

ZHEJIANG  
UNIVERSITY PRESSThe Institution of  
Engineering and Technology

WILEY

## ORIGINAL RESEARCH

# Spherical robot: A novel robot for exploration in harsh unknown environments

Wei Ren | You Wang | Haoxiang Liu | Song Jin | Yixu Wang | Yifan Liu | Ziang Zhang | Tao Hu | Guang Li

State Key Laboratory of Industrial Control Technology, Institute of Cyber Systems and Control, Zhejiang University, Hangzhou, China

**Correspondence**

You Wang,

Email: [king\\_wy@zju.edu.cn](mailto:king_wy@zju.edu.cn)

**Funding information**

Fundamental Research Funds for the Central Universities, Grant/Award Number: No.226-2022-00086

**Abstract**

The authors propose a complete software and hardware framework for a novel spherical robot to cope with exploration in harsh and unknown environments. The proposed robot is driven by a heavy pendulum covered by a fully enclosed spherical shell, which is strongly protected, amphibious, anti-overturn and has a long-battery-life. Algorithms for location and perception, planning and motion control are comprehensively designed. On the one hand, the authors fully consider the kinematic model of a spherical robot, propose a positioning algorithm that fuses data from inertial measurement units, motor encoder and Global Navigation Satellite System, improve global path planning algorithm based on Hybrid A\* and design an instruction planning controller based on model predictive control (MPC). On the other hand, the dynamic model is built, linear MPC and robust servo linear quadratic regulator algorithm is improved, and a speed controller and a direction controller are designed. In addition, based on the pose and motion characteristics of a spherical robot, a visual obstacle perception algorithm and an electronic image stabilisation algorithm are designed. Finally, the authors build physical systems to verify the effectiveness of the above algorithms through experiments.

**KEYWORDS**

mobile robots, robot control, robot perception, robot planning

## 1 | INTRODUCTION

It is highly risky to explore harsh environments and unknown environments, such as wild, fire fields and underground caves. Using mobile robots to replace humans in these exploration tasks can effectively ensure human health and safety. According to different motion principles, mobile robots can be divided into many types, including classic wheeled robots and tracked robots, popular legged robots and unmanned aerial vehicles. Each type of robot has its pros and cons. Wheeled robots have low energy consumption and high speed, but their capabilities of protection and obstacle-crossing are relatively general. Tracked robots have a stronger ability to cross obstacles and are suitable for harsh terrains, but they have the same shortcomings as wheeled robots in that they are afraid of overturning. Legged robots have amazing obstacle-crossing

capabilities, but they also have low energy efficiency [1]. The above three kinds of ground robots cannot be used amphibiously. As aerial robots, Unmanned aerial vehicles are very flexible, and not restricted by ground obstacles, but they are also fragile, with poor payload and endurance [2]. Taking the exploration of unknown underground caves as an example, there might be an underground river and chaotic rocks, which require amphibiousness, protection and anti-overturning performance of the robot. Furthermore, the above environment can be a large confined space that denies Global Navigation Satellite System (GNSS) signal and is difficult to communicate with outside in real-time, requiring the robot explorer to have high autonomy and endurance.

Spherical robot is a new kind of mobile robot, as shown in Figure 1. It has a fully enclosed spherical shell outside and a heavy pendulum installed inside with which the robot realises

This is an open access article under the terms of the [Creative Commons Attribution-NonCommercial-NoDerivs](https://creativecommons.org/licenses/by-nc-nd/4.0/) License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

© 2023 The Authors. *IET Cyber-Systems and Robotics* published by John Wiley & Sons Ltd on behalf of Zhejiang University Press.



**FIGURE 1** One of our designed spherical robots.

forward/backward movement and left/right deflection. Power supply module, drive module, various controllers and sensors are installed inside the spherical shell. This design has several advantages. First, the fully enclosed shell endows spherical robots with strong protection capabilities, such as scratch resistance, collision resistance and drop resistance, so that equipment inside the robot can work properly even in harsh environments. Second, the fully enclosed shell provides convenience for spherical robots to expand their amphibious capabilities so that they can move and work normally even in water. Third, the design of the spherical shell and heavy pendulum ensures that spherical robots can automatically return to normal after being disturbed by external forces—they will never overturn just like tumblers. Fourth, the point contact mode between spherical shell and ground provides flexible motion ability, low energy consumption and long battery life. Therefore, the spherical robot proposed in this paper is very suitable for performing tasks such as security inspections, exploration and investigation in harsh and unknown environments.

Mechanical structure is one of the main research directions of spherical robots. In the earliest studies on spherical robots, the design of mechanical structure was relatively simple, such as the spherical robot whose outer spherical shell was indirectly driven by a differential drive mechanism [3] or by a universal wheel structure [4]. After that, spherical robots developed more complex mechanical structures. Pendulum-driving-structure was most complex, such as BHQ-1 and BHQ-2 robots proposed by Zhan et al. [5, 6] and a four-pendulum-driven spherical robot proposed by DeJong et al. [7]. In addition, some other proposed spherical robots moved by changing the position of mass blocks inside [8], or by the principle of conservation of angular momentum [9].

Non-linear, under-actuated and non-holonomic characteristics lead to complex control problems. In recent years, research focus of spherical robots has gradually shifted to motion control

[10]. More studies were based on proportional–integral–derivative (PID) controllers [11, 12]. They were free from complex dynamic models, but were troubled by low control accuracy, large overshoot and long setting time. Model-based sliding mode control also received attention [13, 14], which was more accurate but suffered from the free-chattering problem. Linear quadratic regulator (LQR) controller achieved a good velocity control effect in simulation environment, but due to model mismatch test results were not satisfactory in physical world [15]. The above researches were mainly aimed at trajectory tracking or velocity control and less at lateral motion control. In addition, most researches were limited to simulation environments and did not take into account the impact of disturbances such as terrain changes in the physical environment.

Spherical shell will shake the robot, which will affect localisation and perception performance. The special way to achieve movement means that path planning algorithm requires more in-depth research, in order to find an optimal followable path. But so far, there are very few researches on localisation, perception and path planning of spherical robots. In response to the above problems, we have conducted comprehensive research on mechanical structure, localisation and perception, path planning and motion control of spherical robots. We also have established several versions of physical spherical robots for experiments.

The main contribution of this paper is to summarise and improve our latest research results on spherical robots, including hardware, perception, planning and control and provide readers with a comprehensive understanding of this new kind of robotic system. In terms of algorithms, the main innovations of this paper are as follows:

- Perception algorithm. For the first time, a wheel odometry algorithm and an electronic image stabilisation (EIS) algorithm are proposed for spherical robot based on motion characteristics. In addition, a spherical robot visual obstacle detection method based on Semi-Global Matching algorithm is developed.
- Planning algorithm. The global path planning algorithm based on Hybrid A\* is improved, and a three-circle coverage strategy is proposed as a collision-free constraint. Based on previous work, an obstacle-free constraint is added to the instruction planning controller.
- Control algorithm. We summarise the latest motion control algorithms based on the robot's dynamic model and make certain improvements on the basis of previous works, such as adding compensation to the velocity controller according to the current roll angle of robot.

The structure of the following content is as follows. Section 2 presents an overview of spherical robot system, including hardware design and algorithm framework. Section 3 introduces localisation, perception and monitoring algorithms. Section 4 presents planning algorithms. Section 5 presents motion control algorithms. Section 6 shows the experimental results. Section 7 gives the conclusion.

