



Article

Path Planning and Energy Efficiency of Heterogeneous Mobile Robots Using Cuckoo–Beetle Swarm Search Algorithms with Applications in UGV Obstacle Avoidance

Dechao Chen ^{1,*}, Zhixiong Wang ^{2,†}, Guanchen Zhou ^{2,†} and Shuai Li ^{3,†}

¹ School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China

² The HDU-ITMO Joint Institute, Hangzhou Dianzi University, Hangzhou 310018, China

³ The College of Engineering, Swansea University, Swansea SA1 7EN, UK

* Correspondence: chdchao@hdu.edu.cn; Tel.: +86-137-7748-1992

† These authors contributed equally to this work.

Abstract: In this paper, a new meta-heuristic path planning algorithm, the cuckoo–beetle swarm search (CBSS) algorithm, is introduced to solve the path planning problems of heterogeneous mobile robots. Traditional meta-heuristic algorithms, e.g., genetic algorithms (GA), particle swarm search (PSO), beetle swarm optimization (BSO), and cuckoo search (CS), have problems such as the tenancy to become trapped in local minima because of premature convergence and a weakness in global search capability in path planning. Note that the CBSS algorithm imitates the biological habits of cuckoo and beetle herds and thus has good robustness and global optimization ability. In addition, computer simulations verify the accuracy, search speed, energy efficiency and stability of the CBSS algorithm. The results of the real-world experiment prove that the proposed CBSS algorithm is much better than its counterparts. Finally, the CBSS algorithm is applied to 2D path planning and 3D path planning in heterogeneous mobile robots. In contrast to its counterparts, the CBSS algorithm is guaranteed to find the shortest global optimal path in different sizes and types of maps.



Citation: Chen, D.; Wang, Z.; Zhou, G.; Li, S. Path Planning and Energy Efficiency of Heterogeneous Mobile Robots Using Cuckoo–Beetle Swarm Search Algorithms with Applications in UGV Obstacle Avoidance.

Sustainability **2022**, *14*, 15137.

<https://doi.org/10.3390/su142215137>

Academic Editors: Raul D.S.G. Campilho and Jun (Justin) Li

Received: 21 September 2022

Accepted: 9 November 2022

Published: 15 November 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Path planning is a very important research and development topic in mobile robots [1]. After obtaining an understanding of the surrounding environment, most of the path planning of mobile robots involves selecting the best or worst collision-free path from the starting point to the endpoint in the environment based on indicators. The final result is the method to solve the problem under the corresponding conditions. The results of path planning have a great influence on the research of mobile robots. Robot path planning research has made some achievements from its beginning until now, but relatively speaking, these developments are far from enough. Thus, based on the research environment, path planning can be divided into global or local planning. According to the study of path planning classification, global path planning is the solving of the environmental planning problem comprehensively and two-part planning can be said to focus on the unknown or known partial path problems. Then, according to the complexity of the working environment of mobile robots, the two categories of robot paths can be described in more detail. Due to the uncertainty of local path planning, it has high flexibility for adjusting according to the environment in the work. However, the path can only represent the local optimum because it is a local environment feature. Global path planning is the determination of the

optimal path in the whole environment, and the final guide path from the start point to the final point is the optimal path.

With the rapid development of internet technology and 5G technology, self-driving has become a popular direction in the automotive field. As a key part of unmanned driving technology, tracking control has also become an important research direction for scholars, and many control methods have also been applied to tracking control. For instance, these methods include traditional PID control [2], adaptive fuzzy control [3,4] and neural network control [5,6], robust sliding mode control [7,8] and robust control [9,10], model predictive control based on a vehicle dynamics model [11,12], etc. Because of its better dynamic characteristics and steering stability, the four-wheel independent drive and independent steering vehicle has become an excellent carrier of unmanned tracking control technology and a research focus of scholars. S Pramanik et al. [13], using the Hooke and Jeeves optimization method, synthesized a central lever steering mechanism to obtain five precision points for a four-wheel vehicle; the steering error, pressure angle, and mechanical advantage of the proposed mechanism were compared with those achieved by the Ackermann steering mechanism. The proposed mechanism had less steering error, a more favorable pressure angle, and increased mechanical advantage. Similarly, the method of compounding the mechanism was also applicable when the central lever was offset from the longitudinal axis of the vehicle. Tu et al. [14] studied a robust controller for four-wheel steering and four-wheel drive agricultural robot vehicles based on reverse sliding mode control, which improved the control ability and robustness of non-holonomic systems with high degrees of freedom. Likewise, Y Tian et al. [15] put forward a type of lateral stability control strategy for four-wheel independent drive electric vehicles. The design of the control system adopts a hierarchical structure. However, unlike the previous control strategy, the research in [15] introduces a method that consists of a combination of sliding mode control and an optimal allocation algorithm. According to a driver's operation commands (steering angle and speed), the steady-state responses of the sideslip angle and yaw rate are obtained. Wang et al. [16] developed a new type of electric vehicle with a four-wheel independent drive and independent steering using wire control and proposed an online reconfigurable steering angle and driving force allocation control method based on optimal distribution of tire force, which improved the stability of the steering controller. Zhenyang Li et al. [17] proposed an active steering control method of superposition of steering torque based on receding horizon control for man-machine collaborative driving of co-drive intelligent vehicles. This control method combines a steering system with a vehicle dynamics system and takes a steering motor torque as the sole control input, which ensures that the driver has the ultimate control right and makes the control of the system more stable.

A biological heuristic algorithm is a type of optimization calculation method which simulates the reproduction habits, foraging characteristics, and living habits of various organisms in natural ecology. Computer systems usually require high energy consumption to solve complex problems, while biological systems improve their performance by studying the habits of biological populations and simulating biological phenomena. Furthermore, compared with computer systems, biological systems can save resources and have higher robustness when solving some optimization problems [18]. The subject knowledge of many fields is used for reference and learning, and is integrated into biological heuristics, forming a set of unique domain models. It is worth mentioning that in the field of deep learning, the natural characteristics of biology inspire people, some deep learning methods benefit from the continuous progress of the biological inspiration principles, and deep learning develops rapidly on this basis. However, the traditional artificial intelligence neural network is different from the biological heuristic algorithm; the intelligent computing method inspired by biological phenomena usually adopts an evolutionary learning method, which is the main subset of natural computing. When writing artificial intelligence algorithms, developers usually use a learning network and activation function composed of neurons to make an algorithm have the ability of intelligent learning. For bio-heuristic technology, the main content of the algorithm usually includes the habits of some biological groups,

and some methods of the habits of individual organisms that can carry out subsequent iterations. Biological heuristic computing takes a bottom-up and decentralized approach, and heuristic algorithms that imitate biological intelligence usually have a relatively simple calculation processes, which is a new method for solving complex system problems [19].

In recent years, many biological heuristic intelligent algorithms have been proposed to solve nonlinear problem optimization. Optimization problems with high degrees of nonlinearity, complex constraints, and many control variables need more time and cost to solve. For example, traditional methods, such as the gradient derivation method, cannot solve for an optimal solution of nonlinear optimization effectively, and the computational complexity is also high. So, biological heuristic intelligence algorithms simulate the reproduction, foraging, and migration characteristics of natural organisms to solve the problem. At the present, the development of biological heuristic intelligent algorithms is relatively mature and has been successfully applied in many fields, such as path planning [20,21], mechanical arms [22,23], neural network optimization [24–27], robots [28–30], etc. This chapter will introduce the biological heuristic intelligent algorithm involved in this paper.

After completing the CBSS algorithm design and numerical verification, the CBSS algorithm is compared with different biological heuristic intelligent algorithms in different simulation scenarios, and the performance of the CBSS algorithm is proven to be better than some traditional biological heuristic intelligent algorithms. Moreover, in order to verify the effect of the application of the CBSS algorithm in actual scenarios, the CBSS algorithm is transplanted to a robot operating system to test whether the algorithm can be effectively applied in the path planning scenarios of real mobile robots and further verify the effectiveness of the algorithm optimization and calculation.

Note that traditional intelligent algorithms [31] lack the ability to escape local optimal values and tend to fall into local optimal values when searching for the global optimization of a fitness function. This kind of defect in the performance of path planning for planning a solution path is too long. If obstacles exist in the environment, it may lead to the path of its planning collision with obstacles, and lead to the failure of planning tasks. Firstly, in order to solve this problem, this paper adopts the characteristics of Levi's flight to make the intelligent algorithm have the ability to combine searching big steps and small steps to jump out of the local optimum. Considering the problem that the calculation time of Levi's flight is too long, the speed update of Levi's flight is carried out by probabilistic switching. After the theoretical convergence proof and the numerical test of the benchmark function, the comparative test proves that the CBSS intelligent algorithm proposed in this paper has a better ability to search the optimal value of the benchmark function. Therefore, it can be proven that the CBSS algorithm presented in this paper is more energy efficient [32]. Before ending this introductory section, the main contributions of the paper are highlighted as follows:

- (1) A new meta-heuristic optimization algorithm, CBSS, is proposed in this paper.
- (2) The cubic spline curve is introduced to solve the robot path planning problem.
- (3) A two-dimensional path planning solution combining a bionics algorithm and the cubic spline curve is proposed in this paper.

2. Kinematic Modeling of Heterogeneous Robots

In this section, three kinematic modelings are proposed for heterogeneous mobile robots, i.e., a mobile robot with four-wheel Ackermann steering structure, a two-wheel differential mobile robot, and a robot chassis control module.

2.1. Kinematics Modeling of Robot Chassis Control Module

The chassis control module is an important module for the robot to carry out forward and backward driving and steering functions. In the autonomous navigation system proposed in this section, the chassis control module is located at the bottom of the autonomous navigation system to control the robot's movement instructions. Usually, the chassis control

module integrates and constructs the kinematics model of the robot, inputs its kinematics parameters, and outputs corresponding specific motion rules. In order to enable the autonomous navigation system to be applied to different types of robots, kinematics models of different robots need to be considered, and the two-wheel differential motion model and four-wheel Ackermann steering motion model are the two most commonly used motion models of mobile robots at present. In order to establish the corresponding robot kinematics chassis control module in the autonomous navigation system, Mathematical derivation of these two kinematic models is made in this section.

2.2. Kinematics Modeling of Two-Wheel Differential Mobile Robot

The two-wheel differential kinematic model can be described as the two isomorphic driving wheels of the robot chassis provide the power for the mobile robot to move or turn. Figure 1 shows a schematic diagram of a two-wheeled differential mobile robot equipped with lidar (blue rectangle) turning at any rotation center point. The rotation radius is R , and the mobile robot is moving at angular speeds w and linear speeds v .

