applied sciences

MDPI

Article

Chaos Particle Swarm Optimization Enhancement Algorithm for UAV Safe Path Planning

Hongyue Chu [1], Junkai Yi [2,*] and Fei Yang [1]

[1] School of Automation, Beijing Information Science & Technology University, Beijing 100192, China
[2] School of Automation, Key Laboratory of Modern Measurement and Control Technology Ministry of Education, Beijing Information Science & Technology University, Beijing 100192, China
* Correspondence: yijk@bistu.edu.cn

Abstract: For safe path planning of unmanned aerial vehicles (UAVs) in a three-dimensional (3D) environment with multiple threats, first, a cost function is introduced according to the terrain constraints and UAV overall performance constraints of the path planning problem. Then, improved nonlinear dynamic inertia weights (INDIW) are introduced into the particle swarm optimization (PSO) algorithm, and when the particles fall into the local optimum, the velocity is perturbed, and the velocity and improved nonlinear dynamic inertia weight PSO (VAINDIWPSO) algorithm are obtained. The algorithm improves the speed of convergence and fitness function value of the PSO algorithm. However, the impact of flyable path optimization is now not obvious. Therefore, to further enhance the overall performance of the VAINDIWPSO algorithm, the adaptive adjustment of the velocity is introduced, the chaotic initialization is carried out, and the improved logistic chaotic map is introduced into the algorithm, and an improved chaotic-VAINDIWPSO (IC-VAINDIWPSO) algorithm is obtained. Then, the corresponding relationship between the algorithm and constraints is used to efficiently search complicated environments and find paths with excessive security and small cost function. The simulation outcomes exhibit that in a complicated environment the IC-VAINDIWPSO algorithm substantially improves the speed of convergence of the algorithm, reduces the fitness function value of the algorithm and the initialization time of the algorithm, and the acquired path is additionally smoother. A near-optimal solution is obtained.

Keywords: unmanned aerial vehicle; path planning; improve particle swarm optimization; improved nonlinear dynamic inertia weights; chaos theory; adaptive adjustment of velocity

Citation: Chu, H.; Yi, J.; Yang, F. Chaos Particle Swarm Optimization Enhancement Algorithm for UAV Safe Path Planning. *Appl. Sci.* **2022**, *12*, 8977. https://doi.org/10.3390/app12188977

Academic Editors: Giancarlo Mauri and Valentino Santucci

Received: 6 August 2022
Accepted: 2 September 2022
Published: 7 September 2022

1. Introduction

UAVs have the advantages of top maneuverability, robust battlefield adaptability, small size, and low cost and are extensively used in military and civilian missions, such as surveillance, reconnaissance operations, search and rescue, emergency communications, and agriculture. The secure path planning of UAVs is of excellent magnitude to the completion rate and efficiency of UAVs performing tasks. Path planning [1,2] refers to calculating the optimal path between the beginning point and the goal point based on totally satisfying the constraints of the UAV itself and exterior threats. UAV secure path planning performs a key function in enhancing the autonomy and intelligence level of unmanned systems, so research on this topic has received greater and greater attention.

For the path planning issue of UAVs or robots, domestic and international scholars have proposed many techniques [3–7], such as the A* algorithm [8], the rapidly exploring random tree [9], the probabilistic roadmap algorithm [10], the Voronoi diagram algorithm [11], etc. However, these algorithms ignore the dynamics and kinematic constraints of UAVs, so they typically cannot be utilized in practical situations. The potential field methods, such as the artificial potential field method [12] and the interfered fluid dynamical system algorithm are additional superb techniques for path planning [13]. However,

when the goal and the obstacle are close, the algorithm can fall into the local optimum, and the optimal flyable path cannot be guaranteed. In recent years, the population-based evolutionary algorithm has made much progress.

Compared with classical algorithms, metaheuristic algorithms [14,15] can effectively solve complex combinatorial optimization problems and have been widely used in multi-UAV path planning research. Metaheuristic algorithms are improved algorithms based on heuristic algorithms., The most famous bio-inspired algorithms are the genetic algorithm (GA) [16] and the differential evolution algorithm (DE). The algorithms based on swarm intelligence include the ant colony optimization algorithm [17], the artificial bee colony (ABC) algorithm [18], the particle swarm optimization (PSO) [19], the moth flame optimization (MFO) [20], and the butterfly optimization algorithm (BOA) [21] etc.

Bio-inspired algorithms have a sturdy capability to search for the optimal solution and are very bendy and efficient, so they are extensively used in the path planning of UAVs. The PSO algorithm has the disadvantages of having a slow convergence speed, being easy fall into local optimum, and having a path that is not smooth, but the PSO algorithm has two vital properties, cognitive consistency and social consistency. These properties allow each particle in the population to find a solution according to its own experience and swarm experience, rather than by using traditional evolution operations such as mutation and crossover. Therefore, in contrast with other bio-inspired algorithms, it can be used to find a global solution with convergence in a shorter time. Moreover, the particle swarm algorithm is not very sensitive to the modifications of initial conditions and objective functions and can adapt to a variety of environmental structures via a small number of parameters such as an acceleration coefficient and two weight factors. Based on these advantages, the PSO algorithm is widely used in path planning, and a variety of variants of the PSO algorithm have emerged. Shikai Shao et al. [22] introduced an improved particle swarm algorithm to make UAV shape formations in a three-dimensional (3D) environment. This approach makes use of logistic chaotic mapping to enhance the preliminary distribution of particles and designs the acceleration coefficient and the maximum velocity as adaptive linearly altering coefficients to reap a better solution. A mutation method for replacing terrible particles with good ones is additionally proposed, which speeds up the convergence of the algorithm. Baoye Song et al. [23] examine the path of the mobile robot by way of combining the improved PSO algorithm with continuous high-order Bezier curves. In the improved PSO algorithm, the adaptive fractional velocity is introduced, and the particle swarm is disturbed by the evolution state of the particle swarm, which enhances the algorithm's capacity to bounce out of the local optimum and can fully explore the search space. Yubing Wang et al. [24] proposed three algorithms for quick convergence, random crossover, and actual search: the most density convergence distributed PSO (MDC-DPSO) algorithm, the quick crossover DPSO (FCO-DPSO) algorithm, and the genuine insurance exploration DPSO (ACE-DPSO) algorithm. Huang C et al. [25] proposed an improved PSO algorithm named GBPSO. To enhance the convergence speed and searchability of particles, a competitive strategy is used in the standard PSO algorithm to optimize the global optimal solution in the particle evolution process. Girija S et al. [26] mixed PSO with an artificial potential field to enhance the velocity of finding out possible and least-cost paths in environments with excessive obstacle density, and the improved algorithm reached the least-cost viable paths in a drastically shorter time. Dongping Tian et al. [27] proposed an improved PSO algorithm, which has a chaos-based initialization and a powerful update mechanism and makes use of the logistic graph to generate uniformly disbursed particles to enhance the quality of the initial population. PSO adaptively adopts inertia weights between linearly decreasing policies and nonlinear decreasing policies for better trade-offs between exploration and exploitation. Xuewen Xia et al. [28] designed a new component in the velocity update rule and adopted a similar view of the references [27] to enhance the overall performance of PSO. Zhuang Shao et al. [29] proposed a distributed collaborative PSO (DCPSO) algorithm with an elite retention strategy. The DCPSO algorithm has higher steadiness and a greater search success rate. Lin Jia et al. [30] proposed an improved PSO algorithm integral

separation PID (IPID) control algorithm primarily based on dynamic inertia weights and crossover operators. Wang, Z. et al. [31] proposed an improved PSO algorithm for parameter optimization. The algorithm introduced adaptive velocity weighting and adaptive population splitting, which accelerated the convergence speed of the algorithm and helped the algorithm bounce out of the local optimal position. Çomak, E. et al. [32] added the alpha parameter to the velocity update equation to control the influence of the global best and global worst particles on the velocity update equation, and the alpha value changed adaptively relative to the diversity measure. Lin, C.-J. et al. [33] proposed the PSO-IAC algorithm, which combines the improved adaptive inertia weights and constraint factors with the standard PSO algorithm. Sabir, Z. et al. [34] combined the modeling strength of artificial neural networks with the global search efficiency of PSO combined with the fast local search method of the interior point scheme. Umar, M. et al. [35] proposed a numerical computing technique for solving nonlinear second-order corneal- shaped models using a feed-forward artificial neural network optimized with PSO and active-set algorithms. These PSO variations share the equal population-based structure as our forthcoming technique. However, they vary in the way the search space and the solution encoded in particles are represented. Therefore, under the identical running environment, dynamic constraints, and goal function conditions, distinctive consequences will appear.

In this research, we first use a PSO algorithm to reap feasible paths. However, its convergence speed and optimal solutions cannot meet our requirements. So how to further speed up the convergence speed and obtain the best fitness function is our principal research motivation. Therefore, the PSO algorithm was improved, and a nonlinear dynamic inertia weight was introduced into the PSO algorithm. When the particles fell into the local optimum, the velocity was slightly disturbed, and an improved nonlinear dynamic inertia weight PSO (VAINDIWPSO) algorithm was obtained. This algorithm hastens the convergence speed and reduces the fitness function value. However, the optimization impact is no longer obvious. Therefore, based on the VAINDIWPSO algorithm, the velocity update formulation of the algorithm is adaptively adjusted. Since the particles with a more uniform initial distribution have less difficulty reaping the optimum solution, chaotic initialization is carried out, and the logistic chaotic map is introduced to attain the improved chaotic-VAINDIWPSO (IC-VAINDIWPSO) algorithm. According to the experimental results, the IC-VAINDIWPSO algorithm solves the problems of the PSO algorithm, such as gradual convergence speed, lengthy initialization time, unsmooth path, and easy descendance into local optimum.