## 2 | SYSTEM OVERVIEW

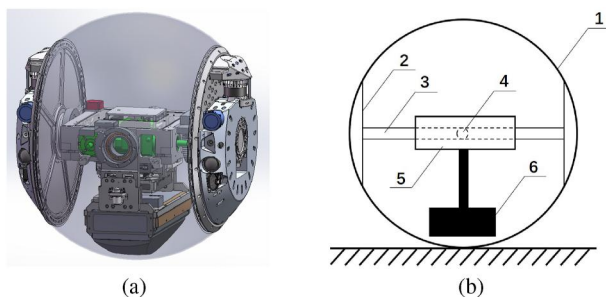
### 2.1 | Hardware design

As mentioned above, the spherical robot proposed in this paper is driven based on a single pendulum structure. Its detailed mechanical structure is shown in Figure 2(a), while the simplified structure is shown in Figure 2(b) [16]. As can be seen, there is a thick main axis across the interior of spherical robot, with a frame structure set in the centre, and flanges set on both sides to connect to the spherical shell. The frame incorporates an auxiliary axis orthogonal to the main axis, and a heavy pendulum is connected to the auxiliary axis under the frame. The two axes are driven by two motors: the main motor and the auxiliary motor, which are installed orthogonally in the frame, so as to realise the front and rear pull up and left and right deflection of the heavy pendulum, thereby changing the position of the centre of mass of spherical robot. The shell of spherical robot consists of three parts. The middle part is in contact with ground. When the position of the robot's centre of mass changes, this part of shell rolls to realise movement of the entire robot. Shells on both sides are rigidly connected to main axis. They are relatively static, sharing the same pose as the main axis, and do not roll as the middle part.

Spherical robot is equipped with various sensors such as a GNSS module, inertial measurement unit (IMU), motor encoder and camera. GNSS module and IMU are set inside the frame structure, encoders are set inside the two motors and cameras are set in the spherical shells on both sides. Due to a lack of space and energy saving considerations, cameras are rigidly connected to main axis instead of being mounted on pan-tilt platforms.

To assemble a spherical robot, we also need an electrical system, a computing system composed of a mini PC and a series of microcontroller unit (MCU), and a system responsible for internal and external communication. These systems are installed on the pendulum or in the frame structure.

Based on the above mechanical structure, a set of general coordinate systems can be established for the spherical robot. By setting the sphere centre as coordinate origin, forward, left and upper directions as the positive directions of  $x$ -axis,  $y$ -axis



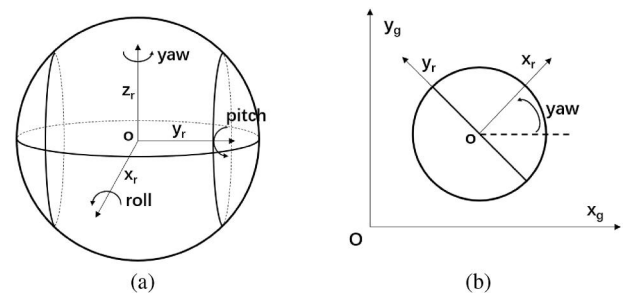
**FIGURE 2** Mechanical structure of a version of spherical robot. (a) Detailed structure. Green parts are motors, red part is inertial measurement unit (IMU), blue parts are cameras. (b) Simplified structure (front view). Component 1 is the shell, 2 is the flange, 3 is the main axis, 4 is the auxiliary axis, 5 is the frame, 6 is the pendulum.

and  $z$ -axis, respectively, we can establish a robot coordinate system, as shown in Figure 3(a). Three Euler angles are formed around each coordinate axis, their positive directions are marked in Figure 3(a). By establishing a global coordinate system, the relationship between the two coordinate systems is shown in Figure 3(b).

In addition, we can define some commonly used physical quantities, as shown in Table 1. In the subsequent part of this paper, the above definition of coordinate systems and physical quantities will be used many times.

### 2.2 | Algorithm framework

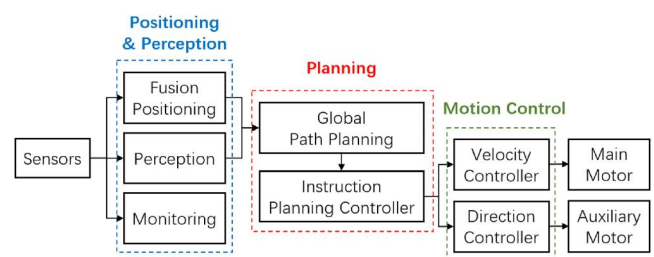
The software data flow diagram of spherical robot is shown in Figure 4. Algorithm framework includes three modules: positioning and perception, planning and motion control. Data read from sensors are processed by those modules and finally uploaded to the background or sent to motors to execute motion instructions. Each algorithm module is summarised as follows.



**FIGURE 3** The established coordinate systems. (a) Design of robot coordinate system and its rotation direction. (b) Relationship between global coordinate system and robot coordinate system (top view).

**TABLE 1** Commonly used physical quantities.

Symbols	Description
$x, y$	X and y position in global coordinate system
$v$	X-direction velocity in robot coordinate system
$\varphi, \psi, \theta$	Roll, pitch and yaw angle of robot
$r$	Radius of spherical shell



**FIGURE 4** Software data flow of spherical robot.

- Positioning and perception. It is divided into three sub-modules. The first sub-module is fusion positioning system, which fuses raw data of sensors such as GNSS, IMU and motor encoders to generate fusion positioning results. The second sub-module is perception system, which is responsible for generating surrounding environment perception information such as obstacles. The third sub-module is a monitoring system, which reads from IMU and camera data, and after image stabilisation processing, the monitoring data are recorded or uploaded to background for operators.
- Planning. Based on the characteristics of spherical robot, an improved Hybrid A\* algorithm and a collision-free constraint judgement method are designed for the global path planning sub-module. As for the instruction planning controller, it is designed based on kinematic model of a spherical robot, and used to tell the robot how to track the reference path given by global planning sub-module with minimum cost.
- Motion control. The velocity controller is designed based on offset-free linear model predictive control (MPC), while the direction controller is designed based on robust servo LQR control with state compensation and velocity feedforward. Instructions from instruction planning controller are put into the two motion controllers, and output instructions from motion controllers are sent into the main and auxiliary motors to realise precise motion control of the spherical robot.

### 3 | POSITIONING AND PERCEPTION

Positioning and perception algorithm solves the problems of “where am I” and “what is around me”, which is the starting point for a spherical robot to realise autonomous action. The fusion positioning system, perception system and monitoring system are introduced below.

#### 3.1 | Fusion positioning system

Basic positioning algorithm is called wheel odometry, which is based on IMU and the encoders in two motors. Wheel odometry is built according to the kinematic model of a spherical robot, and has the advantages of being endogenous, anti-interference and all-weather. According to the coordinate systems established in Figure 3, and the physical quantities defined in Table 1, the kinematic model of a spherical robot can be obtained as shown in Equation (1) [16].

$$\begin{cases} \dot{x} = v \cos \theta - r \dot{\varphi} \sin \theta \\ \dot{y} = v \sin \theta + r \dot{\varphi} \cos \theta \\ \dot{\theta} = \frac{v \tan \varphi}{r} \end{cases} \quad (1)$$

Assuming that there is only rolling but no sliding between the spherical robot and ground, rewrite Equation (1) into

recursive form, we can get the wheel odometry algorithm of spherical robot as following Equation (2).

$$\begin{cases} x_k = x_{k-1} + v \cos \theta_{k-1} - r \dot{\varphi} \sin \theta_{k-1} \\ y_k = y_{k-1} + v \sin \theta_{k-1} + r \dot{\varphi} \cos \theta_{k-1} \\ \theta_k = \theta_{k-1} + \frac{v \tan \varphi}{r} \end{cases} \quad (2)$$

However, in some circumstances, the assumption of non-slip between spherical robot and ground cannot be maintained. For example, wheel odometry will no longer reliably reflect the robot's moving mileage on slippery terrain. Therefore, we introduced a Kalman filter to fuse GNSS and wheel odometry to achieve more robust fusion positioning effects. The algorithm formula is shown in Equation (3), and Table 2 shows the meaning of the symbols. Wheel odometry is used as the predicted value, while GNSS data are used as the observed value. Satellite signals are frequently obstructed or interfered within harsh environments, which results in the loss of observation values. With only the prediction part working, the fusion algorithm degenerates into the wheel odometry algorithm.

$$\begin{cases} \hat{x}_k^- = A \hat{x}_{k-1} + B u_{k-1} \\ P_k^- = A P_{k-1} A^T + Q \\ \hat{x}_k = \hat{x}_k^- + K_k (z_k - H \hat{x}_k^-) \\ K_k = \frac{P_k^- H^T}{H P_k^- H^T + R} \\ P_k = (I - K_k H) P_k^- \end{cases} \quad (3)$$

TABLE 2 Symbols used in Kalman filter.