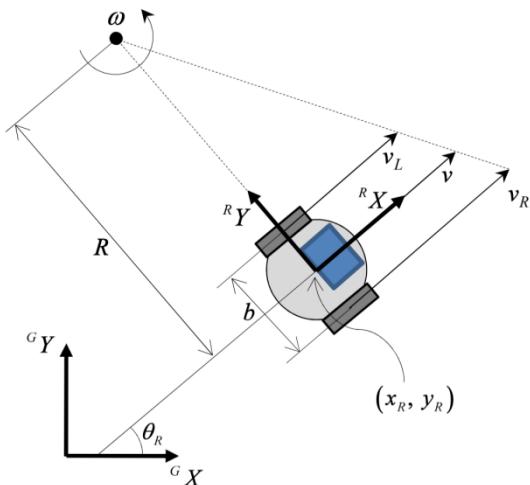


Figure 1. Kinematic model of two-wheel differential robot.

Define the sums of the angular velocity of the two wheels of the robot as w_L and w_R , and the distance between the wheel centers is b . v_L and v_R represent the linear velocity of the left wheel and the right wheel when the robot moves. When $v_L = v_R$, the mobile robot moves in a straight line. When $v_L < v_R$ or $v_L > v_R$, the mobile robot makes a circular turn. When $v_L = -v_R$, the robot rotates in place. The overall linear speed v of robot movement can be expressed as:

$$v = \frac{v_L + v_R}{2} \quad (1)$$

In addition, the robot's turning speed w can be controlled as:

$$w = \frac{v_R - v_L}{b} \quad (2)$$

Therefore, two simultaneous formulas can be obtained:

$$R = \frac{v}{b} = \frac{b}{2} \frac{v_L + v_R}{v_R - v_L} \quad (3)$$

In Figure 1, G_X and G_Y represent the global two-dimensional Cartesian coordinate system in the environment, R_X and R_Y are the robot's coordinate system fixed at the center point of the mobile robot. When the position of the mobile robot relative to the global coordinate system is (x_R, y_R) and the angle relative to the X axis of the global coordinate

system is Θ , the geometric configuration information of the mobile robot can be defined as vector q :

$$q = [x_R \ y_R \ \theta_R]^T \quad (4)$$

The kinematic model of the two-wheel differential mobile robot can be further expressed in the form of matrix:

$$\begin{bmatrix} x_R \\ y_R \\ \theta_R \end{bmatrix} = \begin{bmatrix} \cos(\theta_R) & 0 \\ \sin(\theta_R) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \begin{bmatrix} v_R \\ v_L \end{bmatrix} \quad (5)$$

2.3. Kinematic Modeling of Mobile Robot with Four-Wheel Ackermann Steering Structure

The four-wheel Ackermann model is also called the car steering model because it is the steering model that the car uses to move. Furthermore, a mobile robot with an Ackermann steering structure can complete the steering operation by driving the direction of the front two wheels. When the mobile robot travels in a straight line, the axes of the four wheels remain parallel to the moving direction, while the axes of the tires are perpendicular to the longitudinal center surface of the mobile robot. Figure 2 shows the kinematic model of a four-wheel Ackermann steering structure robot.

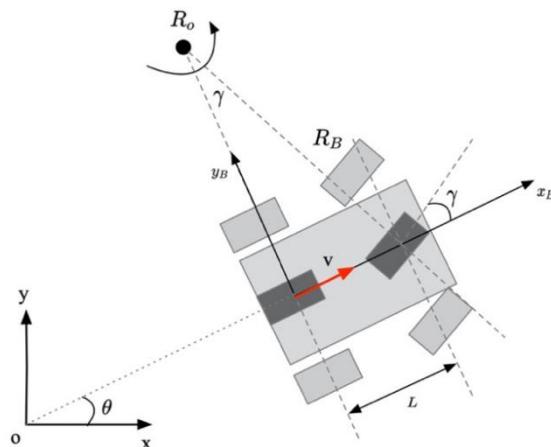


Figure 2. Kinematic model of four-wheel Ackermann steering structure robot.

The global two-dimensional Cartesian coordinate system of the environment is O , and the mobile robot's coordinate system is B . The forward direction of the robot is set as its coordinate axis x_B , the y axis of the robot is perpendicular to the center line of the rear wheel, and the spacing between the two wheels of the mobile robot is L . Under this coordinate axis, the speed of the mobile robot is:

$$v_x = v, v_y = 0 \quad (6)$$

Suppose that a mobile robot is moving in an arc around the center of the circle R_o , and the steering angle of the front wheel of the robot itself is γ . Therefore, from the perspective of the positive direction of the global environment x axis and the forward direction of the robot, the line between the two forms an included angle θ . Then, the angular velocity θ is:

$$\theta = \frac{v}{R_B} \quad (7)$$

which the turning radius R_B is defined as:

$$R_B = \frac{L}{\tan(\gamma)} \quad (8)$$

According to the above conditions, the kinematic equation of the four-wheel Ackermann robot can be defined as:

$$\begin{bmatrix} \theta \\ x \\ y \\ \gamma \end{bmatrix} = \begin{bmatrix} \tan(\gamma) & 0 \\ \frac{L}{\cos(\theta)} & 0 \\ \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} vs. \\ w \end{bmatrix} \quad (9)$$

3. Control Methodology

In this section, three control algorithms are introduced for the path planning and control of heterogeneous mobile robots, namely, the CBSS algorithm, the A* algorithm, and the Dijkstra algorithm.

3.1. CBSS Algorithm

According to bionics theory, Levi's features exist in the trajectory of most biological activities. Similarly, the CBSS algorithm is a group optimization algorithm whose inspiration comes from the biological habits and social behaviors of various organisms. This algorithm aims to solve the problem in most heuristic intelligent algorithms that they become trapped in local optimal value and lack a strategy to escape. Therefore, in this chapter, a biological heuristic intelligent algorithm named the CBSS algorithm is proposed, and Levy flight is integrated into the algorithm to enhance the search ability of individuals in the group. The flow chart of the CBSS algorithm is presented in Figure 3:

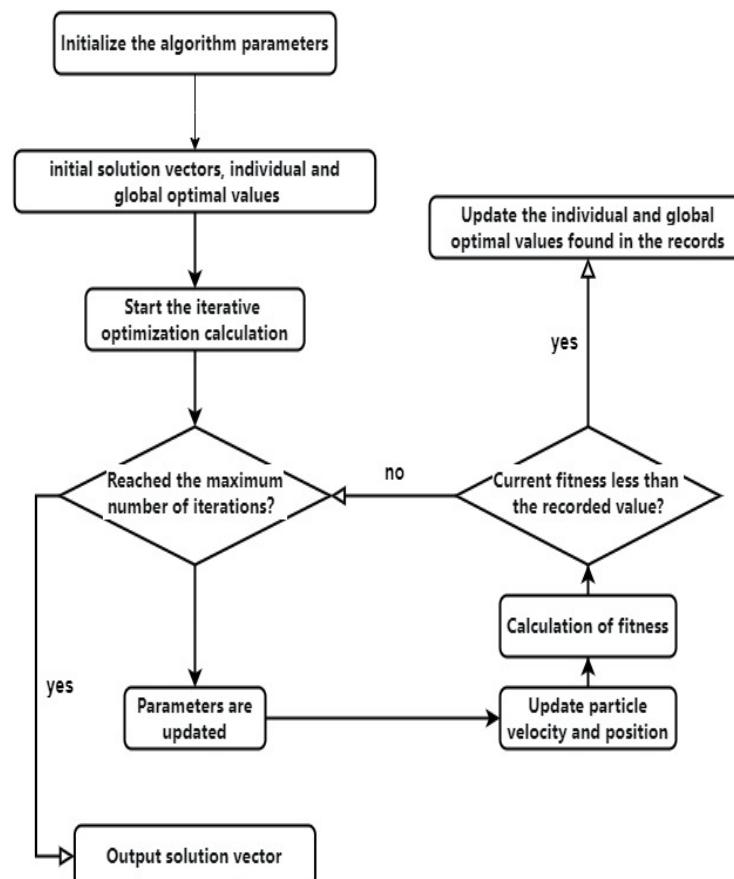


Figure 3. Flow chart of CBSS algorithm.

In the CBSS algorithm, each individual imitates the foraging habit of longicorns: longicorns forage according to the information detected by the two antennae. When individuals

look for targets in each iteration, they will be detected toward the left tentacles X_{lj}^t or right tentacles X_{rj}^t of the individual. In the t iteration of the J th individual, the mathematical description of the left antennal position X_{lj}^t and right antennal position X_{rj}^t is as follows:

$$X_{lj}^t = X_j^t + V_j^t d_0 / 2 \quad (10)$$

$$X_{rj}^t = X_j^t + V_j^t d_0 / 2 \quad (11)$$

In Algorithm 1, the parameters are number of iterations h , number of individuals N , initial step size ξ , speed range V_{min}, V_{max} , width of individuals d_0 , attenuation coefficient τ , probability judgment basis r_s , inertia weight w , cognitive learning factor c_1 , and social learning factor c_2 ;

Algorithm 1 CBSS optimization algorithm

Input: Initialization parameters: $h, N, \xi, V_{min}, V_{max}, d_0, \tau, r_s, w, c_1, c_2$;
 Initial individual initial position X , initial speed V , initial fitness value of each individual $f(x)$;
 Initializing individual optimal solution P_i and algorithmic global optimal solution g_{bst} ;
Output: the global optimal value g_{bst} ;

```

1 : for  $j = 1$  to  $h$ 
2:   First, update the step size  $\xi$ ;
3 :   for  $i = 1$  to  $h$ 
4:      $r_0 = rand()$ ;
5:     if  $r_0 > r_s$  ;
6:       The individual velocity is calculated and updated by multiplying the individual velocity at the last
      moment by the inertial weight  $V$ .
7:       The individual position  $X$  is calculated and updated by multiplying the individual velocity of the
      last moment by the positive constant  $X$ .
8:       Adjust individual positions beyond the solution space;
9 :     else
10:      The individual velocity is calculated and updated by multiplying the inertia weight
        by gaussian random process  $V$ .
11:      Calculate and update the individual position by using the individual position at the last moment
        plus the individual velocity at the last moment times the positive constant;
12:      Adjust individual positions beyond the solution space;
13 :    end
14:    Calculate the current individual fitness value  $g = f(X_i)$ ;
15:    if Individual fitness value  $g <$  global optimal value  $g_{bst}$ ;
16:       $g_{bst} = g$ ;
17:    end
18:    Update the individual's position  $X_j^{t+1}$ ;
19:  end
20:  Update  $V_{max}, V_{min}, \xi, w, c_1, c_2$ ;
21: end
```

3.2. A* Algorithm

Hart et al. [33] proposed the A* algorithm in 1972. The A* algorithm is a heuristic search algorithm in essence, that is, the algorithm is guided to run by certain evaluation indicators. The A* algorithm introduces a cost function and takes it as an evaluation index. The cost function is shown in Formula (12).

$$f(n) = g(n) + h(n) \quad (12)$$

in which $g(n)$ represents the cost from node n to the starting point and is the current actual cost, and $h(n)$ represents the cost from node n to the target point and is the estimated cost. Generally, there are two calculation methods to estimate the cost from node n to the target point: Euclidean distance and Manhattan distance. $g(n)$ represents the total cost of node n .

