{\max})\right)\right)V_4^{\alpha_3}\\ &= -cV_4^{\alpha_3}\end{aligned} \tag{33}$$

where $\alpha_3 = (\gamma_2+1)/2$, $c = |\delta|^{\gamma_2}2^{(\gamma_2+1)/2}/\left(1-\tanh^2(u_c/u_{\max})\right)$, and $\tanh(u_c/u_{\max}) < 1$, so $c > 0$. According to **Lemma 2**, V_4 can converge in finite time.

LEMMA 2: Chu et al. (2022) Suppose that there is a positive definite continuous Lyapunov function $V(x,t)$ defined on $U_1 \times R^+$, where $U_1 \subseteq U \subseteq R_n$. R_n is a neighborhood of the origin, and $V(x,t) \leq -cV^{\alpha}(x,t), \forall x \in U_1 \setminus \{0\}$, where $c > 0$, $0 < \alpha < 1$. Then, the origin of the system is locally finite time stable. The settling time $T \leq V^{1-\alpha}\left(x(t_0),t_0\right)/c(1-\alpha)$ satisfies for a given initial condition $x(t_0) \in U_1$.

3.2. Human decision search algorithm

The human decision search algorithm (HDSA) is a swarm optimization technique that mimics the decision-making process of a human crowd. In many post-apocalyptic survival games or films, the strong group consciousness of humans is often portrayed, but the importance of individual consciousness is also emphasized. In human groups, a small group of individuals called decision-makers make the final decisions based on their experience and personal status. However, the decision of the decision-maker is not necessarily optimal. When the number of individuals in the group is small, it is important to involve more people in the decision-making process to guide the development of the group and to avoid the excessive impact of individual decisions on the group. However, when the number of individuals in the group is large, the proportion of decision-makers should be reduced and only a few elite individuals should be selected to determine the development of the group. This is because too many people involved in the decision-making process may take more time, and the experience of ordinary people may not be as good as that of elite individuals. Because people have emotions, they can think both rationally and emotionally when dealing with problems, and these two opposing ways of thinking must coexist.

Apart from the decision-makers, the rest of the human population is referred to as the executors, consisting of individuals who have no or less ability to make decisions. They carry out the optimal decisions made by the decision-makers. However, individuals among the executors who have some decision-making ability should be encouraged to seek more humane decisions based on the optimal decisions. These decisions should become more adapted to the current environment over time. The number of decision-makers is fixed, and elite individuals in the human population will always be selected as decision-makers. Over time, any individual has the potential to become a decision-maker, and the current decision-maker may become an executor.

In a human population, there are always individuals who question the current decision or believe they have a better one, including the decision-makers themselves. These individuals are known as adventurers, and their numbers and identities are random, making them a source of uncertainty within the population. Although adventurers can lead people to a better life, they can also lead them to disaster. Adventurers, on the other hand, inherit the current optimal choices of the human population and take them into account when making decisions. However, more adventurous individuals will also seek out possible optimal decisions based on their own state. To avoid harming the human population, adventurers must consider whether the decisions they make are more beneficial to their own survival. Additionally, there is a chance that an adventurer will become a decision-maker if they come up with a better or suboptimal decision. Based on the above analysis, the proposed algorithm for optimizing the human decision population consists of three main components: decision updating for decision-makers, decision updating for executors, and decision updating for adventurers.

3.2.1. Decision updates for decision makers

The number of decision-makers is fixed in proportion to the total number of people, and the number of decision-makers is 20–50% of the total number of people. The decision-makers make their decisions based on individual experience as well as individual characteristics. The sine and cosine functions are used to distinguish between rational and emotional decisions by people, and the individuals are randomly updated due to the random adoption of rational and emotional decisions by people.

$$x_i^{t+1} = \begin{cases} r_1 x_i^t \sin\left(r_2\left|r_3 x_{ibest}^t - x_i^t\right|\right), R < 0.5 \\ r_1 x_i^t \cos\left(r_2\left|r_3 x_{ibest}^t - x_i^t\right|\right), R \geq 0.5 \end{cases} \tag{34}$$

where x_i^t denotes the t_{th} iteration of the i_{th} human individual. r_1 is a non-linear term, $r_1 = 2*\left(1 - i/(\alpha_1 * d_{num})\right)$. d_{num} is the number of decision-makers. α_1 is a random number between $(0,1)$. $r_2 = \alpha_2 2\pi$ and α_2 is the random number between $(0,1)$. $r_3 = 2\alpha_3$, α_3 is a random number between $(0,1)$. r is the random number between $(0,1)$. x_{ibest}^t is the individual optimal solution for 1 to t iterations.

3.2.2. Decision updates for executors

Except for the decision-maker, the rest of the individuals are the executors. Among the executors, individuals with a fitness that is higher than the intermediate fitness are ordinary executors that must follow the optimal decision of the decision-maker. Individuals with a fitness below the intermediate fitness are considered as executors with some decision-making ability, and this group can continue to explore the next optimal decision that may exist based on the current optimal decision.

$$x_i^{t+1} = \begin{cases} x_{best}^t + \beta_1\left|\left(x_i^t - x_m^t\right)/\left(f_i^t - f_m^t\right)\right|, f_i^t > f_m^t \\ sgn(x_e^t)exp\left(\left|x_{best}^t - x_i^t\right|/\beta_2\right), f_i^t \leq f_m^t \end{cases} \tag{35}$$

where x_{best}^t is the current global best individual and x_{worst}^t is the current global worst individual. $x_e^t = x_{best}^t - x_{worst}^t$. f_i^t is the fitness of the i_{th} individual, $f_m^t = \left(f_{best}^t + f_{worst}^t\right)/2$, f_{best}^t is the current best fitness, and f_{worst}^t is the current worst fitness. β_1 is the random number of normal distribution with mean 0 and variance 1. The sgn function determines the direction of exploration of individuals. $\beta_2 = t^2/f^t_{best}$ indicates that a more favorable decision result can be obtained over time.

3.2.3. Decision updates for adventurers

The adventurers are random individuals and the number of adventurers is also random. If the adventurer's fitness is less than the average fitness, the adventurer randomly explores based on the current optimal solution. If the adventurer's fitness is higher than the average fitness, the adventurer will continue to explore in the optimal direction according to the current state of the individual.

$$x_i^{t+1} = \begin{cases} x^t_{best} + c_1 \left| x^t_{best} - x^t_i \right|, f^t_i > f^t_{avr} \\ x^t_i + (2c_2 - 1)\left|\left| x^t_e \right|\right|_2 sgn(x^t_e), f^t_i \le f^t_{avr} \end{cases} \tag{36}$$

where c_1 is a normally distributed random number with mean 0. c_2 is a random number between $(0, 1)$ with variance 1. $\|x^t_e\|_2$ is the Euclidean norm of x^t_e and f^t_{avr} is the current mean fitness.

Based on the above discussion, the proposed HDSA has three steps. The first step performs a global random search using the formula 34. In the second step, a local search is performed based on the first step using the formula 35. The third step performs a second global random search using the formula 36 on the basis of the first and second steps. HDSA framework as Algorithm 1.

```
   Input: input parameters $M$, $d_n$, $pop$, $dim$, $l_b$, and $u_b$
   Output: output $x_{best}$, $f_{min}$
 1 Initialize individuals, constrain the upper and
     lower bounds of individuals, calculate the
     individual fitness, initialize the global
     optimal solution $x_{best}$ and the optimal fitness
     $f_{min}$;
 2 while $t < M$ do
 3   Find the current global optimal solution $x_{best}$
       and the current individual optimal solution
       $x_{ibest}$ $r = rand(1)$
 4   for $i : d_n$ do
 5     Use 34 to update the decision-maker's
         decision and calculate the individual
         fitness
 6   Find the current global optimal solution $x_{best}$
       and the global worst solution $x_{worst}$, compute
       the intermediate solution and intermediate
       solution fitness.
 7   for $d_n + 1 : pop$ do
 8     Use 35 to update the executor's decision and
         calculate the individual fitness
 9   Calculate the average fitness, randomly select
       $a_n$ individuals
10   for $1 : a_n$ do
11     Use 36 to update the adventurer's decision
         and calculate the individual fitness
12   Find the current global optimal solution $x_{best}$,
       the individual optimal solution $x_{ibest}$, and the
       optimal fitness $f_{min}$.
13   $t = t + 1$
14 return $x_{best}$, $f_{min}$
```

Algorithm 1. HDSA.