Symbols	Description
$\hat{x}^-$	Prior estimation of state, including position and yaw angle
$\hat{x}$	Posterior estimation of state
$A$	System matrix, equals to identify matrix according to kinematic model
$B$	Input matrix
$u$	System input
$P^-$	Prior error
$P$	Posterior error
$Q$	Prediction covariance matrix
$K$	Kalman gain
$z$	Observation status
$H$	Observation matrix
$R$	Observation covariance matrix
$I$	Identity matrix

### 3.2 | Perception system

Objects 5 cm above the ground that are not suspended and within 10 m of a spherical robot may cause collisions. To avoid collisions, such objects must be spotted in real-time. Therefore, perception system is mainly used to detect obstacles around a spherical robot. As mentioned in Section 2.1, a camera rigidly connected to main axis is installed in spherical shells on both sides, which forms a stereo camera system with a baseline larger than 50 cm to facilitate obstacle detection.

The Semi-Global Matching algorithm is a classic high-precision stereo matching algorithm [17], which is implemented in the form of Semi-Global Block Matching (SGBM) algorithm in OpenCV. Based on this algorithm, we can conveniently and accurately obtain the disparity map of images from two cameras, which means we have obtained 3D camera coordinates  $\tilde{P}_c$  of obstacles. The most commonly used pinhole camera model, which describes the projection process of an object from the global coordinate system to pixel coordinate system, is suitable for cameras on spherical robots. According to the pinhole camera model, projection relationship of obstacles between global coordinate system and camera coordinate system is as follows:

$$\tilde{P}_c = TP_g \quad (4)$$

where  $P_g$  is the global coordinate of the obstacle, and  $T$  is the extrinsic matrix containing camera pose information. The pose of robot is given by the above fusion positioning system, so we can easily obtain the pose of cameras rigidly connected to main axis, that is, the extrinsic matrix  $T$ . Thus, the global coordinates of obstacles can be calculated. We set a certain threshold to filter out suspended, too far obstacles, and false obstacles with negative height values due to ground reflection, leaving only obstacles that may cause collisions, whose global coordinates are put into the following path planning module. So far, the visual obstacle perception work has been completed.

According to requirements, a spherical robot can also be equipped with lidar, and the obstacle perception algorithm based on lidar can be expanded.

### 3.3 | Monitoring system

Spherical robot needs to upload camera images in real-time to backstage operators. However, spherical shells will inevitably cause rapid and large shakes in images [18], which is very likely to cause dizziness for operators. As we introduced in Section 2, cameras are mounted on both sides of the spherical robot without pan-tilt platforms, so the EIS algorithm is necessary to produce stable and comfortable images.

Based on IMU data and perspective translation, the spherical robot's EIS algorithm is designed. IMU and cameras share the same Euler angles since they are rigidly connected to main axis. For roll and pitch angles of cameras, the expected value after EIS is  $0^\circ$ . According to kinematic model in Equation (1),

oscillation of roll angle will cause oscillation of yaw angle. The oscillation component of yaw angle needs to be filtered, but the active steering component should be maintained. Fortunately, the oscillation period is roughly fixed, by which we can design a sliding window mean filter to obtain the expected yaw angle. Thus, we can calculate the amount of three Euler angles to be adjusted by the following Equation (5), and Table 3 shows the meaning of the symbols:

$$\begin{cases} \varphi_{\exp} = \psi_{\exp} = 0 \\ \theta_{\exp} = \frac{\sum \theta_{win}}{n} \\ \varphi_{adj} = \varphi_{\exp} - \varphi_{now} \\ \psi_{adj} = \psi_{\exp} - \psi_{now} \\ \theta_{adj} = \theta_{\exp} - \theta_{now} \end{cases} \quad (5)$$

Then, based on the Equations (6) and (7), the amount to be adjusted for EIS represented by Euler angles is converted into a rotation matrix representation  $R_0$  for subsequent matrix operations:

$$R_0 = \begin{bmatrix} c_y c_r + s_y s_p s_r & -c_y s_r + s_y s_p c_r & s_y c_p \\ c_p s_r & c_p c_r & -s_p \\ -s_y c_r + c_y s_p s_r & s_y s_r + c_y s_p c_r & c_y c_p \end{bmatrix} \quad (6)$$

$$\begin{cases} s_r = \sin \varphi_{adj}, c_r = \cos \varphi_{adj} \\ s_p = \sin \psi_{adj}, c_p = \cos \psi_{adj} \\ s_y = \sin \theta_{adj}, c_y = \cos \theta_{adj} \end{cases} \quad (7)$$

In the next step, we will reproject the shaking images to a stable state based on  $R_0$ .

Optical characteristics and installation errors of the lens will introduce distortion to image. Therefore, the de-distortion process needs to be performed first when we obtain the raw image, based on the distortion model shown in Equation (8), while Table 4 shows the meaning of the symbols. Points in the model are located on the normalised plane of the camera, the origin of the coordinates is in the centre of image.

**TABLE 3** Symbols used in calculating the amount of Euler angles to be adjusted.

Symbols	Description
$\varphi_{\exp}, \psi_{\exp}, \theta_{\exp}$	Expected value of roll, pitch, yaw after stabilisation
$\varphi_{adj}, \psi_{adj}, \theta_{adj}$	The amount to be adjusted of roll, pitch, yaw
$\varphi_{now}, \psi_{now}, \theta_{now}$	Current value of roll, pitch and yaw
$\theta_{win}$	Historical yaw angles within the sliding window
$n$	Length of the sliding window, that is, the discretised oscillation period

$$\begin{cases} x_d = x_u(1 + k_1r^2 + k_2r^4 + k_3r^6) + 2p_1x_u y_u + p_2(r^2 + 2x_u^2) \\ y_d = y_u(1 + k_1r^2 + k_2r^4 + k_3r^6) + p_1(r^2 + 2y_u^2) + 2p_2x_u y_u \\ r^2 = x_u^2 + y_u^2 \end{cases} \quad (8)$$

For an undistorted image, projection relationship of an object between the global coordinate system and pixel coordinate system is shown in Equation (9):

$$zP_{uv} = KTP_g \quad (9)$$

where  $P_{uv}$  is the pixel coordinate of the projected object,  $K$  is intrinsic matrix of camera,  $z$  is pixel depth (scalar). When  $K$  and  $T$  are multiplied in Equation (9), a transformation from homogeneous coordinates to non-homogeneous coordinates is implied. In order to reproject the image, first multiply  $K^{-1}$  on both sides of Equation (9) to project the image from pixel coordinate system to camera coordinate system, as shown in Equation (10):

$$zK^{-1}P_{uv} = TP_g \quad (10)$$

Multiplying  $R_0$  on both sides of Equation (10) can change the shaking camera to a stable state. This is equivalent to transforming the extrinsic matrix of camera, limiting the three degrees of freedom of rotation to our desired state, as shown in Equation (11):

$$zR_0K^{-1}P_{uv} = (R_0T)P_g \quad (11)$$

Similarly, when  $R_0$  and  $T$  are multiplied in Equation (11), a transformation from homogeneous coordinates to non-homogeneous coordinates is implied. Then, multiply  $K$  to the left on both sides, project image from camera coordinate system back to pixel coordinate system, and finally obtain the pixel coordinate  $P'_{uv}$  of the object after EIS, as shown in Equation (12):

$$zP'_{uv} = zKR_0K^{-1}P_{uv} = K(R_0T)P_g \quad (12)$$

Therefore, we obtain the perspective transformation matrix  $T_p$  corresponding to the images before and after image stabilisation as follows:

$$T_p = KR_0K^{-1} \quad (13)$$

**TABLE 4** Symbols used in camera distortion model.

Symbols	Description
$x_d, y_d$	The x-coordinate and y-coordinate of point in distorted image
$x_u, y_u$	The x-coordinate and y-coordinate of point in undistorted image
$k_1, k_2, k_3$	Radial distortion parameters
$p_1, p_2$	Tangential distortion parameters

In summary, perspective transformation based on  $T_p$  can be used to reconstruct shaking images to the desired stable state. In order to reduce the black data-free area at the edge of images after reprojection, we can crop images appropriately. Finally, stabilised images are sent to a video compression card, and communication system is responsible for recording or uploading the compressed video stream together with other status data.