The advantages of the A* algorithm are its simple calculation method and short planning path, but this method has a large amount of calculation and many inflection points in planning path. Aiming to mitigate the problem of an unsmooth path, Min Haitao [34] added the cost of path curvature into the design of heuristic function to improve the smoothness of path. Furthermore, in order to allow the robot to achieve the optimal navigation path and real-time obstacle avoidance under the condition of complex and bumpy roads, X Ji [35] et al. proposed an optimization algorithm based on the fusion of optimized A* algorithm and the dynamic window approach. There are also some studies on optimizing the generated path to increase the smoothness of the path. Aiming to mitigate the problem that the mobile robot may collide or fail along the planned path in an environment with random obstacles, X Ji et al. [36] proposed a robot path planning scheme that combines the improved A* algorithm with an enhanced dynamic window method. Secondly, the traditional dynamic window algorithm is optimized by adding a state function that is applied to local path planning obstacle avoidance and movement, which also improves the speed of path acquisition.

3.3. Dijkstra Algorithm

Johnson et al. [37] proposed Dijkstra in 1973. The Dijkstra algorithm is mainly used to solve for the shortest distance between one vertex and other vertices in a power graph. Furthermore, the algorithm saves the shortest distance from the starting point to each vertex through an array and saves the vertex corresponding to the traversed shortest path through a set. When the algorithm is implemented, the vertex with the smallest distance from the starting point is obtained from the vertices outside the set every time by iteration, this point is added to the set, and the values in the array are refreshed through this point until the set contains all vertices. $W(index, k)$ is the weight of edge $[index, k]$, and $D(k)$ is the shortest path length from the source node to node K . If the value of $D(k)$ is greater than the sum of $D(index)$ and $W(index, k)$, then

$$D(k) = D(index) + W(index, k) \quad (13)$$

Otherwise, the value of D of k does not change at all.

The Dijkstra algorithm has a strong robustness and can calculate the optimal path solution between two points. However, this algorithm is an undirected search algorithm. As the number of nodes increases, the computational efficiency of this algorithm decreases. Based on the shortcomings of the Dijkstra algorithm, scholars have made many improvements. For example, aiming to solve the path planning problem of an automated guided vehicle (AGV) in intelligent storage, Sun et al. [38] proposed an improved Dijkstra algorithm that combines the eight-angle search method and the Dijkstra algorithm for path optimization. In comparison to the traditional Dijkstra algorithm, the path length planned by the improved Dijkstra algorithm is shorter and the turning angle is less, indicating that the improved algorithm is correct, feasible, effective, and has a strong global search ability. Aiming to improve the path planning and smoothing of mobile robot, Li et al. [39] proposed an improved artificial fish swarm algorithm combined with a continuous piecewise Bezier curve. To solve the problems of low accuracy, the many inflection points, and the long planning path of the traditional artificial fish swarm path planning algorithm, a feasible solution and step size range were introduced on the basis of the Dijkstra algorithm. Moreover, in order to solve the problem of poor convergence and degradation of the algorithm, dynamic feedback range and adaptive step size were introduced. Based on the improved Dijkstra algorithm, Alshammrei et al. [40] designed and implemented an optimal collision free algorithm.

4. Simulations and Comparisons

Robot Operating System (ROS) is a widely used robot application development platform which provides a series of out-of-the-box libraries and toolkits to help software developers develop and build robot SLAM mapping, path navigation, path tracking, and user-defined algorithms, and it is compatible with physical robots. Furthermore, ROS provides functions of function library, visual simulation, message data interaction, hardware interaction, and device drivers. The visual simulation tool of its experiment can import many models that simulate real scenes, making the simulation experiment infinitely close to the real environment. Furthermore, ROS can also be deployed on mobile robots, enabling mobile agents to have a unique development environment on which mobile robots can achieve more functions.

The autonomous navigation of mobile unmanned vehicle in a given environment needs to integrate several modules and realize them at the same time so as to develop an autonomous navigation system that can drive to preset goals and have no collisions in the process. What is more, these modules include: environment map detection and drawing, robot map positioning, path planning, and path tracking modules. In this chapter, the CBSS algorithm is applied in the ROS platform to develop an autonomous navigation system that can navigate in a given environment, and the system can be transplanted to a physical mobile robot for application in a real-world environment.

The autonomous navigation system is set up in such a way that each subsystem operates a specific task to achieve the tasks of the unmanned vehicle. Rviz, a visualization software, can display the moving process of the unmanned vehicle in real time. To put it simply, the laser radar map is used to scan and detect the given environment and provide the corresponding environment information for the subsequent tasks. The CBSS algorithm is used to calculate and optimize the robot's path, which provides several waypoints to minimize the distance traveled by the robot. Furthermore, trajectory tracking is used to drive the mobile robot forward according to the planned path and reach the given target point. The integration of these module functions results in the operation flow of the autonomous navigation system, as shown in Figure 4.

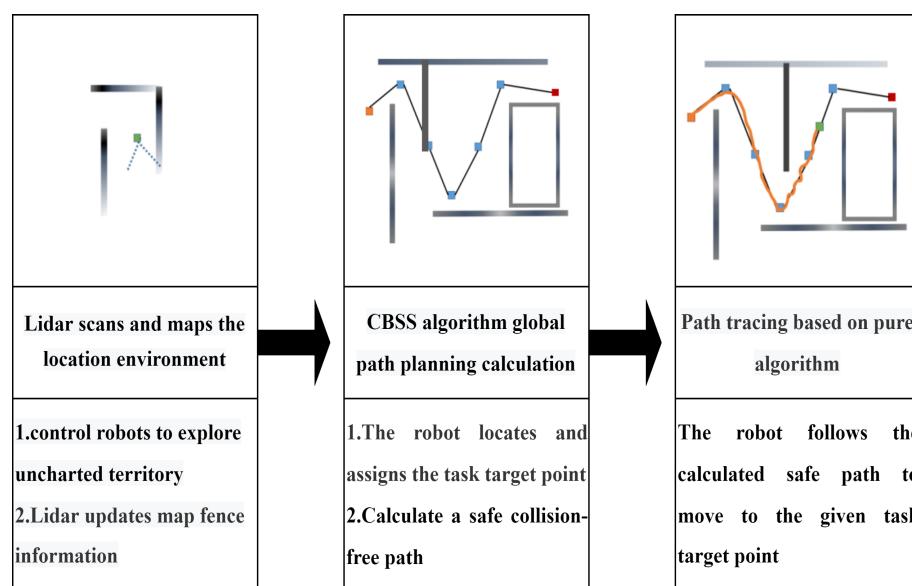


Figure 4. Path Planning process of heterogeneous mobile robots.

4.1. Autonomous Navigation System Settings

The architecture of the autonomous navigation system is divided into four layers: the hardware layer, the data interaction layer, the navigation algorithm layer, and the application layer. Among them, the data interaction layer, navigation algorithm layer, and

application layer can be summarized as the software system in the autonomous navigation system. The overall architecture of the autonomous navigation system is shown in Figure 5.

First of all, when the user needs to perform a robot navigation task to make the mobile robot drive to the user-specified target position, the interaction between the user and the system can be realized through the application layer of the autonomous navigation system. Then, users can set the target points of robot navigation tasks on Rviz visualization software, and Rviz can collect data in real time, dynamically display the process of map construction, and display the driving process of mobile robots on the constructed map in real time.

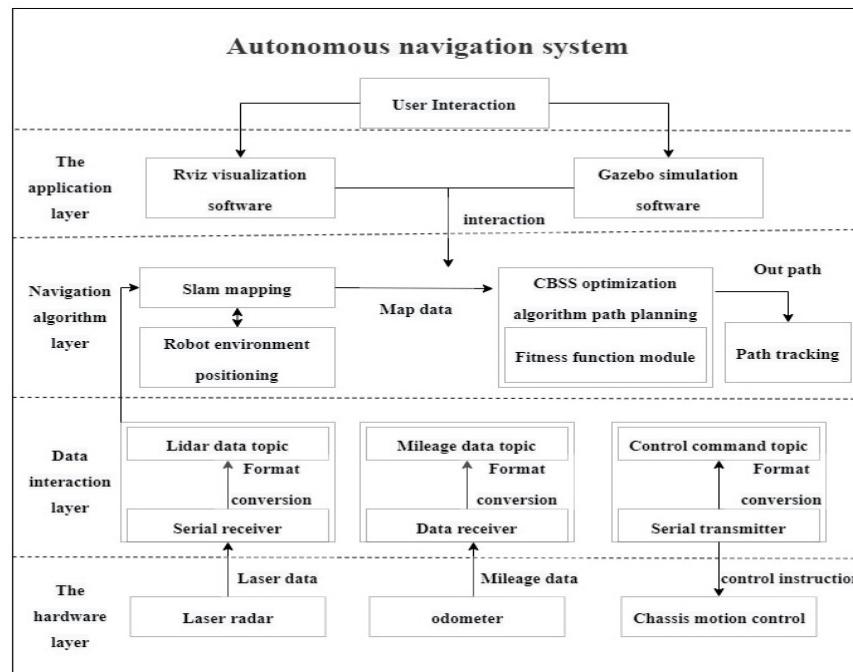


Figure 5. Autonomous navigation system architecture.

Gazebo simulation software allows users to build a task environment in a simulation environment, simulate the entire process of task running, debug code according to the task execution status, and optimize algorithm design. Among the architecture of the autonomous navigation system, the navigation algorithm layer includes key algorithms in the autonomous navigation system, including mobile robot autonomous positioning, map construction, CBSS global path planning algorithm, and pure tracking algorithm, among which the CBSS global path planning algorithm can calculate the shortest path from the robot's current position to the target point according to the known map environment.

The data interaction layer is responsible for interacting with each sensor in the hardware system, converting the data into a format that the hardware system and the software system can receive and process, and publishing the lidar data topics, mileage topics, and control command topics to realize the data communication between modules. Then, the hardware layer mainly obtains information or executes the chassis motion control command of the robot through the hardware. In addition, the laser data of map can be obtained from the laser radar, the mileage data can be obtained from the odometer, the command of serial transmitter can be accepted, and the chassis control command of the mobile robot can be executed according to the kinematic model of the mobile robot.