3.3. Yaw controller and linear velocity controller

According to the control algorithm in the "RBFNN-Based Active Fault-Tolerant Control Algorithm" section, the AFTC is used to design controllers in this section to follow the desired yaw angle ψ_d and desired linear velocity v_d. The robot linear velocity sliding mode surface is: $S_v = \alpha_v e_v + \beta_v {e_v}^{\lambda_v}$, where $e_v = v_d - v$. The sliding mode convergence law is $\dot{S}_v = -k_{2v} S_v - k_{3v}|S|^{\gamma_{1v}} \text{sgn}(S_v)$.

The proof of convergence for the velocity controller is similar to that for the general-purpose controller in the "RBFNN-Based Active Fault-Tolerant Control Algorithm" section. The unconstrained control law is designed as

$$F_{uc} = m\left(\dot{v}_d - \hat{w}_v^T h_v + k_{2v} S_v + k_{3v}|S_v|^{\gamma_{1v}} \text{sgn}(S_v)\right) \tag{37}$$

The anti-input saturation controller of linear velocity is designed as

$$\begin{cases} F_{uf} = \begin{cases} \int \left(k_{4v} e_F + |e_F|^{\gamma_{2v}} sgn(e_F) + \dot{F}_{uc}\right) \\ \quad / \left(1 - \tanh^2(F_{uf}/F_{\max})\right) dt \quad , \delta_v \ge \Delta_v \\ \int |\delta_v e_F|^{\gamma_{2v}} sgn(e_F) + \dot{F}_{uc} / \left(1 - \tanh^2(F_{uf}/F_{\max})\right) dt \\ \quad , \delta_v < \Delta_v \end{cases} \\ F_{ucon} = F_{\max} \tanh(F_{uf}/F_{\max}) \end{cases} \tag{38}$$

Where $e_F = F_{uc} - F_{ucon}$.

The yaw angle controller is $\omega_d = k_\psi e_\psi + \dot{\psi}_d$, where $e_\psi = \psi_d - \psi$. The yaw angle sliding mode surface is $S_\omega = e_\omega + \alpha_\psi e_\psi + \beta_\psi {e_\psi}^{\lambda_\psi}$. The sliding mode convergence law is $\dot{S}_\omega = -k_{2\omega} S_\omega - k_{3\omega}|S_\omega|^{\gamma_{1\omega}} \text{sgn}(S_\omega)$.

The unconstrained control law is designed as

$$T_{rc} = I\left(\dot{\omega}_d - \hat{w}_\omega^T h_\omega + k_{2\omega} S_\omega + k_{3\omega}|S_\omega|^{\gamma_{1\omega}} \text{sgn}(S_\omega)\right) \tag{39}$$

The anti-input saturation controller of the yaw angle is designed as

$$\begin{cases} T_{rf} = \begin{cases} \int \left(k_{4\omega} e_T + |e_T|^{\gamma_{2\omega}} sgn(e_T) + \dot{T}_{rc}\right) \\ \quad / \left(1 - \tanh^2(T_{rf}/T_{\max})\right) dt \quad , \delta_\omega \ge \Delta_\omega \\ \int |\delta_\omega e_T|^{\gamma_{2\omega}} sgn(e_T) + \dot{T}_{rc} / \left(1 - \tanh^2(T_{rf}/T_{\max})\right) dt, \\ \quad \delta_\omega < \Delta_\omega \end{cases} \\ T_{rcon} = T_{\max} \tanh(T_{rf}/T_{\max}) \end{cases} \tag{40}$$

where $e_T = T_{rc} - T_{rcon}$. The controller parameters are not described in this section as they have been discussed in the "RBFNN-Based Active Fault-Tolerant Control Algorithm" section.

The input to the angular velocity neural network is both the yaw error and the angular velocity error, and the output is the

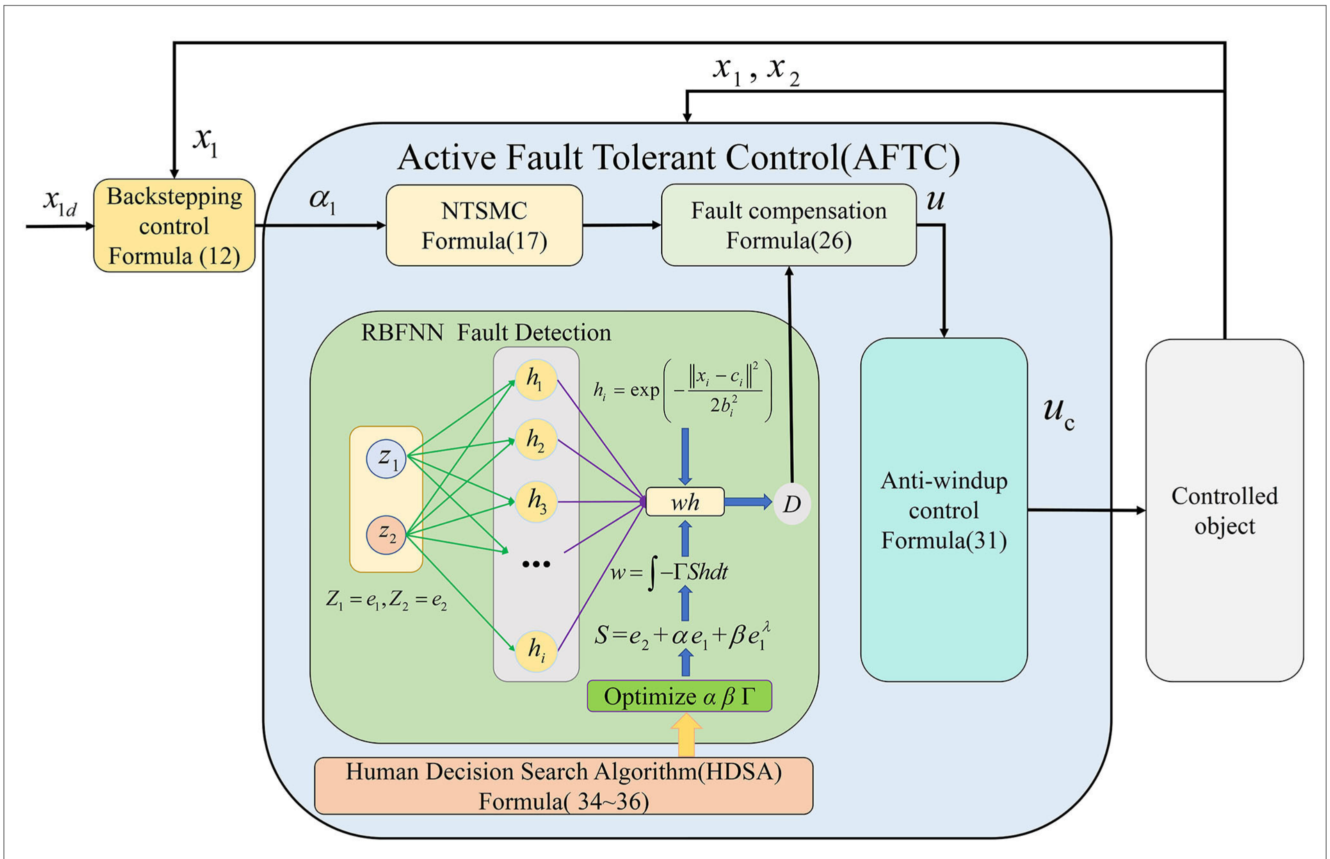
x_1, x_2
x_1
x_{1d}
Backstepping control Formula (12)
α_1
Active Fault Tolerant Control(AFTC)
NTSMC Formula(17)
Fault compensation Formula(26)
u
RBFNN Fault Detection
h_1
h_2
h_3
h_i
z_1
z_2
$Z_1 = e_1, Z_2 = e_2$
$h_i = \exp\left(-\frac{\|x_i - c_i\|^2}{2b_i^2}\right)$
wh
D
$w = \int -\Gamma S h dt$
$S = e_2 + \alpha e_1 + \beta e_1^{\lambda}$
Optimize $\alpha\ \beta\ \Gamma$
Human Decision Search Algorithm(HDSA) Formula(34~36)
Anti-windup control Formula(31)
u_c
Controlled object