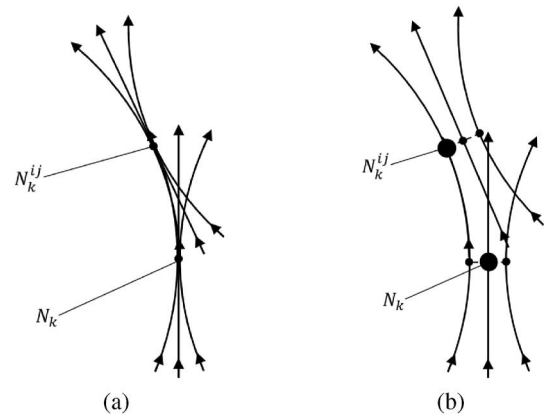
## 4 | PLANNING

Planning algorithm solves the problem of “how do I get to the target point”, which is the core of thinking and top-level design for spherical robots. As mentioned in Section 2, planning module is divided into two sub-modules, the global path planning and the instruction planning controller.

### 4.1 | Global path planning

Equation (1) illustrates the kinematic model of a spherical robot, which shows that it is a robot system subject to non-holonomic constraints. In design of the global path planning algorithm, the above kinematic constraints must be taken into account. Hybrid A\* algorithm adds non-holonomic constraints based on a wheeled robot [19], but since spherical robot's kinematic model differs from that of wheeled robot, spherical robot frequently finds it challenging to follow the planned path. By fully considering the law of lateral motion and tilt of spherical robots, we designed an online global path planning algorithm suitable for spherical robots by improving Hybrid A\* algorithm's node expansion rule in our previous work [20], and in this paper we further improve it.

According to the kinematic model in Equation (1), the changes of  $x, y$  coordinates and yaw angle of spherical robot is related to roll angle. That means, if roll angle of the parent node and the child node are different when the node expands in path planning algorithm, the sphere's centre will move laterally as a result, as shown in Figure 5. Since each node needs to consider



**FIGURE 5** Comparison of node expansion. (a) Wheeled robot. (b) Spherical robot.

more features, on the basis of Hybrid A\*, we add some necessary attributes to each node, as shown in Table 5.

For each child node  $N_c$ , its state update method is as follows, where child node attributes are represented by variables with the subscript c, while parent node attributes are represented by variables with the subscript p.

$$\begin{cases} x_c = x_p + v_c \Delta t (\sin \theta_c - \sin \theta_p) - r \Delta \varphi \sin \varphi_p & (14a) \\ y_c = y_p + v_c \Delta t (-\cos \theta_c + \cos \theta_p) + r \Delta \varphi \cos \varphi_p & (14b) \\ \theta_c = \theta_p + v_c r^{-1} \tan \varphi_c \Delta t & (14c) \\ \Delta t = t_c - t_p & (14d) \\ \Delta \varphi = \varphi_c - \varphi_p & (14e) \\ g_c = g_p + g_l + g_d + g_r & (14f) \\ h_{1c} = A^*_{\text{method}}(N_c, \text{goal}, \text{map}).\text{cost} & (14g) \\ h_{2c} = \text{Dubinspath}(N_c, \text{goal}).\text{cost} & (14h) \\ h_c = \max(h_{1c}, h_{2c}) & (14i) \\ f_c = g_c + h_c * (1 + \text{greed}) & (14j) \end{cases}$$

With the given desired roll angle  $\varphi$  and velocity  $v$ , physical attributes of node will be updated over time in the way described in Equations (14a) to (14e). This is essentially the kinematic recurrence relation of the spherical robot shown in Equation (1).

Calculation method of costs in Equations (14f) to (14j) is similar to the Hybrid A\* algorithm. Spent cost  $g$  is composed of its parent node's spent cost  $g_p$  and cost of current expansion: path length cost  $g_b$ , orientation cost  $g_d$  and forward/reversing switching cost  $g_r$ . Calculate heuristic cost of holonomic A\* path and non-holonomic Dubins path ignoring obstacles from current node to goal position, and take the maximum value them to obtain  $h$ . Total cost  $f$  is the sum of  $g$  and  $h$ . Parameter  $\text{greed}$  adjusts the weight of  $g$  and  $h$ , making the expansion more or less aggressive. So far, we have

**TABLE 5** Attributes of each node.

Symbols	Description
$x, y, \theta$	X-coordinate, y-coordinate and yaw angle
$\varphi, v$	Given desired roll angle and velocity
$t$	Time cost to expand from start node to current node
$N_p$	Parent node of current node
$g$	Spent cost from start node to current node
$h_1$	Heuristic cost of holonomic A* path cost from current node to goal position
$h_2$	Heuristic cost of non-holonomic dubins path ignoring obstacles From current node to goal position
$h$	Estimated heuristic cost from current node to goal position
$f$	Total cost of current node

completed the improvement of the node expansion rule to ensure that planned paths are kinematically feasible.

The preliminary planned path, however, may overlap with obstacle areas. Violation of collision-free constraints means there is a risk of collision. Therefore, we need to establish an efficient collision-free constraint judgement method. The triangle area judgement method [21] is a relatively intuitive method for collision detection, but for a spherical robot, it is difficult to accurately describe the spherical shell with multiple triangles—the number of triangles will be too large negatively affecting the algorithm's real-time performance.

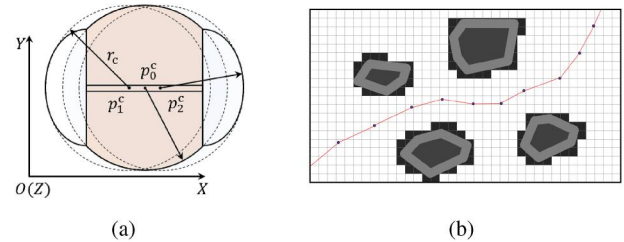
In order to improve computational efficiency, we shrink the robot into several collision constraint points  $P_i^c$  and inflate obstacles accordingly [22]. Set an appropriate radius  $r_c$  for each collision constraint point to generate a corresponding covering circle, ensuring that their union can completely cover the area occupied by the robot. Then, with  $r_c$  as the radius, expand obstacles so that collision avoidance can be realised by simply keeping those collision constraint points out of obstacle areas, as shown in Figure 6(b). This circle coverage strategy will cover part of the redundant area around the robot. Increasing the number of coverage circles can reduce redundant area while increasing computational cost. To cover a spherical robot, we usually use three covering circles, as shown in Figure 6(a).