The main contributions of this paper are as follows. To obtain an approximate optimal path to quickly obtain two points, two variants of the particle swarm algorithm are proposed, and then the particle swarm algorithm and the two variants are subjected to path planning experiments. The experimental results show that although the VAINDIW-PSO algorithm improves the convergence speed of the algorithm and reduces its fitness function value, the path optimization is not obvious. The IC-VAINDIWPSO algorithm greatly improves the convergence speed of the algorithm, reduces its fitness function value and initialization time, and the obtained path is also smoother. Then, to verify that the algorithm obtains paths under different threat numbers, seven groups of experiments were carried out. The experimental results show that the results obtained by the three algorithms in a simple environment are almost the same, and with the increase in the complexity of the environment, the IC-VAINDIWPSO algorithm achieves better results, The advantages are fully revealed. To reduce the randomness of the experimental results, 10 groups of experiments were carried out by changing the parameters of the threat. Finally, it is compared with other algorithms in the existing literature to fully verify the superiority of the IC-VAINDIWPSO algorithm in the paper.

The rest of this paper is organized as follows: the Section 2 introduces the threat environment model for path planning; the Section 3 introduces the PSO algorithm for UAV safe path planning; The Section 4 introduces the two newly proposed variants of PSO algorithms, namely the VAINDIWPSO algorithm and the IC-VAINDIWPSO algorithm. The

Section 5 discusses the experimental results and their analysis; the Section 6 describe the experimental conclusions and future work.

2. Threat Environment Model

2.1. Optimal Path

When the UAV is performing a task, it is essential to choose different criteria to achieve the optimum according to different application conditions. Here we choose to minimize the path length and use the flight path X_i to represent the listing of n waypoints that the UAV needs to fly through. Each waypoint corresponds to a path node on the map. Let the path node coordinates be $P_{ij} = (x_{ij}, y_{ij}, z_{ij})$, and two path nodes the Euclidean distance is $\| \overrightarrow{P_{ij}P_{i,j+1}} \|$, then the cost function of the path length is:

$$F_1(X_i) = \sum_{j=1}^{n-1} \| \overrightarrow{P_{ij}P_{i,j+1}} \| \tag{1}$$

2.2. Terrain and UAV Performance Constraints

In UAV path planning, it is additionally imperative to guide the UAV to safely pass the threats triggered using obstacles. Let K be the set of all threats and every threat be represented by way of a cylinder. As shown in Figure 1, let the projected middle coordinate of the obstacle be C_k and the radius be R_k. Due to the notably small dimension of the UAV, most research approximates the UAV as a particle. To make the threat cost of the UAV more accurate, the size of the UAV is considered here, the diameter of the UAV is set as D, and d_k is the distance between the UAV and C_k.

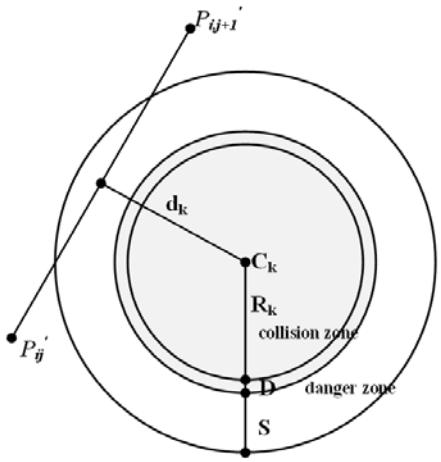

Figure 1. Threat.

Given a path segment $\| \overrightarrow{P_{ij}P_{i,j+1}} \|$, by way of thinking about the diameter D of the drone and the hazard distance S from the danger zone to the collision zone, calculate the threat cost F_2 at waypoint P_{ij}:

$$\begin{cases} F_2(X_i) = \sum\limits_{j=1}^{n-1} \sum\limits_{k=1}^{K} T_k(\overrightarrow{P_{ij}P_{i,j+1}}) \\ T_k(\overrightarrow{P_{ij}P_{i,j+1}}) = \begin{cases} 0 & d_k > S + D + R_k \\ (S + D + R_k) - d_k, & D + R_k < d_k \leq S + D + R_k \\ \infty & d_k \leq D + R_k \end{cases} \end{cases} \tag{2}$$

Due to the running environment, positioning accuracy, and different reasons, the UAV nonetheless has the possibility of colliding with obstacles, so the distance from the collision area S is known as the danger zone. For example, in a static surrounding with excellent

positioning signals, S can be chosen as several tens of meters, and in the case of shifting objects or weak positioning signals, S can be chosen as several hundred meters.

The flight height of the UAV is usually constrained between two given height extremes, namely the maximum height and the minimum height. Let the minimal height be $h_{\min}$ and the most height be $h_{\max}$, then the height cost related to the waypoint P_{ij} is calculated as follows:

$$H_{ij} = \begin{cases} \left| h_{ij} - \frac{(h_{\max}+h_{\min})}{2} \right|, h_{\min} \le h_{ij} \le h_{\max} \\ \infty, \quad \text{otherwise} \end{cases} \tag{3}$$

Here h_{ij} is the flying height of the drone relative to the ground, and H_{ij} is acquired according to (3), maintaining the average height and penalizing the out-of-range values. So the altitude cost can be expressed as:

$$F_3(X_i) = \sum_{j=1}^{n} H_{ij} \tag{4}$$

The assessment of the smoothing cost needs to calculate the turning angle and the climbing angle. As shown in Figure 2, the turning angle $]\varphi_{ij}$ is the angle between $\overrightarrow{P_{ij}{}'P_{i,j+1}{}'}$ and $\overrightarrow{P_{i,j+1}{}'P_{i,j+2}{}'}$ between the two line segments projected on the surface Oxy, and let $\vec{k}$ be the unit vector on the z-axis, The projected vector can be calculated as:

$$\overrightarrow{P_{ij}{}'P_{i,j+1}{}'} = \vec{k} \times (\overrightarrow{P_{ij}P_{i,j+1}} \times \vec{k}) \tag{5}$$

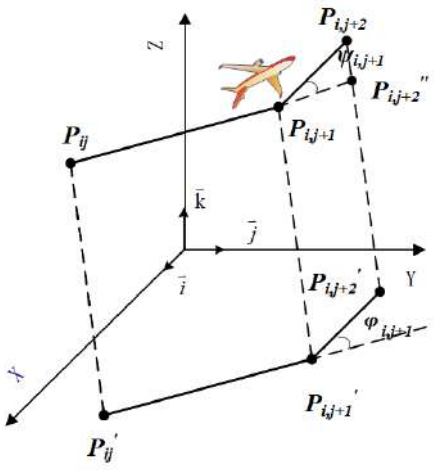

Figure 2. Turning and climbing angle calculation.

So, the turning angle is calculated as:

$$\varphi_{ij} = \arctan\left(\frac{\left\| \overrightarrow{P_{ij}{}'P_{i,j+1}{}'} \times \overrightarrow{P_{i,j+1}{}'P_{i,j+2}{}'} \right\|}{\overrightarrow{P_{ij}{}'P_{i,j+1}{}'} \cdot \overrightarrow{P_{i,j+1}{}'P_{i,j+2}{}'}}\right) \tag{6}$$

The climb angle is the angle between the UAV's ascent trajectory and the horizontal line; that is, the climb angle ψ_{ij} refers to the angle between the path segment $\overrightarrow{P_{ij}P_{i,j+1}}$ and its projection $\overrightarrow{P_{ij}{}'P_{i,j+1}{}'}$ on the horizontal plane.

$$\psi_{ij} = \arctan\left(\frac{z_{i,j+1} - z_{ij}}{\|\overrightarrow{P_{ij}{}'P_{i,j+1}{}'}\|}\right) \tag{7}$$

The smoothing cost is calculated as follows:

$$F_4(X_i) = a_1 \sum_{j=1}^{n-2} \varphi_{ij} + a_2 \sum_{j=1}^{n-1} \left|\psi_{ij} - \psi_{i,j-1}\right| \tag{8}$$

where a_1 and a_2 are the penalty coefficients for turning and climbing angles, respectively.

2.3. Total Cost Function

By thinking about the optimality, safety, and feasibility constraints related to the path X_i, the overall cost function can be defined according to (1) to (8) in it:

$$F(X_i) = \sum_{k=1}^{4} b_k F_k(X_i) \tag{9}$$

Among them, b_k is the weight coefficient. Equations (1)–(9) are acquired from reference [36].

The choice of the fitness function is vital in UAV path planning [37]. In this paper, the optimal overall cost function in the complicated environment of more than one threat in UAV navigation is taken as the fitness function.

3. PSO Algorithm

The PSO algorithm [38] is a biological evolutionary algorithm, which was inspired by the foraging process of birds, randomly initializing a team of particles in the solution area of the optimization problem. Each particle represents a feasible solution in the goal function, and its position is decided by way of the particle motion velocity, which is affected by using the particle history optimal solution and the group history optimal solution.

In the n-dimensional search space, the position of the particle i ($i = 1, 2, \cdots, N$) is denoted using $x_i(k) = (x_{i1}(k), x_{i2}(k), \cdots, x_{in}(k))$, and the velocity is denoted using $v_i(k) = (v_{i1}(k), v_{i2}(k), \cdots, v_{in}(k))$. The parameter of the personal best position searched via the particle i is $p_{best} = (p_{i1}, p_{i2}, \cdots, p_{in})$, and the parameter of the global best position of the population is $g_{best} = (p_{g1}, p_{g2}, \cdots, p_{gn})$. We use the acquired global best and personal best position information to calculate the corrected position and velocity of the particle, as shown in (10) and (11):

$$v_i(k+1) = \omega v_i(k) + c_1 r_1 (p_i(k) - x_i(k)) + c_2 r_2 (p_g(k) - x_i(k)) \tag{10}$$

$$x_i(k+1) = x_i(k) + v_i(k+1) \tag{11}$$

where t is the number of iterations, ω is the inertia weight, c_1 and c_2 are learning factors, and $c_1 = c_2 = 1.5$, r_1 and r_2 are random numbers.