4.2. Global Path Planning via CBSS Algorithm on Gazebo

This section introduces the application of CBSS in the simulation environment of an autonomous navigation system. Users can build a mobile robot model and environment model through Gazebo, the visual simulator of application layer in autonomous navigation, and test the effectiveness of the algorithm in the simulation environment. Furthermore,

the mobile robot model in Gazebo emulator is consistent with the real robot model. The Gazebo emulator can build a complete 3D rendering environment and support sensor noise simulation, robot cluster, and dynamic physical models. These features make experimental results on the Gazebo simulator more realistic.

In the simulation experiment, MBOT, a mobile robot equipped with lidar and a two-wheel differential motion model, is constructed and applied to perform path planning tasks in this environment. In this experiment, the MBOT mobile robot can move forward and backward, and the robot does not need a steering motor and steering wheel. However, it can turn left and right in place and drive in a narrow environment only by controlling the speed difference between the two driving wheels. The appearance of the mobile robot is shown in Figure 6.

The environment of the simulation experiment was built. First, a wall model was used to build a closed indoor environment, and common indoor furniture such as makeup table and sofa were introduced. Then, a series of obstacles are set in the feasible passage area to construct a $25\text{ m} \times 15\text{ m}$ indoor environment with obstacles, as is shown in Figure 6.

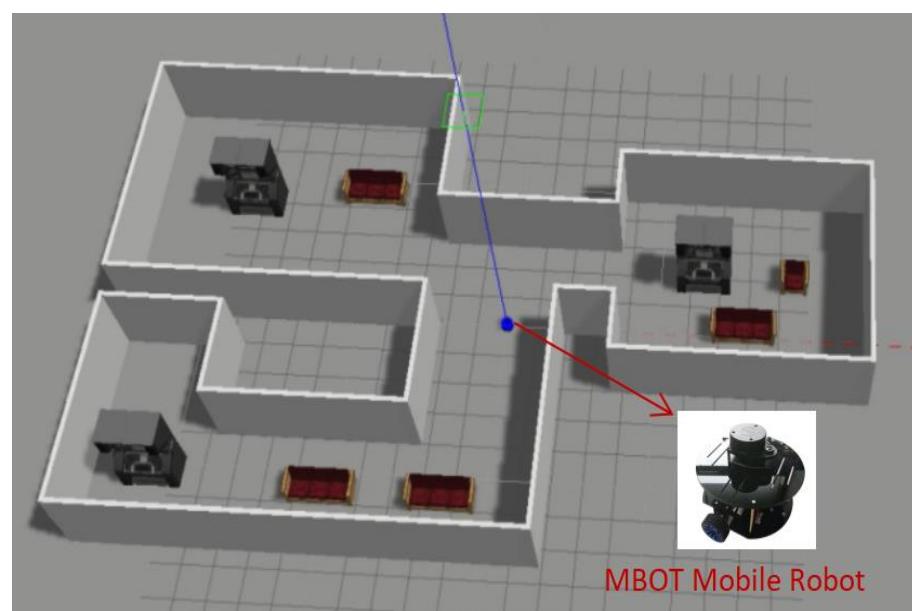


Figure 6. Test scenario for MBOT mobile robot path planning and control.

The preset environment is a more realistic simulation of the indoor environment. Before the path planning experiment, the mobile robot MBOT is placed in the simulation environment under the coordinates of the global environment reference frame, and then the laser scanning SLAM process is started to create a map that can be used for path planning tasks in the future. So, by controlling the moving trajectory of the mobile robot, the mobile robot can gradually explore the whole space. Furthermore, according to the input of laser radar scanning, clearing figure grid will gradually be passable and obstacle information filled in the region and the exploration by six exploratory stage show, as shown in Figure 7, a two-dimensional coordinate system represent the ground plane of the map, when robot to complete the exploration to the environment, a two-dimensional grid map can be obtained.

In global path planning, the CBSS algorithm is used to calculate the path point set of feasible paths from the starting point to the end point of evolution, and a fitness function taking Euclidean distance and obstacle collision into consideration is used as the standard to minimize the search. In addition, two different tasks are performed on the environment map to verify the planning effect of the autonomous navigation system and tracking effect of mobile robots after planning.

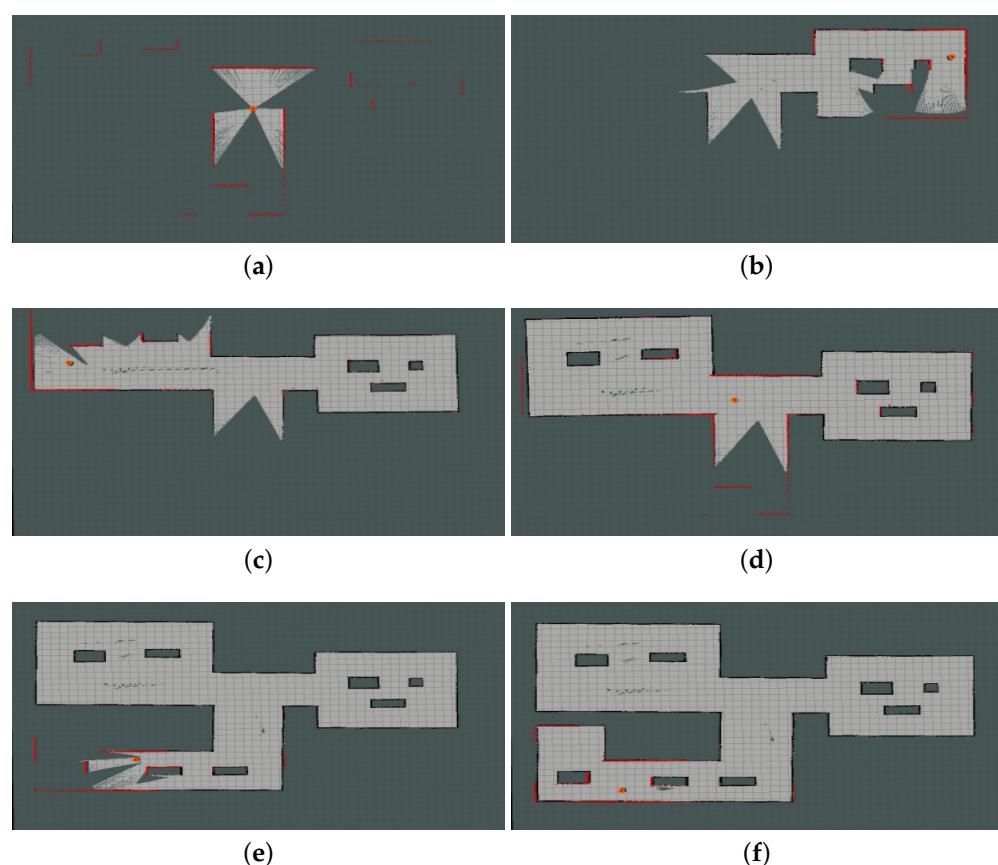


Figure 7. SLAM task process. (a) Phase one. (b) Phase two. (c) Phase three. (d) Phase four. (e) Phase five. (f) Phase six.

The experiment specifies two tasks for the path planning experiment. In the two tasks, the CBSS global path planning algorithm is used to optimize and solve the path planning stage, and it is used as the driving path connecting the current position and target position of mobile robot MBOT in the simulation environment. Furthermore, users can assign tasks by specifying moving target points and robot poses in Rviz visualization software with the mouse. Notably, Gazebo mainly displays mobile robot movement in a 3D environment, while Rviz mainly displays mobile robot path planning results and path tracking process on a 2D raster map. When the car needs to turn or turn for a period of time, it will judge based on the probability judgment according to r_s and update the individual speed V and individual position X .

The effects of task execution were shown in six stages in Rviz visualization software, respectively. In Task 1, the starting point and end point of the task are $(-11.4 \text{ m}, -6.62 \text{ m})$ and $(10.4 \text{ m}, 2.12 \text{ m})$, respectively. The experimental results are shown in Figure 8. Among them, in the global planning algorithm using A* and Dijkstra algorithm, the time required to complete Task 1 is also different, as shown in Table 1. The table contains the times for each algorithm to execute once and the average elapsed time for ten executions, which can better reflect the advantages of algorithm. The average value of ten executions of CBSS is 61.129 s, while the values of the Dijkstra and A* algorithm are 64.938 s and 72.365 s, respectively, which shows that the time and energy consumption of CBSS are better than those of the other algorithms.

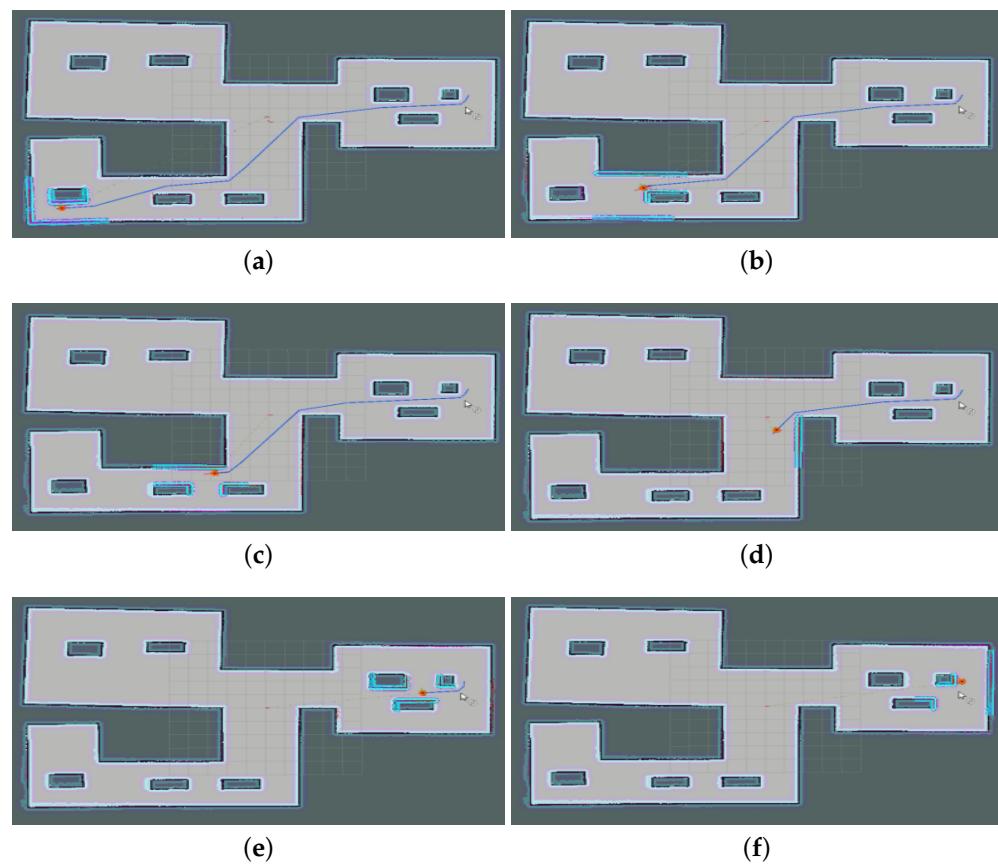


Figure 8. Execution of Task 1 in Rviz. (a) Phase one. (b) Phase two. (c) Phase three. (d) Phase four. (e) Phase five. (f) Phase six.