FIGURE 6
AFTC framework.

uncertainty term in the angular velocity control. The coordinate vector matrix of the centroids of the Gaussian basis function neurons in the angular velocity neural network is

$$c_\psi = \begin{bmatrix} -1.6 & -0.8 & -0.4 & -0.2 & -0.1 & 0 & 0.1 & 0.2 & 0.4 & 0.8 & 1.6 \\ -1.6 & -0.8 & -0.4 & -0.2 & -0.1 & 0 & 0.1 & 0.2 & 0.4 & 0.8 & 1.6 \end{bmatrix}_{2*11}$$

The width of the Gaussian basis function $b_\psi = 0.1, i = 1 \cdots 11$.

The input to the linear velocity neural network is the velocity error and the output is the linear velocity control uncertainty term. The coordinate vector matrix of the centroids of the Gaussian basis function of the neurons in the linear velocity neural network is $c_v = \begin{bmatrix} -1.6 & -0.8 & -0.4 & -0.2 & -0.1 & 0 & 0.1 & 0.2 & 0.4 & 0.8 & 1.6 \end{bmatrix}_{1*11}$. The width of the Gaussian basis function $b_v = 0.1, i = 1 \cdots 11$.

Based on the above discussion, the proposed framework for the AFTC is shown in Figure 6.

4. Simulation results

In the section entitled "HDSA's Related Work", we have demonstrated the advantages of the proposed HDSA; therefore, in this section, the HDSA is used to optimize the sliding mode surface parameters of the yaw controller and the linear velocity controller. As the weight update parameters of the RBFNNs are related to the sliding mode parameters, this also indirectly optimizes the RBFNNs.

The parameters to be optimized for yaw angle control are the sliding mode surface coefficients α_ω , β_ω and the neural network update coefficient Γ_ω. According to the idea of AFTC, the presence of $-3N.m$ of disturbance torque in the robot model simulates the worst case. The initialized optimization algorithm parameters are as follows: dimension is 3, the number of populations is 20, the number of max iterations is 10, and the upper limit of parameters is 20 and the lower limit is -20.

The evaluation function of the yaw controller is designed as $f_{obj} = 0.8 * |e_\psi| + 0.1 * |e_\omega| + 0.01 * |T_{rc}|$. For yaw control, we want to reduce both the yaw error and the yaw velocity error with the smallest control input. As the control objective is to eliminate the yaw error, the yaw error is given the largest weight in the evaluation function. To keep the control input and yaw error in the same order, the control input weight is reduced. The optimization parameters for the yaw controller are shown in Figure 7.

As shown in Figure 7, the optimized parameters converge after eight iterations. The values of $\Gamma_\omega = 20$, $\alpha_\omega = 7.4407$, and $\beta_\omega = 2.9369$ are obtained through the optimization process.

The optimized parameters are substituted into the AFTC and the control results are compared with the unoptimized AFTC, NTSMC, and SMC. Before 10 s, the yaw angle is influenced by a torque with a mean value of $-1N.m$ and a mean square error of 0.1. After 10 s, the yaw angle is influenced by a torque with a mean value of $-3N.m$ and a mean square error of 0.1. The control parameters are given in Table 3.

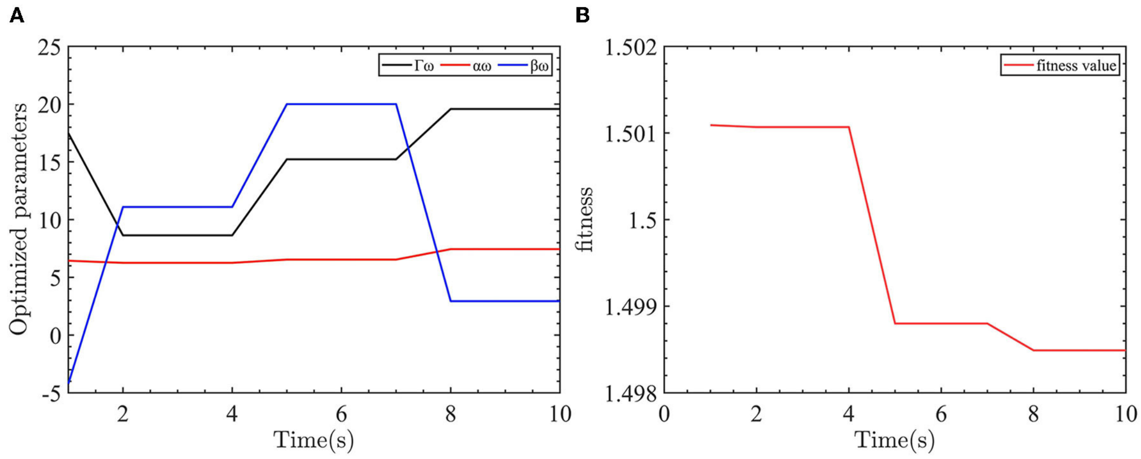

FIGURE 7
Yaw control parameter optimization and fitness of the yaw controller objective function. **(A)** The optimized parameters of yaw controller. **(B)** The objective function output value.

TABLE 3 Parameters of yaw angle controllers.

Controllers	Parameters	Value
Proposed AFTC	$k_{1\psi}$	2
	$\alpha_\omega, \beta_\omega, \lambda_\omega$	1, 2, 3
	$k_{2\omega}, k_{3\omega}, \gamma_\omega$	1, 5, 0.5
	$k_{4\omega}$	5
	Γ_ω	10
NTSMC	$k_{1\psi}$	2
	$\alpha_\omega, \beta_\omega, \lambda_\omega$	1, 2, 3
	$k_{2\omega}, k_{3\omega}, \gamma_\omega$	5, 20, 0.5
SMC	$k_{1\psi}$	2
	$k_{2\omega}, k_{3\omega}$	5, 5

The results of the yaw angle controller are shown in Figure 8.

In Figure 8A, the optimized AFTC has a significantly faster response speed (pink line). Despite being influenced by a $-1\ N.m$ torque disturbance in the range of 0–10 s, the AFTC, NTSMC (green line), and SMC (red line) maintain their robustness and are not affected by the disturbance. After 10 s, the yaw angle is subjected to a torque of $-3N.m$, in which case reliance on the robustness of the controller can no longer guarantee yaw angle control performance, as shown in the 10–11 s enlargement in Figure 8A. The SMC is unable to follow the desired yaw angle with a static error of $\sim$0.05 *rad*, and the NTSMC also has a small static difference.

As shown in Figure 8B, the proposed AFTC (pink line) and the optimized AFTC (orange line) do not enter the driver saturation state. The NTSMC (purple line) and the SMC (green line) enter the driver saturation state. Compared with the conventional SMC (green line) and NTSMC (purple line) control inputs, which have high-frequency input chatter, the control input of the proposed AFTC is more stable. This suggests that the robustness achieved by the conventional SMC comes at the expense of control input performance. In Figure 8C, the output of the radial basis function neural network (RBFNN) is displayed, showing a value of 1 before 10 s and 3 after 10 s. The RBFNN can estimate the unknown yaw disturbances online. The RBFNN weights are updated accordingly, as shown in Figure 8.

