Mobile Networks and Applications (2023) 28:1806–1817
https://doi.org/10.1007/s11036-023-02247-y

RESEARCH

Shortest Path Control for Target Searching Using Robot in Complex Task with Large Range

Jinyin Peng[1] · **Li Zhao**[2]

Accepted: 4 September 2023 / Published online: 16 September 2023

Abstract
For target searching with robot in large search area, path planning spends long time and is affected by the surrounding environment, which makes the efficiency of the whole searching is not high. In this way, a shortest path control method for target searching using robot in a large range of complex tasks is proposed. This method utilizes environmental information collected by laser sensors to establish an environmental map for searching, and uses slime mold optimization algorithm to find the shortest path and avoids collision with obstacles. The objective function for robot's target path searching is established to obtain the shortest path. To ensure the robot's movement is optimal, a path controller is applied during the robot's target searching process. The experimental results show that the proposed method effectively avoids obstacles in the path, the planned path distance is kept within 5 m, and the path planning time is less than 50 s, which indicates that the proposed method has high path planning efficiency.

Keywords Environmental map · Slime optimization algorithm · Target searching · Path planning · Mobile control

1 Introduction

Robots are mainly used in the industrial field to replace people to complete search, rescue, and other complex works in dangerous or harsh environment. They can also be used in the military field to complete reconnaissance, demining and other tasks, as well as in the other fileds like medical and aerospace [1]. The tasks in the above fields are classified from the perspectives, mainly including target searching, rescuing, mechanical activities, exploration and humanoid applications like football etc. [2]. When working in complex and dangerous environments, there are more dangerous factors in manual operations, which threaten the life safety of workers. The multi-robot system collaborative target searching has broad application prospects in resource exploration, military anti-terrorism, post-disaster search and rescue, and replacing human beings to perform dangerous tasks in special environments [3].

Wang et al. [4] simplified obstacles into two situations: continuous and non-continuous obstacles, and moved to a specific boundary according to the situation. Particle swarm optimization (PSO) for target position estimation used the acquired target signal to estimate the target position, and reached the vicinity of the target, so as to achieve target search. There were obstacles in the planned path of this method, and the obstacle avoidance ability was poor. Gao et al. [5] provided a probability calculation model based on the actual working environment, built a probability map based on the model, and then proposed a robot target search path planning method with the expected shortest time as the optimization index. The method adopted a hierarchical planning mode and carried out topological point sequence planning in the upper topological map. The local path planning between topological points in the lower feature map was too long, and the path planning effect was not ideal. Fan et al. [6] proposed a target search and rounding framework for swarm robots based on consensus initiative, which improved the anti-ant colony algorithm and added a variety of pheromones to help swarm robots cooperate to explore the environment and generate pheromone maps. At the same time, the framework combined the pheromone map

✉ Li Zhao
zhaoli@mail.hniu.cn

[1] School of Mechanical and Electrical Engineering, Hainan Vocational University of Science and Technology, Haikou 570000, China

[2] Software Department, Hunan College of Information, Changsha 410000, China

generated in the previous stage with the hierarchical gene regulatory network model to complete the task of searching and rounding up the dynamic target of swarm robots in the scene with unknown environmental information and limited communication. The path planning time required by this method was too long, and there were problems of low planning efficiency. Kumar et al. [7] proposed a path planning method for autonomous robots based on an improved grey wolf optimization method. By modifying and optimizing the traditional grey wolf optimization algorithm, the improved method can find the optimal path and maintain a safe distance from other objects and robots, with high efficiency and feasibility. However, this method required multiple iterations to achieve good results, resulting in a longer execution time. Nascimento et al. [8] proposed probabilistic foam based safe path planning algorithm for mobile robots, which ensured a safe path through a series of bubble structure constraints and provided a safe area. However, when performing path planning, the probabilistic foam method needed to build a series of bubble structures to limit the safe path and cover the free configuration space. This process required much calculation and time. Durakl et al. [9] proposed a new path planning method for mobile robots based on Bezier curves, using grid graphs to model the environment and traditional algorithms to find paths between the starting and ending points. By pruning based on Bezier curves to discard excess nodes and smoothing peaks, path planning was achieved. This method had high computational complexity and required a significant amount of computing resources and time, resulting in reduced planning efficiency.

The existing methods have some problems such as poor obstacle avoidance ability, long path and low planning efficiency. Therefore, the shortest path control method for robot target search in a large range of complex tasks is proposed. The main contributions of this article's method are shown as below.

(1) Establishment of environmental map: Collect environmental information through laser sensors, and use line extraction, line matching, and line fitting techniques to establish an environmental map for target search. This method can accurately describe obstacles and target points in the environment, providing an important data foundation for subsequent path planning.
(2) Application of slime mold optimization: By using slime mold optimization algorithm, the shortest path is found by establishing an objective function for target searching path, avoiding collision with obstacles. Compared to other shortest path planning algorithms, slime mold optimization algorithms has certain advantages in handling complex environments and large-scale tasks.
(3) Path control design: By designing a controller, the robot is ensured to move along the optimal path and complete the target search task. This controller can respond in real-time to environmental changes and obstacle dynamics, ensuring that the robot can efficiently and safely complete tasks.

2 The Construction of Environment Map

The sensor carried by the robot detects environmental information, extracts the features of obstacles and targets, and generates a map that the robot can recognize. Maps are represented differently due to different sensors. The proposed method uses a laser sensor [10, 11], which collects a large number of information points per cycle with high accuracy, and can quickly collect data points at a higher frequency, making it suitable for extracting straight line features. Moreover, the linear map has low-level topological information and is suitable for a wide range of environments, a straight line is a line segment with a simple geometric shape, which is more convenient in description and modeling. In contrast, curves have complex shapes that require more data to describe and are more complex in processing and calculation. And line maps provide intuitive path information, making it easier for robots to plan and navigate, improving planning efficiency, so the linear map is chosen as the description method of the environmental model.

The construction of environmental maps is mainly divided into three steps: line extraction, line matching, and line fitting. The information about obstacles and target points collected through laser sensors is processed to extract straight line segments. Match these line segments with the generated line segments in the global map to determine if they are related. If there is no correlation, directly import the global map; If there is correlation, perform line fitting, generate new line segments, and update the global line map.

2.1 Straight-Line Extraction

2.1.1 Data Preprocessing

The laser sensor can obtain the distance information of the object within a range of 180°to ensure the accuracy of the measurement, only the monitoring data of the object within a certain range of the distance robot is extracted. These data are represented by the polar coordinates of the robot. In order to facilitate calculation, the data represented by polar coordinates are converted into data information in Cartesian coordinate system. The schematic diagram of coordinate transformation is shown in Fig. 1.

