U.P.B. Sci. Bull., Series C, Vol. 86, Iss. 2, 2024 ISSN 2286-3540

A NEW ALGORITHM ON PATH PLANNING FOR ROBOTICS

Hongqiang LI[1,*], Weimin KUANG[2], Zhijia CHEN[3]

Path planning process using sweeping robots has been a hot spot in the research community. It is noteworthy that conventional robots adopt random collision and infrared detection to avoid obstacles, which results in high repetition and low coverage of the cleaning area. In this research, we have proposed an intelligent obstacle avoidance area recovery algorithm. This algorithm was built on U-shaped trajectory. However, unlike conventional obstacle avoidance algorithms, the developed algorithm established a mechanism to allow the robots to remember the processed areas in order to avoid the cleaning of the same area again. Second, based on the kinematics analysis of the sweeping robot, the front side of the robot adopted universal wheels to address the limitation of incomplete linear motion and realize omni-directional motion. Experimental results of robots operating using the proposed algorithm revealed that 94.61% of the total area was covered, which was higher than those of robots using random cleaning technique. Furthermore, compared with conventional robots, those using the proposed algorithm consumed less power and presented high efficiencies.

Keywords: Sweeping robot; Coverage path planning; Kinematic analysis; Navigation

1. Introduction

In robotics, sweeping robots have been a hot spot in the research community [1]. The aim of developing a sweeping robot was to replace manual ground cleaning. Path planning and path keeping abilities are major topics in the research on sweeping robots [2]. It is noteworthy that complete area path planning requires the robots to cover every part of the workspace, which is a critical issue in cleaning robots [3]. Recently, with the development of intelligent houses, sweeping robots are greatly becoming research hot spot [4], but they are still not widely popularized.

The factors limiting the development of sweeping robots include hardware-based and software-based restrictions. Regarding hardware-based restrictions, traditional sweeping robots generally employ a single infrared [5] or ultrasonic sensor for obstacle avoidance. Walking mode is random reciprocating due to which cleaning robots cannot completely clean the target surface, thus making the process

[1] Guangzhou Industry and Trade Technician College, Guangzhou, China, e-mail: 471918248@qq.com
[2] Guangzhou Industry and Trade Technician College, Guangzhou, China
[3] Guangzhou Industry and Trade Technician College, Guangzhou, China

less efficient and highly repetitive [6]. Therefore, it is difficult to develop an intelligent planning and obtain orderly cleaning. Fig. 1 illustrates this process. It should be noted that although internal spiral route cleaning had comprehensive coverage and low repetition rate, its cleaning time was longer, power consumption was higher and path planning was not optimal in the presence of obstacles [7]. Consequently, robots were unable to meet the requirements of the user. High-end sweeping robots, such as L10 PRO launched by Chomie, used double line laser to avoid obstacles during navigation [8]. Fig. 2 depicts this process. These sweeping robots had high precision; however, they were not cost effective. In terms of software, map construction (SLAM) [9] and path planning of sweeping robots involved operating system, deep learning, communication transmission and several other software features required for robot operation. These complex control algorithms have a variety of technical bottlenecks. Hence, the scavenging efficiency of current floor sweeping robots was not efficient. In addition, SLAM relied on lidar sensors, most of which were not cost effective and continuous rotation seriously affected their service life.

Although sweeping robots usually maneuver their actuators along a predetermined path to reach a target location or to cover a designated target area, this approach is not optimized to avoid static or dynamic obstacles in the path space domain. Therefore, autonomous robots must overcome the obstacles of interacting in complex environments by solving the covered path planning (CPP) problem[1][2]. The goal of the CPP algorithm is to compute optimal paths and project collision-free trajectories to ensure that the robot completely covers the area of interest (AOI) within a certain period of time. The robustness and performance of CPP efficiency is based on several parameters such as percentage of area covered, travel time, path overlap rate, and energy consumption of the robot.

CPP is the core of processing area coverage optimization in mobile robot exploration. Area coverage is generalized by robots as a fully or partially enclosed area with non-overlapping paths. According to the prior knowledge of the surrounding environment by airborne sensors, CPP algorithms can be divided into offline algorithms [3] [4]and online algorithms[5]. Offline algorithms allow mobile robots to perform coverage in static, known environments. CPP is generally based on global sequential point-to-point coverage, where the robot travels along a route on a given map and avoids obstacles[6][7]. However, in practice, robots need to deal with unknown or partially known environments[8][9]. Therefore, online algorithms are preferred to optimize the exploration strategy and explore unknown areas within the area of interest while the robot are moving in the environment. The robot will choose a suitable path by acquiring real-time data from local sensors and extracting unique features in the dynamic environment[10]. Finally, the robot must create a limited mapping of the probed environment using CPP techniques [11].

Current CPP algorithms mainly focus on classical algorithms and heuristic algorithms for solving optimization problems[12]. Collision-free paths[13], covering cost functions [14] (shortest paths and smooth paths) and covering sequences (set covering problem, SCP and traveling salesman problem, TSP) are directly related to the CPP problem, in which the optimization problem is considered. It also includes the characteristics of the CPP optimization algorithm, as well as various technical characteristics, namely, search time, path optimality, dynamic performance, convergence speed, and computational complexity. CPP remains an open problem in the field of robotics in improving the efficiency of planning optimal paths covering target areas and generating collision-free paths with less computational effort. The generated coverage paths should be optimal to ensure minimal logistical costs such as overlap, number of turns, travel time, and energy consumption [15]. CPP problems include potential uncertain failures, unknown obstacles in complex environments, and path optimality, which are considered major challenges in robotics.

SLAM is challenging in real-world applications. It requires to estimate the robot’s motion and the state of its surroundings from the sensor data and construct an accurate map at the same time. This process requires processing a large amount of data and needs to take into account sensor errors as well as the complexity of the environment[16]. In addition, the practical application of SLAM involves a variety of complex factors. For example, robots need to adapt to different sensors and algorithms in different environments. There are lots of issues such as dynamic obstacles, light changes, and sensor failures which needs to be taken into account in different environments. These factors increase the difficulty and complexity of SLAM implementation. In addition, practical applications of SLAM also should consider issues such as efficiency and real-time performance. In some cases, robots cannot construct maps and localize themselves in dynamic environments. In summary, SLAM is a hard to apply in practice.