Table 1. The elapsed time of different algorithms.

Algorithm	Elapsed Time (s)	Average Elapsed Time (s)
Dijkstra	66.125	64.938
A*	71.441	72.365
CBSS	62.268	61.129

In Task 2, the mobile robot needs to drive from the room at the upper left corner of the map to the room at the lower left corner, and the starting point and ending point are set to be $(-11.8 \text{ m}, -6.34 \text{ m})$ and $(-11.2 \text{ m}, 6.36 \text{ m})$, respectively. In addition, the task execution effect of the robot is displayed in six stages in the Gazebo visual simulator and Rviz visual software. The experimental results are shown in Figure 9.

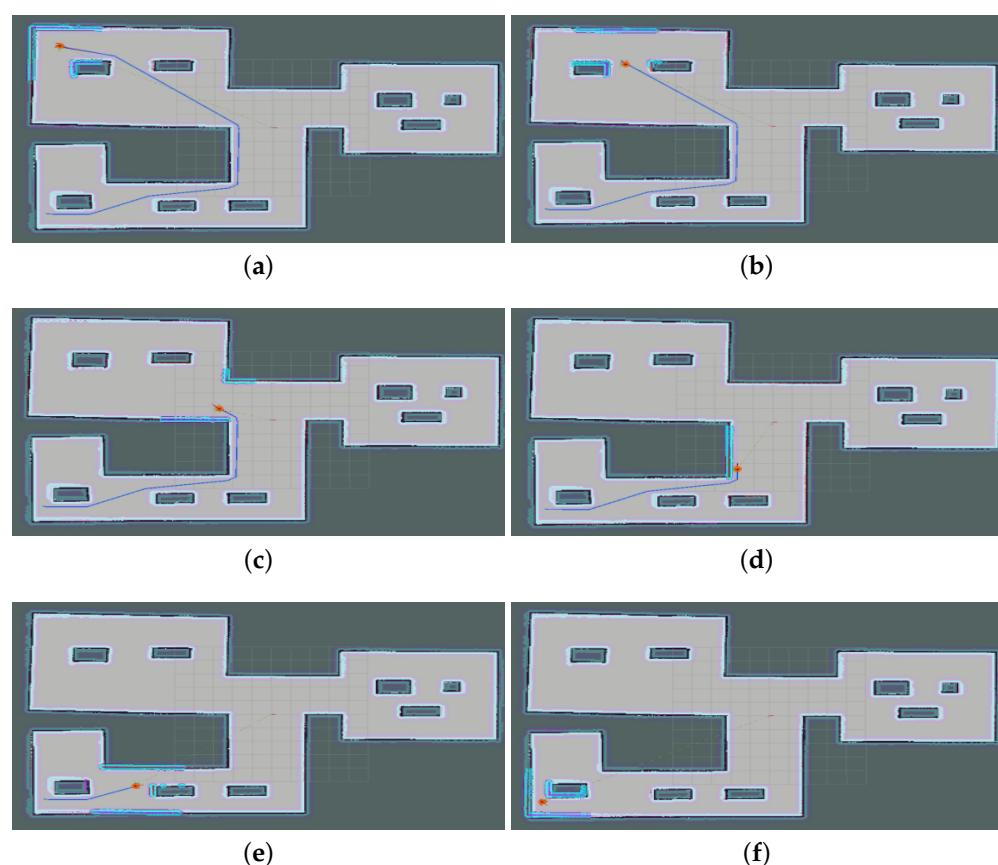


Figure 9. Execution of Task 2 under Rviz. (a) Phase one. (b) Phase two. (c) Phase three. (d) Phase four. (e) Phase five. (f) Phase six.

In Task 1, the mobile robot needs to drive to the environment behind the obstacles in the upper-left corner; these obstacles create narrow-feasibility channels through the obstacles that are required as part of the optimal collision-free path. In the absence of the CBSS algorithm's exploration ability, these narrow channels may not be found in the accessible path, and other path schemes calculate a long distance to the target point. In the planning results, it can be seen that the CBSS global path planning algorithm calculates a safe and successful path through the obstacles. Moreover, with the obstacles on the map under the influence of the expansion coefficient, the planned path always maintains a safe distance from the obstacles, and the mobile robot successfully reaches the end after avoiding the obstacles.

In Task 2, the mobile robot needs to drive from the upper-left corner of the map to the lower-left corner, and there was a wide space in the environment of the middle of the road map. Task 2 has many feasible solution paths, but the optimal path should remain in the broad road on the left. More decentralized problems exist in the environment: path planning needs to find the optimal solution for obstacle avoidance and the short circuit diameter length index. Furthermore, in the solution computed by the CBSS global path planning, the path always maintains a wide distance from the obstacles and the left wall and guarantees a safe distance. In task two, the CBSS algorithm finds an optimal solution path, and the mobile robot can drive to the goal without collision.

In order to verify the reliability of the autonomous navigation system and eliminate the interference of accidental errors, 50 repeated tests were carried out in each of the two tasks. Furthermore, the success times of the mobile robot successfully planning the global path and driving safely to the specified position according to the path were recorded, and the success rate was calculated.

According to the results in Table 2, the overall success rate of both tasks reached more than 95%. The sensor error and random noise will interfere with the overall task execution, but the global path planning based on the CBSS algorithm can still guarantee effectiveness and stability in the autonomous navigation system and enable mobile robots to complete the navigation tasks assigned by users. Hence, in order to make the experimental portion better and more effective, Table 3 shows the planning and execution times of Task 1 and Task 2.

Table 2. The number and success rate of experiments for Task 1 and Task 2.

Experiment Task	Number Successful	Success Rate
Task 1	48	96%
Task 2	49	98%

Table 3. The planning and execution times of Task 1 and Task 2.

Experiment Task	Planning Time (s)	Execution Time (s)
Task 1	1.34	62.268
Task 2	1.94	69.569

Below, the algorithm of this paper and various other bionic algorithms, such as BSO and PSO, are compared with GA in order to verify that the reliability of the system of the autonomous navigation algorithm is higher than the other bionic types. To rule out the influence of other factors in the experiment, this experiment also used 20 repetitions for both the algorithm presented in this paper and the other bionic algorithms. In addition, the path planning fitness value proportion and time proportion results for Tasks 1 and 2 of this paper's algorithm and other bionic algorithms were recorded, as shown in Table 4. In order to more intuitively compare the fitness values and planning times calculated by the CBSS algorithm and the other algorithms, the experimental values are plotted in the form of histograms in Figure 10. The histograms more clearly show the experimental data differences between the CBSS algorithm and other algorithms. In addition, in order to verify that this algorithm is the most energy efficient and that the generated path is the shortest, Table 5 shows that the path generated by this algorithm is significantly shorter than the paths generated by other the algorithms.

Table 4. The proportion of fitness scale of each algorithm's path planning for different tasks.

Task	Fitness Scale				Time Scale			
	CBSS	BSO	PSO	GA	CBSS	BSO	PSO	GA
task1	85.03%	92.28%	93.46%	100.00%	24.23%	38.15%	50.65%	100.00%
task2	78.53%	85.36%	92.18%	100.00%	32.92%	41.02%	45.35%	100.00%

Table 5. Different algorithms generate path lengths for different tasks.

Task	CBSS	Dijkstra	A*	BSO	PSO	GA
Task 1	37.46	41.96	48.31	39.68	42.26	46.43
Task 2	29.86	33.48	36.24	32.18	34.69	38.52

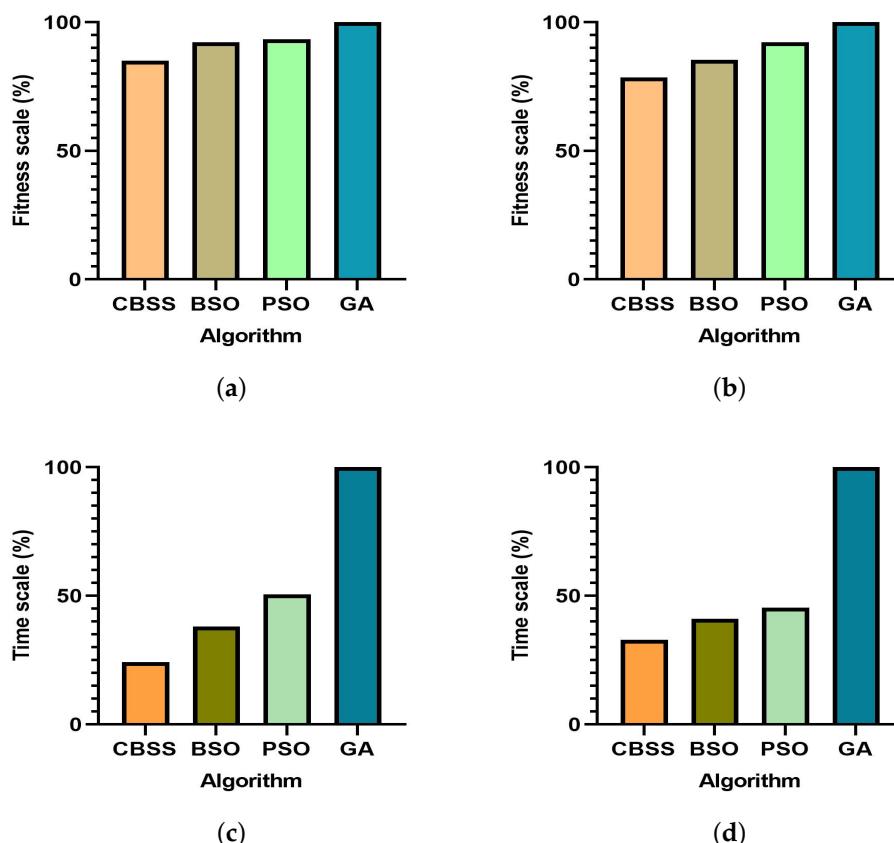


Figure 10. The time scale and fitness scales of the algorithms performing different tasks. (a) Task 1. (b) Task 2. (c) Task 1. (d) Task 2.