When the PSO algorithm is in the stagnation stage of evolution, the particles in the particle swarm will "aggregate", and the particles will not be dispersed until the stagnation of evolution is broken. Equation (12) can be obtained by transforming Equations (10) and (11):

$$x_i(k+1) = x_i(k) + \omega \times [x_i(k) - x_i(k-1)] + \tau \times [\sigma - x_i(k)] \tag{12}$$

in, $\tau = c_1 r_1 + c_2 r_2$, $\sigma = \frac{c_1 r_1 p_{best} + c_2 r_2 g_{best}}{c_1 r_1 + c_2 r_2}$.

Equation (12) further leads to (13):

$$x_i(k+1) = (1+\omega-\tau) \times x_i(k) + \omega \times x_i(k-1) + \tau \times \sigma \tag{13}$$

According to (13), we obtain:

$$\lim_{t \to +\infty} x_i(k) = \sigma = \frac{c_1 r_1 p_{best} + c_2 r_2 g_{best}}{c_1 r_1 + c_2 r_2} \tag{14}$$

Since r_1 and r_2 are uniformly distributed, Equation (15) can be obtained:

$$\lim_{t \to +\infty} x_i(k) = \sigma = \frac{c_1 p_{best} + c_2 g_{best}}{c_1 + c_2} \tag{15}$$

Since $c_1 = c_2 = 1.5 > 0$, Equation (16) can be obtained:

$$\lim_{t \to +\infty} x_i(k) = \sigma = \frac{c_1 p_{best} + c_2 g_{best}}{c_1 + c_2} = \frac{p_{best} + g_{best}}{2} \tag{16}$$

It can be seen from (10) to (16) that the particles will gather on the extreme values obtained by their extreme value p_{best} and the global extreme value g_{best} of the group. If the particle cannot discover a better position p_{best}, the evolution will be in a stagnant state. The particles will gather at $(p_{best}+g_{best})/2$ and converge to the local optimal solution.

4. Proposed PSO Variant

Using the maneuvering characteristics of the UAV, the VAINDIWPSO algorithm and the IC-VAINDIWPSO algorithm for solving the path planning problem are proposed, and the realization scheme is given.

4.1. VAINDIWPSO Algorithm

To speed up the convergence of the algorithm and make the algorithm leap out of the local optimum as soon as possible, the VAINDIWPSO algorithm is proposed. According to (19), the improved nonlinear dynamic inertia weight (INDIW) was once introduced into the VAINDIWPSO algorithm based on the PSO algorithm, and when particles fell into the local optimum, the velocity of the global optimum particles would be slightly disturbed, as shown in (20).

4.1.1. Improvements Nonlinear Dynamic Inertial Weights

Because the PSO algorithm has the trait of fast convergence, the algorithm effortlessly falls into the local optimum in the later stage of iteration, so the optimal solution cannot be obtained. The linear inertia weight update approach cannot efficiently resolve this problem [39], so the nonlinear dynamic inertia weight is delivered [40], as (17) shows:

$$\omega(k+1) = (\omega_{\max} - \omega_{\min}) \times \left(\frac{\omega(k)}{\omega_{\max}}\right)^{\frac{k}{T_{\max}}} \tag{17}$$

It can be seen from (17) that $\omega(k)$ is descending alongside the nonlinear curve, which reduces the probability of the algorithm falling into the local optimum. However, due to the complicated terrain environment throughout path planning, an improved nonlinear dynamic inertia weight is introduced, such as (18):

$$\omega(k+1) = (\omega(k) - \omega_{\min}) \times \left(\frac{\omega_{\min}}{\omega_{\max}}\right)^{\frac{k^2}{T_{\max}{}^2}} \tag{18}$$

Compared with (17), in (18), the variable $\omega(k)$ is placed in the coefficient term, and the $\frac{k}{T_{\max}}$ term is squared so that the inertia weight can rapidly converge to meet the real-time

necessities of the algorithm for the system. For further optimization, a random disturbance term is delivered based on (18), and (19) is introduced again:

$$\omega(k+1) = (\omega(k) - \omega_{\min}) \times (\frac{\omega_{\min}}{\omega_{\max}})^{\frac{k^2}{T_{\max}{}^2}} + R(k) \tag{19}$$

Among them, $R(k)$ is a disturbance term whose range is a uniformly distributed random number $[-\omega_{\min}/2, \omega_{\min}/2]$.

In the algorithm solution process, the inertia weight suggests a general downward trend. It cannot be proven that the inertia weight in the subsequent iteration ought to be smaller than the preceding one, and the algorithm solely explores locally in the later stage and cannot precisely resolve the optimal solution. Therefore, the inertia weight is improved to make the nonlinearity decrease, and random disturbances are added, as shown in (19), so that tiny oscillations take place during the iterative update process, and when the algorithm falls into a local optimum, it can rely on this tiny oscillation to leap out of the local exploration state.

4.1.2. Update Velocity

The state of particle movement is decided through the velocity, and the particle adjusts the velocity according to the experience received through itself and the experience acquired through different particles and searches for the optimal solution. The greater the iterations are, the more likely the particles will produce local optima. Therefore, to make the convergence speed of the algorithm quicker and keep it from obtaining the local optimal value, the particles that are nearer to g_{best}, being fine-tuned, (20) are proposed as a new velocity update formula. That is, when the particle cannot locate the global optimal solution, the velocity of the global optimal particle is slightly disturbed, and the convergence speed of the algorithm is improved.

$$v_i(k+1) = g_{best} - x_i(k) + \alpha v_i(k) + \beta r_3 \tag{20}$$

where r_3 is a unit random value, and α and β are the disturbed coefficients, which are constant $\alpha = 0.2$ and $\beta = 0.3$.

4.2. IC-VAINDIWPSO Algorithm

To speed up the convergence speed and initialization speed of the algorithm and obtain a smoother path and forestall the algorithm from falling into local optimal, the IC-VAINDIWPSO algorithm is proposed. The IC-VAINDIWPSO algorithm is primarily based on the VAINDIWPSO algorithm, which first of all introduces the adaptive adjustment of the velocity, such as (21). Then it consists of out chaos initialization. Finally, according to (22), improved logistic chaos mapping is introduced into the algorithm [41].

4.2.1. Adaptive Adjustment of Velocity

The approach of adaptive velocity adjustment is introduced into the algorithm [32]. According to the evaluation of the actual situation, it can be seen that when the particle moves in the global optimal direction and the particle can consistently locate a better point, if the flight direction is no longer changed, the particle will fly to the optimal particle faster, thus rushing the convergence speed of the algorithm, as shown in (21):

$$v_i(k+1) = \begin{cases} \omega v_i(k) + c_1 r_1 (p_i(k) - x_i(k)) + c_2 r_2 (p_g(k) - x_i(k)), & Fit(k) \geq Fit(k-1) \\ v_i(k), & Fit(k) < Fit(k-1) \end{cases} \tag{21}$$

As shown in (21), if the fitness value of the current particle in this iteration is greater than the preceding one, there is the potential that the particle moving alongside the current direction is conducive to discovering the global optimal particle. If the fitness value of the current particle in this iteration is higher than the last value, the particle is no longer

moving in the direction of the global optimal particle. The next time the velocity of the particle will be updated according to the particle velocity update method of the standard PSO algorithm.

4.2.2. Chaos Initialization

The widespread PSO algorithm randomly initializes the population. Although the technique is simple, it is easy to make the initial particle distribution uneven or even away from the optimal solution, which appreciably reduces the execution efficiency. Therefore, the ergodicity and randomness of chaos theory are introduced to intervene in the initialization of the population. The initial particle swarm is formed by using a chaotic sequence to make a certain uniform distribution. To improve the quality of the initialized particles, double the particles are generated at some point during initialization, and then half of the particles with the lower fitness function value are selected. This how the initial particles become evenly allotted in the solution area. It also makes the obtained initial particle swarm nearer to the optimal solution, which improves the convergence speed of the algorithm.

4.2.3. Logistic Chaos Map

The chaotic sequence can be generated using five sorts of chaotic maps, such as the logistic map [42], the lorenz map [43], the piecewise linear chaotic map [44], the tent map [45], and the Henon map [46]. Among them, logistic mapping is a relatively easy mapping technique in terms of its mathematical shape and is broadly used. We apply the improved logistic chaotic map to deal with the optimal particle of the population.

When most particles acquire approximate solutions or the identical solution, the population falls into a local optimum and loses diversity. So it is imperative to mutate the optimal particles so that the rest of the particles can rapidly leap out of the local optimum state. This is why the logistic chaotic map was introduced into the improved algorithm [32]. The logistic chaotic map can additionally generate a logistic of a certain length, and its randomness and ergodicity can be used to enhance searchability. Chaos is an unstable phenomenon springing up from a deterministic system, indicating that its movement state is random. The ergodicity of chaotic motion can traverse all the states without repetition of its laws. Therefore, a purposeful search for chaotic mutation will yield better outcomes than a random search. The adopted logistic chaotic map is shown in (22):

$$z_i(k) = \mu z_i(k-1)(1-\mathrm{z}_i(k-1)) \tag{22}$$

Among them, μ is the control parameter of the chaotic state, $\mu \in (0, 4]$, when $\mu = 4$, the control system is in a completely chaotic state, which is conducive to leaping out of the local optimum. When using logistic chaotic mapping, the position of the particle must be mapped in the range of [0, 1], that is, $0 \leq z_i(k) \leq 1$.

When using chaos theory for local search, the particle position is shown in (23):

$$\mathrm{x}_i(k) = x_i(k-1) + \eta \mathrm{z}_i(k-1) \tag{23}$$

Here η is the adjustment coefficient. In the search process, setting η slightly larger in the early stage is beneficial to the algorithm leaping out of the local optimum. When the optimal solution is approached, the value η step-by-step turns smaller. So η is adaptive as the algorithm iterates, as shown in (24):

$$\eta = \gamma[(k_{\max} - k + 1)/T_{max}]^2 x_i(k-1) \tag{24}$$

Among them, γ is the neighborhood radius, $\gamma = 0.1$.