In Fig. 1, the point J is the information point of an object detected by the laser sensor, β indicates the angle of OJ to the pole axis, OX and the point J Cartesian coordinates are:

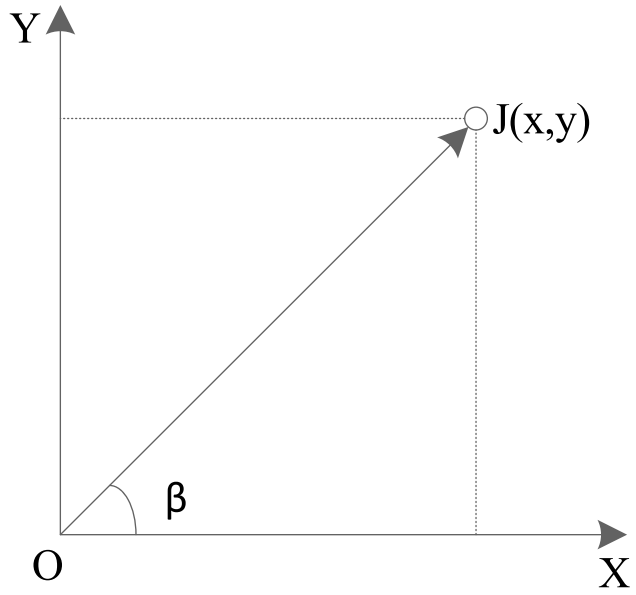


Fig. 1 Diagram of coordinate transformation

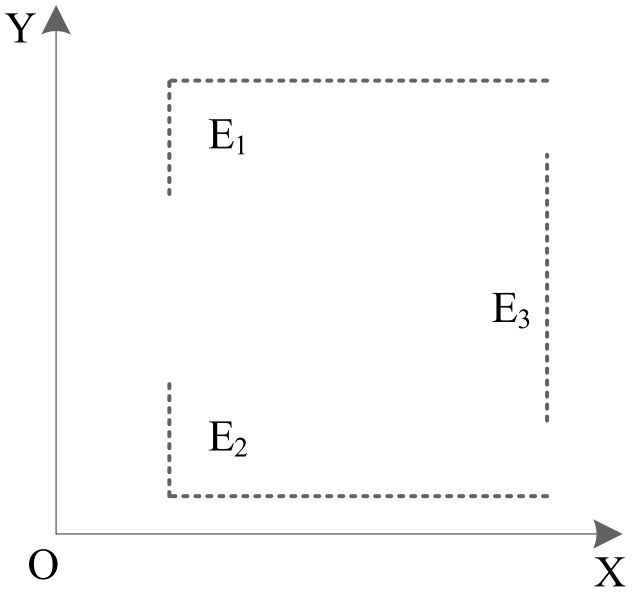


Fig. 2 Diagram of region division

$$\begin{cases} x = r\cos\beta \\ y = r\sin\beta \end{cases} \tag{1}$$

where r represents the distance between the point J and the origin O of the coordinate system. Through the above method, the polar coordinates of the objects detected by the laser sensor are converted to Cartesian coordinates.

2.1.2 Regional Division

After obtaining the right-angle coordinates of the obstacle, the coordinates of these points are divided into disconnected areas according to the size of the straight-line distance between the two points, and they are processed separately. The schematic diagram of regional division is shown in Fig. 2. There are obvious spacing between different obstacles E_1, E_2 E_3, which belong to the three regions that are not connected.

For the robot, after extracting the information of the points, a threshold F_1 is set to judge whether they are the same object. If the distance between two points is less than the threshold F_1, the two points are considered to be continuous and belong to the same object. If the distance between two points is greater than the threshold value F_1, the two points are considered to be the division points of the region, respectively representing an object, is not connected.

The specific judgment process is as follows:

(1) Calculate the distance f_i between the two points:
(2) To judge the relationship between f_i and the threshold value F_i, the region is divided into two parts when $f_i > F_1$ takes (x_i, y_i) as the split points, the regions $E_1\{(x_0, y_0), (x_1, y_1), Z(x_1, y_1)\}$ and $E_N\{(x_{i+1}, y_{i+1}), (x_{i+2}, y_{i+2}), Z\}$ can be obtained, and the E_N is treated in the same way, thus obtaining E_2 and E_3. And so on, and the whole area $(E_1, E_2, E_3, L, E_{n-1}, E_n)$ can be eventually get.

2.1.3 Straight Line Segmentation

The IEPF algorithm is used to segment the line segment. The algorithm is an iterative algorithm, and the line segment is segmented by setting a fixed threshold. The region segmentation diagram is shown in Fig. 3.

All the points in Fig. 3(a) are the set of one of the regions E_i. Take the first point A and the last point D as the line segment Z_1, and calculate the distance from the remaining points in this region to Z_1. As can be seen from the figure, the maximum value P_1 is obtained at point P, and a distance threshold F_2 for line segment segmentation is set. If P_1 is less than threshold F_2, the region can be represented by a straight line segment; if P_1 is greater than threshold F_2, the region is divided into S_1 and S_1 by using point C as the segmentation point, and then the region S_1 and S_1 are processed by the same method.

As shown in Fig. 3(b), connect points A and C to form A straight segment Z_2 in the region S_1, the maximum distance from B point to Z_2 is P_2, and then the size of it and threshold F_2 is determined. By this on, until the maximum distance between the beginning and the end of the region is less than the threshold value F_2, to obtain a set of N points that can be represented by a straight segment. The schematic representation of the region E_i segmentation is shown in Fig. 4.