Deep learning is a powerful tool in a wide range of applications. However, for path planning problems, deep learning may not be the best choice. First of all, deep learning requires a large amount of data to train the model. For path planning problems, big data may lead to increased difficulty and cost of data collection and processing [17]. In addition, it would be impractical to collect all the map data. Secondly, deep learning models usually require long training time and expensive computing resources. In path planning problems, real-time response and fast calculation of paths are required, which requires the model to be able to make accurate decisions in a short period of time. The training and prediction time of deep learning models is long, which is difficult to meet the real-time demand. In addition, path planning problems usually need to consider multiple factors, such as scene conditions, obstacle restrictions, and driving distances. These factors are often not simple features but need to consider the interaction between multiple factors. Deep learning models have difficulty dealing with such complex interactions, which may lead to inaccurate or uninterpretable predictions from the model. Therefore, although deep learning has a wide range of applications in many fields, it may not be the best choice in path planning problems.

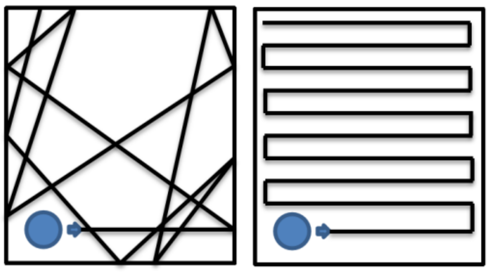


Fig. 1. The paths finding performance of random collision and path planning robots.

Fig. 2. A seeping robot with SLAM.

In this research, we improved traditional sweeping robots. We carried out kinematics analyses on robots based on U-shaped paths. This was accomplished by combining dead reckoning and gyroscope positioning schemes [10]. We employed infrared sensors, ultrasonic sensors, and photoelectric encoders to construct cost effective hardware, which effectively implemented path planning. The designed robot was cost effective and highly efficient. Also, it had low repetition rate and covered the complete area while successfully avoiding obstacles.

2. Kinematics analysis of sweeping robots

The kinematics analysis and trajectory planning of robots are the most crucial processes for robot controlling, managing robot motion, and path planning [9,13,20].

2.1. Motion model analysis

Sweeping robot model was developed using a two-wheel robot. A simplified sweeping robot model is illustrated as Fig. 3(a).

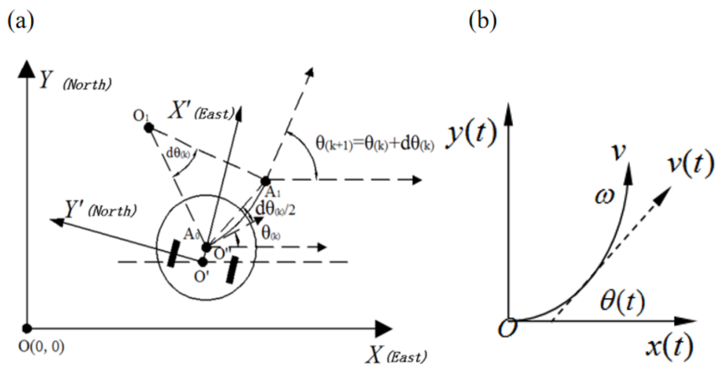


Fig. 3. The schematic diagram of robot motion model. (a) kinematic model for sweeping robot and (b) positional relationship.

XOY in Fig. 3(a) represents world coordinates. World coordinate described robots in their current environment. Generally, robot cleaning starting point is denoted by world coordinate origin *O*. In Fig. 3(a), *X'O'Y'* is robot body coordinates. The central position *O''* of the axes of the two wheels of the robot denoted the origin of local coordinate. Then, we established the relationship between the position ***(x,y)*** in global coordinates and position ***(x',y')*** in body coordinates. Local coordinates was assumed to be ***(x0,y0)*** and relative rotation angle between two coordinate systems was taken as α. Then, according to the rotation matrix, it could be expressed that:

$$\begin{pmatrix} x \\ y \end{pmatrix} = (\begin{matrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{matrix}) \cdot \begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{1}$$

Robot position was represented by axis midpoint and yaw angle theta. Sweeping robot position was described by the coordinate vector $\boldsymbol{A=(x,y,\theta)^T}$ in the global coordinate system, where ***(x,y)*** denotes robot coordinates and *θ* is the angle between travel direction of robot and positive direction of X axis; i.e., yaw angle. When the sliding of the driving wheel of the robot was neglected, kinematics model was stated as the following matrices.

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} v \\ \omega \end{pmatrix} \tag{2}$$

2.2. Positioning model analysis

In simplified robot motion control, the center of mass of the robot was considered to coincide with its geometric center. Considering this premise, the three most common robot motions were as follows [21]:

(1) Forward and backward motion: maintaining similar speeds for both side motors in the same direction.

(2) Circular motion of any radius: allowing both side motors with different speeds along the same or opposite directions.

(3) Zero radius circular motion: allowing both side motors with the same speed and in opposite directions.

Considering the right wheel as an example, wheel diameter was ***D***, encoder line number was ***n***, and record pulse readings was ***Nr***. During time ***Δt***, the distance traveled by the right wheel was ***ΔSr***. This was formulated as:

$$\Delta S_r = \frac{\pi D N_\mathrm{r}}{\mathrm{n}} \tag{3}$$

During time ***Δt***, the distances covered by the left and right wheels were ***ΔSl*** and ***ΔSr***, respectively. Then, the distance ***ΔS*** and angle ***Δθ*** of the rotation of the robot (with ***O ′*** as reference point) were mathematically stated as:

$$\begin{pmatrix} \Delta S \\ \Delta\theta \end{pmatrix} = \begin{pmatrix} 1/2 & 1/2 \\ 1/B & -1/B \end{pmatrix} \cdot \begin{pmatrix} \Delta S_r \\ \Delta S_l \end{pmatrix} \tag{4}$$