4.2.4. The Specific Steps of the IC-VAINDIWPSO Algorithm

The fundamental steps are as follows:

(a) Modeling the threat environment for drone flight;

(b) Initializing the particle swarm with improved chaos theory;
(c) Evaluating particles: according to the constraints in the threat environment, the fitness value of the particles is evaluated;
(d) Updating particles: the particle positions and velocities are updated according to (11) and (19)–(21);
(e) Calculating the rate of change of fitness (FCR) according to (25). If $k < 2T_{max}/3$, and FCR is less than the set threshold, the local position of the particle changes slightly, and it falls into the local optimum. Go to (f), otherwise, go to (g);

$$FCR = \frac{F(k) - F(k-1)}{F(k-1)} \tag{25}$$

(f) According to the logistic chaotic map (23), the mutation operation is performed on the global best position;
(g) Stop condition; by iterating continuously, adjust their velocity and position to keep them inside the feasible range until the algorithm reaches the maximum number of iterations and obtains the optimal solution; otherwise, return to (c).

For a clearer understanding of the IC-VAINDIWPSO algorithm, pseudocode is added, as shown in Algorithm 1.

Algorithm 1 IC-VAINDIWPSO algorithm

```
      /* Initialization: */
1  Obtain search map and initial path planning information;
2  Set swarm parameters, swarm_size;
3  for each particle i in swarm do
4     Create a random path;
5     Assign to particle's position;
6     Compute fitness of the particle;
7     Set local_best of the particle to its fitness;
8  end
9  Chaos initialization
10 Set global_best to the best-fit particle;
    /* Evolutions: */
11  for k ← 1 to max_generation do
12     for each particle i in swarm do
13        Compute velocity;                          /* Equation (21)*/
14        if Particles become stuck in local optima
15           Compute velocity;                       /* Equation (20)*/
16        end
17        Compute new position;                      /* Equation (11) */
18        Update fitness;                            /* Equation (9) */
19        if k is less than 2/3 * max_generation
20           Compute FCR                             /* Equation (25) */
21           if FCR is less than the set threshold
22           Compute new position;                   /* Equation (23) */
23           end
24        end
25        Update local_best;
26     end
27     Update global_best;
28     Update inertia weights                         /* Equation (19) */
29     Save best_position associated with global_best; /* the best path */
30  end
```

To evaluate the three algorithms extra clearly, Table 1 compares the three algorithms.

Table 1. The difference between the three algorithms.

Algorithm	PSO Algorithm	VAINDIWPSO Algorithm	IC-VAINDIWPSO Algorithm
Velocity and position	Equations (10) and (11)	Equations (10) and (11)	Equations (21) and (11)
inertia weight	$\omega(k+1) = \omega(k) * \omega_{damp}$	Equation (19)	Equation (19)
update velocity	no	Equation (20)	Equation (20)
Chaos initialization	no	no	yes
Logistic chaos map	no	no	Equation (22)

5. Experimental Simulation Analysis

To evaluate the overall performance of the improved PSO algorithm, we conduct comparisons and experiments via a sequence of computational simulations.

5.1. Experimental Parameters

In this paper, the UAV path planning is simulated in the Matlab environment and is run on a server with a 2.1 GHz CPU, 16.00 GB of RAM, and 64-bit operating system, and the effectiveness of the proposed technique is verified. Table 2 shows the parameters of PSO and its variant applications. The experimental environment model is an actual digital elevation model map acquired from the lidar sensor. A 3D terrain environment with x, y, and z coordinates is generated by augmenting a terrain structure area on Christmas Island, Australia. When setting up threats of radius r in these scenarios, the number and location of threats (indicated by way of red cylinders) are chosen according to the environment of different complexity, as shown in Table 3 and Figure 3.

Table 2. Parameter setting.

Parameter Name	Parameter Notation	Parameter Value
learning factors	$c_1 = c_2$	1.5
disturbed coefficients	α	0.2
disturbed coefficients	β	0.3
control parameter of the chaotic state	μ	$\mu \in (0, 4]$
neighborhood radius	γ	0.1

Table 3. Threats parameter setting.

Threat Parameter \ Threat Number	x	y	z	r
1	900	100	250	80
2	300	750	150	80
3	700	550	150	70
4	350	200	150	70
5	500	350	150	80
6	600	200	150	70
7	400	500	100	80

There are three algorithms in the experiment, specifically the PSO algorithm, the VAINDIWPSO algorithm, and the IC-VAINDIWPSO algorithm. The damping rate of the PSO algorithm is a constant value ω_{damp}, the maximum and minimum values of the inertia weight of the VAINDIWPSO algorithm and the IC-VAINDIWPSO algorithm are $\omega_{\max}$ and $\omega_{\min}$, respectively, and the value of ω of the corresponding algorithm is acquired from Table 4. When the three algorithms draw the path curve in the environment, n = 10 path points are received between the coordinates of the starting point and the target point according to the flight regulation of the UAV to structure the flight path of the UAV. The coordinates of the starting point are (200,100,150), and the coordinates of the target point are (800,800,150).

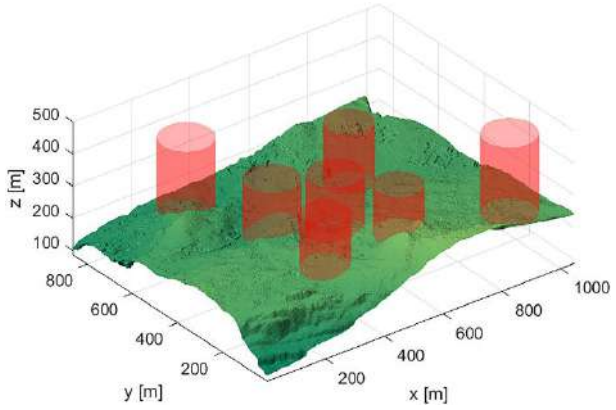

Figure 3. 3D terrain environment.

Table 4. The inertia weight formula and parameters of the three algorithms.

Parameter \ Algorithm	Inertia Weight	$\omega(1)$	$\omega_{\max}$	$\omega_{\min}$	ω_{damp}
PSO algorithm	$\omega(k+1) = \omega(k) * \omega_{damp}$	1	-	-	0.98
VAINDIWPSO algorithm	$\omega(k+1) = (\omega(k) - \omega_{\min}) \times (\frac{\omega_{\min}}{\omega_{\max}}) \frac{k^2}{T_{\max}^2} + R(k)$	1	0.9	0.4	-
IC-VAINDIWPSO algorithm	$\omega(k+1) = (\omega(k) - \omega_{\min}) \times (\frac{\omega_{\min}}{\omega_{\max}}) \frac{k^2}{T_{\max}^2} + R(k)$	1	0.9	0.4	-

5.2. Analysis of Results

The path planning of the UAV needs to bypass the threat area from the coordinates of the starting point and fly to the coordinates of the target point. Considering the randomness of the heuristic algorithm, every tested algorithm was independently executed 30 times, and the experimental results were used to evaluate and compare its performance. For the comfort of comparison, Figures 4–7 exhibit the top view and fitness function value curves of the optimal UAV paths generated by using the PSO algorithm, the VAINDIWPSO algorithm, and the IC-VAINDIWPSO algorithm in 30 independent runs, respectively. To visualize the differences between the tested algorithms, Table 5 shows the data results of the three algorithm runs.

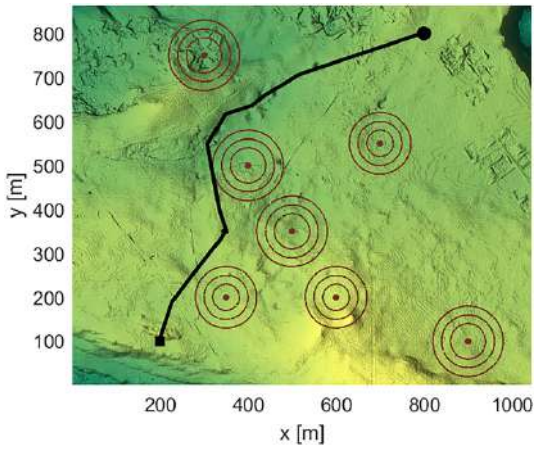

Figure 4. Top view of the path generated by the PSO algorithm.

According to Figures 4–6, all three algorithms can generate feasible paths that meet the necessities of path length, threat, turn angle, climb/dive angle, and altitude. However, the PSO algorithm is susceptible to falling into the local optimum and cannot locate an optimal solution. The VAINDIWPSO algorithm introduces improved nonlinear dynamic

inertia weight and interferes with the velocity, the convergence algebra is decreased by 44.3%, the fitness function value is additionally reduced, and a better path than the PSO algorithm is obtained. The IC-VAINDIWPSO algorithm, which introduces chaos theory and adaptively adjusts the velocity, can precisely acquire near-optimal solutions.

The iterative convergence of the PSO algorithm is very slow. So the number of iterations is chosen to be 1000, and the number of iterations of the VAINDIWPSO algorithm and the IC-VAINDIWPSO algorithm is chosen to be 500 times. The IC-VAINDIWPSO algorithm performs a chaotic initialization operation. So 1000 particles are generated in the course of initialization. Then 500 particles with low fitness function values are chosen as the initial population, and the initial population of the other two algorithms is also additionally chosen as 500.