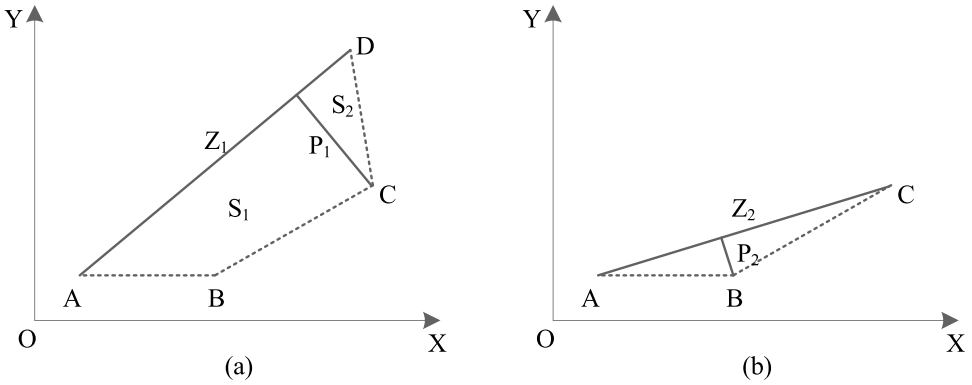


Fig. 3 Schematic diagram of region segmentation

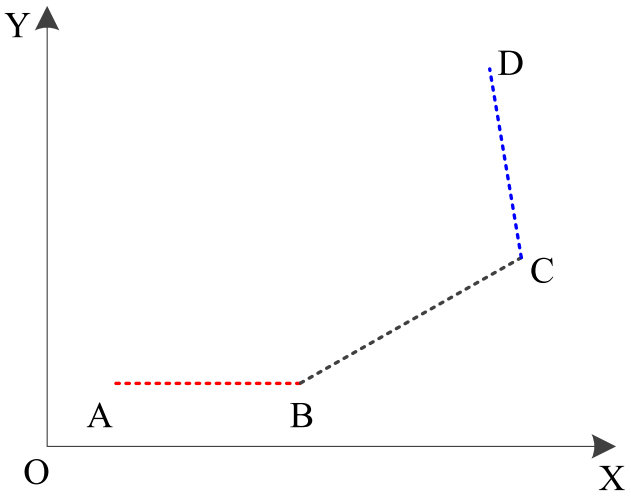


Fig. 4 Schematic representation of region E_i segmentation

In Fig. 4, two segmentation points B and C are obtained after the segmentation of E_i, thus dividing the region into three point sets, which are red, black and blue respectively.

After the above method, N point set Z_i can be represented by a straight line segment is obtained. However, due to the existence of noise, these points are not strictly linear and need to be fitted. The least square method [12] was used to complete the parameter estimation of line segment features.

Suppose that the function relation of x and y is:

$$y = s_0 + s_1 x \tag{2}$$

There are two undetermined parameters in the above equation, s_0 represents the intercept and s_1 represents the slope. For the N group of measurements to be fitted, their x values are considered to be fixed, and the error is related to y.

When using the least square method to estimate the parameters, the deviation weighted sum of squares of y_i should be minimized. For each set of observations, minimizing formula (2) is the optimal estimate $\sum_{i=1}^{N} y_i - \left(s_0 + s_1 x_1\right)^2\Big|_{s=\hat{s}}$ of s (s_0 and s_1). In order to minimize the deviation-weighted sum of squares of y_i, the following requirements need to be met:

$$\begin{cases} \frac{\partial}{\partial_{s0}} \sum_{i=1}^{N} y_i - \left(s_0 + s_1 x_i\right)^2\Big|_{s=\hat{s}} = -2\sum_{i=1}^{N}\left(y_i - \hat{s}_0 - \hat{s}_1 x_1\right) = 0 \\ \frac{\partial}{\partial_{s0}} \sum_{i=1}^{N} y_i - \left(s_0 + s_1 x_i\right)^2\Big|_{s=\hat{s}} = -2\sum_{i=1}^{N}\left(y_i - \hat{s}_0 - \hat{s}_1 x_1\right) = 0 \end{cases} \tag{3}$$

Solve the above formula to obtain the best estimates $\hat{s}_0$ and $\hat{s}_1$ of $\hat{s}_0$ and s_1:

$$\begin{cases} \hat{s}_0 = \left[\left(\sum x_i^2\right)\left(\sum y_i\right) - \left(\sum x_i\right)\left(\sum x_i y_i\right)\right] / \left[N\left(\sum x_i^2\right) - \left(\sum x_i\right)^2\right] \\ \hat{s}_1 = \left[N\left(\sum x_i y_i\right) - \left(\sum x_i\right)\left(\sum y_i\right)\right] / \left[N\left(\sum x_i^2\right) - \left(\sum x_i\right)^2\right] \end{cases} \tag{4}$$

After fitting calculation by least square method, the optimal solution $y = \hat{s}_0 + \hat{s}_1 x$ of the line is obtained, which is converted into the standard linear equation $Ax + By + C = 0$, and the two endpoints of the fitted line segment are the projection point [13, 14] from the beginning and end point of the point set Z_i to the line, that is, the vertical point from the beginning and end point of the line. The calculation of vertical coordinates is shown in Fig. 5. The vertical line is drawn from point $A\left(x_0, y_0\right)$ to line, Z_i the vertical foot is point B, and the coordinates of point B are:

$$\begin{cases} x_1 = \left(B^2 x_0 - AB y_0 - AC\right) / \left(A^2 + B^2\right) \\ y_1 = \left(A^2 y_0 - AB x_0 - AC\right) / \left(A^2 + B^2\right) \end{cases} \tag{5}$$

After the above steps, parameters of the fitted line segment are obtained: slope, intercept, starting coordinates, ending coordinates and length of the line segment. The linear segment is extracted by the object data collected by the laser sensor, and the local map generated by the object information collected in the current period is obtained.

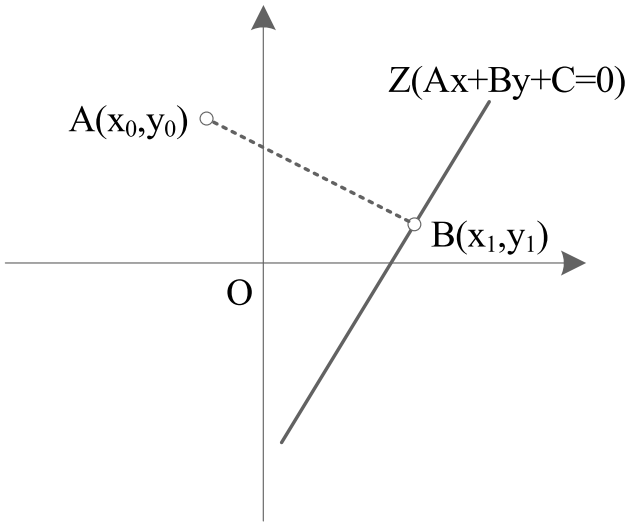


Fig. 5 Calculation of vertical coordinates

2.2 Straight Line Matching

After extracting the line segments, these line segments are matched with the existing line segments in the global map according to the correlation conditions. Suppose that the two endpoints of a line segment in the global map are P_{k1} and P_{k2}, the two endpoints of a newly detected line segment in the local map are P_1 and P_2, and the Angle between the two line segments is η. The relationship between two endpoints and another line segment exists in the following two cases:

(1) The vertical point is in the line segment.
(2) The vertical point is on the extension of the line segment.

The line where the line segment AB is located at point C is the vertical segment, and the vertical foot is D. If $AD > BD$ and $AD > AB$ D, or $BD > AD =$ and $BD > AB$, the vertical point is on the extension of the line segment, otherwise the vertical point is on the line segment.

The vertical distance from the point to the line segment is represented by d_v, and the projection distance from the point to the line segment is represented by d_p [15]. The vertical distance and projection distance are shown in Fig. 6. In Fig. 6(a), no matter in the first or second case, vertical distance represents the distance d_{v1} and d_{v2} from P_{k1} and P_{k2} to the line where line segment Z_n is located. Vertical distance d can be calculated by the following formula:

$$d = \left|Ax_0 + B_{y_0} + C\right| / \sqrt{A^2 + B^2} \tag{6}$$

In Fig. 6(b), for the first case, the projected distance is the distance d_{v2} from point P_{k2} to line segment Z_n, which can be calculated by formula (8). For the second case, the projected distance d_{p1} is the distance between P_{k1} and P_1.