Figs. 3(a)-(b) illustrate the positional analysis of the sweeping robot. At first, the robot was in the position presented as $\boldsymbol{A_0(x_{(k)}, y_{(k)}, \theta_{(k)})}$. In time ***Δt*** (sampling period) of the control system, the robot moved from point $\boldsymbol{A_0(x_{(k)}, y_{(k)}, \theta_{(k)})}$ to point $\boldsymbol{A_1(x_{(k+1)}, y_{(k+1)}, \theta_{(k+1)})}$ where, $\boldsymbol{\Delta x_{(k)}}$, $\boldsymbol{\Delta y_{(k)}}$, and $\boldsymbol{d\theta_{(k)}}$ denoted increment along horizontal direction, increment along vertical direction, and yaw angle of the robot, respectively. It should be noted that yaw angle represented the angle between horizontal axis as the starting position and counterclockwise direction as the positive direction. $\boldsymbol{\Delta S_{(k)}}$ represents robot movement along the curved path from point $\boldsymbol{A_0}$ to point $\boldsymbol{A_1}$ and $\boldsymbol{R_{(k)}}$ is the arc radius of the curved track during this time. Therefore, $\boldsymbol{\Delta x_{(k)}}$, $\boldsymbol{\Delta y_{(k)}}$, and $\boldsymbol{d\theta_{(k)}}$ were computed using the following equation.

$$\begin{aligned} \Delta x_{(k)} &= l_{A_0A_1} \cdot \cos(\theta_{(k)} + \frac{d\theta_{(k)}}{2}) = \Delta S_{(k)} \cdot [\sin(\frac{d\theta_{(k)}}{2}) / \frac{d\theta_{(k)}}{2}) \cdot \cos(\theta_{(k)} + \frac{d\theta_{(k)}}{2}) \\ \Delta y_{(k)} &= l_{A_0A_1} \cdot \sin(\theta_{(k)} + \frac{d\theta_{(k)}}{2}) = \Delta S_{(k)} \cdot [\sin(\frac{d\theta_{(k)}}{2}) / \frac{d\theta_{(k)}}{2}) \cdot \sin(\theta_{(k)} + \frac{d\theta_{(k)}}{2}) \end{aligned} \tag{5}$$

Due to positioning calculation, the control system of ***Δt*** (sampling period) was small; therefore, $\boldsymbol{d\theta_{(k)}}$ was smaller. At this time,

$$d\theta_{(k)} \to 0, \lim \frac{\sin(\theta_{(k)})/2}{\theta_{(k)}/2} = 1 \tag{6}$$

Then, the pose at moment ***k*** was calculated using (5). The current pose information was expressed as:

$$\begin{aligned} x_{(k+1)} &= x_{(k)} + \Delta S_{(k)} \cdot \cos(\theta_{(k)} + \frac{d\theta_{(k)}}{2}); \\ y_{(k+1)} &= y_{(k)} + \Delta S_{(k)} \cdot \sin(\theta_{(k)} + \frac{d\theta_{(k)}}{2}); \\ \theta_{(k+1)} &= \theta_{(k)} + d\theta_{(k)}; \end{aligned} \tag{7}$$

3. Algorithm design

In this research, we presented an improved intelligent obstacle avoidance region recovery algorithm based on traditional U-shaped route [23,24]. The proposed algorithm aimed to increase cleaning range while minimizing the cost. However, unlike traditional obstacle avoidance algorithms, the proposed algorithm provided a mechanism which allowed the robot to memorize swept area by setting the variable Time-Swerve. The developed algorithm prevented robot from cleaning any certain area multiple times. This also ensured that the cleaning robot cleaned all target surfaces.

3.1. Traditional U-shaped trajectory path algorithm

Cleaning robots use yaw angle as reference angle. The robot int his research moved along forward direction and adjusted yaw direction based on reference direction. This was done to ensure that the robot did not deviate from its original direction [25,26,27]. When sensors detected an obstacle, the cleaning robot reduced its speed, rotated 90°, and then kept moving unless the width limit of the seeping area was reached. Then, the robot rotated 90° again and moved forward. Thus, the robot kept moving in the same fashion following a rotational trend. Fig. 4 illustrates this process.

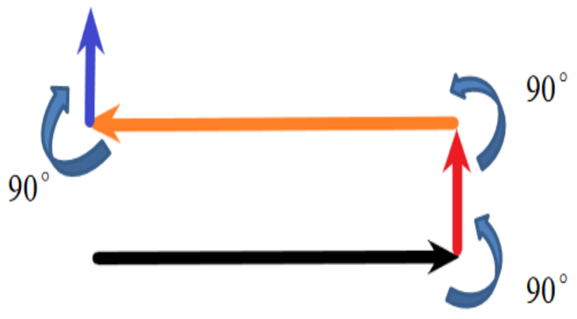


Fig. 4. The traditional U-shaped motion trajectory .

It is noteworthy that the proposed algorithm worked effectively in a simple environment. In a slightly complex environment comprising obstacles, the robot could get trapped in a particular place [22,27].

3.2. Improved intelligent obstacle avoidance algorithm based on traditional algorithm

In real-world environments, there are different obstacle types. We ignored the irregular shape of obstacles and assumed their shape to be rectangular. Due to the obstruction caused by obstacles, traditional path algorithms might not be able to completely cover the sweeping area. Hence, improved intelligent algorithms were applied to clean the areas missed by the traditional algorithms.

The improved algorithm developed in this research introduced additional variables into traditional U-shaped path algorithm, namely W, T, D, Prev-D, and Φ. These variables enabled the robot to temporarily remember the uncovered areas covered by obstacles. These variables are defined in Table 1. Here, the proposed algorithm calculated length for the uncovered area by computing the product W*T. Based on this information, the robot moved to a suitable position, as illustrated in Fig. 5.

Table 1

The definitions of variables

Variable	Definition	Initial value
Φ	Robot diameter	32cm
W	Robot cleaning mouth width	16cm
T	Robot turning times	0cm
D	The distance the robot now travels	0cm
Prev-D	The last time the robot traveled	0cm

(1) Sweeping path planning

The autonomous path planning exploration strategy for the robot starts from the initial point and sweeps from left to right. When the front ultrasonic sensor detects a wall, the robot stops moving and records the current distance traveled as D. It should be noted that at this point, the previous distance traveled by the robot, Prev-D, is equal to the current distance D. After turning and moving forward by a cleaning width W, the robot turns again to proceed with the next path sweep, while recording the current number of turns T. Fig.5 illustrates the robot freely planning and sweeping in space according to this path strategy.