## 4.2 | Instruction planning controller

An instruction planning controller is necessary to tell the spherical robot how to track the reference path given by the global path planning algorithm with minimum cost. We designed the controller based on the principle of MPC to find the optimal control strategy in our previous work [23], and we refine it in this paper. Cost function  $J(\cdot)$  to be minimised is defined in the following equations, and meaning of symbols is shown in Table 6.

$$\min_{\bar{U}(\cdot)} J(\cdot) = \sum_{i=1}^N \|\bar{X} - X_{ref}\|_Q + \sum_{i=1}^{N-1} \|\bar{U} - U_{ref}\|_R \quad (15)$$

$$\text{s.t. } \dot{\bar{X}}(i|k) = F(\bar{X}(i|k), \bar{U}(i|k)), \quad (15a)$$



**FIGURE 6** Collision-free constraint judgement method. (a) Three-circle-coverage-strategy for spherical robot. (b) Grid map representation of robot path (red line) and obstacles (grey black area) after obstacle expansion.

$$\overline{\mathbf{X}}(0|k) = \overline{\mathbf{X}}_k, \quad \overline{\mathbf{U}}(0|k) = \overline{\mathbf{U}}_k, \quad (15b)$$

$$\overline{\mathbf{X}}(i|k) \in [\overline{\mathbf{X}}_{\min}, \overline{\mathbf{X}}_{\max}], \quad (15c)$$

$$\overline{\mathbf{U}}(i|k) \in [\overline{\mathbf{U}}_{\min}, \overline{\mathbf{U}}_{\max}], \quad (15d)$$

$$G(\overline{\mathbf{X}}(i|k), \overline{\mathbf{U}}(i|k)) \leq 0 \quad (15e)$$

$$\|\overline{\mathbf{X}}(i|k) - \mathbf{X}_{obj}(j)\|^2 - (r + r_{obj} + \xi_s)^2 \geq 0 \quad (15f)$$

Equation (15a) is established based on kinematic model in Equation (1). Equation (15b) expresses the initial condition. Limitations for states and inputs are, respectively, presented in Equations (15c, d). Inequality constraint in Equation (15e) is employed to indicate the boundary value problem of direct multi-shooting method Equation (15f) represents the obstacle-free constraint of instruction planning controller—the distance between the centre of the robot and the centre of the obstacle envelope circle cannot be less than the sum of robot radius, obstacle envelope circle radius and an extension distance for safety. The above non-linear optimisation problem can be solved as a Sequential Quadratic Programming (SQP) problem. In practice, we use SQP problem solver CasADi [24] to effectively obtain the optimal controller instruction  $\overline{\mathbf{U}}(\cdot)$ . The instruction is put into subsequent motion control module so that spherical robot can move and work in a predictable manner.

## 5 | MOTION CONTROL

Precise and robust motion control is the basis for achieving advanced motion functions. However, a spherical robot is a non-linear, under-actuated and non-holonomic robot system because of the spherical shell and the heavy pendulum's drive mode. The Design of motion controllers for spherical robots is challenging, and requires in-depth research.

Main motor and auxiliary motor are respectively responsible for the forward/backward pull-up and left/right pull-up of heavy pendulum. Therefore, motion controller of spherical robot can be decomposed into forward/backward velocity controller and left/right direction controller. A fuzzy PID

**TABLE 6** Symbols used in cost function.

Symbols	Description
$N$	Prediction horizon
$Q, R$	Definite weighting coefficient matrix
$\overline{\mathbf{X}}, \overline{\mathbf{U}}$	States and inputs at moment $i+k$
$\mathbf{X}_{ref}, \mathbf{U}_{ref}$	Reference states and reference inputs at moment $i+k$
$\overline{\mathbf{X}}_k, \overline{\mathbf{U}}_k$	Initial states and inputs
$\mathbf{X}_{obj}$	States of $j$ th obstacle
$r_{obj}$	Obstacle envelope circle radius
$\xi_s$	Extension distance for safety

controller was tried in our earlier research [16], which was model-independent and exhibited high robustness to terrain changes but relatively low control accuracy. We also tried sliding mode controllers [23, 25, 26], which were model-based that improved control accuracy over fuzzy PID controllers, but free-chattering problem resulted in higher motor energy consumption and lower motor life. In the latest works, we started from the dynamic model of spherical robot, designed an offset-free linear MPC for velocity control in Ref. [27], and a robust servo LQR controller based on state compensation and velocity feedforward for direction control in Ref. [28], which enabled motion control with higher precision and lower energy consumption. In this paper, we refine algorithms based on Refs. [27, 28] to varying degrees.

### 5.1 | Dynamic model of spherical robot

Dynamic model is built up by Euler-Lagrange method [23, 27, 28]. The definitions of the symbols are shown in Tables 1 and 7, and Euler-Lagrange equations can be written as Equation (16).

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_k} \right) - \frac{\partial L}{\partial q_k} + \frac{\partial \Psi}{\partial \dot{q}_k} = \tau_k \quad (16)$$

where  $q_k \in \mathbf{q} = [\alpha \quad x_c \quad \beta \quad \varphi]^T$  is one of the states of the controlled system, that is, the input of controllers, and  $\tau_k \in \boldsymbol{\tau} = [\tau_p \quad \tau_r]^T$  is the input torque of the controlled system, that is, the output of controllers. As can be seen, the system's input is simply the output torques  $\tau_p$  and  $\tau_r$  of the

**TABLE 7** Symbols used in the dynamic model.

Symbols	Description
$L, \Psi$	Lagrangian function and Rayleigh's dissipation function
$\alpha, \beta$	Pitch and roll angle of pendulum
$\tau_p, \tau_r$	Output torque of main motor and auxiliary motor
$F_f$	Friction between spherical shell and ground
$x_c$	Distance that centre of robot moves along $x$ -axis, $x_c = \varphi r$
$M(q)$	Inertia matrix
$N(\mathbf{q}, \dot{\mathbf{q}})$	Nonlinear matrix
$E(q)$	Input transformation matrix
$m_p, m_f, m_s$	Mass of pendulum, frame and spherical shell
$m$	Total mass of the spherical robot, $m = m_p + m_f + m_s$
$I_{pp}, I_{pr}$	Moment of inertia of pendulum in pitch and roll direction
$I_{fp}, I_{fr}$	Moment of inertia of frame in pitch and roll direction
$I_{sp}, I_{sr}$	Moment of inertia of spherical shell in pitch and roll direction
$l$	Distance between centre of robot and centre of pendulum
$\zeta$	Viscous damping coefficient

main and auxiliary motors, but there are four state variables in  $q$ , indicating that the spherical robot is an underactuated system. Transforming Equation (16) into matrix form, we obtain the following:

$$\begin{cases} \mathbf{M}(q)\ddot{q} + \mathbf{N}(q, \dot{q}) = \mathbf{E}(q)\tau \\ \mathbf{M}(q) = \begin{bmatrix} \mathbf{M}_p & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_r \end{bmatrix}, \mathbf{N}(q, \dot{q}) = \begin{bmatrix} \mathbf{N}_p \\ \mathbf{N}_r \end{bmatrix} \\ \mathbf{M}_p = \begin{bmatrix} m_p l^2 + I_{pp} + I_{fp} & m_p l \cos \alpha \\ m_p r l \cos \alpha & m r^2 + I_{sr} + I_{fr} \end{bmatrix} \\ \mathbf{M}_r = \begin{bmatrix} m_p l^2 + I_{pr} & m_p r l \cos \beta \\ m_p r l \cos \beta & m r^2 + I_{sr} + I_{fr} \end{bmatrix} \\ \mathbf{N}_p = \begin{bmatrix} m_p g l \sin \alpha \cos \beta + \zeta(\dot{\alpha} + \dot{x}_c \cos \alpha / r) \\ -m_p r l \dot{\alpha}^2 \sin \alpha + \zeta(\dot{\alpha} \cos \alpha + \dot{x}_c / r) + F_f r \end{bmatrix} \\ \mathbf{N}_r = \begin{bmatrix} m_p g l \sin \beta \\ -m_p r l \dot{\beta}^2 \sin \beta \end{bmatrix}, \mathbf{E}(q) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}^T \\ q = \begin{bmatrix} q_p \\ q_r \end{bmatrix}, q_p = \begin{bmatrix} \alpha \\ x_c \end{bmatrix}, q_r = \begin{bmatrix} \beta \\ \varphi \end{bmatrix}, \tau = \begin{bmatrix} \tau_p \\ \tau_r \end{bmatrix} \end{cases} \quad (17)$$

The above dynamic model can be decomposed into two sub-models, which are used to design the velocity controller and direction controller, as shown in the Equations (18) and (19).

$$\mathbf{M}_p \cdot \ddot{q}_p + \mathbf{N}_p = [1 \quad 1]^T \cdot \tau_p \quad (18)$$

$$\mathbf{M}_r \cdot \ddot{q}_r + \mathbf{N}_r = [1 \quad 1]^T \cdot \tau_r \quad (19)$$

It can be seen that the dynamic model of the spherical robot shows strong non-linearity, which is much more complex than the kinematic model used in Equation (15a). If we design the controller directly based on the original model as in Section 4.2, the huge amount of calculation will cause problems with real-time control (the control frequency of the spherical robot is 50 Hz). Fortunately, the roll angle of the heavy pendulum is always less than  $15^\circ$  (0.262 rad). Therefore, in the following velocity and direction controller design, the non-linear dynamic model will be linearised near the origin first, followed by subsequent design processes.