The distance between two points p_i and p_j is denoted by $(p_i, p_j) = \|p_i - p_j\|$. The projected distance from point P_i to line segment z is represented by $(p_i, z) = \min_{p_i \epsilon z} \|p_i - p_j\|$, and the vertical distance from point p_i to line segment z is represented by $d_p = \min_{p_i \epsilon z \cup z+} \|p_i - p_j\|$, where $z+$ is the extension line of z.

The distance relationship between two line segments is defined by the Hausdorff distance, which represents the maximum distance value in the projected distance between any point on one and another line segments [16, 17], that is, $d_H(z_i, z_j) = \min_{p_i \epsilon z_i} \left\{ \min_{p_i \epsilon z_i} d(p_i, p_j) \right\}$. The Hausdorff distance is oriented rather than symmetric, that is, $d_H(z_i, z_j) \neq d_H(z_i, z_j)$. Hausdorff distance is used to define an undirected distance measure $d_H(z_i, z_j) = \max\left[d_H(z_i, z_j), d_H(z_j, z_i)\right]$.

The straight-line matching process is as follows:

(1) Take out a newly detected line segment, calculate its length, if less than 20 mm, it is considered to be interference line segment, delete it, and then take the next line segment for calculation until the line segment that meets the conditions is found;
(2) Match the qualified line segments with each line segment in the global map in turn. If the following formula is met, it is considered that there is correlation between them and the matching is successful. Otherwise, the match fails:

$$d(z_i, z_j) < F_3 \tag{7}$$

where, F_3 represents the distance threshold.
(3) Repeat the above steps to complete the correlation judgment for each newly detected straight line segment.

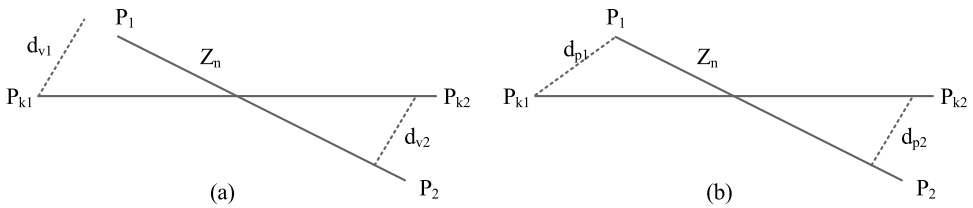


Fig. 6 Vertical and projected distance

2.3 Line Fitting

By matching the lines, determine the correlation between each newly detected line segment and the line segment in the global map. If the matching fails, it indicates that the line segment has not been detected before and is directly imported to the global map. If the match is successful, the least square method will be used to fit the two straight lines [18, 19], the fitted straight lines will be added to the straight line map, and the corresponding two straight lines in the global map and the local map will be deleted. The specific flow chart of the map update is shown in Fig. 7.

3 Robot Target Search Shortest Path Control

3.1 Shortest Path Planning

3.1.1 Path Planning Objective Function

The slime mold optimization algorithm (SMOA) is used to plan the shortest path of the robot in the process of target search. The slime mold optimization algorithm is an algorithm for finding the shortest path. Compared with other methods, this algorithm transforms the robot search path problem into a problem of finding the optimal solution in the search space by simulating the behavior of slime molds in nature when searching for food. During this process, slime molds continuously update their location and environmental information, and use pheromones to guide other slime molds to better locations. In this process, this method can greatly avoid falling into local optima and find the shortest path.

It is assumed that the robot moves in a finite area on a two-dimensional plane where there are finite static and dynamic obstacles. According to the map established in Section 2, the information of obstacles and targets within the range can be obtained, and the size and location of obstacles can be judged and estimated, and the moving speed and direction of moving objects can be estimated.

The task of path planning is to obtain the best path to avoid all obstacles between the robot's position and the local target position according to the real-time information of the environment [20, 21].

With a balance between convergence speed, faster decision-making ability and accuracy to achieve superior foraging ability and avoid falling into local minima, slime molds adapt to random search situations based on the source of food quality. In the case of abundant food quality, slime molds will forage near it and concentrate on the encirclement of the food source. When food quality is insufficient, slime molds explore other spaces in search of food.

The main stage of slime is the period of nutrient capture. During this period, the cytoplasm in the slime mold seeks out food, surrounds it, and secretes enzymes to digest it. During movement, the front end extends into a fan and then forms a network of interconnected veins that allow the cytoplasm to flow inside. Due to their unique patterns and characteristics, they can utilize multiple food

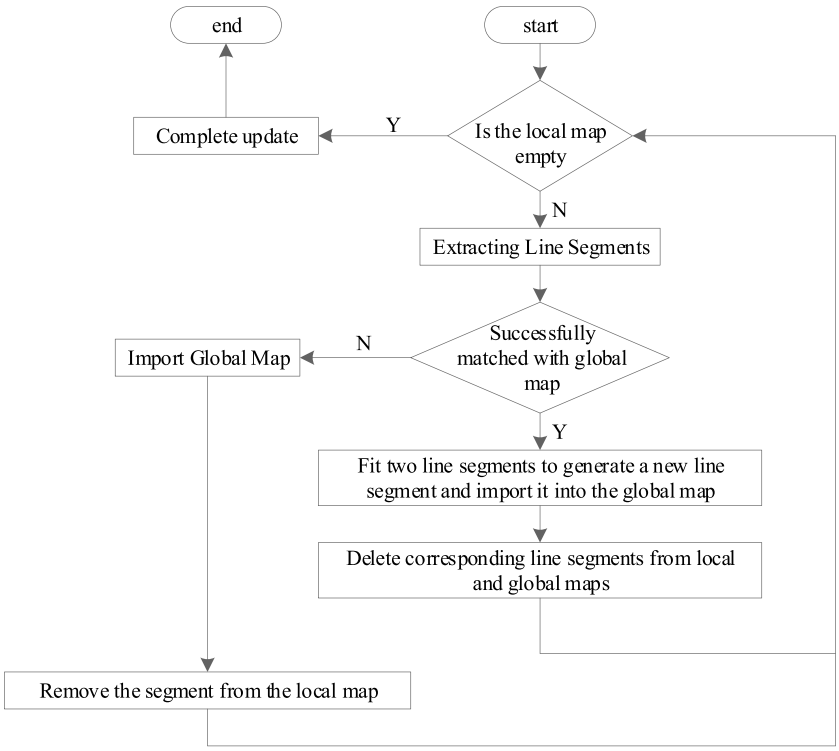


Fig. 7 Map update flow chart

sources at the same time, forming a network of interconnected veins.