4.3. Global Path Planning via CBSS Algorithm on 3D Mountain Topography

In order to further study the performance of the CBSS algorithm in 3D path planning, 3D path planning experiments were carried out on the maps obtained from 3D elevation data modeling in this section, so as to fully verify the quality of CBSS algorithm's path planning under 3D maps of different sizes and then obtain more perfect empirical conclusions. The 3D path planning environmental model is constructed from elevation data, and the distance unit of the 3D elevation map is meters. The map data with elevation data of 30×30 were selected to build the environmental model of route planning. Because of the small map size and gentle terrain, these data were used to test the feasibility of CBSS algorithm in 3D path planning. The CBSS algorithm can successfully plan the path in the map, and the quality of the planned path is better in comparison to the other algorithms in the experiment. To verify the effectiveness of the CBSS algorithm in the path planning of three-dimensional high-level map, in this section, the CBSS algorithm is compared with three heuristic intelligent algorithms, ACO algorithm, PSO algorithm, and BSO algorithm, in four types of maps.

The experiment records the optimal path planned by each algorithm on different maps and draws it into Figure 11 to show the final effect of its planned path. Furthermore, when processing the experimental data, the experiment was repeated for 30 times, and the final fitness value and time cost calculated by each algorithm were recorded, and their average values were taken to eliminate the interference of random errors. The experimental data are shown in Table 6.

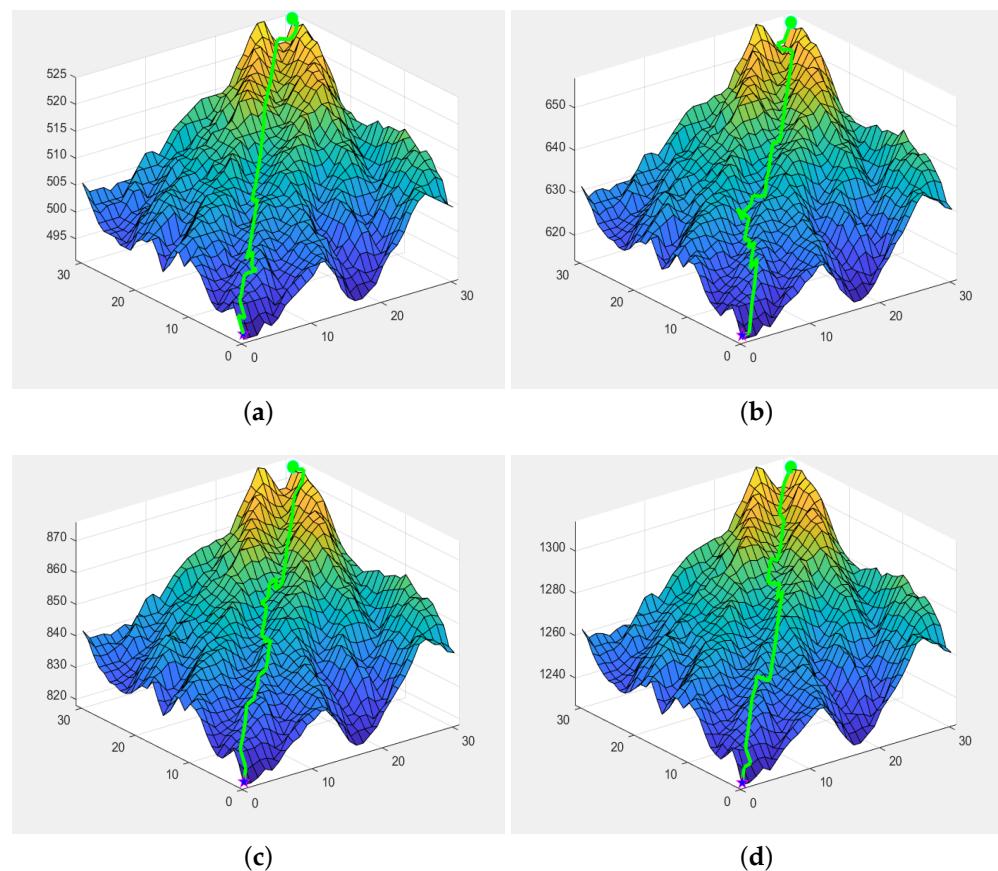


Figure 11. Path planning results of each algorithm in 3D mountain terrain. (a) CBSS. (b) BSO. (c) PSO. (d) GA.

Table 6. The path generation time and distance length of each algorithm in 3D mountain terrain.

Planning Time (s)				Distance Length (m)			
CBSS	BSO	PSO	GA	CBSS	BSO	PSO	GA
13.45	14.15	14.89	15.23	68.4	76.4	94.1	113.6

5. Real-World Mobile Robot

In order to more fully verify the reliability of the CBSS global path planning algorithm and the effect of the autonomous navigation system applied to physical mobile robots and different mobile robot motion models, the Gazebo simulation platform was used to complete a series of verification experiments. The autonomous navigation system was applied to the NanoCar, a solid mobile robot with a four-wheel Ackermann kinematic model, and the corresponding path planning task was completed.

Path Planning of Autonomous Navigation System Using the NanoCar

NanoCar is a mobile robot with a four-wheel Ackermann steering motion model. The robot comes with a full-featured development kit consisting of: 3B Raspberry Pi, CSI camera, battery, encoder, motor, steering gear, and lidar. In addition, the kit can be used for ROS multi-machine communication, OpenCV machine vision, laser SLAM map construction, path tracking, path planning, and other experiments and developments. Component names and locations of the NanoCar mobile robot are shown in Figure 12.

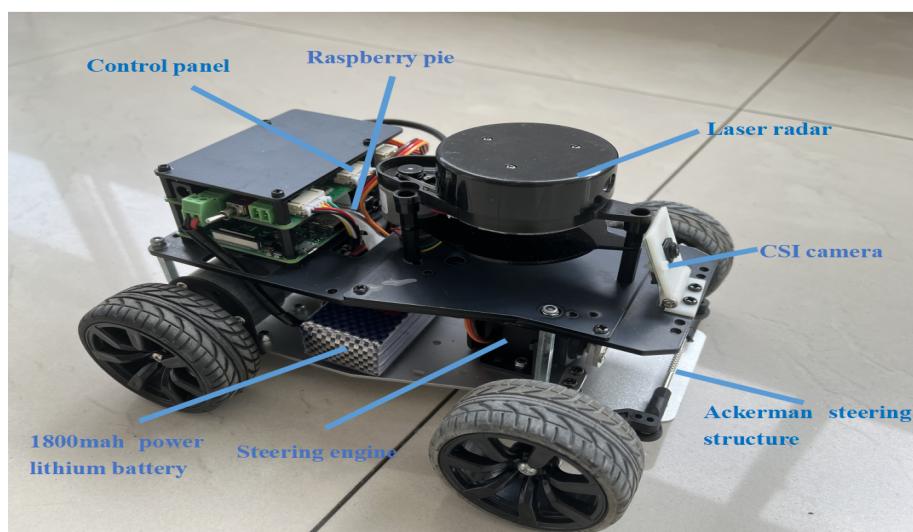


Figure 12. Hardware architecture of NanoCar in experiment.

The NanoCar is 145 mm wide and 152 mm high, with a front and rear length of 210 mm, a front and rear wheelbase of 160 mm, and a mass of about 1 kg. The NanoCar structure is divided into two layers, with the top layer housing the CSI camera, lidar, Raspberry Pi, and motor control panel. Raspberry Pi is equipped with a robot operating system which can be connected to the PC to perform task assignments, and the motor control board can move according to the command speed, execute the speed feedback data report, provide power supply, and perform current detection. The second layer houses the Ackermann steering wheels and motion drive motors controlled by the bottom layer and is equipped with an 1800 mah power lithium battery.

The hardware connection between the NanoCar's scanning and mapping of the environment through Silan radar and the physical navigation process is shown in Figure 13. Remote hosting is provided by the Raspberry Pi with an SSH network connection. When issued a built figure or navigation command, the Raspberry Pi and the robot's operating system receive the command corresponding to the processing calculation, and the instruction is sent by cable to the control panel to control speed. The steering gear control steering commands are sent to the bottom of motor structure realize the movement of the robot and complete the response to the navigation task. The car tracking controller is a PID controller. When the robot is moving, the processor can obtain the calculation information of the accelerometer and lidar and provide feedback to the remote host through the network. So, the user can check the corresponding data obtained by NanoCar during operation in the visualization software on the host.

In terms of the NanoCar navigation system, the autonomous navigation system in this chapter adopts the navigation framework provided by robot operating system platform ROS for developers under the existing architecture, which is mainly distributed in the move_BASE function package. The basic functions are shown in Figure 14. The framework provides path planning, mapping SLAM algorithm, navigation point positioning, data conversion and transmission, transmission control instructions, motor driving, and other contents, and developers can write their own global path planning algorithm or local path planning algorithm.

In the system, the CBSS global path planning algorithm is used as the global path planning algorithm. In the actual operation, a local path planning algorithm must be added to ensure that the optimal path trajectory conforms to the kinematic constraints and ensures the operation efficiency. The dynamic window method is used as the local path planning algorithm in the system, the sensor information is sent to MOVE_base in tf/tfMessage format, the path planning system information is sent to the robot controller in CMD_VEL format, and the hardware platform receives the specific speed information and starts to execute the task. In addition to path planning, the move_base feature pack also provides a

series of plugins for recovery behavior. For example, the cost map recovery plugin can be used to clear and reset the cost map, and the slow movement reset plugin can be used to reset and restore the slow movement behavior of the robot.

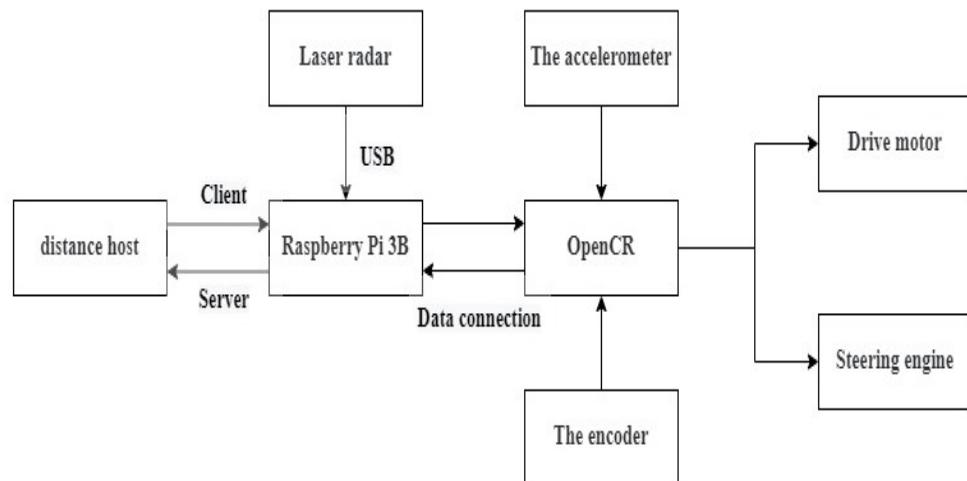


Figure 13. NanoCar Hardware Interaction.