The parameters to be optimized for the velocity controller are the sliding mode surface coefficients α_v and β_v and the neural network update coefficients Γ_v. The presence of $-5N$ force in the robot model simulates the worst case. The initialized optimization algorithm parameters are as follows: the dimension is 3, the number of populations is 20, the number of maximum iterations is 10, and the upper limit of parameters 20 and the lower limit is 20.

The evaluation function is designed as $f_{obj} = 0.8 * |e_v| + 0.02 * |F_{uc}|$. When controlling the linear velocity, we want to minimize the linear velocity error with the smallest control input. Therefore, the linear velocity error has the largest weight in the evaluation function. The weight of the control input is reduced to keep the control input and the linear velocity error at the same level. The linear velocity controller optimization parameters are shown in Figure 9.

As shown in Figure 9, the optimization parameters converge after two iterations. The optimized parameters are $\Gamma_v = 15.6467$, $\alpha_v = 16.1866$, and $\beta_v = 20$.

These parameters are used in the proposed AFTC, and the control results are compared and analyzed with the unoptimized AFTC, NTSMC, and SMC controllers. Before 10 s, the linear velocity is affected by a force with a mean value of $-2N$ and a mean square error of 0.1. After 10 s, the velocity is influenced by a force with a mean value of $-5N$ and a mean square error of 0.1. The velocity controller parameters are given in Table 4.

The control results of linear velocity controllers are shown in Figure 10.

Similar to the performance of the yaw control, in Figure 10A, the optimized AFTC (pink line) responds faster compared with the proposed AFTC (purple line) and SMC (red line). Between 0 and 10 s, when the line speed is subjected to -2N force, AFTC (purple line), NTSMC (green line), and SMC (red line) are not affected

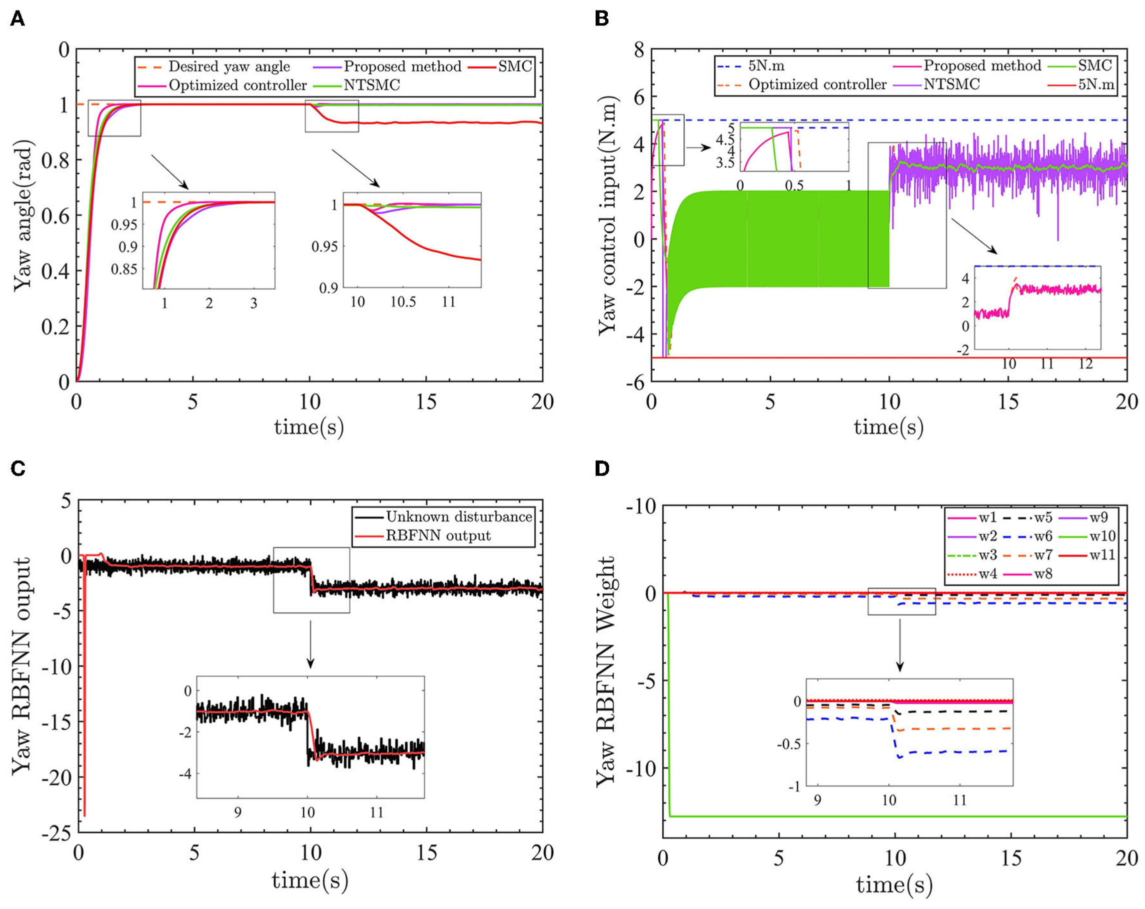

FIGURE 8
(A) The yaw angle control results. **(B)** Control input torque. **(C)** Yaw angle RBFNN output value. **(D)** Yaw angle RBFNN weight.

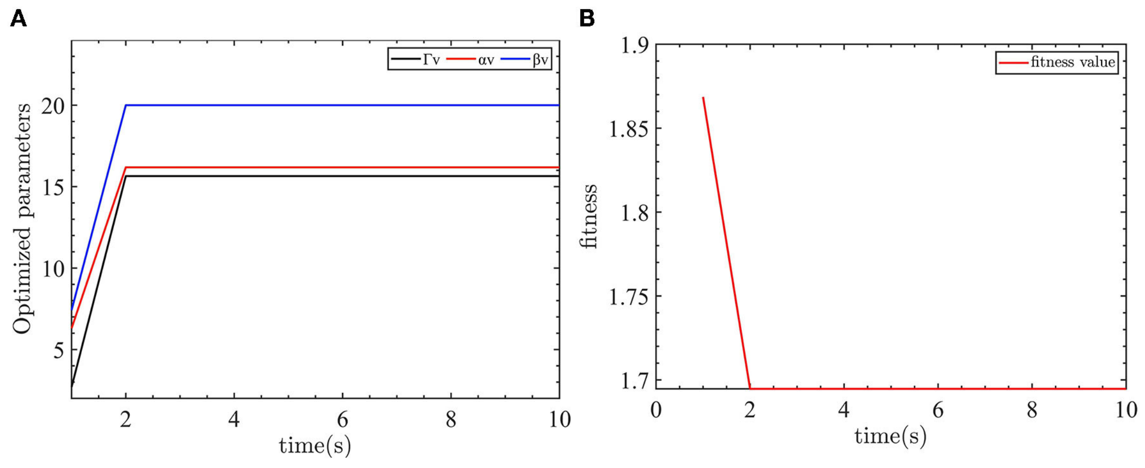

FIGURE 9
Velocity control parameter optimization and fitness of the velocity controller objective function. **(A)** The optimized parameters of velocity controller. **(B)** The objective function output value.

by the disturbances. After 10 s, the linear velocity is subjected to a force of $-5N$ and the velocity control performance cannot be guaranteed by the NTSMC and SMC. There is a static error of $\sim 0.05m/s$ for the NTSMC and $\sim 0.6m/s$ for the SMC, as shown in the 9–12 s enlargement in Figure 10A. Both the proposed AFTC and the optimized AFTC can follow the desired linear velocity, and the velocity controller is almost unaffected by the $-5N$ force using the optimized parameters. The proposed AFTC and the optimized AFTC can effectively track the desired linear velocity, with minimal impact from the $-5N$ force disturbance. The velocity controller of the AFTC is almost unaffected by the disturbance, indicating its robustness and ability to maintain precise control performance.