Explore and find nutrients and food sources: slime bacteria follow the food odor in the air. Circular, this structure allows slime obacteria to explore food in all potential directions near the optimal solution. The population of myxobacteria is treated by initialization below $X_i(t)$:

$$X_i(t) = unifrnd(L_B, U_B, 1, N) \tag{8}$$

where, L_B, U_B are the upper and lower bound of the population, and N indicates the population size.

The food search behavior of the population can be described by:

$$\vec{X}(t+1) = \begin{cases} \vec{X}_b(t) + \vec{v}_b + [W\vec{X}_A(t) - \vec{X}_B(t)] & r < p \\ \vec{v}_c\vec{X}(t) & r \geq p \end{cases} \tag{9}$$

where, $\vec{X}(t)$ and $\vec{X}(t+1)$ are the positions of individual myxobacteria before and after the search respectively. $\vec{X}_A(t)$ and $\vec{X}_B(t)$ are randomly selected individuals. W represents the weight coefficient; $\vec{X}_b(t)$ is the position of the best fitness; $\vec{v}_b$ and $\vec{v}_c$ both represent the learning factor.

The parameters r and p are calculated by the following formula:

$$\begin{cases} r = arctanh[-(t/\max_t) + 1] \\ p = tan|D(i) - D_F| \end{cases} \tag{10}$$

where, t is the number of current iterations. $\max_t$ represents the maximum number of iterations. $D(i)$ is the fitness of the next generation. D_F is the current best fitness.

Wrapping food: In this step, slime molds surround and devour food sources by extending their veins. The positive and negative feedback $\overline{W}S(i)$ between myxomycetes vein width and food quality was calculated by the following formula

$$\overline{W}S(i) = \begin{cases} 1 + r\log\{[b_F - D(i)]/[b_F + w_F]\} & condition \\ 1 - r\log\{[b_F - D(i)]/[b_F + w_F]\} & other \end{cases} \tag{11}$$

where, b_F is the best fitness of this generation. w_F indicates the worst fitness of this generation.

The search population X at iteration i is updated with reference to the best position X_b at iteration i, whose position can be changed by fine-tuning $\vec{v}_b$, $\vec{v}_c$, and W. The logarithmic factor increases the rate of change of the value to avoid random perturbations in the frequency of venous contraction. Slime molds adjust their exploration process according to the quality of the food. When the food source is larger, the weight near it becomes larger. When food is insufficient, the weight of the nearby area is reduced and other areas are explored.

Prey capture/Oscillation: Biological oscillators emit waves to locations with higher concentrations of food to alter cytoplasmic flow in the veins of the slime mold. The oscillation frequency in the SMO algorithm depends on W. Where the concentration of food is higher, the oscillation frequency is higher. And where the concentration of food is lower, the oscillation frequency is lower, so the speed of reaching the food is slower. This process increases the proficiency of the SMO algorithm and results in a globally optimal solution (high concentration food source). The advantage of this approach is that even after the best food source is obtained, it still strives to explore nearby locations to obtain a higher optimal solution, thus avoiding local minima and early erroneous convergence.

Taking finding the shortest path and avoiding obstacles and collision [22, 23] as targets, the robot target search path planning objective function F is established

$$F(OV) = P_L + \eta p P_L D \tag{12}$$

where, OV represents the target value, P_L represents the path length, η represents the scale factor, and D represents the average distance value between all insertion points and all obstacles. This value is zero when the maximum number of insertion points is outside the obstacle.

The above objective function is intended to be minimized through all iterations of nature-inspired algorithm evolution. This minimizes the path length that satisfies the first constraint. The second goal is achieved by minimizing D. Scale factors are used to determine the influence of path distance obstacles. The larger the scale factor, the more emphasis is placed on excluding paths that contain collisions.

3.1.2 Implementation of Path Planning Based on SMO

The path planning task is decomposed into multiple local planning tasks, and the local path planning is performed by moving the window in the established environment map.

The current position of the robot is the starting point S, the target position is the target point T, and a set of equally spaced points $L_1, L_2, \cdots, L_{n-1}$ on the starting point and the target point line ST is defined as the subtarget.

Define a moving window Q_i, $i = 0, 1, \cdots, n-1$, with length $||L_{i+1} - L_i||$ and width $2I$, where $L_0 = S$, $L_n = T$, and the window size is determined by the scope of the environment map.

The robot performs path planning in each scrolling window $Q_0, Q_1, \cdots, Q_{n-1}$ in turn. When the robot reaches the sub-target L_i, the following procedure is performed:

(1) Perceive obstacles in window Q_i and save parameters about obstacles.
(2) With L_i as the starting point and L_{i+1} as the target point, the slime mold optimization algorithm is used to plan the path, and a collision-free path with the shortest distance was planned.
(3) The robot moves from the starting point L_i to the target point L_{i+1} at a constant speed along the shortest path planned in the previous step.
(4) $i+1 \rightarrow i$, return to (1) for the path planning for the next window.

This process continues until the robot reaches its target.

3.2 Translational Control

During the robot target search process, the controller shown in Fig. 8 is designed to control the robot to move along the optimal path obtained in Section 3.1. Based on the optimal path information, the controller will generate motion instructions for the robot, including forward, backward, turning, and other actions. Based on the node order of the optimal path, the controller will determine when the robot will turn and move forward to maintain movement on the optimal path, while guiding the robot to bypass obstacles to ensure that the robot can continue to move along the optimal path.