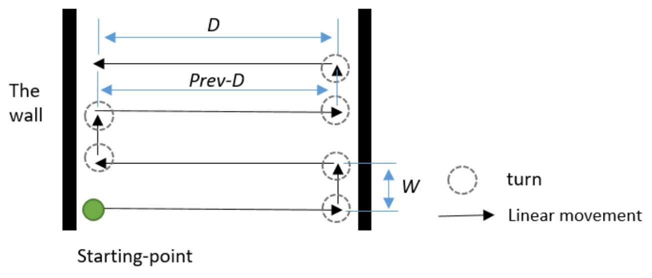


Fig.5 Robot sweeping path planning

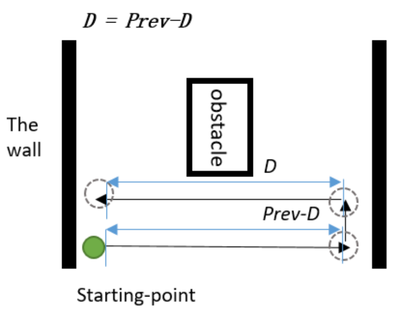
The pseudo-code for sweeping path planning is as follows:

```
1:  while true do
2:    move_forward
3:    if ultrasonic = trigger then
4:        $D$ = moved_distance
5:        turn
6:        move distance $W$
7:      turn
8:      $T = T + 1$
9:          ……do something
10:     $Prev\text{-}D = D$
11:     end
12: end
```

Obstacle Uncovered Area Length Calculation and Avoidance:

Case 1: D = Prev-D, where the current distance D traveled by the robot is equal to the previous distance, indicating that the robot is in a parallel passage (Fig.6(a)). The turn count T is incremented and recorded.

Case 2: D + Φ < Prev-D, where the current distance D traveled by the robot plus the robot's diameter Φ is less than the previous distance Prev-D. This indicates that the robot has encountered an obstacle. In this case, the turn count T is reset and counting starts again (Fig.6(b)). The robot then turns and continues sweeping according to the cleaning path.



Scenario 1: Parallel passage

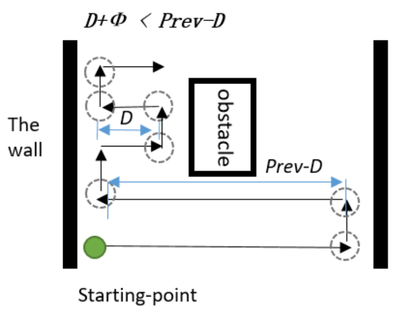


Fig.6(b) Scenario 2: Obstacle

Case 3: D -Φ > Prev-D, the distance of the current robot moving D minus the diameter of the robot Φ is greater than the distance of the last movement Prev-D, indicating that the robot has stepped out of the obstacle Fig.7. At this time, the length of the uncovered area of the obstacle is obtained by the steering number T and the cleaning width W.

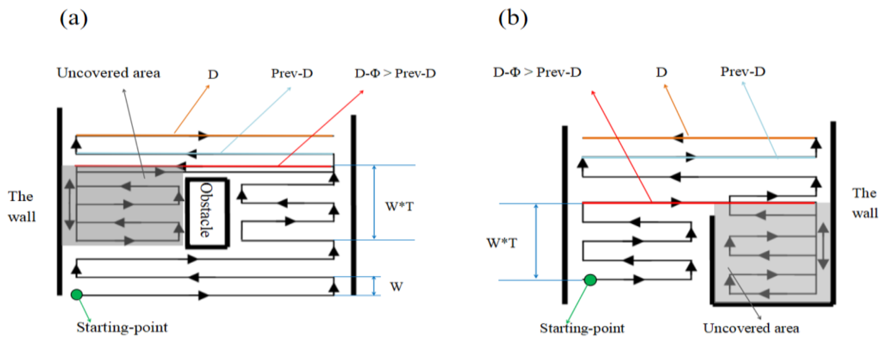


Fig. 7. The robot encountering two different obstacles. (a) Obstacle 1. (b) Obstacle 2.

After obtaining the length of the obstacle uncovered area in Scenario 3, the robot moves in the opposite direction by that length, reaches the uncovered area, and performs cleaning. This approach achieves full coverage and eliminates issues such as multiple cleaning of the same area.

Pseudo-code for obstacle uncovered area length calculation is as follows:

```
1    Sweeping path planning…
2:   if $D = Prev\text{-}D$ then
3:     $T = T + 1$
4:   end
5:   if $D + \Phi < Prev\text{-}D$ then
6:     $T = 0$
7:   end
8:   if $D - \Phi > Prev\text{-}D$ then
10:    Length = W * T
11:  end
```

Program logic flow chart of the proposed improved intelligent obstacle avoidance algorithm based on traditional algorithm is illustrated in Fig. 8.