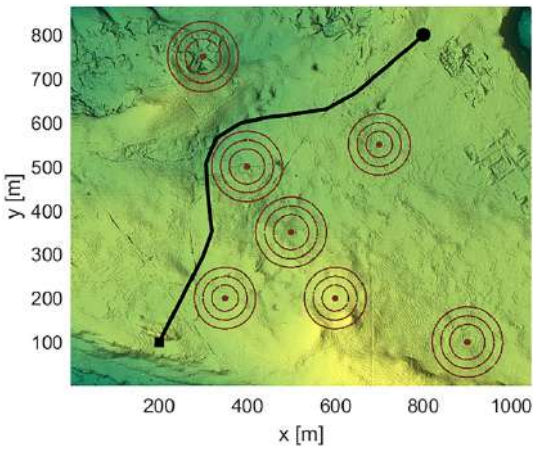

Figure 5. Top view of the path generated by the VAINDIWPSO algorithm.

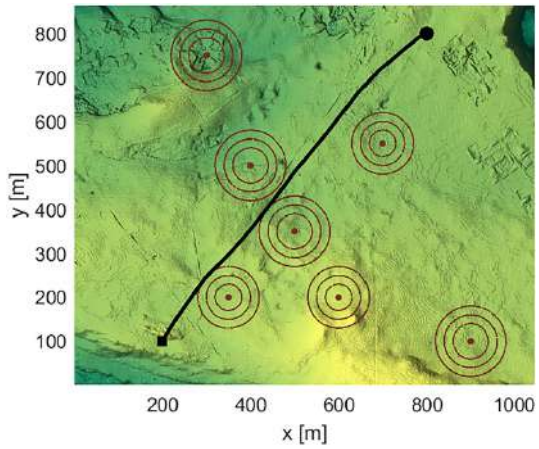

Figure 6. Top view of the path generated by the IC-VAINDIWPSO algorithm.

According to the combination of Table 5 and Figure 7, it can be seen precisely and intuitively that the iterative convergence times of the VAINDIWPSO algorithm are smaller than those of the PSO algorithm, and the average fitness value is additionally decreased a lot, which indicates that the VAINDIWPSO algorithm has relatively proper performance. However, the number of iterative convergences is nonetheless large. Therefore, we introduced chaos theory and adaptively adjusted the velocity to structure the IC-VAINDIWPSO algorithm. The iteration convergence times of the IC-VAINDIWPSO algorithm are 20, the initialization running time is 0.644 s, and the average fitness value is 5575.7793. Compared with the preceding two algorithms, the iterative convergence times of the IC-VAINDIWPSO algorithm are a whole lot smaller than the other two algorithms, and the average fitness is additionally the lowest. The IC-VAINDIWPSO initialization running time algorithm is

additionally noticeably reduced, which is shorter than the PSO algorithm. It is 86.35%, which is 86.43% shorter than the VAINDIWPSO algorithm. The running velocity and efficiency of the algorithm are improved, and it can rapidly and precisely bounce out of the local optimum, attain the global optimum solution, and reap the optimum path.

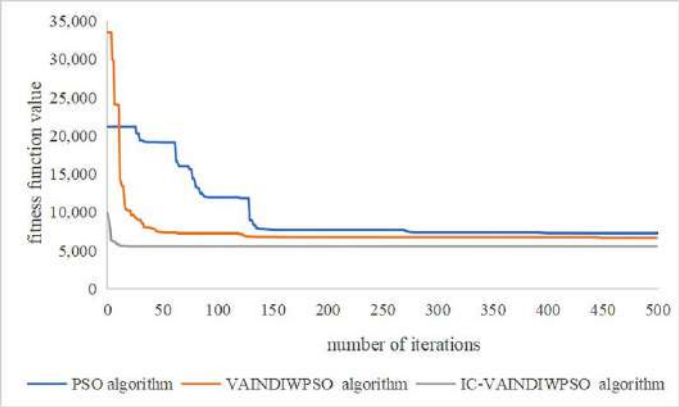

Figure 7. Convergence graph of fitness function of three algorithms.

Table 5. Algorithm data comparison.

Algorithm Name	Initial Population	Number of Iterations	Iterative Convergence Times	Optimal Fitness	Average Fitness	Initialization Runtime/s	Iteration Running Time/s
PSO algorithm	500	1000	449	7256.4168	7350.4576	4.745	130.492
VAINDIWPSO algorithm	500	500	250	6635.2626	6704.6754	4.717	62.648
IC-VAINDIWPSO algorithm	1000	500	20	5518.1756	5575.7793	0.644	58.997
IC-VAINDIWPSO algorithm (stop running 50 generations after convergence)	1000	73	23	5520.1236	5579.7346	0.654	9.424

Figure 8 shows that when the algorithm converges, the fitness function value will continue to be unchanged. Moreover, it is acknowledged from the above that the IC-VAINDIWPSO algorithm does not easily fall into the local optimum, so according to Figure 8, the algorithm stops running after 50 iterations after it converges. This technique can attain the optimal fitness value with a small number of iterations, which considerably reduces the running time of the algorithm, and permits the UAV to rapidly and precisely design near-optimal solutions.

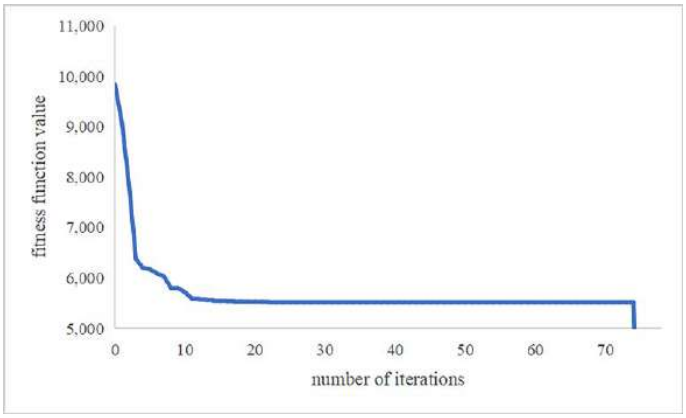

Figure 8. IC-VAINDIWPSO algorithm fitness function convergence graph (50 generations stop running after convergence).

Figures 9–12 exhibit the 3D path planning diagrams and side views of the paths acquired by using the PSO algorithm and the IC-VAINDIWPSO algorithm in complicated terrain. It can be seen that the path acquired via the PSO algorithm is relatively bad and not smooth. The IC-VAINDIWPSO algorithm can rapidly discover the near-optimal solutions and keep a suitable flight height with the terrain, and the path is additionally smooth and effective.

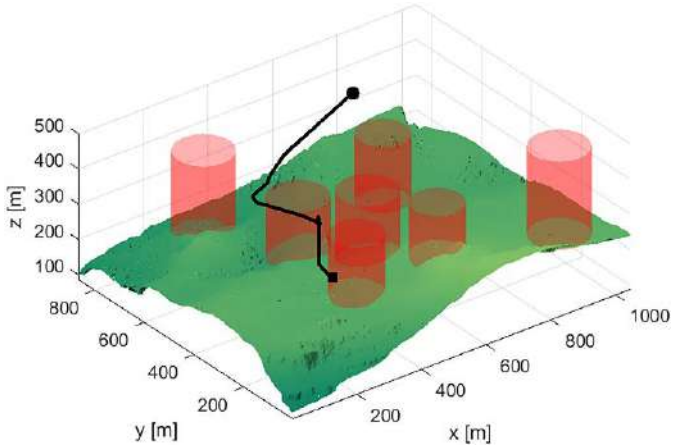

Figure 9. The 3D path planning diagram of the PSO algorithm.

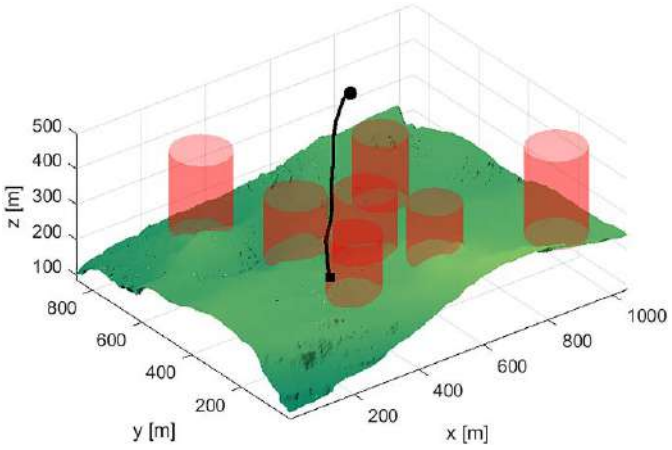

Figure 10. The 3D path planning diagram of the IC-VAINDIWPSO algorithm.

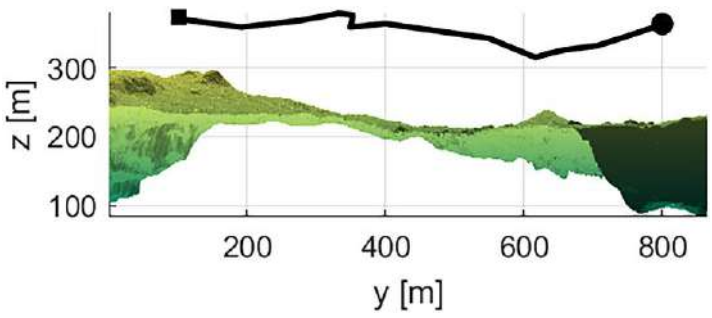

Figure 11. Convergence graph of the fitness function of the three algorithms.

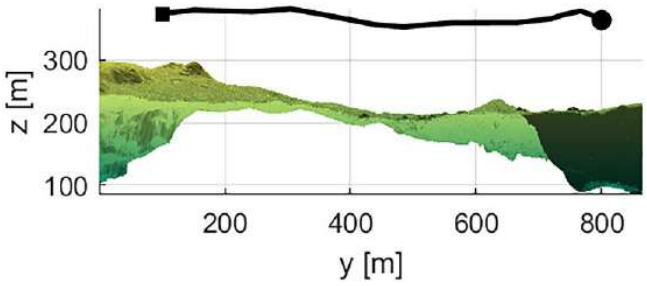

Figure 12. The 3D path side view of the IC-VAINDIWPSO algorithm.