The previous discussion has highlighted the improved responsiveness and robustness of the optimized AFTC. To further emphasize the advantages of the optimized AFTC, the output value of the evaluation function is used as a criterion to evaluate the performance of the four controllers. A smaller output value of the evaluation function indicates better controller performance. The output values of the evaluation functions for the four controllers are depicted in Figure 11.

As shown by the green lines in Figures 12A, B, the optimized AFTC controller exhibits the smallest value of the evaluation function. This signifies that the optimized AFTC achieves the best performance among the four controllers. As the linear velocity and yaw angle are consistently subjected to external disturbances, the output value of the evaluation function continually increases. This is because of the fact that the control inputs are not equal to zero. In the case of large external disturbances, the NTSMC and SMC controllers can no longer eliminate the yaw angle error and the linear velocity error. Consequently, the output value of the evaluation function rapidly increases, as indicated by the red and blue lines.

To further verify the effectiveness of the proposed algorithm, the AFTC is used to design the yaw angle controller and the velocity controller. The desired yaw angle and the desired linear velocity is planned by the LOS algorithm. The optimized parameters are selected as the controller's parameters. The LOS algorithm

TABLE 4 The parameters of velocity controllers.

Controllers	Parameters	Value
Proposed AFTC	$\alpha_v, \beta_v, \lambda_v$	1, 2, 3
	k_{2v}, k_{3v}, γ_v	1, 5, 0.5
	k_{4v}	5
	Γ_v	10
NTSMC	$\alpha_v, \beta_v, \lambda_v$	1, 2, 3
	k_{2v}, k_{3v}, γ_v	5, 20, 0.5
SMC	k_{2v}, k_{3v}	5, 5

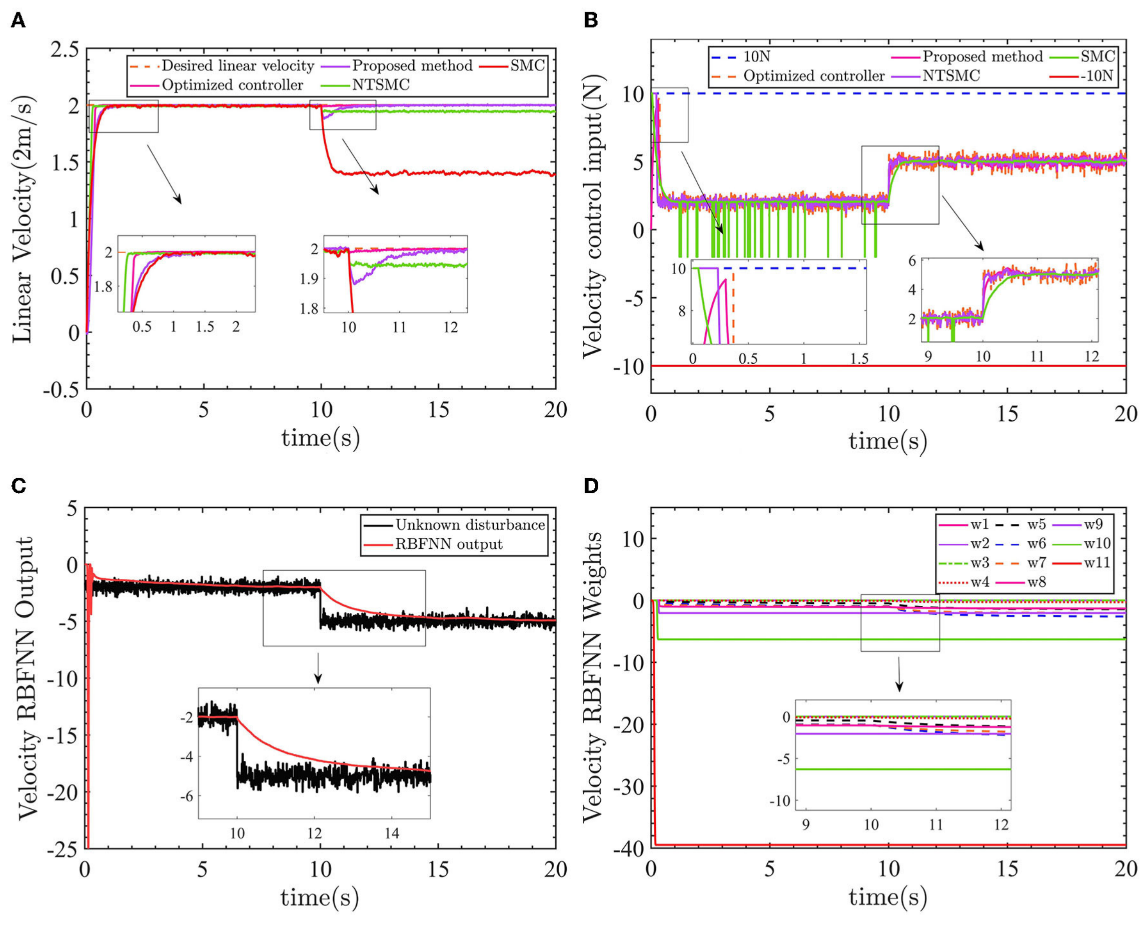

FIGURE 10
Linear velocity control results. **(A)** Velocity control results. **(B)** Control input force. **(C)** Velocity RBFNN output value. **(D)** Velocity RBFNN weight.

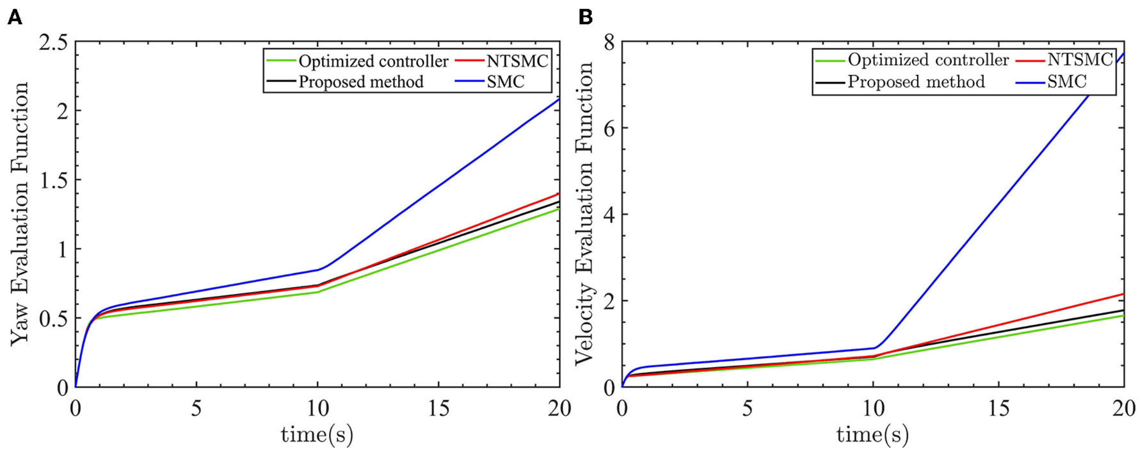

FIGURE 11
Four control evaluation function outputs. **(A)** Yaw angle evaluation function outputs. **(B)** Velocity evaluation function outputs.