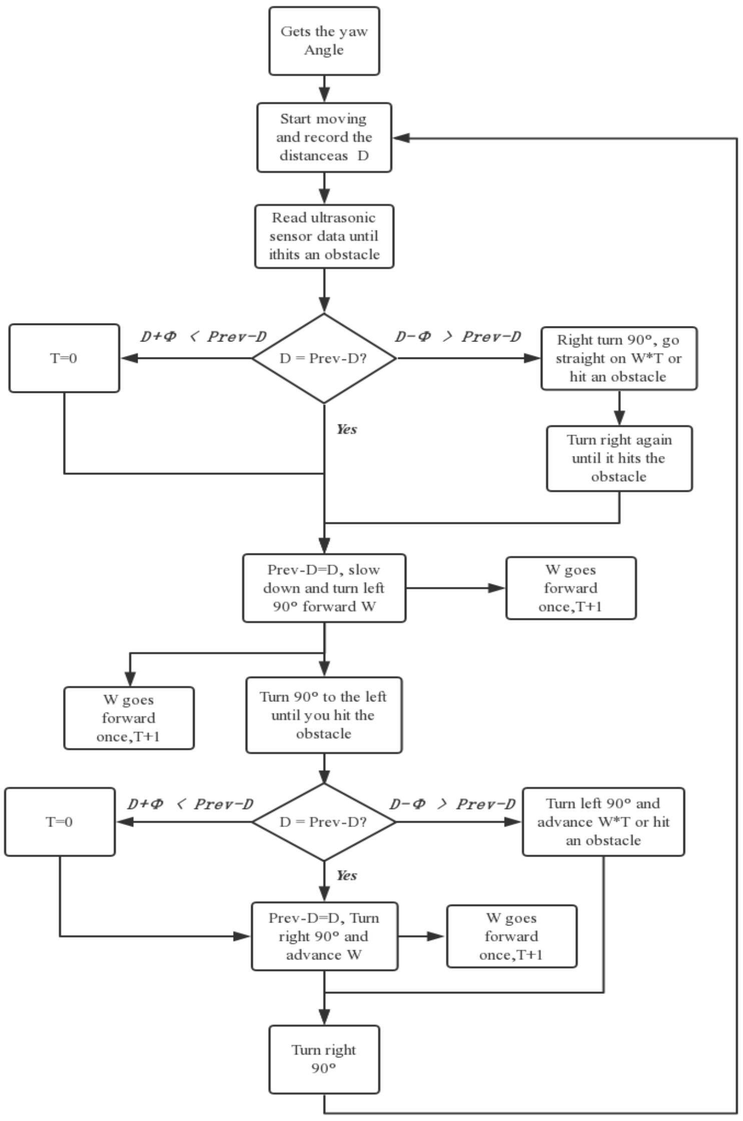

Gets the yaw Angle
Start moving and record the distanceas D
Read ultrasonic sensor data until ithits an obstacle
D+Φ < Prev-D
D = Prev-D?
D-Φ > Prev-D
T=0
Right turn 90°, go straight on W*T or hit an obstacle
Yes
Turn right again until it hits the obstacle
Prev-D=D, slow down and turn left 90° forward W
W goes forward once,T+1
W goes forward once,T+1
Turn 90° to the left until you hit the obstacle
D+Φ < Prev-D
D = Prev-D?
D-Φ > Prev-D
T=0
Turn left 90° and advance W*T or hit an obstacle
Yes
Prev-D=D, Turn right 90° and advance W
W goes forward once,T+1
Turn right 90°

Fig.8 The flow chart of the proposed algorithm.

3.3 Motion Path Tracking Control Algorithm

The robot uses linear motion trajectories and rotational motion to complete the entire cleaning path during the cleaning process (Fig.9). The robot always moves in a fixed horizontal direction and stops moving and rotates 90° when it detects a wall before proceeding with the next linear trajectory.

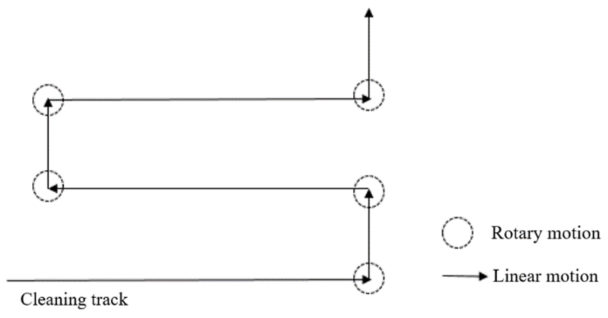


Fig.9 Robot cleaning path

(2) Closed-loop control algorithm

PID controller is a negative feedback control algorithm, the output quantity composed of proportional, integral, differential linear combination. PID is commonly used in industrial process control, has a simple structure, robustness and other characteristics, PID control algorithms can usually be composed of a P controller, PI controller, PD controller and PID controller, Fig.10.

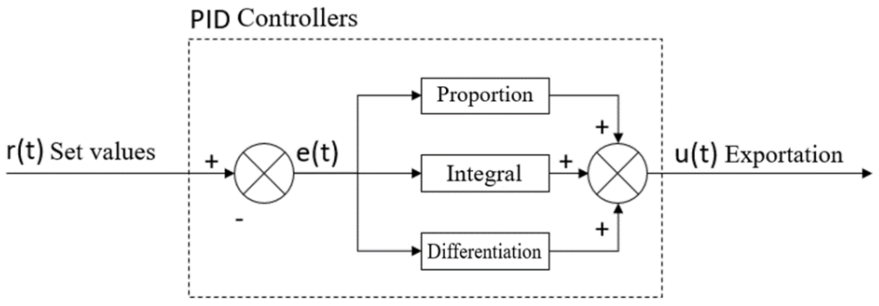


Fig.10 PID Control algorithm block diagram

PID control algorithm:

$$\begin{aligned}\mathrm{u}(t) &= K_p\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\,\frac{de(t)}{dt}\right] \\ &= K_p e(t) + K_i\int_0^t e(t)dt + K_d\,\frac{de(t)}{dt}\end{aligned} \quad \textbf{(1)}$$

Where: r(t) is the set value and u(t) is the output.

(3) Closed-loop control of rotary motion

The symbols are defined below:

Variable	Definition	Initial value
θ_{fb}(t)	Robot yaw angle at time t	
θ_{set}	Setting the robot yaw angle	
$\dot{\theta}$(t)	The robot's linear velocity z at time t	

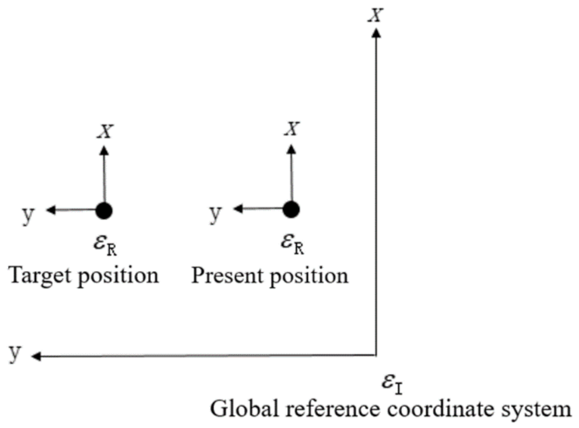
Use the current robot attitude as a feedback value to set the angle at which the robot needs to rotate.(90°,-90°…)The angular velocity of the robot rotating around the z-axis is calculated by the PD control algorithm (Eq. 1) $\dot{\theta}$(t)。

$$e(t) = \theta_{\mathrm{set}} - \theta_{fb}(t)$$

$$\dot{\theta}(t) = K_p \cdot e(t) + (K_p \cdot T_d \cdot \frac{e(t) - e(t-1)}{dt}) \quad \textbf{(2)}$$

(4) Linear point-to-point control algorithm

Given the target coordinates of the robot, combine the robot coordinates with the feedback to calculate the robot's linear velocity, and set the target coordinates in the x-direction to infinity to make the robot always move forward during the sweeping movement.