According to the data in Table 5, we choose different numbers of threats according to the serial numbers of the threats to acquire different environmental models, and then we use the above three algorithms to conduct 30 simulation experiments in different environments to acquire the average value of the fitness function. As shown in Figure 13, when the number of threats is less than three, the three algorithms can acquire the optimal solution, and the average value of the acquired fitness function is now not very different. However, as the environment will become extra complex, the PSO algorithm and the VAINDIWPSO algorithm will fall into a local optimum, and the IC-VAINDIWPSO algorithm overcomes this shortcoming. So the gap between the average values of the fitness functions of the three algorithms will grow larger, and the degree of pros and cons of the acquired path will also be different. It can be seen from Table 6 that when the number of threats is small, the influence of threats on the acquired path is relatively small. The VAINDIWPSO algorithm and the IC-VAINDIWPSO algorithm can directly acquire near-optimal solutions, whilst the PSO algorithm still needs 153 iterations to converge, its average fitness function is the lowest, and the initialization time is additionally the shortest. As the number of threats progressively increases, the benefits of the IC-VAINDIWPSO algorithm steadily emerge. It can be seen from Table 7 that the iteration convergence times of the IC-VAINDIWPSO algorithm are 25 times, the average fitness function is 5229.1847, and the initialization time is 0.629 s, which is 8.2% shorter than the VAINDIWPSO algorithm and 44.53% shorter than the PSO algorithm. Compared with Table 5, it can be seen that the more complex the environment is, the more obvious the optimization effect of the IC-VAINDIWPSO algorithm is, and the greater the gap between the three algorithms is. Figures 14–17 are the top views of the PSO algorithm and the IC-VAINDIWPSO algorithm in three threats and five threat era paths. It can be seen that the IC-VAINDIWPSO algorithm acquires better flyable paths.

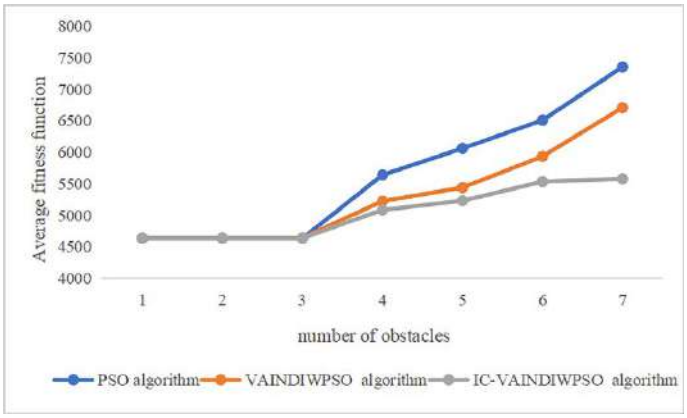

Figure 13. Average of fitness functions for different numbers of threats.

Table 6. Data comparison of three algorithms with three threats.

Algorithm Name	Initial Population	Number of Iterations	Iterative Convergence Times	Optimal Fitness	Average Fitness	Initialization Runtime/s	Iteration Running Time/s
PSO algorithm	500	1000	153	4618.3186	4631.182	0.167	82.169
VAINDIWPSO algorithm	500	500	1	4637.3815	4637.3815	0.104	46.221
IC-VAINDIWPSO algorithm	1000	500	1	4637.3815	4637.3815	0.604	43.509

Table 7. Data comparison of three algorithms with five threats.

Algorithm Name	Initial Population	Number of Iterations	Iterative Convergence Times	Optimal Fitness	Average Fitness	Initialization Runtime/s	Iteration Running Time/s
PSO algorithm	500	1000	360	5721.9739	6058.1241	1.134	103.802
VAINDIWPSO algorithm	500	500	158	5422.4877	5437.8974	0.754	53.884
IC-VAINDIWPSO algorithm	1000	500	25	5196.9241	5229.1847	0.629	51.382

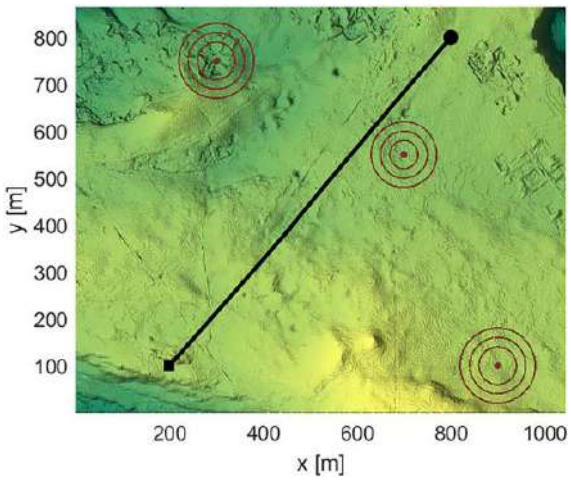

Figure 14. The top view of the paths generated by the PSO algorithm under three threats.

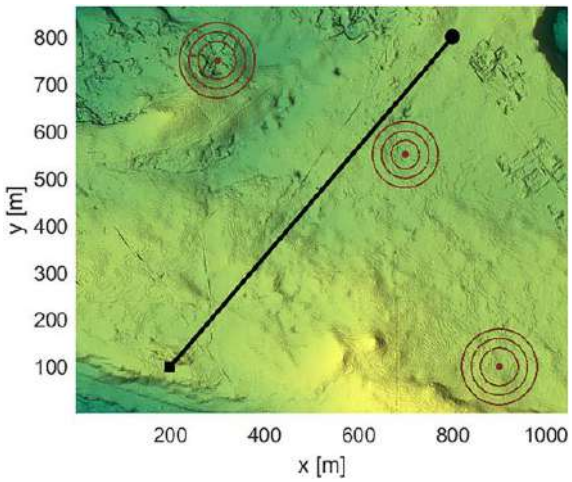

Figure 15. The top view of the paths generated by the IC-VAINDIWPSO algorithm under three threats.

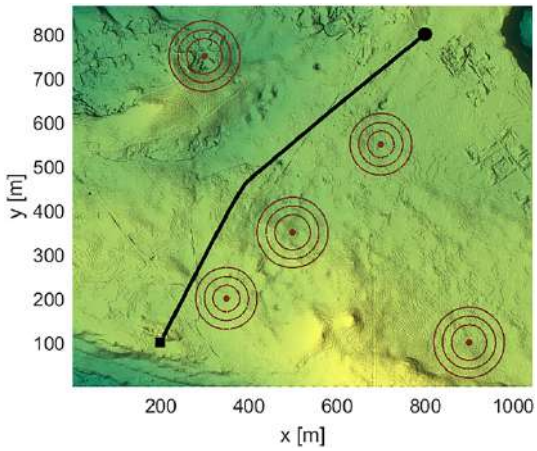

Figure 16. The top view of the paths generated by the PSO algorithm under five threats.

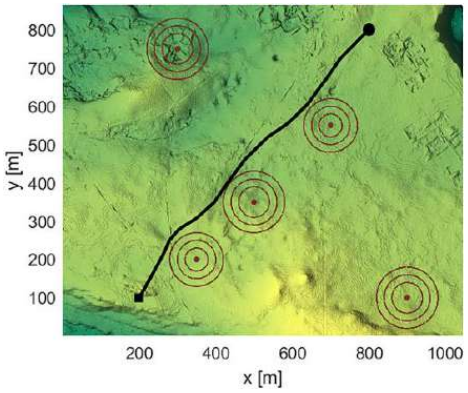

Figure 17. The top view of the paths generated by the IC-VAINDIWPSO algorithm under five threats.

The positions of the two threats are randomly and not repeatedly changed in every experiment, and 10 groups of distinctive environmental models are generated, performing 30 simulation tests on the above three algorithms in distinct environmental models and calculating the average value of the fitness function. The outcomes are shown in Figure 18. It can be seen from the analysis of the simulation consequences that in contrast with the PSO algorithm and the VAINDIWPSO algorithm, the fitness function value of the IC-VAINDIWPSO algorithm is typically lower, displaying a greater optimization ability. Figures 19–22 are the top views of the PSO algorithm and the IC-VAINDIWPSO algorithm in the fourth set of data and the ninth set of data generation paths.

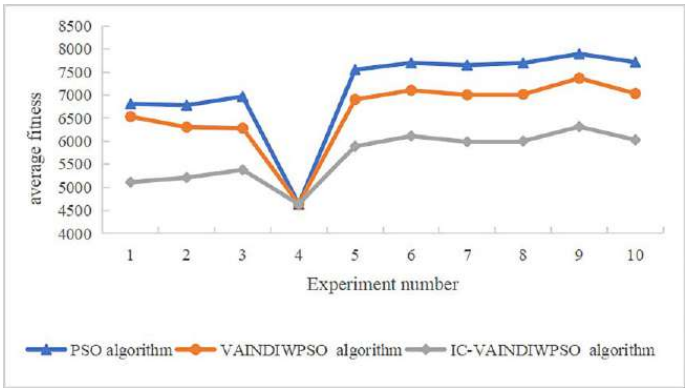

Figure 18. The average value of fitness functions for different environments.

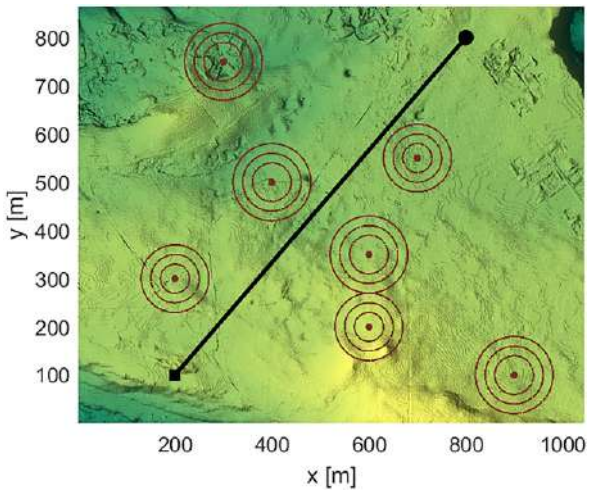

Figure 19. The top view of the path generated by the fourth set of data for the PSO algorithm.