## 5.2 | Velocity controller

Select state vector  $\mathbf{x}_p = [\alpha \quad \dot{\alpha} \quad x_c \quad \dot{x}_c]^T$ , system input  $u_p = \tau_p$ , and system output vector  $y_p$  is the same as the above state vector  $x_p$ . By linearising the model shown in Equation (17) near the origin, the continuous state-space model can be established:

$$\begin{cases} \dot{\mathbf{x}}_p = \mathbf{A}_c \mathbf{x}_p + \mathbf{B}_c u_p + \mathbf{V}_c \\ \mathbf{y}_p = \mathbf{x}_p \end{cases} \quad (20)$$

in which

$$\mathbf{A}_c = \frac{1}{|\mathbf{M}_p|} \begin{bmatrix} 0 & -m_p g l \cos \beta M_{p4} & 0 & m_p g l \cos \beta M_{p2} \\ 1 & \zeta(M_{p3} - M_{p4}) & 0 & \zeta(M_{p2} - M_{p1}) \\ 0 & 0 & 0 & 1 \\ 0 & \zeta(M_{p3} - M_{p4})/r & 1 & \zeta(M_{p2} - M_{p1})/r \end{bmatrix}^T, \\ \mathbf{B}_c = \frac{1}{|\mathbf{M}_p|} \begin{bmatrix} 0 \\ M_{p4} - M_{p3} \\ 0 \\ M_{p1} - M_{p2} \end{bmatrix}, \quad \mathbf{V}_c = \frac{1}{|\mathbf{M}_p|} \begin{bmatrix} 0 \\ M_{p3} F_f r \\ 0 \\ -M_{p1} F_f r \end{bmatrix}, \quad \mathbf{M}_p = \begin{bmatrix} M_{p1} & M_{p3} \\ M_{p2} & M_{p4} \end{bmatrix}$$

Then, the state equation is discretised based on the sampling time  $T_s$ :

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d u_k + \mathbf{V}_d \\ \mathbf{y}_k = \mathbf{x}_k \end{cases} \quad (21)$$

where  $\mathbf{A}_d = \mathbf{I} + \mathbf{A}_c T_s$ ,  $\mathbf{B}_d = \mathbf{B}_c T_s$ ,  $\mathbf{V}_d = \mathbf{V}_c T_s$ .

Modelling error, linearisation error and discretisation error of the dynamic model will cause mismatch between model and actual robot, and the size of the resulting deviation is related to target velocity and terrain conditions—we define it as disturbance  $w$  whose effect on state and output are, respectively,  $B_w w$  and  $C_w w$ . Considering  $w$  as an additional system state, augment the system state  $\mathbf{x}_k$  to  $\tilde{\mathbf{x}}_k = [\mathbf{x}_k \quad \mathbf{w}_k]$ , an augmented state space model can be obtained as Equation (22):

$$\begin{cases} \tilde{\mathbf{x}}_{k+1} = \mathbf{A}_a \tilde{\mathbf{x}}_k + \mathbf{B}_a \tilde{u}_k + \mathbf{V}_a \\ \mathbf{y}_k = \mathbf{C}_a \tilde{\mathbf{x}}_k \end{cases} \quad (22)$$

where

$$\mathbf{A}_a = \begin{bmatrix} \mathbf{A}_d & \mathbf{B}_w \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \quad \mathbf{B}_a = \begin{bmatrix} \mathbf{B}_d \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{V}_a = \begin{bmatrix} \mathbf{V}_d \\ \mathbf{0} \end{bmatrix}, \\ \mathbf{C}_a = \begin{bmatrix} \mathbf{I} \\ \mathbf{C}_w \end{bmatrix}^T$$

By introducing a Kalman filter as Equation (3) with the output  $y_k$  measured at time  $k$ , the augmented system state  $\tilde{\mathbf{x}}_k$  can be optimally estimated, which includes the disturbance  $w_k$ .

Then, we can start to construct the optimisation of cost function. In order to achieve the comprehensive optimal control of velocity, attitude, current and other states, the cost function is designed as shown in Equation (23), including three parts: output error, output process error and input error, where the output includes not only velocity but also pitch angle, pitch angular velocity, etc.

$$\min_{u, \mathbf{x}, \mathbf{y}} \bar{\mathbf{y}}_{N_p}^T \mathbf{P}_p \bar{\mathbf{y}}_{N_p} + \sum_{i=0}^{N_p-1} \bar{\mathbf{y}}_i^T \mathbf{Q}_p \bar{\mathbf{y}}_i + \sum_{i=0}^{N_u-1} \bar{\mathbf{u}}_i^T \mathbf{R}_p \bar{\mathbf{u}}_i \quad (23)$$

$$\text{s.t. } \tilde{\mathbf{x}}_0 = \tilde{\mathbf{x}}^*, \quad (23a)$$

$$\tilde{\mathbf{x}}_{k+1} = \mathbf{A}_a \tilde{\mathbf{x}}_k + \mathbf{B}_a \tilde{u}_k + \mathbf{V}_a, \quad (23b)$$

$$\mathbf{y}_k = \mathbf{C}_a \tilde{\mathbf{x}}_k, \quad (23c)$$

$$u_{\min} \leq u \leq u_{\max}, \quad (23d)$$

where  $\bar{y}_i = y_i - \bar{y}_{ref}$ ,  $\bar{u}_i = u_i - \bar{u}_{ref}$ , they are the differences between the current output/input and the reference output/input.  $P_p$ ,  $Q_p$  and  $R_p$  are weight matrices,  $N_p$ ,  $N_u$  represent the prediction horizon and control horizon,  $N_p \leq N_u$ ,  $u_k = u_{N_u-1}$  when  $k \geq N_u$ , and  $\bar{x}^*$  means the optimal estimation of current augmented states from Kalman filter.

Assuming that the estimate of disturbance  $w^*$  in Equation (23) is constant, the following operations in Equation (24) can be performed, in order to remove the disturbance from cost function (23) and transform it into a finite-horizon optimal control problem, which is only related to state  $x$  and control variable  $u$ .

$$\begin{cases} \mathbf{x}_{ref} = \mathbf{y}_{ref} - \mathbf{C}_w \mathbf{w}^* \\ \mathbf{V}_p = \mathbf{V}_d - \mathbf{B}_w \mathbf{w}^* \end{cases} \quad (24)$$

Furthermore, since a linear state-space model is used in the dynamic model, the system state at time  $k+1$  can be obtained by continuous recursion from initial state  $x^*$  and control sequence  $u$ . Therefore, the optimal control problem about  $x$  and  $u$  can be transformed into quadratic programming (QP) problem only with control sequence  $u$ , as shown in Equation (25):

$$\min_u \frac{1}{2} \mathbf{u}^T \mathbf{H}_p \mathbf{u} + \mathbf{u}^T \mathbf{f}, u_{\min} \leq u \leq u_{\max} \quad (25)$$

where  $\mathbf{H}_p = 2(\bar{\mathbf{B}}^T \bar{\mathbf{Q}} \bar{\mathbf{B}} + \bar{\mathbf{R}})$  is the Hessian matrix,  $\mathbf{f} = 2\left[\bar{\mathbf{B}}^T \bar{\mathbf{Q}}(\bar{\mathbf{A}}\mathbf{x}^* + \bar{\mathbf{V}} - \bar{\mathbf{x}}_{ref}) - \bar{\mathbf{R}}\bar{\mathbf{u}}_{ref}\right]$  is the Jacobi matrix, and

$$\bar{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q}_p & 0 & \cdots & 0 \\ 0 & \mathbf{Q}_p & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{P}_p \end{bmatrix}, \quad \bar{\mathbf{R}} = \begin{bmatrix} \mathbf{R}_p & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{R}_p \end{bmatrix},$$

$$\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_d \\ \vdots \\ \mathbf{A}_d^{N_p} \end{bmatrix}, \quad \bar{\mathbf{B}} = \begin{bmatrix} \mathbf{B}_d & \cdots & 0 \\ \mathbf{A}_d \mathbf{B}_d & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \mathbf{A}_d^{N_p-1} \mathbf{B}_d & \cdots & \mathbf{B}_d \end{bmatrix}$$

Some popular QP problem solvers such as qpOASES can be used to effectively solve the problem, now, we finally obtain the optimal control sequence  $u$  for velocity control.