Symbol Definition:

Variable	Definition	Initial value

$\varepsilon_{\mathrm{set}} = [\mathrm{x} \quad y \quad \theta]^T$	Target coordinate	
$\dot{\varepsilon}_R = [x \quad y \quad \theta]^T$	Linear velocity of the robot's reference coordinate system	
$\varepsilon_{\mathrm{I}} = [x \quad y \quad \theta]^T$	Current machine position (global reference coordinate system)	

Calculate target position point $\rho = \begin{bmatrix} \mathrm{r} \\ \theta \end{bmatrix} = \begin{bmatrix} \sqrt{(\varepsilon_{set}.x - \varepsilon_I.x)^2 + (\varepsilon_{set}.y - \varepsilon_I.y)^2} \\ \arctan(\frac{\varepsilon_{set}.y - \varepsilon_I.y}{\varepsilon_{set}.x - \varepsilon_I.x}) \end{bmatrix}$

Calculate the target position point, calculated using the PD control algorithm, with a set value of 0 i.e. the expected straight-line distance between the two points before is equal to 0.

Calculation bias

$$\boldsymbol{e(t) = 0 - \rho.r}$$

Calculate output

$$\boldsymbol{v_{sum}(t) = K_p \cdot e(t) + (K_p \cdot T_d \cdot \frac{e(t) - e(t-1)}{dt})}$$

Calculate the angular velocity z of the robot using the PD algorithm $\dot{\theta}$。

Calculation bias

$$\boldsymbol{e(t) = \rho.\theta - \varepsilon_I.\theta}$$

Calculate output

$$\boldsymbol{\dot{\theta} = K_p \cdot e(t) + (K_p \cdot T_d \cdot \frac{e(t) - e(t-1)}{dt})}$$

Transforms the closing speed to the linear speed in the robot's reference coordinate system. $\dot{\varepsilon}_R$

$$\dot{\varepsilon}_{\mathrm{R}} = \begin{bmatrix} \cos(\varepsilon_I.\theta) & \sin(\varepsilon_I.\theta) & 0 \\ -\sin(\varepsilon_I.\theta) & \cos(\varepsilon_I.\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v_{sum} \cdot \cos(\rho.\theta) \\ v_{sum} \cdot \sin(\rho.\theta) \\ \dot{\theta} \end{bmatrix}$$

(5) Velocity Smoothing Algorithm

When the robot encounters an obstacle or a wall and stops during the cleaning process, the acceleration limiting algorithm combined with the first-order low-pass filtering algorithm is used to filter the linear velocity, which ensures smooth motion speed and reduces the positional projection error due to slippage. The details are shown in Figure 11 below.

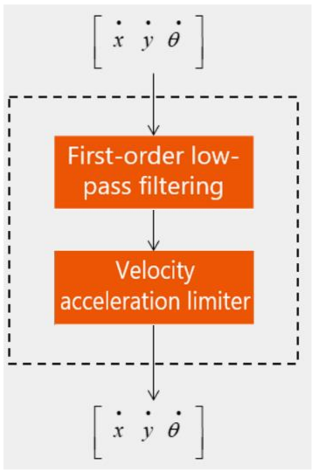

$\begin{bmatrix} \dot{x} & \dot{y} & \dot{\theta} \end{bmatrix}$
First-order low-pass filtering
Velocity acceleration limiter
$\begin{bmatrix} \dot{x} & \dot{y} & \dot{\theta} \end{bmatrix}$

Fig. 11 Velocity smoothing method

Symbol Definition:

Variable	Definition	Initial value
$v(t)$	Current speed at time t	
acc_{max}	maximum acceleration	
$acc(t)$	Current acceleration at time t	
$v(t)_{fiter}$	Current filtered velocity at time t	
$k_{lowpass}$	Low-pass filtering factor	

Algorithmic step:

Low-pass smoothing filtering for current input speeds

$$v_{lowpass}(t) = (1 - k_{lowpass}) \cdot v(t-1) + k_{lowpass} \cdot v(t)$$

included among these: $k_{lowpass} \in [0,1]$

Calculate current acceleration

$$acc(t) = \frac{dv}{dt} = \frac{v(t) - v(t-1)}{dt}$$

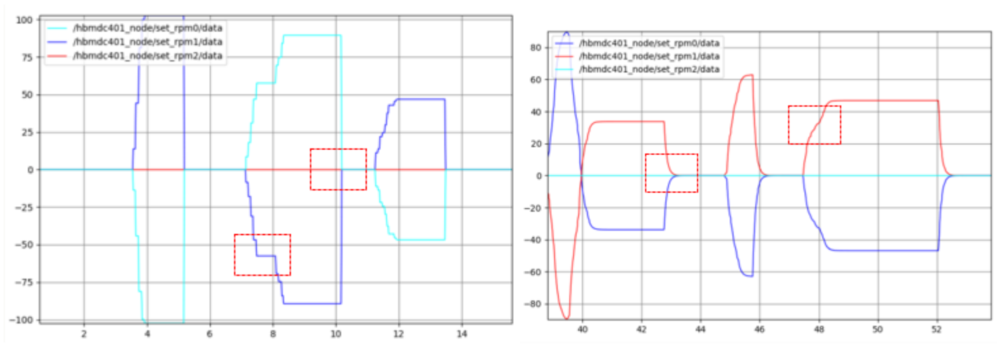
Magnitude limiting of acceleration

```
$if \quad \|acc(t)\| > acc_{\max} \quad then$
  $acc(t) = sig(acc(t)) \cdot acc_{\max}$
$end$
```

Recalculate the velocity that satisfies the maximum acceleration constraint

$$v_{\mathrm{filter}}(t) = acc(t) \cdot dt + v(t-1)$$

The comparison before and after speed smoothing is shown below***:***



(a) Raw speed (b) Speed after smoothing

Fig.12 Comparison before speed smoothing

4. The design of experiment platform

We used a conventional sweeping robot for the evaluation of the developed algorithm. Fig. 13(a) shows a conventional sweeping robot. This robot used infrared and ultrasonic sensors to detect obstacles. In addition, this robot was equipped with a gyroscope MPU6050 to calculate rotation angle [28].

The robot adjusted the direction based on the path planning function of the code plate and gyroscope to ensure trajectory accuracy. Robot motion control was handled by MPU6050 module. This was a 6-axis module and used sensors such as gyroscopes. MPU6050 module was able to accurately calculate parameters such as wheel speed, providing a reference basis for robot path planning.