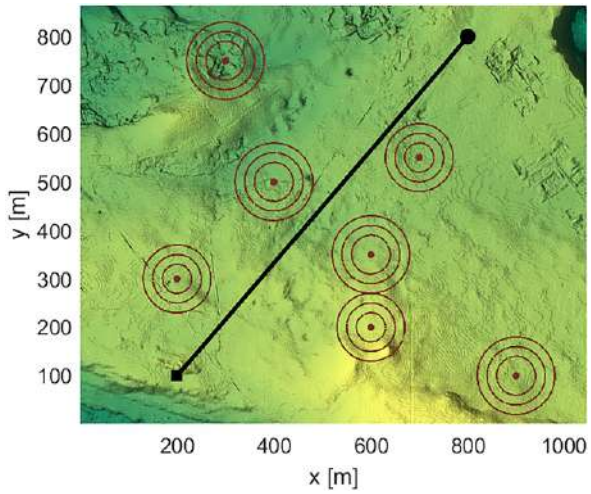

Figure 20. The top view of the path generated by the fourth set of data for the IC-VAINDIWPSO algorithm.

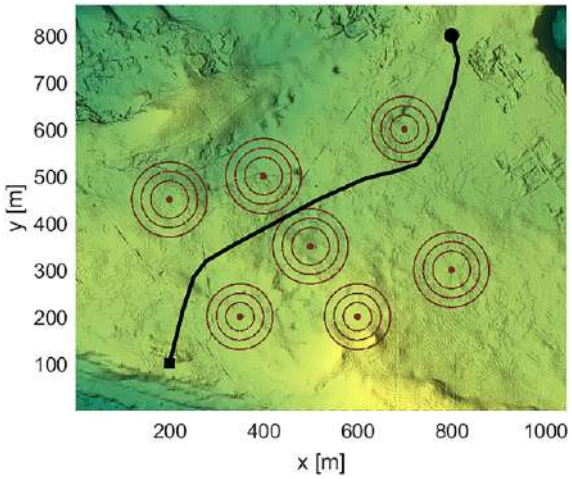

Figure 21. The top view of the path generated by the ninth group of data for the PSO algorithm.

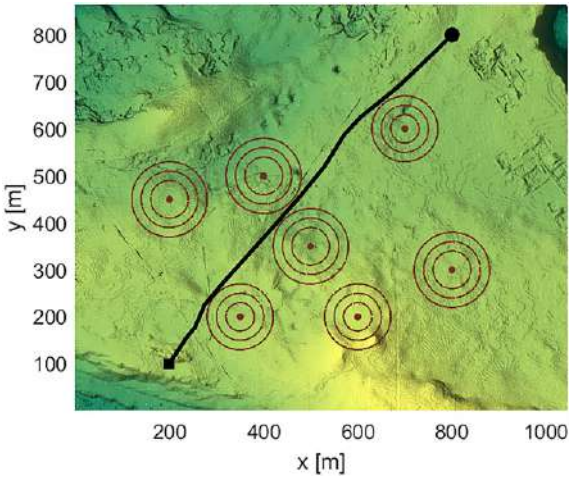

Figure 22. The top view of the path generated by the ninth group of data for the IC-VAINDIWPSO algorithm.

Table 8 shows the data in Scenario 4 of Reference [21]. It can be seen from the data that the fitness function value and convergence algebra of the PSO algorithm are higher than the quantum-behaved PSO algorithm (QPSO), the genetic algorithm (GA), and differential evolution (ABC). From the data in Table 5, it can be considered that the fitness function cost and convergence algebra of the IC-VAINDIWPSO are much higher quality than the PSO algorithm. Although the environment of this paper and the literature [21] are different, they are each chosen from Christmas Island, Australia, so evaluating Tables 6 and 8 horizontally, it can be seen that the IC-VAINDIWPSO algorithm is higher than the QPSO algorithm, the GA algorithm, and the ABC algorithm, so the IC-VAINDIWPSO algorithm, a variant of the PSO algorithm, is very valuable for in-depth study.

Table 8. Algorithm data comparison.

Algorithm	Average Fitness	Iterative Convergence Times
PSO algorithm	5781	56
QPSO algorithm	7120	761
GA algorithm	6325	224
ABC algorithm	5325	118

According to all the experimental results and analysis, it can be concluded that the VAINDIWPSO algorithm introduces nonlinear decreasing dynamics inertia weight and adds random disturbances so that the algorithm can jump out of the local optimum. It also introduces the velocity update formula when the particles cannot fall into the local optimum. During the optimal solution, the speed of the global optimal particle is slightly disturbed, and the algorithm can also jump out of the local optimum. Therefore, the VAINDIWPSO algorithm plays a great role in jumping out of the local optimum and reducing the fitness function value of the algorithm. The IC-VAINDIWPSO algorithm introduces the strategy of adaptive speed adjustment so that the particles continuously fly to the optimal solution, and then it introduces the chaotic initialization. The chaotic initialization can initially generate a uniform distribution of particles even close to the optimal solution and can maintain the initialization time. In a shorter time, the initialization time of the algorithm is reduced and convergence is sped up. Finally, the logistic chaotic map is introduced, and the randomness and ergodicity are used to improve the searchability. Finally, the approximate optimal solution is obtained.

To sum up, regardless of the complexity of the environment, the IC-VAINDIWPSO algorithm can discover the most appropriate flight path. It has the advantages of solving the problems of slow convergence speed, far initialization time, unsmooth path, and easy

descendance into the local optimum. Therefore, the IC-VAINDIWPSO algorithm can remedy the UAV path planning problem.

6. Conclusions

Aiming at the research on UAV security path planning, the IC-VAINDIWPSO algorithm is proposed. Based on the PSO algorithm, the IC-VAINDIWPSO algorithm first introduces when particles fall into local improved nonlinear dynamic inertia weights and the velocity is perturbed. Then the adaptive adjustment of the velocity is introduced, and finally, chaos initialization is carried out, and the improved logistic chaotic map is introduced. The IC-VAINDIWPSO algorithm can efficaciously carry out UAV path planning in a complicated 3D environment. When setting up the threat environment model, the algorithm also considers many constraints such as optimality, safety, and feasibility. According to the acquired path planning results in complicated environments, the IC-VAINDIWPSO algorithm can attain near-optimal solutions in most scenarios, whilst the PSO algorithm and the VAINDIWPSO algorithm only operate well in easy scenarios. By evaluating the three algorithms, the superiority of the IC-VAINDIWPSO algorithm in fixing issues such as sluggish convergence speed, lengthy initialization time, unsmooth path, and easy descendance into the local optimum phenomenon is verified. The experimental consequences exhibit that the IC-VAINDIWPSO algorithm is an efficient and possible approach for UAV route planning. However, this experiment was only carried out in a static environment, without thinking about the effect of dynamic obstacles, and the constraints are extraordinarily few. In the future, we will think about more constraints and then conduct in-depth research on the UAV path planning issues in complicated terrain environments with dynamic obstacles.

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

Funding: This research was funded by the National Natural Science Foundation of China, grant number U1636208.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

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

References

1. Yu, X.; Li, C.; Zhou, J. A Constrained Differential Evolution Algorithm to Solve UAV Path Planning in Disaster Scenarios. *Knowl.-Based Syst.* **2020**, *204*, 106209. [CrossRef]
2. Zhao, Y.; Zheng, Z.; Liu, Y. Survey on Computational-Intelligence-Based UAV Path Planning. *Knowl.-Based Syst.* **2018**, *158*, 54–64. [CrossRef]
3. Patle, B.K.; Babu, L.G.; Pandey, A.; Parhi, D.R.K.; Jagadeesh, A. A Review: On Path Planning Strategies for Navigation of Mobile Robot. *Def. Technol.* **2019**, *15*, 582–606. [CrossRef]
4. Aggarwal, S.; Kumar, N. Path Planning Techniques for Unmanned Aerial Vehicles: A Review, Solutions, and Challenges. *Comput. Commun.* **2020**, *149*, 270–299. [CrossRef]
5. Sánchez-Ibáñez, J.R.; Pérez-del-Pulgar, C.J.; García-Cerezo, A. Path Planning for Autonomous Mobile Robots: A Review. *Sensors* **2021**, *21*, 7898. [CrossRef] [PubMed]
6. Gul, F.; Mir, I.; Abualigah, L.; Sumari, P.; Forestiero, A. A Consolidated Review of Path Planning and Optimization Techniques: Technical Perspectives and Future Directions. *Electronics* **2021**, *10*, 2250. [CrossRef]
7. Zafar, M.N.; Mohanta, J.C. Methodology for Path Planning and Optimization of Mobile Robots: A Review. *Procedia Comput. Sci.* **2018**, *133*, 141–152. [CrossRef]
8. Zhang, Z.; Wu, J.; Dai, J.; He, C. A Novel Real-Time Penetration Path Planning Algorithm for Stealth UAV in 3D Complex Dynamic Environment. *IEEE Access* **2020**, *8*, 122757–122771. [CrossRef]