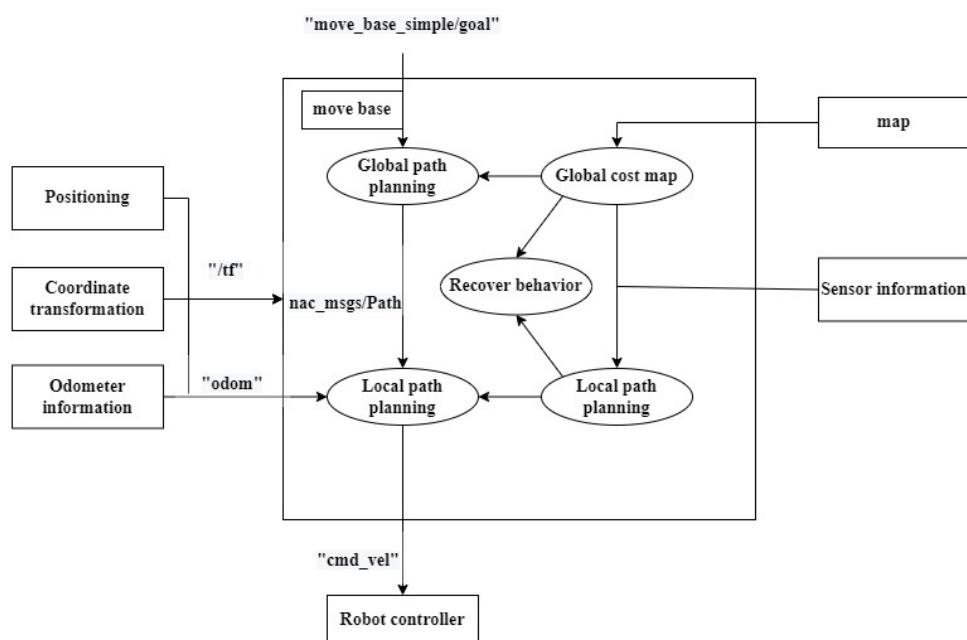


Figure 14. Navigation framework module interaction.

Mobile robots use a lidar-scanned environment map for path planning, which is based on the cost map of the raster map. The global cost map and local cost map constitute the cost map. Among them, the global cost map and local cost map are used for global path planning and local path planning, respectively. If the mobile robot is only regarded as a point on the map when it is working, the actual size of the robot will be ignored, resulting in a collision between the edges of the robot and environmental obstacles. However, the cost map considers the actual size of the mobile robot, and the edges of the obstacle will expand outward into an expansion region based on the expansion coefficient, which is also considered as an obstacle. When the path planning algorithm uses a cost map to calculate, it can ensure that the edge of the mobile robot always maintains a safe distance from the edge of the actual obstacle due to the influence of obstacle expansion.

The test site adopts a laboratory environment. Objects such as cartons and cardboard were placed in the laboratory to simulate the obstacle environment, and the scene was set up and the path planning experiment was carried out. NanoCar needed to bypass Obstacle 1 and Obstacle 2 from the starting position and finally drive to the end point. The experimental scene is shown in Figure 15.

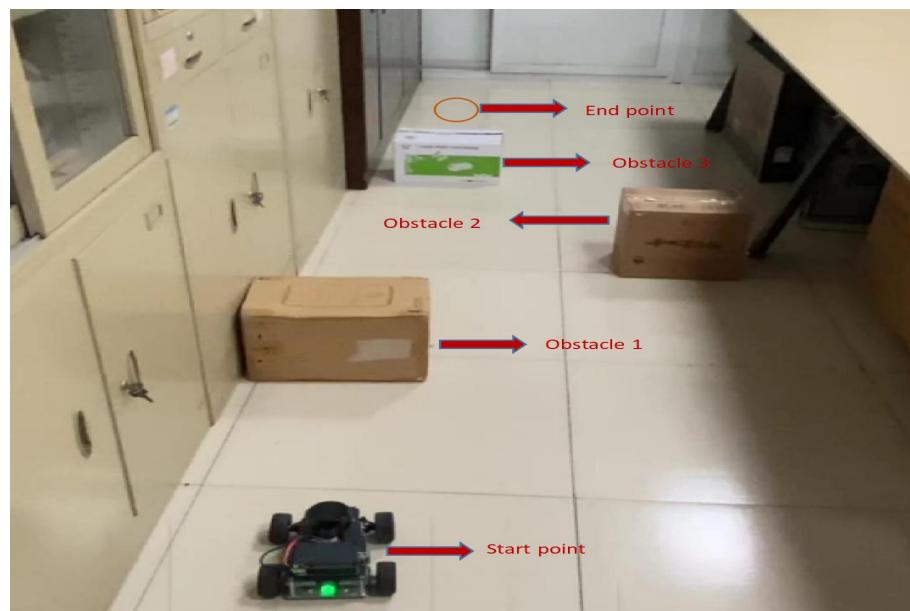


Figure 15. Experimental environment of NanoCar.

Once the navigation end point is set, the mobile robot will encounter obstacles in the process of driving to the end point. Note that the robot avoids the obstacles through the path planning algorithm, and finally drives to the set goal. The driving process of mobile robot is shown in Figure 16. The planned path of mobile robot is displayed in the blue line segment, and its traveling track is drawn in a red curve. As a result, experiments show that the mobile robot equipped with autonomous navigation system can effectively avoid obstacles in driving and plan a safe path to drive safely to the target point given by the user. In order to verify that the performance of the algorithm in this paper is better than that of other algorithms in the real scene, 20 experiments were repeatedly conducted with the CBSS, A*, Dijkstra, BSO, PSO, GA algorithms. As shown in Table 7, the experimental results show that the average planning and average execution time of the algorithm in this paper are better than other algorithms in the real scene.

Table 7. The average planning and average execution time of experiment.

Algorithm	Average Planning Time (s)	Average Execution Time (s)
CBSS	0.68	4.23
Dijkstra	1.56	4.94
A*	2.35	5.32
BSO	1.64	5.04
PSO	1.86	5.29
GA	2.93	5.98

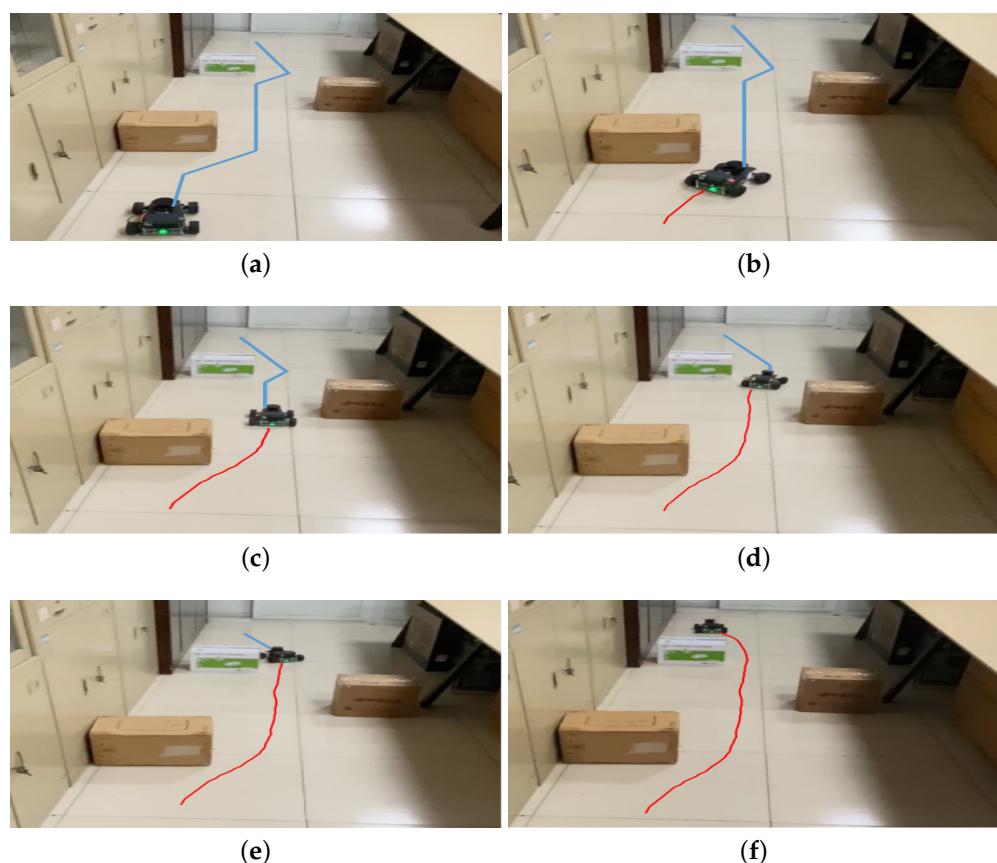


Figure 16. Task execution process of NanoCar. (a) Phase one. (b) Phase two. (c) Phase three. (d) Phase four. (e) Phase five. (f) Phase six.

6. Conclusions

In this paper, a new meta-heuristic path planning algorithm called CBSS has been introduced to solve the path planning problems of heterogeneous mobile robots. In addition, comprehensive computer simulations were conducted to verify the accuracy, search speed, energy efficiency, and stability of the CBSS algorithm. The results of the real-world experiment have proven that the proposed CBSS method is much better than its counterparts.

Although the CBSS algorithm performs well in numerical verification, 2D and 3D environments, and the algorithm has also been successfully applied in the robot operating system and the physical mobile robot NanoCar, there are still some limitations in the research, mainly involving the following aspects:

- (1) The path planning algorithm in this paper is executed in a static environment, and the path planning solution in the dynamic environment path scenario is not considered.
- (2) The mobile robot has a relatively single sensor system.