Conventional sweeping robots have a 16V battery with the capacity of 9000mAh. This battery provided the necessary power to drive the motor. The robots also had another battery of 5V which was used to power up the control unit.

STM32F429IGT6 was applied as control chip. This chip comprised a floating-point unit (FPU) which had high performance and low power consumption [29]. It processed the real-time data collected by MPU6050 sensor [30]. L298N motor driving chip was used which realized the driving control of the motor used for robot motion. Throughout the entire control process, the main control unit played the most important roles, such as controlling robot motion, driving robot wheels, etc.

The hardware control scheme of the entire robot is illustrated in Fig. 13(b).

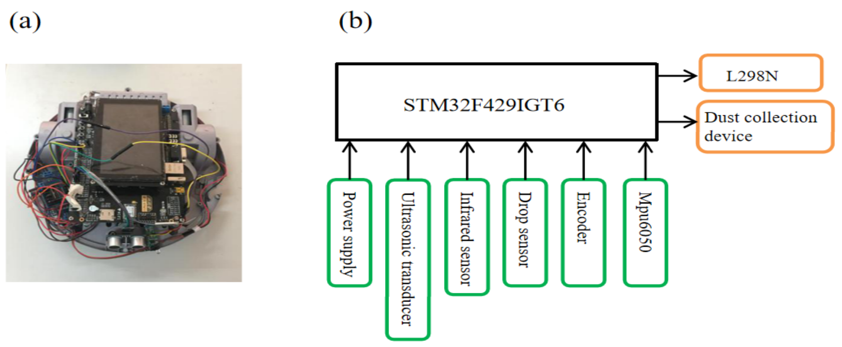


Fig.13. Design principle of the experimental system used in this work for evaluation. (a) Experiment platform. (b) Hardware control system.

5. Experimental results

As described in this section, several experiments were conducted to verify the performance of the proposed algorithm. On the one hand, the practicality of path planning methods for robots, especially the accuracy of algorithms, needed to be verified through experiments. On the other hand, handling the missing areas is also an important task for robots. In addition, robots functions in indoor environments also needed to be verified.

The operating system used in the experiment is Windows 10, and the software is programmed using C++ language.

We calculated the search range of robots using the following expression.

$$R_c = \frac{d_1 \times W + d_2 \times W + \cdots + d_{n-1} \times W + d_n \times W}{S - S_O} \qquad (8)$$

where dn is the distance passed by the robot in each horizontal movement, S denotes target cleaning area, and S_O is obstacles area.

It is noteworthy that, in real-world scenarios, a cover exists among different robot paths, thus rendering Eq. (8) inappropriate. In order to address this problem, the search range was changed to SC which should include areas that had not been processed previously. In this case, Rc was calculated by using the following equation.

$$R_c = \frac{S - S_C}{S} \qquad (9)$$

5.1. Testing of U-shaped path algorithm

During the experiment, it was necessary to fully consider robot hardware, especially its control system. The motor used to control robot motion could reach a maximum speed of over 30cm/s. Considering robot motion safety, only 20cm/s was taken here. Throughout the entire experiment, the robot followed a U-shaped route, as illustrated in Fig. 14.

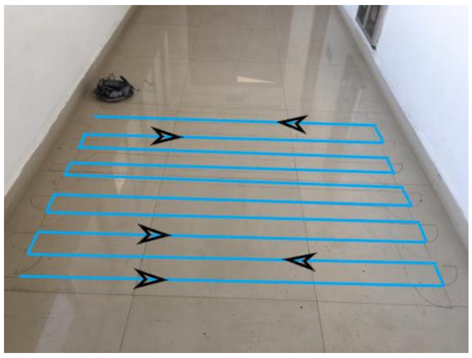


Fig. 14. U-shaped route followed by the robot in this experiment.

By repeating the same experiment multiple times, robot motion stability could be better examined. Based on this testing method, the robot was allowed to take 7 U-shaped routes during its motion and then examine the deviations occurring in each action. The corresponding results are illustrated in Fig. 9.

It was seen from the figure that the robot experienced cumulative errors during the process of continuously repeating the path, with a maximum cumulative cost of 7mm after 7 iterations. The reason for this deviation was related to robot hardware control error.

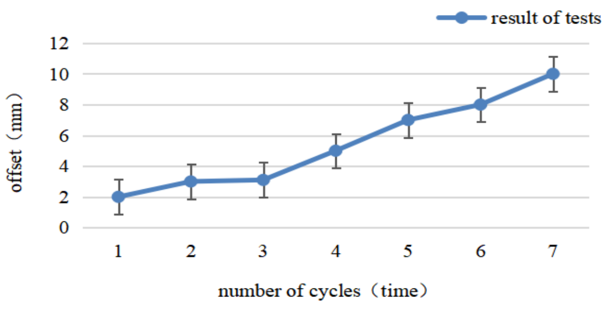


Fig.15. Offset in the distance computed after the completion of 7 cycles.

5.2. Improved intelligent obstacle avoidance algorithm test

In this research, we evaluated the proposed algorithm under two different obstacle situations.

We conducted five experiments to evaluate the first case and calculated coverage rate. Fig. 10 shows the route followed by the robot during this experiment. Please note that the box in the hallway presented in Fig. 16 covered 0.4 m^2. On the other hand, the area of the test field was about 4 m^2. Corresponding results are summarized in Table 1. The error depicted in the results was a cumulative offset due to multiple repeated motion processes.

Then, five repeated experiments were performed again to verify the coverage of the search method, as illustrated in Fig. 11. Based on the statistical results given in Table 2, it was seen that the coverage obtained from the five repeated experiments fluctuated around 94%.

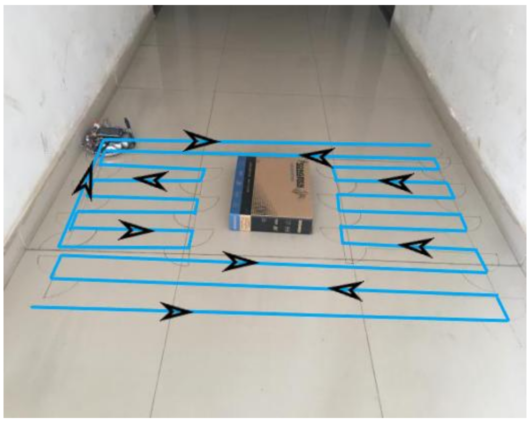


Fig. 16. The route followed by the robot in case 1.