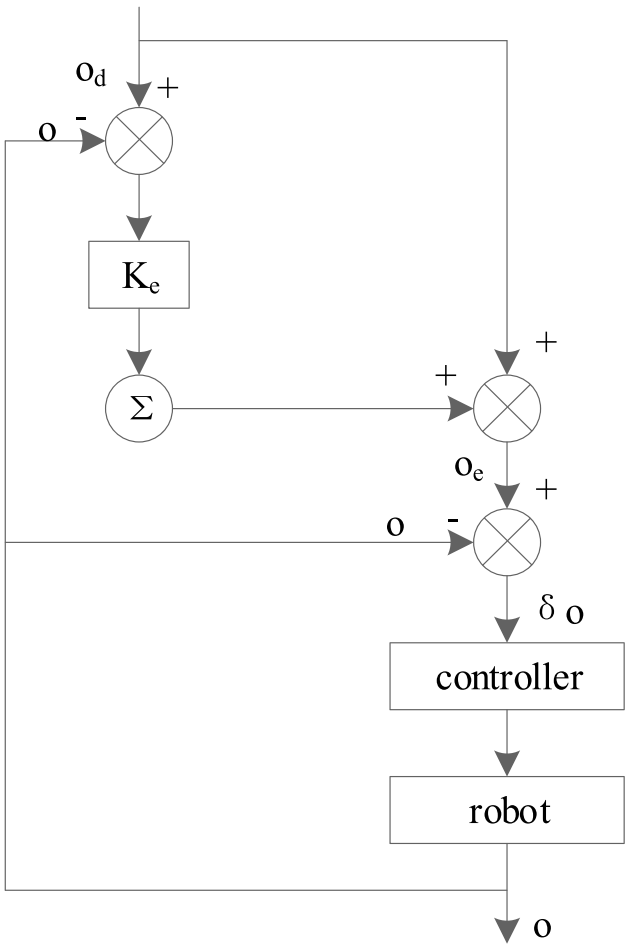


Fig. 8 Diagram of Controller

In Fig. 8, o_d represents the desired position vector. It can be expressed either as a fixed point or as a continuous trajectory. o represents the current position vector of the robot. The error e between o_d and o is called the tracking error. K_e is a diagonal matrix with dimensions consistent with the degree of freedom of the robot, called the compensation gain. The symbol $\sum$ represents the accumulator, which adds up the the points $K_e e$ on the trajectory that have been tracked. The output of the accumulator is added to o_d to obtain a new command or a corrected reference trace o_e. The controller then adjusts the robot motion according to the command error δo, and δo is the error between o_e and o. When $K_e = 0$, $o_e = o_d$, $\delta o = e$.

Assuming that the trajectory of each joint is discrete to an infinite number of points, the sampling time interval between adjacent points is T. According to Fig. 8, the error between the first desired point and its current position is $e_1 = o_d^{(1)} - o^{(1)}$, and the new instruction $o_e^{(1)} = o_d^{(1)} + K_e^{(1)} e_1$ and instruction error $\delta o^{(1)} = o_e^{(1)} - o^{(1)}$ after the first compensation.

The relationship between the first point tracking error and the instruction error is obtained from the above analysis:

$$e_1 = \delta o^{(1)} - K_e^{(1)} e_1 \tag{13}$$

After the first point compensation, the joints of the robot reach position $o^{(2)}$ after the action of the controller. Similar to before, we can get:

$$\begin{cases} e_2 = o_d^{(2)} - o^{(2)} \\ o_e^{(2)} = o_d^{(2)} + K_e^{(1)} e_1 + K_e^{(2)} e_2 \\ \delta o^{(2)} = o_e^{(2)} - o^{(2)} \end{cases} \tag{14}$$

On the basis of the following:

$$e_2 = \delta o^{(2)} - K_e^{(1)} e_1 - K_e^{(2)} e_2 \tag{15}$$

By the same token, we obtain:

$$\begin{cases} e_n = \delta o^{(n)} - K_e^{(1)} e_1 - K_e^{(2)} e_2 - \cdots - K_e^{(n-1)} e_{n-1} - K_e^{(n)} e_n \\ e_{n+1} = \delta o^{(n+1)} - K_e^{(1)} e_1 - K_e^{(2)} e_2 - \cdots - K_e^{(n)} e_n - K_e^{(n+1)} e_{n+1} \end{cases} \tag{16}$$

Subtract the two formulas in formula (16) to obtain the following formula:

$$e_{n+1} = (I + K_e^{(n+1)})^{-1} (e_n + E_{n+1}) \tag{17}$$

where, I is the identity matrix. E_{n+1} represents the difference of the instruction error.

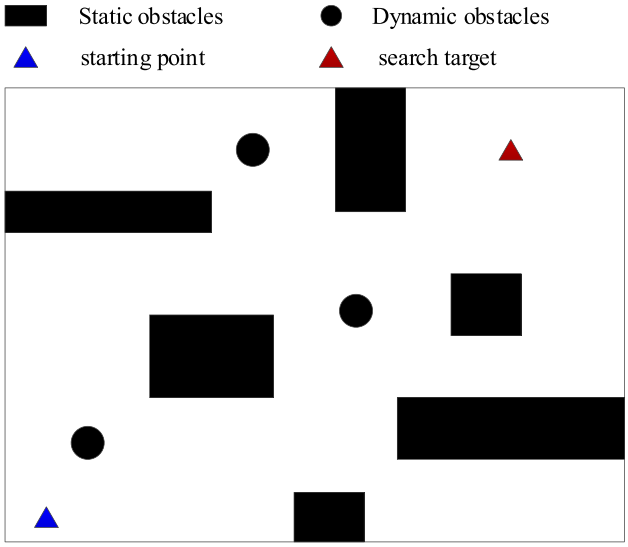


Fig. 9 The target search environment

For convenient description, E is used to represent the difference of the instruction error obtained in each compensation.

Expand the formula (18) iteratively to obtain:

$$e_{n+1} = ({}^{I}\cdots({}^{I}(e_1 + E_1)\cdots + ({}^{I}\cdots({}^{I}E_j\cdots + ({}^{I}E_{n+1} \tag{18}$$

where, e_1 is the error between the joint position in the initial state and the expected trajectory. The above equation shows that as long as K_e is a positive definite matrix and the difference E of the instruction error equals zero, the tracking error e_1 will converge and will not be affected by e_1, and the larger the element value in K_e, the faster the convergence rate.

4 Experiment and Discussion

In order to verify the overall effectiveness of the shortest path control method for robot target search in a large range of complex tasks, it is necessary to test it, select the Robotics System Toolbox in MATLAB for simulation, and establish an environment with a size of 50 m × 50 m, including obstacles and target devices. Select the LiDAR sensor and set its scanning range to 360 degrees with a maximum detection range of 30 m. Reference [4], reference [5] and reference [6] methods are introduced into the test process as comparison methods to ensure the authenticity of the test. The robot target search environment is shown in Fig. 9.

In the environment shown in Fig. 9, the proposed method, the reference [4] method, the reference [5] method and the reference [6] method are used, and the path planning results of the four methods are shown in Fig. 10

Analyzing Fig. 10, it was found that the path planned by the proposed method can successfully avoid dynamic and static obstacles in the environment, which proves that the