### 5.3 | Direction controller

As for direction control, we select  $\mathbf{x}_r = [\beta \quad \dot{\beta} \quad \varphi \quad \dot{\varphi}]^T$  as state vector, with system input  $u_r = \tau_r$  and system output  $y_r = \varphi$ . We can linearise the model in Equation (17) using the same method as in Section 5.2. The system needs to be able to maintain a stable state in addition to ensuring the optimal response to the initial state in a limited time. In such cases where stability and optimality are both required, infinite-time regulators should be used to find the solution. With

constraints of the dynamic model, LQR formula minimises the following infinite quadratic cost function:

$$\min_u \frac{1}{2} \mathbf{e}^T \mathbf{F}_r \mathbf{e} + \frac{1}{2} \int_0^\infty (\mathbf{e}^T \mathbf{Q}_r \mathbf{e} + \mathbf{u}^T \mathbf{R}_r \mathbf{u}) \quad (26)$$

where  $\mathbf{e}$  is the difference between the reference output and actual output,  $\mathbf{F}_r$ ,  $\mathbf{Q}_r$  and  $\mathbf{R}_r$  are terminal weight matrix, error weight matrix and control weight matrix, respectively. It can be proven that the system is fully controllable, so based on solving the continuous time algebraic Riccati equation, the only optimal control law  $\mathbf{u}_L = -\mathbf{K}_r \mathbf{x}_r$  with the goal of zero roll angle can be obtained as follows:

$$\mathbf{u}_L = -\mathbf{K}_r \mathbf{x}_r = -k_{r1}\beta - k_{r2}\dot{\beta} - k_{r3}(\varphi_d - \varphi) - k_{r4}\dot{\varphi} \quad (27)$$

where  $k_{r1}$ ,  $k_{r2}$ ,  $k_{r3}$  and  $k_{r4}$  are members of  $\mathbf{K}_r$ , and  $\varphi_d$  is the desired roll angle of the robot.

The difficulty of eliminating steady-state error is the disadvantage of LQR controller. Therefore, Wise proposed robust servo LQR (RSLQR) in Ref. [29]. By defining the state deviation as a new state vector and introducing the integral link into forward loop of control law, RSLQR eliminates the steady-state error. Based on this, we updated the control law as follows, in which  $\mathbf{K}_r$  and  $\mathbf{K}_{ri}$  are also obtained by solving the continuous time algebraic Riccati equation:

$$\mathbf{u}_s = -\mathbf{K}_r \mathbf{x}_r - \mathbf{K}_{ri} \int_0^\infty \mathbf{e} \quad (28)$$

So far, we have obtained the offset-free LQR direction controller of spherical robot. Further, in order to optimise the dynamic process, we add more compensation terms to the control law. First, we define the high-order terms ignored when linearising dynamic model as a new state vector, and introduce a state compensation  $\delta$  as follows:

$$\delta = f(\mathbf{x}_r, \dot{\mathbf{x}}_r) = \frac{m_p r l \dot{\beta}^2 \sin \beta}{M_{r2} + M_{r4}} \quad (29)$$

where  $M_{r2}$  and  $M_{r4}$  are elements in the inverse of the linearised  $\mathbf{M}_r$  in Equation (17):

$$\mathbf{M}_{rL}^{-1} = \begin{bmatrix} M_{r1} & M_{r2} \\ M_{r2} & M_{r4} \end{bmatrix}, \quad (30)$$

Other definitions of the symbols in Equation (29) can be found in Table 7. In addition, for the convenience of calculation and solution, the motions of the main axis and auxiliary axis are fully decoupled when modelling. However, the direction control of spherical robot is affected by velocity. When velocity is higher, more centripetal force is required to achieve the same turning radius. Therefore, it is necessary to perform feed-forward compensation according to velocity and set angle. Experiments have shown that this compensation is a quadratic

function of velocity. For velocity feedforward  $\rho$ , set the value according to the following equation:

$$\rho = \text{sgn}(\varphi) * (\alpha_{r1}v^2 + \alpha_{r2}v), \quad (31)$$

where  $\text{sgn}$  is the sign function,  $\alpha_{r1}$  and  $\alpha_{r2}$  are feedforward coefficients. In summary, the final  $u$  is given by the following equation:

$$u = u_s + \delta + \rho, \quad (32)$$

Now, we finally obtain the control sequence  $u$  from the improved RSLQR controller we designed for direction control.

## 6 | EXPERIMENTAL RESULTS

### 6.1 | Experimental setup

In order to verify the above algorithms, we physically built spherical robots, including two types with diameters of 60 cm or 80 cm, as shown in Figure 1. Among them, the 60 cm one is equipped with a TI TMS320F28069 MCU, which is mainly used for the operation of motion control algorithms; a mini PC (Intel i7-8559U, 2.70 GHz, Quad-core 64-bit), for positioning, perception and planning algorithms running under robot operation system; in addition, it is equipped with sensor components such as motor encoders, IMU, wide-angle cameras and GNSS modules. It should be noted that we built a ground base station for GNSS module. In open areas, thanks to real-time kinematic technology, the GNSS receiver in a spherical robot can achieve a high-precision positioning effect at centimeter-level or even millimetre-level. The 80 cm one has a larger internal space and battery capacity, and the most important upgrade is in a mini PC (Intel i7-1165G7, 2.8–4.7 GHz, Quad-core 64-bit, RTX2060 GPU). Since the monitoring system requires GPU acceleration for real-time EIS, the test was conducted using an 80 cm spherical robot, while other algorithms were tested on the a 60 cm spherical robot.

### 6.2 | Fusion positioning system

The test of the fusion positioning system proposed in Section 3.1 was conducted in a flat and open field. We drew a 16.2 m\*8.4 m rectangular track on the ground, and operated the 60 cm spherical robot to patrol around the track clockwise from the origin by remote control. Positioning results of wheel odometry, RTK-GNSS and fusion positioning system are shown in Figure 7. The reference ground truth is also marked in Figure 7 - since the spherical robot will inevitably shake when moving, and cannot run strictly and uniformly according to the drawn orbit, the given ground truth can only be used for qualitative reference but not quantitative analysis.

It can be seen in Figure 7 that within a total distance of about 50 m, although the wheel odometry finally returned to the starting point, a certain rotation deviation occurred between it

and the reference ground truth due to the cumulative error of the yaw angle. After adding GNSS data and completing sensor fusion, the fusion positioning result eliminates the cumulative error, which is in good agreement with the reference ground truth. The high efficiency of the Kalman filter ensures that the fusion positioning system can provide high-frequency (100 Hz) positioning results without cumulative errors.