The future directions are listed as the following points:

- (1) The application of this algorithm to two-wheel differential mobile robots or four-wheel differential mobile robots is to be considered. The effectiveness of the algorithm can be further verified by lateral comparison experiments on different kinematic models and different types of solid mobile robots.
- (2) The camera picture data, lidar data, and IMU, etc., are to be fused to improve the accuracy and the robustness of the algorithm in the navigation system.
- (3) The local dynamic planning problem is to be considered.

Author Contributions: Conceptualization, D.C., Z.W., G.Z. and S.L.; methodology, D.C., Z.W., G.Z. and S.L.; software, D.C., Z.W., G.Z. and S.L.; validation, D.C., Z.W., G.Z. and S.L.; formal analysis,

D.C., Z.W., G.Z. and S.L.; investigation, D.C., Z.W., G.Z. and S.L.; resources, D.C., Z.W., G.Z. and S.L.; data curation, D.C., Z.W., G.Z. and S.L.; writing—original draft preparation, D.C.; writing—review and editing, D.C., Z.W., G.Z. and S.L.; visualization, D.C., Z.W., G.Z. and S.L.; supervision, D.C., Z.W., G.Z. and S.L.; project administration, D.C., Z.W., G.Z. and S.L.; funding acquisition, Z.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant 62276085 and Grant 61906054, and in part by the Natural Science Foundation of Zhejiang Province under Grant LY21-F030006.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Not applicable.

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

References

1. Zhou, G.; Chen, D.; Gu, R.; Li, S. Cuckoo-Beetle Swarm Search for Nonlinear Optimization: A New Meta-Heuristic Algorithm. In Proceedings of the 2021 7th International Conference on Systems and Informatics (ICSAI) IEEE, Chongqing, China, 13–15 November 2021; pp. 1–6
2. Pati, C.S.; Kala, R. Vision-based robot following using pid control. *Technologies* **2017**, *5*, 34. [[CrossRef](#)]
3. Yang, Z.; Sun, Z.; Piao, H.; Zhao, Y.; Zhou, D.; Kong, W.; Zhang, K. An autonomous attack guidance method with high aiming precision for UCAV based on adaptive fuzzy control under model predictive control framework. *Appl. Sci.* **2020**, *10*, 5677. [[CrossRef](#)]
4. Mun, P.T.; Ryong, L.S.; Hak, Y. Workspace mapping with adaptive fuzzy control for robotic manipulator in teleoperation. *J. Mech. Sci. Technol.* **2020**, *34*, 2171–2178.
5. Lewis, F.W.; Jagannathan, S.; Yesildirak, A. *Neural Network Control of Robot Manipulators and Non-Linear Systems*; CRC Press: Boca Raton, FL, USA, 2020.
6. Jisoo, K.; Keeyoung, C. Development of an Artificial Neural Network Control Allocation Algorithm for Small Tailless Aircraft Based on Dynamic Allocation Method. *Int. J. Aeronaut. Space Sci.* **2022**, *23*, 363–378
7. Han, H.; Wu, X.; Qiao, J. Design of robust sliding mode control with adaptive reaching law. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *50*, 4415–4424. [[CrossRef](#)]
8. Mechali, O.; Xu, L.; Xie, X.; Iqbal, J. Theory and practice for autonomous formation flight of quadrotors via distributed robust sliding mode control protocol with fixed-time stability guarantee. *Control Eng. Pract.* **2022**, *123*, 105150. [[CrossRef](#)]
9. Buddhadeva, S.; Keshari, R.S.; Kumar, R.P. Robust control approach for stability and power quality improvement in electric car. *Int. Trans. Electr. Energy Syst.* **2020**, *30*, e12628.
10. Osadchy, S.I.; Zozulya, V.A.; Ladanyuk, A.P.; Vikhrova, L.G.; Kalich, V.M. Optimal Robust Control of a Robots Group. *Autom. Control Comput. Sci.* **2019**, *53*, 298–309. [[CrossRef](#)]
11. Guo, H.; Cao, D.; Chen, H.; Sun, Z.; Hu, Y. Model predictive path following control for autonomous cars considering a measurable disturbance: Implementation, testing, and verification. *Mech. Syst. Signal Process.* **2019**, *118*, 41–60. [[CrossRef](#)]
12. Mohammad, R.; Navid, M.; Saeid, N.; Shady, M. Model predictive control with learned vehicle dynamics for autonomous vehicle path tracking. *IEEE Access* **2021**, *9*, 128233–128249.
13. Santiranjan, P.; Shrikant, T.S. Kinematic synthesis of centrallever steering mechanism for four wheel vehicles. *Acta Polytech.* **2020**, *60*, 252–258. [[CrossRef](#)]
14. Tu, X.; Gai, J.; Tang, L. Robust navigation control of a 4WD/4WS agricultural robotic vehicle. *Comput. Electron. Agric.* **2019**, *164*, 104892. [[CrossRef](#)]
15. Tian, Y.; Cao, X.; Wang, X.; Zhao, Y. Four wheel independent drive electric vehicle lateral stability control strategy. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 1542–1554. [[CrossRef](#)]
16. Wang, C.; Heng, B.; Zhao, W. Yaw and lateral stability control for four-wheel-independent steering and four-wheel-independent driving electric vehicle. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2020**, *234*, 409–422. [[CrossRef](#)]
17. Li, Z.; Zhao, Y.; Xu, K.; Wu, J.; Kong, Q.; Liu, X. Research on Stability Control Method of Vehicle Based on Steering Torque Superposition. *IEEE Access* **2020**, *8*, 80518–80526. [[CrossRef](#)]
18. Zhang, H.; Jiang, H.; Luo, Y.; Xiao, G. Data-driven optimal consensus control for discrete-time multi-agent systems with unknown dynamics using reinforcement learning method. *IEEE Trans. Ind. Electron.* **2016**, *64*, 4091–4100. [[CrossRef](#)]
19. Ibrahim, G.J.; Rashid, T.A.; Akinsolu, M.O. An energy efficient service composition mechanism using a hybrid meta-heuristic algorithm in a mobile cloud environment. *J. Parallel Distrib. Comput.* **2020**, *143*, 77–87. [[CrossRef](#)]
20. Jiang, X.; Lin, Z.; He, T.; Ma, X.; Ma, S.; Li, S. Optimal path finding with beetle antennae search algorithm by using ant colony optimization initialization and different searching strategies. *IEEE Access* **2020**, *8*, 15459–15471. [[CrossRef](#)]

21. Nasir, G.H.; Károly, J. Dynamic differential annealed optimization: New metaheuristic optimization algorithm for engineering applications. *Appl. Soft Comput.* **2020**, *93*, 106392.
22. Chen, D.; Li, S.; Wu, Q. A novel supertwisting zeroing neural network with application to mobile robot manipulators. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 1776–1787. [[CrossRef](#)]
23. Chen, D.; Li, S.; Wu, Q.; Luo, X. New disturbance rejection constraint for redundant robot manipulators: An optimization perspective. *IEEE Trans. Ind. Inf.* **2019**, *16*, 2221–2232. [[CrossRef](#)]
24. Chen, D.; Li, S. DRDNN: A robust model for time-variant nonlinear optimization under multiple equality and inequality constraints. *Neurocomputing* **2022**, *511*, 5939–5954. [[CrossRef](#)]
25. Ali, S.; Hassan, S.; Anupam, Y. A dynamic metaheuristic optimization model inspired by biological nervous systems: Neural network algorithm. *Appl. Soft Comput.* **2018**, *71*, 747–782.
26. Han, F.; Jiang, J.; Ling, Q.-H.; Su, B.-Y. A survey on metaheuristic optimization for random single-hidden layer feedforward neural network. *Neurocomputing* **2019**, *335*, 261–273. [[CrossRef](#)]
27. Chen, D.; Li, X.; Li, S. A novel convolutional neural network model based on beetle antennae search optimization algorithm for computerized tomography diagnosis. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**. [[CrossRef](#)]
28. Chen, D.; Li, S.; Li, W.; Wu, Q. A multi-level simultaneous minimization scheme applied to jerk-bounded redundant robot manipulators. *IEEE Trans. Autom. Sci. Eng.* **2019**, *17*, 463–474. [[CrossRef](#)]
29. Chen, D.; Zhang, Y.; Li, S. Tracking control of robot manipulators with unknown models: A jacobian-matrix-adaption method. *IEEE Trans. Ind. Inf.* **2017**, *14*, 3044–3053. [[CrossRef](#)]
30. Chen, D.; Li, S.; Lin, F.-J.; Wu, Q. New super-twisting zeroing neural-dynamics model for tracking control of parallel robots: A finite-time and robust solution. *IEEE Trans. Cybern.* **2019**, *50*, 2651–2660. [[CrossRef](#)]
31. Zhu, Y.; Huang, C.; Wang, Y.; Wang, J. Application of bionic algorithm based on CS-SVR and BA-SVR in short-term traffic state prediction modeling of urban road. *Sustainability* **2022**, *23*, 1141–1151. [[CrossRef](#)]
32. Alexey, B.; Alexander, V.; Vadim, B.; Vladimir, P. Conductor Identification Using Acoustic Signal Method. *Int. J. Automot. Technol.* **2022**, *60*, 1141–1151.
33. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [[CrossRef](#)]
34. Min, H.; Xiong, X.; Wang, P.; Yu, Y. Autonomous driving path planning algorithm based on improved A* algorithm in unstructured environment. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2021**, *235*, 513–526. [[CrossRef](#)]
35. Ji, X.; Feng, S.; Han, Q.; Yin, H.; Yu, S. Improvement and fusion of a* algorithm and dynamic window approach considering complex environmental information. *Arab. J. Sci. Eng.* **2021**, *46*, 7445–7459. [[CrossRef](#)]
36. Yang, H.; Teng, X. Mobile Robot Path Planning Based on Enhanced Dynamic Window Approach and Improved A Algorithm. *J. Robot.* **2022**, *2022*, 2183229. [[CrossRef](#)]
37. Johnson, D.B. A note on Dijkstra's shortest path algorithm. *J. ACM (JACM)* **1973**, *20*, 385–388. [[CrossRef](#)]
38. Yinghui, S.; Ming, F.; Yixin, S. AGV path planning based on improved Dijkstra algorithm. *J. Phys. Conf. Ser.* **2021**, *1746*, 012052.
39. Li, F.-F.; Du, Y.; Jia, K.-J. Path planning and smoothing of mobile robot based on improved artificial fish swarm algorithm. *Sci. Rep.* **2022**, *12*, 659. [[CrossRef](#)]
40. Shaher, A.; Sahbi, B.; Lioua, K. Improved Dijkstra Algorithm for Mobile Robot Path Planning and Obstacle Avoidance. *CMC-Comput. Mater. Contin.* **2022**, *18*, 5939–5954.