Fig. 10 Path planning results

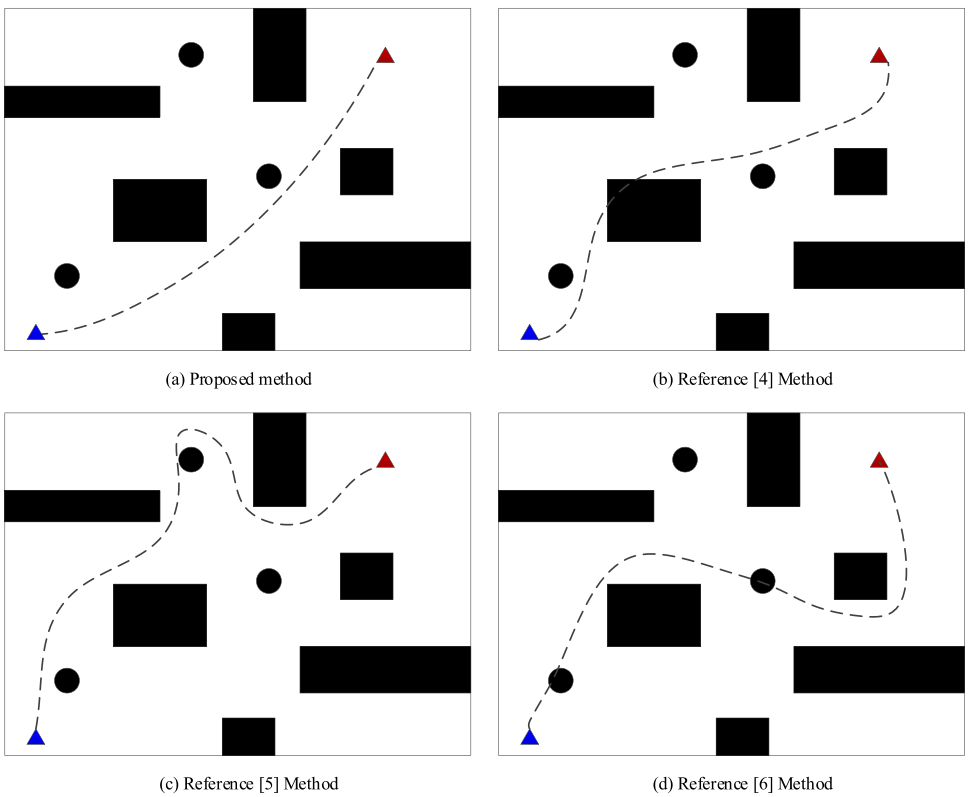
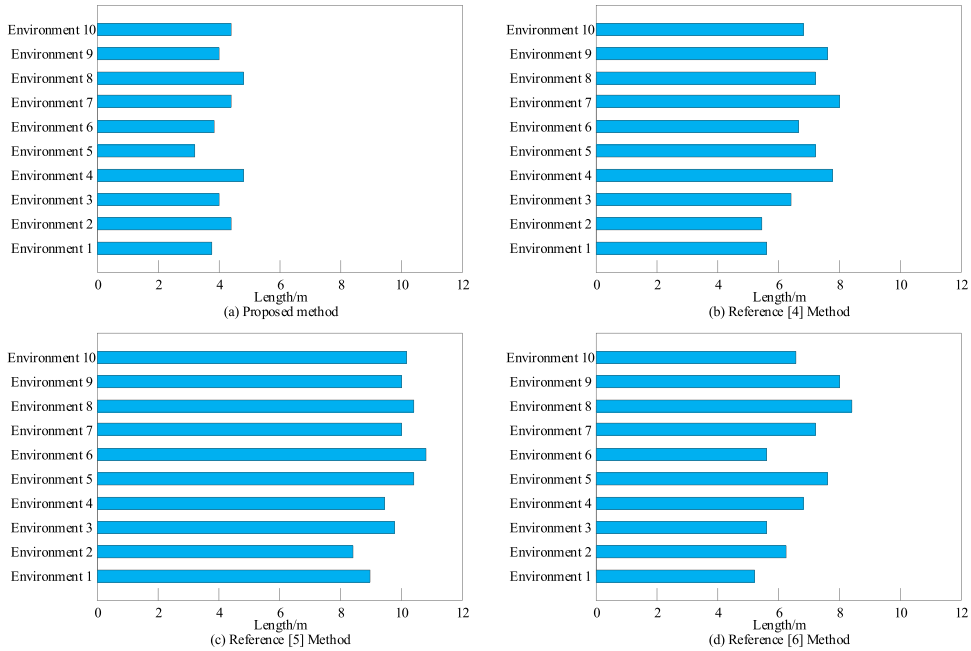
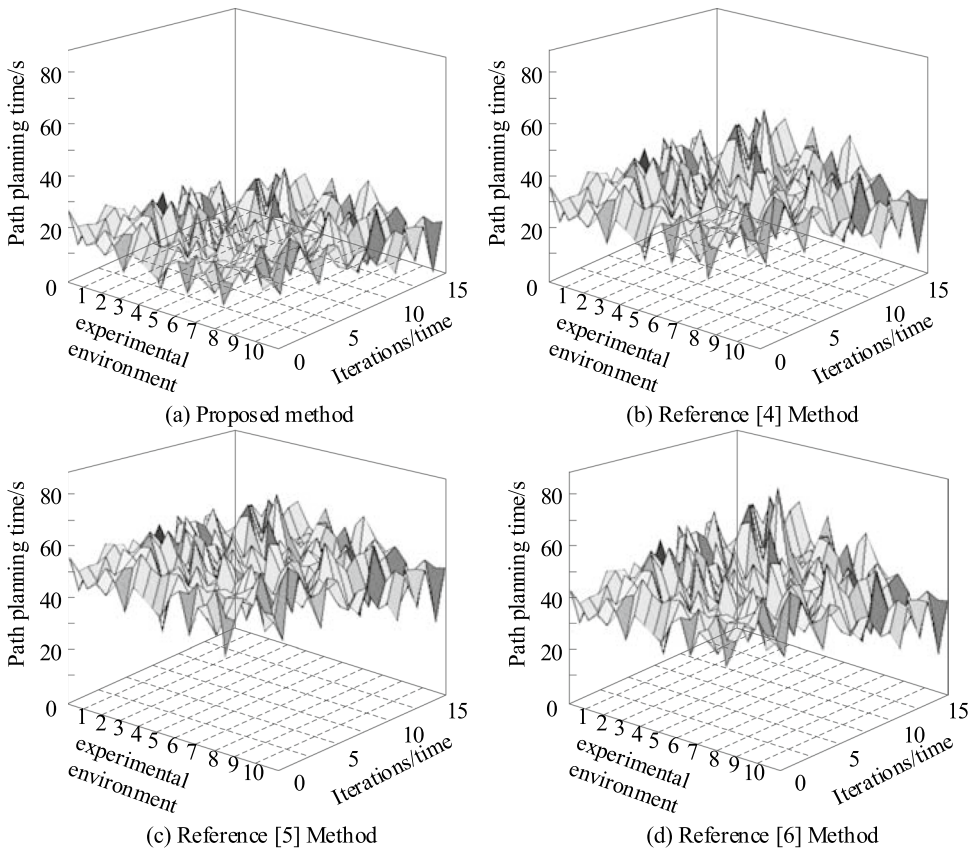


Fig. 11 Path length



proposed method has strong obstacle avoidance ability. The path planned by the method in reference [4] did not successfully avoid static obstacles, indicating that this method has certain limitations in avoiding static obstacles. Although the method in reference [5] can successfully avoid obstacles in the environment, the path is too long, which can affect the operational efficiency and time of the robot. The path planned by the method in reference [6] did not successfully avoid dynamic obstacles, indicating that this method has certain shortcomings in avoiding dynamic obstacles. Through

Fig. 12 Path planning time



the above tests, it has been verified that the proposed method has strong obstacle avoidance ability. In the field of industrial automation, this method can autonomously navigate through busy production lines, avoiding workpieces or other mobile devices, to ensure the stability and safety of the production process.