Table 2

The results of coverage tests for case 1

Test	No. 1	No. 2	No. 3	No. 4	No. 5
S-S_C	3.659	3.588	3.686	3.621	3.607
S	3.837				
R_c	95.36%	93.57%	96.08%	94.38%	94.11%

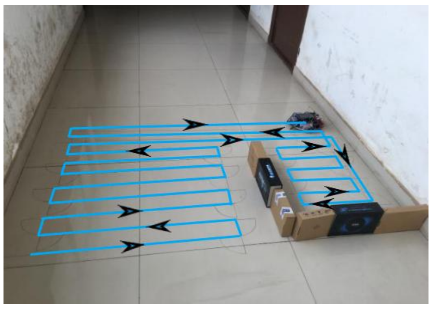


Fig. 17. The route followed by the robot in case 2.

5.3. Comprehensive Test

We performed the final test to evaluate and compare the U-shaped path and proposed algorithms. This experiment aimed to examine the level that the proposed method could achieve for robot motion in indoor environments, especially in complex situations.

Fig. 18 illustrates the route followed by the robot during this test. It was seen that the coverage rate in this experiment was 94.61%.

When the robot detected the wall, which indicated that the robot could not go further, the cleaning task was ended.

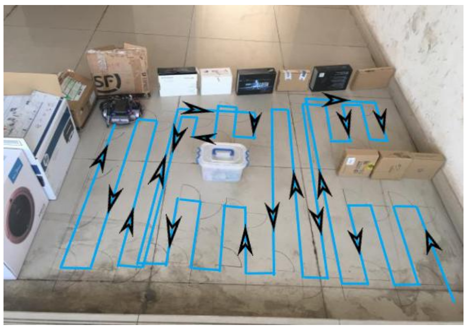


Fig. 18. The route followed by the robot during the comprehensive test.

6. Conclusions

According to the analysis of sweeping robot motion model, in this work, we proposed a new method for sweeping robot motion. Also, U-shaped path was applied as the reference path for robot motion. Robot route was mainly based on U-shaped route. It is noteworthy that, based on the proposed algorithm, the robot memorized the area behind the obstacle by comparing the existing state with previous state. It also remembered the area that was cleaned in order to avoid repeated cleaning. During the experiment, random programming was adopted as the reference method. Compared to this method, the proposed method enabled the robot to achieve 94.61% coverage of cleaning area. This experiment showed that the proposed method had significantly improved robot efficiency. The experimental results of stability also confirmed that the maximum cumulative offset of the proposed method was 7mm. These results indicated that the robot using the proposed algorithm had good path-keeping ability. The proposed algorithm was evaluated for two types of obstacles to show that it had the ability to control the robot during cleaning process behind the obstacles and enabled it to return to the main path after cleaning. In order to comprehensively test the proposed algorithm, we set up a complex experimental environment to test whether the robot could work in this environment. The experimental results fully confirmed that the proposed method enabled robots to have better adaptability, especially for complex environments where cleaning work could be completed. In future research, we will examine algorithm performance improvement from different perspectives to further enhance robot motion performance.

REFERENCE

[1]. Miao X, Lee H S, Kang B Y. Multi-cleaning robots using cleaning distribution method based on map decomposition in large environments. IEEE Access, 2020, 8: 97873-97889.

[2]. Sharma G, Dutta A, Kim J H. Optimal online coverage path planning with energy constraints//Proceedings of the 18th international conference on autonomous agents and multiagent systems. 2019: 1189-1197.

[3]. Sung I, Choi B, Nielsen P. On the training of a neural network for online path planning with offline path planning algorithms. International Journal of Information Management, 2021, 57: 102142.

[4]. Wang L, Liu L, Qi J, et al. Improved quantum particle swarm optimization algorithm for offline path planning in AUVs. IEEE Access, 2020, 8: 143397-143411.

[5]. Schmid L, Pantic M, Khanna R, et al. An efficient sampling-based method for online informative path planning in unknown environments. IEEE Robotics and Automation Letters, 2020, 5(2): 1500-1507.

[6]. Laghmara H, Boudali M T, Laurain T, et al. Obstacle avoidance, path planning and control for autonomous vehicles//2019 IEEE intelligent vehicles symposium (IV). IEEE, 2019: 529-534.

[7]. Wang P, Gao S, Li L, et al. Obstacle avoidance path planning design for autonomous driving vehicles based on an improved artificial potential field algorithm. Energies, 2019, 12(12): 2342.

[8]. Chang L, Shan L, Jiang C, et al. Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment. Autonomous Robots, 2021, 45: 51-76.

[9]. Schmid L, Pantic M, Khanna R, et al. An efficient sampling-based method for online informative path planning in unknown environments. IEEE Robotics and Automation Letters, 2020, 5(2): 1500-1507.

[10]. Qi J, Yang H, Sun H. MOD-RRT*: A sampling-based algorithm for robot path planning in dynamic environment. IEEE Transactions on Industrial Electronics, 2020, 68(8): 7244-7251.

[11]. Lluvia I, Lazkano E, Ansuategi A. Active map** and robot exploration: A survey. Sensors, 2021, 21(7): 2445.

[12]. Ab Wahab M N, Nefti-Meziani S, Atyabi A. A comparative review on mobile robot path planning: Classical or meta-heuristic methods?. Annual Reviews in Control, 2020, 50: 233-252.

[13]. Shin H, Chae J. A performance review of collision-free path planning algorithms. Electronics, 2020, 9(2): 316.

[14]. Cabreira T M, Brisolara L B, Paulo R F J. Survey on coverage path planning with unmanned aerial vehicles. Drones, 2019, 3(1): 4.

[15]. Wai R J, Prasetia A S. Adaptive neural network control and optimal path planning of UAV surveillance system with energy consumption prediction. Ieee Access, 2019, 7: 126137-126153.

[16]. Lutz P, Schuster M J, Steidle F. Visual-inertial SLAM aided estimation of anchor poses and sensor error model parameters of UWB radio modules//2019 19th International Conference on Advanced Robotics (ICAR). IEEE, 2019: 739-746.

[17]. Gao J, Ye W, Guo J, et al. Deep reinforcement learning for indoor mobile robot path planning. Sensors, 2020, 20(19): 5493.