9. Kiani, F.; Seyyedabbasi, A.; Aliyev, R.; Gulle, M.U.; Basyildiz, H.; Shah, M.A. Adapted-RRT: Novel Hybrid Method to Solve Three-Dimensional Path Planning Problem Using Sampling and Metaheuristic-Based Algorithms. *Neural Comput. Appl.* **2021**, *33*, 15569–15599. [CrossRef]
10. Ravankar, A.A.; Ravankar, A.; Emaru, T.; Kobayashi, Y. HPPRM: Hybrid Potential Based Probabilistic Roadmap Algorithm for Improved Dynamic Path Planning of Mobile Robots. *IEEE Access* **2020**, *8*, 221743–221766. [CrossRef]
11. Ayawli, B.B.K.; Mei, X.; Shen, M.; Appiah, A.Y.; Kyeremeh, F. Mobile Robot Path Planning in Dynamic Environment Using Voronoi Diagram and Computation Geometry Technique. *IEEE Access* **2019**, *7*, 86026–86040. [CrossRef]
12. Jayaweera, H.M.; Hanoun, S. A Dynamic Artificial Potential Field (D-APF) UAV Path Planning Technique for Following Ground Moving Targets. *IEEE Access* **2020**, *8*, 192760–192776. [CrossRef]
13. Yao, P.; Wang, H.; Su, Z. Cooperative Path Planning with Applications to Target Tracking and Obstacle Avoidance for Multi-UAVs. *Aerosp. Sci. Technol.* **2016**, *54*, 10–22. [CrossRef]
14. Zamani, H.; Nadimi-Shahraki, M.H.; Gandomi, A.H. QANA: Quantum-Based Avian Navigation Optimizer Algorithm. *Eng. Appl. Artif. Intell.* **2021**, *104*, 104314. [CrossRef]
15. Zamani, H.; Nadimi-Shahraki, M.H.; Gandomi, A.H. Starling Murmuration Optimizer: A Novel Bio-Inspired Algorithm for Global and Engineering Optimization. *Comput. Methods Appl. Mech. Eng.* **2022**, *392*, 114616. [CrossRef]
16. Nazarahari, M.; Khanmirza, E.; Doostie, S. Multi-Objective Multi-Robot Path Planning in Continuous Environment Using an Enhanced Genetic Algorithm. *Expert Syst. Appl.* **2019**, *115*, 106–120. [CrossRef]
17. Chen, J.; Ling, F.; Zhang, Y.; You, T.; Liu, Y.; Du, X. Coverage Path Planning of Heterogeneous Unmanned Aerial Vehicles Based on Ant Colony System. *Swarm Evol. Comput.* **2022**, *69*, 101005. [CrossRef]
18. Hu, Y.; Sun, Z.; Cao, L.; Zhang, Y.; Pan, P. Optimization Configuration of Gas Path Sensors Using a Hybrid Method Based on Tabu Search Artificial Bee Colony and Improved Genetic Algorithm in Turbofan Engine. *Aerosp. Sci. Technol.* **2021**, *112*, 106642. [CrossRef]
19. Xia, G.; Han, Z.; Zhao, B.; Wang, X. Local Path Planning for Unmanned Surface Vehicle Collision Avoidance Based on Modified Quantum Particle Swarm Optimization. *Complexity* **2020**, *2020*, 3095426. [CrossRef]
20. Dai, X.; Wei, Y. Application of Improved Moth-Flame Optimization Algorithm for Robot Path Planning. *IEEE Access* **2021**, *9*, 105914–105925. [CrossRef]
21. Wu, P.; Wang, Z.; Jing, H.; Zhao, P. Optimal Time–Jerk Trajectory Planning for Delta Parallel Robot Based on Improved Butterfly Optimization Algorithm. *Appl. Sci.* **2022**, *12*, 8145. [CrossRef]
22. Shao, S.; Peng, Y.; He, C.; Du, Y. Efficient Path Planning for UAV Formation via Comprehensively Improved Particle Swarm Optimization. *ISA Trans.* **2020**, *97*, 415–430. [CrossRef]
23. Song, B.; Wang, Z.; Zou, L. An Improved PSO Algorithm for Smooth Path Planning of Mobile Robots Using Continuous High-Degree Bezier Curve. *Appl. Soft Comput.* **2021**, *100*, 106960. [CrossRef]
24. Wang, Y.; Bai, P.; Liang, X.; Wang, W.; Zhang, J.; Fu, Q. Reconnaissance Mission Conducted by UAV Swarms Based on Distributed PSO Path Planning Algorithms. *IEEE Access* **2019**, *7*, 14. [CrossRef]
25. Huang, C.; Fei, J. UAV Path Planning Based on Particle Swarm Optimization with Global Best Path Competition. *Int. J. Patt. Recogn. Artif. Intell.* **2018**, *32*, 1859008. [CrossRef]
26. Girija, S.; Joshi, A. Fast Hybrid PSO-APF Algorithm for Path Planning in Obstacle Rich Environment. *IFAC-PapersOnLine* **2019**, *52*, 25–30. [CrossRef]
27. Tian, D.; Shi, Z. MPSO: Modified Particle Swarm Optimization and Its Applications. *Swarm Evol. Comput.* **2018**, *41*, 49–68. [CrossRef]
28. Xia, X.; Xing, Y.; Wei, B.; Zhang, Y.; Li, X.; Deng, X.; Gui, L. A Fitness-Based Multi-Role Particle Swarm Optimization. *Swarm Evol. Comput.* **2019**, *44*, 349–364. [CrossRef]
29. Shao, Z.; Yan, F.; Zhou, Z.; Zhu, X. Path Planning for Multi-UAV Formation Rendezvous Based on Distributed Cooperative Particle Swarm Optimization. *Appl. Sci.* **2019**, *9*, 2621. [CrossRef]
30. Jia, L.; Zhao, X. An Improved Particle Swarm Optimization (PSO) Optimized Integral Separation PID and Its Application on Central Position Control System. *IEEE Sens. J.* **2019**, *19*, 7064–7071. [CrossRef]
31. Wang, Z.; Fu, Y.; Song, C.; Zeng, P.; Qiao, L. Power System Anomaly Detection Based on OCSVM Optimized by Improved Particle Swarm Optimization. *IEEE Access* **2019**, *7*, 181580–181588. [CrossRef]
32. Çomak, E. A Particle Swarm Optimizer with Modified Velocity Update and Adaptive Diversity Regulation. *Expert Syst.* **2019**, *36*, e12330. [CrossRef]
33. Lin, C.-J.; Li, T.-H.S.; Kuo, P.-H.; Wang, Y.-H. Integrated Particle Swarm Optimization Algorithm Based Obstacle Avoidance Control Design for Home Service Robot. *Comput. Electr. Eng.* **2016**, *56*, 748–762. [CrossRef]
34. Sabir, Z.; Raja, M.A.Z.; Guirao, J.L.G.; Shoaib, M. A Neuro-Swarming Intelligence-Based Computing for Second Order Singular Periodic Non-Linear Boundary Value Problems. *Front. Phys.* **2020**, *8*, 224. [CrossRef]
35. Umar, M.; Amin, F.; Wahab, H.A.; Baleanu, D. Unsupervised Constrained Neural Network Modeling of Boundary Value Corneal Model for Eye Surgery. *Appl. Soft Comput.* **2019**, *85*, 105826. [CrossRef]
36. Phung, M.D.; Ha, Q.P. Safety-Enhanced UAV Path Planning with Spherical Vector-Based Particle Swarm Optimization. *Appl. Soft Comput.* **2021**, *107*, 107376. [CrossRef]

37. Teng, H.; Ahmad, I.; Msm, A.; Chang, K. 3D Optimal Surveillance Trajectory Planning for Multiple UAVs by Using Particle Swarm Optimization With Surveillance Area Priority. *IEEE Access* **2020**, *8*, 86316–86327. [CrossRef]
38. Wang, B.; Li, S.; Guo, J.; Chen, Q. Car-like Mobile Robot Path Planning in Rough Terrain Using Multi-Objective Particle Swarm Optimization Algorithm. *Neurocomputing* **2018**, *282*, 42–51. [CrossRef]
39. Huang, X.; Li, C.; Chen, H.; An, D. Task Scheduling in Cloud Computing Using Particle Swarm Optimization with Time Varying Inertia Weight Strategies. *Clust. Comput.* **2020**, *23*, 1137–1147. [CrossRef]
40. Kiani, A.T.; Nadeem, M.F.; Ahmed, A.; Khan, I.; Elavarasan, R.M.; Das, N. Optimal PV Parameter Estimation via Double Exponential Function-Based Dynamic Inertia Weight Particle Swarm Optimization. *Energies* **2020**, *13*, 4037. [CrossRef]
41. Tharwat, A.; Elhoseny, M.; Hassanien, A.E.; Gabel, T.; Kumar, A. Intelligent Bézier Curve-Based Path Planning Model Using Chaotic Particle Swarm Optimization Algorithm. *Clust. Comput.* **2019**, *22*, 4745–4766. [CrossRef]
42. Lian, J.; Yu, W.; Xiao, K.; Liu, W. Cubic Spline Interpolation-Based Robot Path Planning Using a Chaotic Adaptive Particle Swarm Optimization Algorithm. *Math. Probl. Eng.* **2020**, *2020*, 1849240. [CrossRef]
43. Masood, F.; Ahmad, J.; Shah, S.A.; Jamal, S.S.; Hussain, I. A Novel Hybrid Secure Image Encryption Based on Julia Set of Fractals and 3D Lorenz Chaotic Map. *Entropy* **2020**, *22*, 274. [CrossRef] [PubMed]
44. Luo, Y.; Zhou, R.; Liu, J.; Cao, Y.; Ding, X. A Parallel Image Encryption Algorithm Based on the Piecewise Linear Chaotic Map and Hyper-Chaotic Map. *Nonlinear Dyn.* **2018**, *93*, 1165–1181. [CrossRef]
45. Wangsheng, F.; Chong, W.; Ruhua, Z. Application of Simulated Annealing Particle Swarm Optimization in Complex Three-Dimensional Path Planning. *J. Phys. Conf. Ser.* **2021**, *1873*, 012077. [CrossRef]
46. Gohari, P.S.; Mohammadi, H.; Taghvaei, S. Using Chaotic Maps for 3D Boundary Surveillance by Quadrotor Robot. *Appl. Soft Comput.* **2019**, *76*, 68–77. [CrossRef]