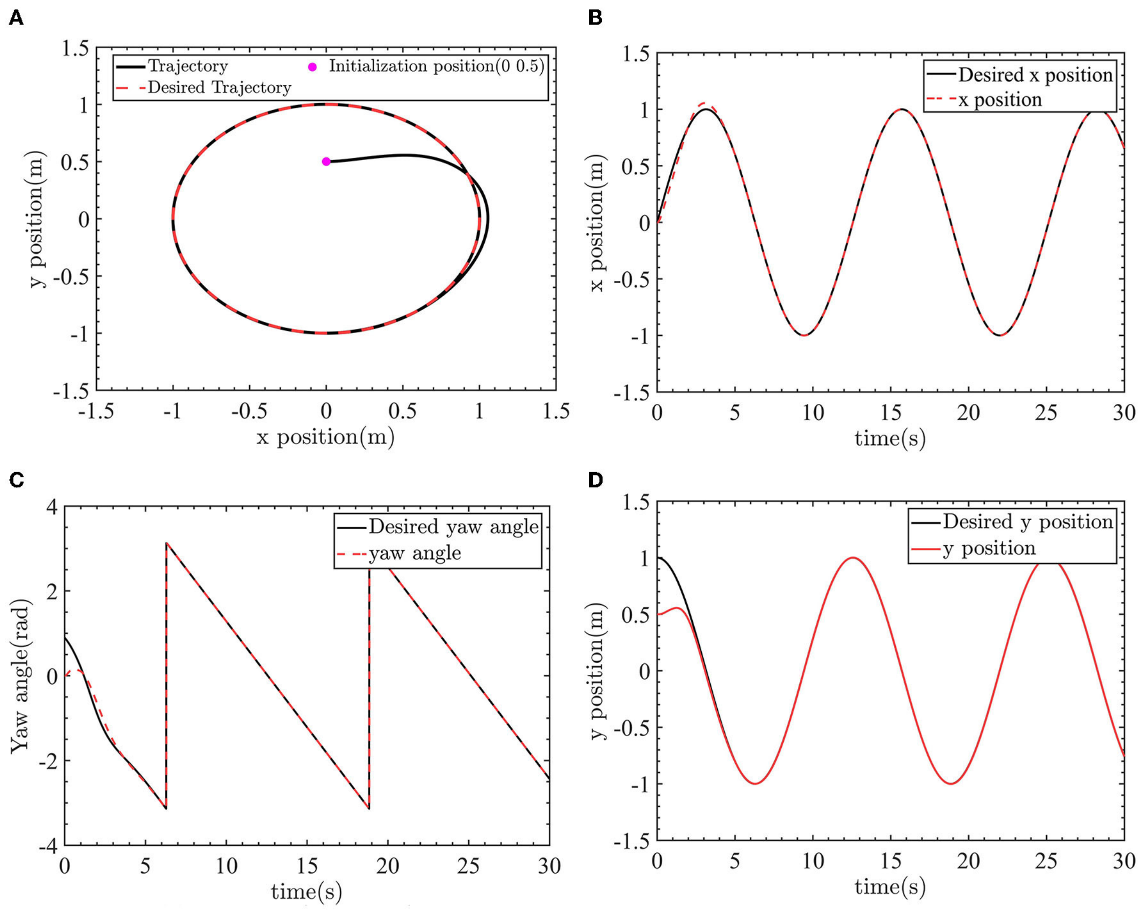

FIGURE 12
The robot tracks the desired trajectory. **(A)** Tracking the circle desired trajectory. **(B)** X-position control. **(C)** Yaw angle control. **(D)** Y-position control.

and the improved LOS algorithm can be found in the author's previous work (Wang et al., 2022b). The desired trajectory is a circular trajectory with radius $R = 1m$, angular velocity $\omega_r = 0.5rad/s$, and linear velocity $v_r = 0.5m/s$. The initial position and pose of the robot is $\left[0\text{m}, 0.5\text{m}, 0\text{rad}\right]$. A drag force of $-2N$ and a torque of $-1N.m$ are applied to the robot. The

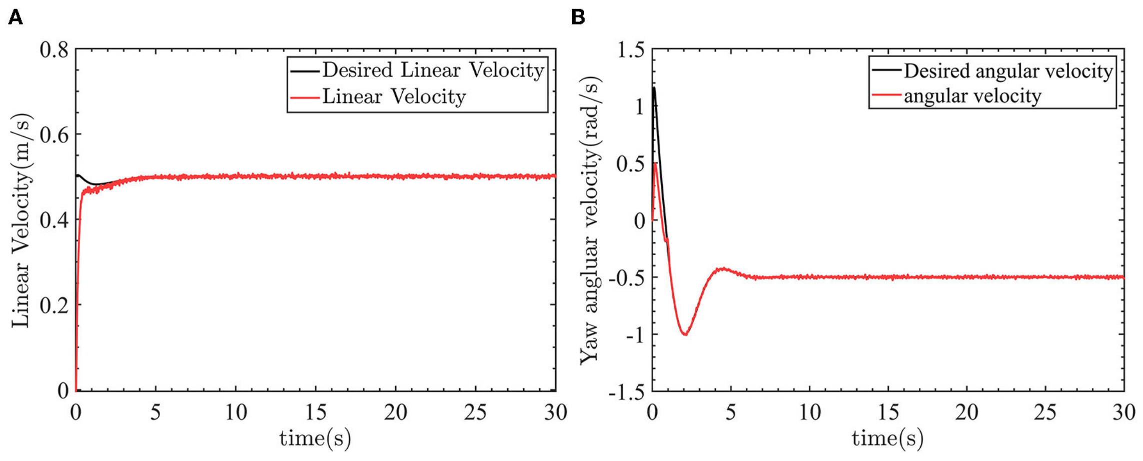

FIGURE 13
The control results of linear velocity and yaw angular velocity. **(A)** Linear velocity control. **(B)** Yaw angle velocity control.

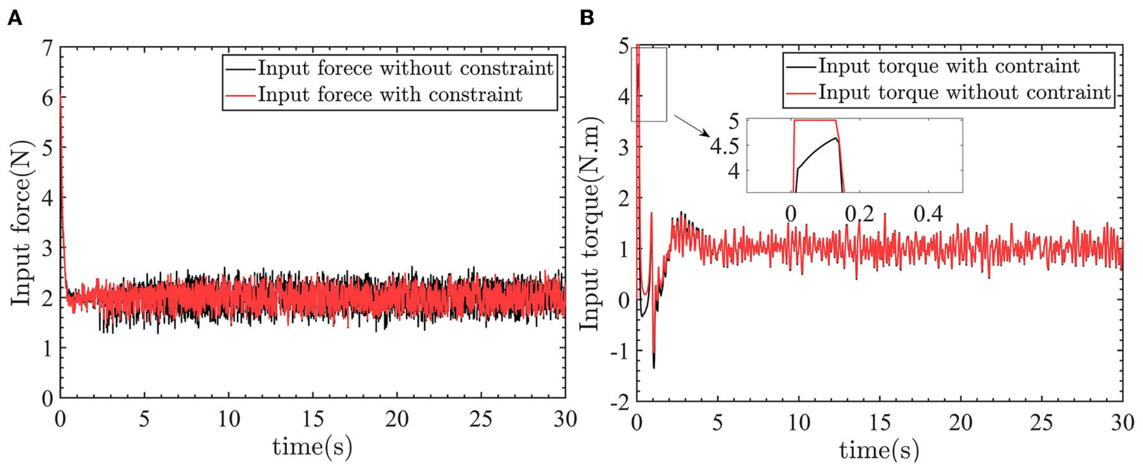

FIGURE 14
The linear velocity control input and yaw angular velocity control input. **(A)** Control input force. **(B)** Control input torque.

LOS algorithm is

$$\begin{cases} \psi_L = \psi_r - \alpha \\ \alpha = \arctan(e_y/\Delta) \\ v_L = v_r + ke_x \end{cases} \tag{41}$$

where ψ_L, v_L are the desired yaw angle and desired linear velocity planned by the LOS algorithm. e_x, e_y is the position error in Frenet-Serret (F-S) frame. Δ and k are the positive adjustable parameters.

The control results of the robot tracking the desired circle trajectory are shown as Figures 12–14. The robot position control and yaw angle control are shown in Figure 12.

The robot can track the desired trajectory. The actual position pose of the robot is consistent with the desired position pose. The linear velocity control and angular velocity control are shown in Figure 13.

In Figure 13A, the linear velocity can track the desired linear velocity of $0.5m/s$. In Figure 13B, the angular velocity can track the desired angular velocity of $-0.5rad/s$. Figure 14 shows the linear velocity control input and yaw angle velocity control input.

TABLE 5 The single-peak test functions.