10 experimental environments are set up, use the above method to expand the path planning, and test the path length of the four methods. The results are shown in Fig. 11.

From Fig. 11, it can be seen that in different environments, the path length planned by the proposed method is the shortest, always within 6, while the path length planned by the methods in reference [4, 5], and [6] are all greater than the proposed method. Through comparison, it can be seen that the path length planned by the proposed method is the shortest. This is because the proposed method takes the shortest path length as the optimization objective when establishing the path planning objective function, so it can reach the search target with the shortest distance in complex environments. In practical applications, the proposed method has broad application potential in path planning. It can be applied in fields such as autonomous vehicles, drones, industrial robots, etc. to achieve more efficient, safe, and reliable path planning and navigation functions.

In the above path planning test, the time required by the four methods to plan the path is as following Fig. 12.

Analyzing Fig. 12, it can be seen that in path planning testing, under the same testing environment conditions, the proposed method takes much less time to plan the path than the methods in reference [4], reference [5], and reference [6], and overall remains within 55 s. This is because the proposed method establishes an environmental map before path planning, which can directly obtain the positions of obstacles and search targets in the environment. Based on this, the shortest path is planned, thus shortening the path planning time and improving the efficiency of path planning. In fields such as emergency rescue and industrial automation, rapid path planning can help robots quickly reach target points, execute corresponding tasks, and improve application effectiveness.

5 Conclusion

At present, robot target search path control methods have problems such as poor obstacle avoidance ability, too long path and low planning efficiency. A shortest path control method for robot target search under a large range of complex tasks is proposed. The environment map of robot target search is established by the proposed method, and the objective function of path planning is established on this basis, so as to obtain the shortest distance and effective obstacle avoidance path. The controller is designed to control the robot to move along the optimal path. It is proved that the proposed method can effectively realize the obstacle avoidance of the robot, and the planned path length is short and the path planning efficiency is high, which provides a guarantee for the development of robot technology.

Author Contributions Jinyin Peng wrote the main manuscript text, Li Zhao guided and reviewed the whole paper. All authors reviewed the manuscript.

Funding The paper was supported by Science Foundation of Institute of Education Department of Hunan Province with No. 21C1373.

Data Availability N/A.

Declarations

Ethics Approval N/A.

Competing Interests N/A.

References

1. Mu X, Liu Y, Guo L et al (2021) Intelligent Reflecting Surface Enhanced Indoor Robot Path Planning: A Radio Map based Approach. IEEE Trans Wireless Commun 20(7):4732–4747
2. Zhao W, Lin R, Dong S et al (2021) Dynamic node allocation based multirobot path planning. IEEE Access 9:106399–106411
3. Zhang Z, Wan Y, Wang Y et al (2021) Improved hybrid A* path planning method for spherical mobile robot based on pendulum. Int J Adv Rob Syst 18(1):671–680
4. Wang M, Zhou S, Zhang H et al (2022) Multi-target search of swarm robots cooperative control in an unknown environment. Control Theory Appl 39(04):750–760
5. Gao B, Zhang B, Wang J et al (2022) An expected-time optimal target search method based on probabilistic maps. Control Decis 37(04):944–952
6. Fan Z, Sun F, Ma P et al (2022) Stigmergy-Based Swarm Robots for Target Search and Trapping. Trans Beijing Inst Technol 42(02):158–167
7. Kumar R, Singh L, Tiwari R (2021) Path planning for the autonomous robots using modified grey wolf optimization approach[J]. J Intell Fuzzy Syst 40(5):9453–9470
8. Nascimento LBP, Barrios-Aranibar D, Santos VG et al (2021) Safe path planning algorithms for mobile robots based on probabilistic foam[J]. Sensors 21(12):4156
9. Duraklı Z, Nabiyev V (2022) A new approach based on Bezier curves to solve path planning problems for mobile robots[J]. J Comput Sci 58:101540
10. Liu S, Huang S, Wang S et al (2023) Visual Tracking in Complex Scenes: A Location Fusion Mechanism Based on the Combination of Multiple Visual Cognition Flows. Information Fusion 96:281–296
11. Büttner L, Krmer V, Dues M et al (2022) Flow-measurements in the wake of an oscillating sessile droplet using laser-Doppler velocity profile sensor[J]. tm-Technisches Messen 89(3):178–188
12. Zhang L, Zhang T, Shin H (2021) An Efficient Constrained Weighted Least Squares Method With Bias Reduction for TDOA-Based Localization. IEEE Sens J 21(8):10122–10131

13. Bai Y, Yuen C, Liu Q (2021) A modified least-squares method for quantitative analysis in Raman spectroscopy. IEEE J Sel Top Quantum Electron 27(4):1–9
14. Tominec I, Breznik E (2021) An unfitted RBF-FD method in a least-squares setting for elliptic PDEs on complex geometries. J Comput Phys 436(4):110283
15. Liu S, Huang S, Xu X, et al. (2023) Efficient Visual Tracking Based on Fuzzy Inference for Intelligent Transportation Systems, IEEE Transactions on Intelligent Transportation Systems, online first, https://doi.org/10.1109/TITS.2022.3232242
16. Fancourt H, Lynch J, Byrd J et al (2021) Next-generation osteometric sorting: Using 3D shape, elliptical Fourier analysis, and Hausdorff distance to optimize osteological pair-matching. J Forensic Sci 66(3):821–836
17. Rodrigues RO (2021) An Efficient and Locality-Oriented Hausdorff Distance Algorithm: Proposal and Analysis of Paradigms and Implementations. Pattern Recogn 117(1):107989
18. Yao J, Yu K, Fu Q et al (2021) Computational Method for Heat Partition at the Rail-Armature Interface Based on Least Squares Regression. IEEE Trans Plasma Sci 49(6):2008–2014
19. Wang C, Chen X, Yuan G et al (2021) Semisupervised Feature Selection With Sparse Discriminative Least Squares Regression. IEEE Trans Cybern 52(8):8413–8424
20. Liu S, Wang S, Liu X et al (2021) Human Memory Update Strategy: A Multi-Layer Template Update Mechanism for Remote Visual Monitoring. IEEE Trans Multimedia 23:2188–2198
21. Belge E, Altan A, Hacolu R (2022) Metaheuristic Optimization-Based Path Planning and Tracking of Quadcopter for Payload Hold-Release Mission. Electronics 11(8):1208
22. Zhang W, Cheng H, Hao L et al (2021) An obstacle avoidance algorithm for robot manipulators based on decision-making force. Robot Comput-Integr Manuf 71:102114
23. Shi F, Hu X (2022) Fuzzy Dynamic Obstacle Avoidance Algorithm for Basketball Robot Based on Multi-Sensor Data Fusion Technology. Int J Found Comput Sci 33(06n07):649–666