Function	Initial range	Fmin
$f_1(x) = \sum_{i=1}^{30} x_i^2$	$-100 \le x_i \le 100$	0
$f_2(x) = \sum_{i=1}^{30} \lvert x_i\rvert + \prod_{i=1}^{30} \lvert x_i\rvert$	$-10 \le x_i \le 10$	0
$f_3(x) = \sum_{i=1}^{30} \left(\sum_{j=1}^{i} x_j\right)^2$	$-100 \le x_i \le 100$	0
$f_4(x) = \max_i \{\lvert x_i\rvert\ 1 \le i \le 30\}$	$-100 \le x_i \le 100$	0
$f_5(x) = \sum_{i=1}^{29} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	$-30 \le x_i \le 30$	0
$f_6(x) = \sum_{i=1}^{29} (\lvert x_i + 0.5\rvert)^2$	$-100 \le x_i \le 100$	0
$f_7(x) = \sum_{i=1}^{30} ix_i^4 + random\,[0, 1)$	$-1.28 \le x_i \le 1.28$	0

TABLE 6 The multi-peak test functions.

Function	Initial range	Fmin
$f_8(x) = -\sum_{i=1}^{30}\left(x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right)$	$-500 \le x_i \le 500$	-12569.5
$f_9(x) = \sum_{i=1}^{30}\left[x_i^2 - 10\cos\left(2\pi x_i + 10\right)\right]$	$-5.12 \le x_i \le 5.12$	0
$f_{10}(x) = -20exp^{\left(-0.2\sqrt{\frac{1}{30}\sum_{1}^{30} x_i^2}\right)} - exp^{\left(\frac{1}{30}\sum_{1}^{30}\cos 2\pi x_i\right)} + 20 + c$	$-100 \le x_i \le 100$	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$-600 \le x_i \le 600$	0
$f_{12}(x) = \frac{\pi}{30}\left\{100\sin^2\left(\pi y_1\right) + \sum_{i=1}^{29}\left(y_i - 1\right)^2 \times \left[1 + 10\sin^2\left(\pi y_{i+1}\right)\right] + \left(y_n - 1\right)^2\right\} + \sum_{i=1}^{30} u\left(x_i, 10, 100, 4\right)$	$-50 \le x_i \le 50$	0
$f_{13}(x) = 0.1\left\{\sin^2\left(\pi 3x_1\right) + \sum_{i=1}^{29}\left(x_i - 1\right)^2\left[\sin^2\left(3\pi x_{i+1}\right)\right] + \left(x_n - 1\right)^2\left[1 + \sin^2\left(2\pi x_{30}\right)\right] + \sum_{i=1}^{30} u\left(x_i, 5, 100, 4\right)$	$-50 \le x_i \le 50$	0

TABLE 7 The fixed-dimensional multi-peak test functions.

Function	Initial range	Fmin
$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}\left(x_i - a_{ij}\right)^6}\right]^{-1}$	$-65.536 \le x_i \le 65.536$	1
$f_{15}(x) = \sum_{i=1}^{11}\left[a_i^2 - \frac{x_1\left(b_i^2 + b_i x_2\right)}{b_i^2 + b_i x_3 + x_4}\right]$	$-5 \le x_i \le 5$	0.0003075
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 - \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	$-5 \le x_i \le 5$	-1.0316
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	$-5 \le x_1 \le 10$ $0 \le x_2 \le 15$	0.398
$f_{18}(x) = \left[1 + \left(x_1 + x_2 + 1\right)^2 \times \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \times \left[30 + \left(2x_1 - 3x_2\right)^2 \times \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$	$-2 \le x_i \le 2$	3
$f_{19}(x) = -\sum_{i=1}^{4}\exp\left[-\sum_{j=1}^{n} a_{ij}\left(x_j - p_{ij}\right)^2\right]$	$0 \le x_i \le 1$	-3.86
$f_{20}(x) = -\sum_{i=1}^{4}\exp\left[-\sum_{j=1}^{n} a_{ij}\left(x_j - p_{ij}\right)^2\right]$	$0 \le x_i \le 1$	-3.32

In Figures 14A, B, the $-2N$ force and $-1N.m$ torque are applied to the robot. So the control inputs are $2N$ and $1N.m$ to counteract the effect of the external force and torque on the robot.

The test functions for swarm intelligence optimization algorithms are shown in Tables 5–7.

5. Conclusion

This paper proposes an RBFNN-based anti-input saturation AFTC to solve the problem of degraded control performance of the CDR during movement on the water surface caused by drive faults, uncertain water resistance, and uncertain model parameters. The AFTC incorporates a fast NTSMC, which ensures the robustness of the robot against external disturbances and the effects of uncertain model parameters. The RBFNN is used to estimate drive faults and compensate for the controller output. Additionally, an anti-input saturation control algorithm is introduced to prevent controller input saturation. Furthermore, the traditional approach of manually tuning controller parameters based on the designer's experience and iterative debugging is replaced with an optimization method called HDSA. The HDSA algorithm optimizes the controller parameters to ensure the optimal control performance of the robot.

In further work, adaptive algorithms are necessary for the adjustment of the upper limit of the maximum control input to the robot on the ground and on the water surface.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

KW implementation and execution of the theory research and experiment and writing of the manuscript. YL theoretical support on the idea and helped write the manuscript. CH preliminary work and revising the manuscript. All authors actively contributed to the preparation of the content of this paper.

Funding

This work was supported in part by the Sharing Technology Project (41412040102), the China National Science Foundation (61473155), the Jiangsu Technology Department under Modern Agriculture (BE2017301), and the Six Talent Peaks Project in Jiangsu Province (GDZB-039).

Acknowledgments

We thank the editors and reviewers of the journal.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Ali, N., Tawiah, I., and Zhang, W. (2020). Finite-time extended state observer based nonsingular fast terminal sliding mode control of autonomous underwater vehicles. *Ocean Eng.* 218, 108179. doi: 10.1016/j.oceaneng.2020.108179

Chen, G., Tu, J., Ti, X., Wang, Z., and Hu, H. (2021). Hydrodynamic model of the beaver-like bendable webbed foot and paddling characteristics under different flow velocities. *Ocean Eng.* 234, 109179. doi: 10.1016/j.oceaneng.2021.109179

Chen, L., Cui, R., Yang, C., and Yan, W. (2019). Adaptive neural network control of underactuated surface vessels with guaranteed transient performance: theory and experimental results. *IEEE Transact. Ind. Electron.* 67, 4024–4035. doi: 10.1109/TIE.2019.2914631

Chu, R., Liu, Z., and Chu, Z. (2022). Improved super-twisting sliding mode control for ship heading with sideslip angle compensation. *Ocean Eng.* 260, 111996. doi: 10.1016/j.oceaneng.2022.111996

Cohen, A., and Zarrouk, D. (2020). "The amphistar high speed amphibious sprawl tuned robot: design and experiments," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Las Vegas, NV: IEEE), 6411–6418.

Deng, Y., Zhang, X., Im, N., Zhang, G., and Zhang, Q. (2020). Adaptive fuzzy tracking control for underactuated surface vessels with unmodeled dynamics and input saturation. *ISA Trans.* 103, 52–62. doi: 10.1016/j.isatra.2020.04.010

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transact. Syst. Man Cybernet. Part B* 26, 29–41. doi: 10.1109/3477.484436

Fister, I., Fister Jr, I., Yang, X.-S., and Brest, J. (2013). A comprehensive review of firefly algorithms. *Swarm Evol. Comput.* 13, 34–46. doi: 10.1016/j.swevo.2013.06.001

Gao, B., Liu, Y.-J., and Liu, L. (2022). Adaptive neural fault-tolerant control of a quadrotor uav via fast terminal sliding mode. *Aerospace Sci. Technol.* 129, 107818. doi: 10.1016/j.ast.2022.107818

Gheisarnejad, M., and Khooban, M. H. (2020). An intelligent non-integer pid controller-based deep reinforcement learning: Implementation and experimental results. *IEEE Transact. Ind. Electron.* 68, 3609–3618. doi: 10.1109/TIE.2020.2979561

Guo, J., Zhang, K., Guo, S., Li, C., and Yang, X. (2019). "Design of a new type of tri-habitat robot," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)* (Tianjin: IEEE), 1508–1513.

Guo, X., Huang, S., Lu, K., Peng, Y., Wang, H., and Yang, J. (2022). A fast sliding mode speed controller for PMSM based on new compound reaching law with improved sliding mode observer. *IEEE Trans. Transp. Elect.* 9, 2955–2968.

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., and Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Fut. Gen. Comp. Syst.* 97, 849–872. doi: 10.1016/j.future.2019.02.028

Hou, Q., and Ding, S. (2021). Finite-time extended state observer-based super-twisting sliding mode controller for pmsm drives with inertia identification. *IEEE Transact. Transport. Electrif.* 8, 1918–1929. doi: 10.1109/TTE.2021.3123646

Huang, J., Wang, W., Wen, C., and Li, G. (2019). Adaptive event-triggered control of nonlinear systems with controller and parameter estimator triggering. *IEEE Trans. Automat. Contr.* 65, 318–324. doi: 10.1109/TAC.2019.2912517

Jiang, T., and Lin, D. (2020). Fast finite-time backstepping for helicopters under input constraints and perturbations. *Int. J. Syst. Sci.* 51, 2868–2882. doi: 10.1080/00207721.2020.1803438

Liao, Y.-,l., Zhang, M.-,j., Wan, L., and Li, Y. (2016). Trajectory tracking control for underactuated unmanned surface vehicles with dynamic uncertainties. *J. Cent. South Univ.* 23, 370–378. doi: 10.1007/s11771-016-3082-4

Liu, K., Gao, H., Ji, H., and Hao, Z. (2020). Adaptive sliding mode based disturbance attenuation tracking control for wheeled mobile robots. *Int. J. Control Automat. Syst.* 18, 1288–1298. doi: 10.1007/s12555-019-0262-7

Liu, X., Zhang, M., and Yao, F. (2018). Adaptive fault tolerant control and thruster fault reconstruction for autonomous underwater vehicle. *Ocean Eng.* 155, 10–23. doi: 10.1016/j.oceaneng.2018.02.007

Mirjalili, S. (2016). Sca: a sine cosine algorithm for solving optimization problems. *Knowl. Based Syst.* 96, 120–133. doi: 10.1016/j.knosys.2015.12.022

Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Adv. Eng. Softw.* 69, 46–61. doi: 10.1016/j.advengsoft.2013.12.007

Najafi, A., Vu, M. T., Mobayen, S., Asad, J. H., and Fekih, A. (2022). Adaptive barrier fast terminal sliding mode actuator fault tolerant control approach for quadrotor uavs. *Mathematics* 10, 3009. doi: 10.3390/math10163009

Nan, F., Sun, S., Foehn, P., and Scaramuzza, D. (2022). Nonlinear mpc for quadrotor fault-tolerant control. *IEEE Robot. Automat. Lett.* 7, 5047–5054. doi: 10.1109/LRA.2022.3154033

Shen, Q., Yue, C., Goh, C. H., and Wang, D. (2018). Active fault-tolerant control system design for spacecraft attitude maneuvers with actuator saturation and faults. *IEEE Transact. Ind. Electron.* 66, 3763–3772. doi: 10.1109/TIE.2018.2854602

Song, M.-P., and Gu, G.-C. (2004). "Research on particle swarm optimization: a review," in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826)*, (Shanghai: IEEE), 2236–2241.

Wang, F., Ma, Z., Gao, H., Zhou, C., and Hua, C. (2022). Disturbance observer-based nonsingular fast terminal sliding mode fault tolerant control of a quadrotor UAV with external disturbances and actuator faults. *Int. J. Cont. Autom. Syst.* 20, 1122–1130. doi: 10.1007/s12555-020-0773-2

Wang, H., Shi, J., Wang, J., Wang, H., Feng, Y., and You, Y. (2019a). Design and modeling of a novel transformable land/air robot. *Int. J. Aero. Eng.* doi: 10.1155/2019/2064131

Wang, K., Liu, Y., Huang, C., and Bao, W. (2022a). Water surface flight control of a cross domain robot based on an adaptive and robust sliding mode barrier control algorithm. *Aerospace* 9, 332. doi: 10.3390/aerospace9070332

Wang, K., Liu, Y., Huang, C., and Cheng, P. (2022b). Water surface and ground control of a small cross-domain robot based on fast line-of-sight algorithm and adaptive sliding mode integral barrier control. *Appl. Sci.* 12, 5935. doi: 10.3390/app12125935

Wang, N., and Deng, Z. (2019). Finite-time fault estimator based fault-tolerance control for a surface vehicle with input saturations. *IEEE Trans. Ind. Informat.* 16, 1172–1181. doi: 10.1109/TII.2019.2930471

Wang, N., Xie, G., Pan, X., and Su, S.-F. (2019b). Full-state regulation control of asymmetric underactuated surface vehicles. *IEEE Trans. Ind. Informat.* 66, 8741–8750. doi: 10.1109/TIE.2018.2890500

Wu, G., Chen, G., Zhang, H., and Huang, C. (2021). Fully distributed event-triggered vehicular platooning with actuator uncertainties. *IEEE Transact. Vehic. Technol.* 70, 6601–6612. doi: 10.1109/TVT.2021.3086824

Wu, L.-B., Park, J. H., Xie, X.-P., Gao, C., and Zhao, N.-N. (2020). Fuzzy adaptive event-triggered control for a class of uncertain nonaffine nonlinear systems with full state constraints. *IEEE Transact. Fuzzy Syst.* 29, 904–916. doi: 10.1109/TFUZZ.2020.2966185

Xing, H., Shi, L., Hou, X., Liu, Y., Hu, Y., Xia, D., et al. (2021). Design, modeling and control of a miniature bio-inspired amphibious spherical robot. *Mechatronics* 77, 102574. doi: 10.1016/j.mechatronics.2021.102574

Xue, J., and Shen, B. (2020). A novel swarm intelligence optimization approach: sparrow search algorithm. *Syst. Sci. Control Eng.* 8, 22–34. doi: 10.1080/21642583.2019.1708830

Xue, J., and Shen, B. (2022). Dung beetle optimizer: a new meta-heuristic algorithm for global optimization. *J. Supercomput.* 1–32. doi: 10.1007/s11227-022-04959-6

Yu, X.-N., Hao, L.-Y., and Wang, X.-L. (2022). Fault tolerant control for an unmanned surface vessel based on integral sliding mode state feedback control. *Int. J. Control Automat. Syst.* 20, 2514–2522. doi: 10.1007/s12555-021-0526-x

Zhang, G., Chu, S., Zhang, W., and Liu, C. (2022). Adaptive neural fault-tolerant control for usv with the output-based triggering approach. *IEEE Transact. Vehic. Technol.* 71, 6948–6957. doi: 10.1109/TVT.2022.3167038

Zhang, H., Xi, R., Wang, Y., Sun, S., and Sun, J. (2021). Event-triggered adaptive tracking control for random systems with coexisting parametric uncertainties and severe nonlinearities. *IEEE Trans. Automat. Contr.* 67, 2011–2018. doi: 10.1109/TAC.2021.3079279

Zhao, Y., Qi, X., Ma, Y., Li, Z., Malekian, R., and Sotelo, M. A. (2020). Path following optimization for an underactuated usv using smoothly-convergent deep reinforcement learning. *IEEE Transact. Intell. Transport. Syst.* 22, 6208–6220. doi: 10.1109/TITS.2020.2989352

Zhong, G., Cao, J., Chai, X., and Bai, Y. (2021). Design and performance analysis of a triphibious robot with tilting-rotor structure. *IEEE Access* 9, 10871–10879. doi: 10.1109/ACCESS.2021.3050182