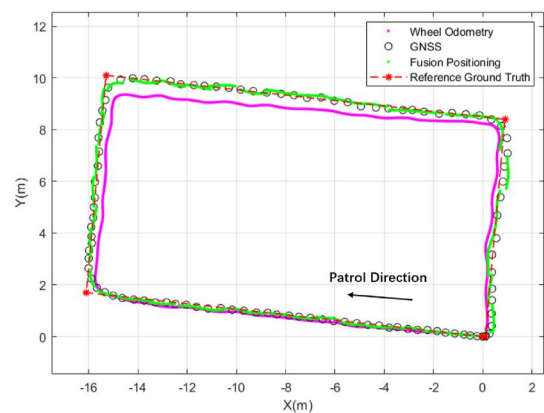
### 6.3 | Perception system

The perception system proposed in Section 3.2 was tested in the lobby of an office building. We placed a 60 cm spherical robot tilted to the right, and a tester as an “obstacle” was set at a distance of 10 m. The original pictures taken by stereo camera, disparity map obtained by SGBM algorithm and disparity map after threshold filtering are shown in Figure 8.

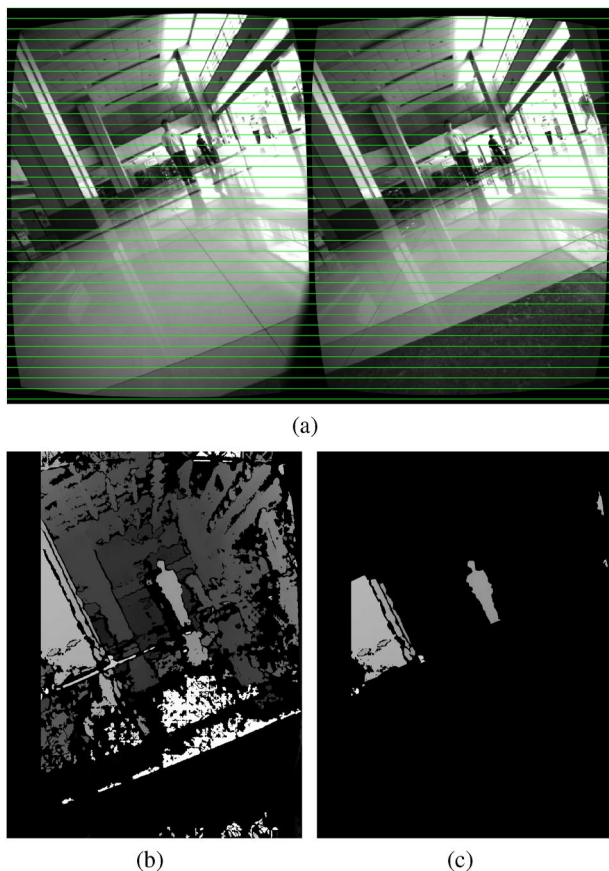
Relative error between the measured distance and ground truth of “obstacle” is about 2%. When frame resolution is 1624\*1240, the processing time for a single group of frames from stereo camera is 2 s. Since the accuracy requirement for obstacle detection is not high, we can sacrifice a certain detection accuracy to obtain higher frame rate by reducing image resolution. Experiments show that when resolution is reduced to 812\*620, the average single-frame processing time is shortened to 0.3 s; when it is further reduced to 541\*413, the average single-frame processing time is shortened to 0.1 s, which is perfectly acceptable for obstacle detection. With GPU acceleration, the contradiction between precision and processing time will be more effectively resolved.

### 6.4 | Monitoring system

Monitoring system proposed in Section 3.3 was tested in a bumpy construction site. Road condition of the construction site is terrible, bringing large shake for spherical robot. Images before and after EIS of left camera from the test video are shown in Figure 9.



**FIGURE 7** Positioning test result for spherical robot. GNSS, Global Navigation Satellite System.

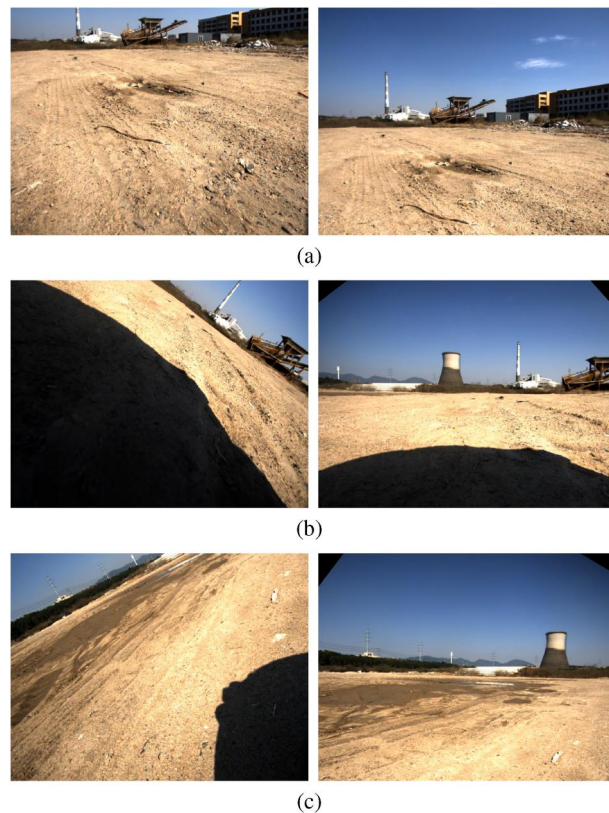


**FIGURE 8** Test result of visual obstacle perception. (a) Original pictures taken by stereo camera, green lines show the horizontal alignment of images. (b) Disparity map by Semi-Global Block Matching (SGBM) algorithm. (c) Disparity map after threshold filtering.

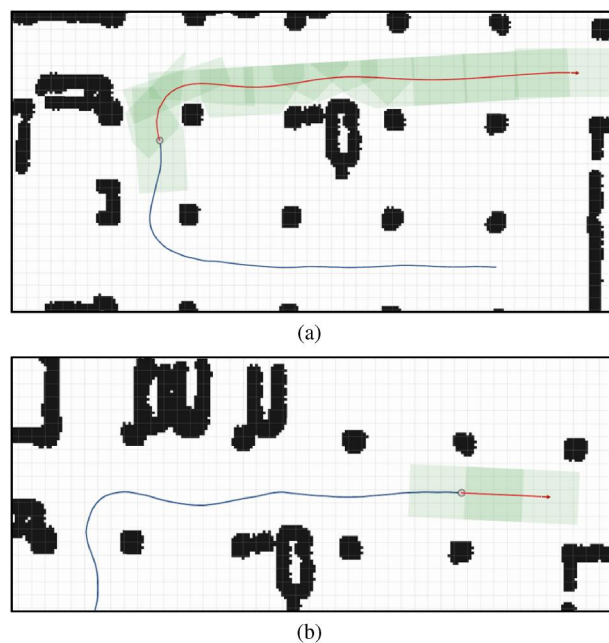
It can be seen that the stability of image was greatly improved after EIS. Relying on GPU acceleration, the single-frame processing time of EIS algorithm is about 10 ms, which fully meets the real-time requirements.

## 6.5 | Planning

Planning algorithms given in Section 4 were tested in an underground garage. We pre-established the prior grid map of the garage and inflated obstacles with the circle coverage strategy proposed in Section 4.1, as shown in Figure 10, where the black obstacle grids were marked as impassable areas, and the robot performed path planning and collision detection tasks in the remaining white passable areas. Since the garage was a GNSS-denied environment, spherical robot used the wheel odometry algorithm proposed in Section 3.1 to obtain its own positioning information. Pre-set target point scripts endowed spherical robot with the ability to patrol autonomously. Figure 10(a) and 10(b) show trajectories at the 45th and 75th seconds of the patrol mission, respectively. Spherical robot began in the lower right corner of Figure 10(a), with the blue curve representing the robot's actual trajectory, the red curve representing the subsequent reference trajectory, and the green



**FIGURE 9** Three typical frames before and after electronic image stabilisation (EIS). Original frames are on the left and corrected frames are on the right, and they are cropped to reduce the black data-free area. (a) 291st frame. (b) 325th frame. (c) 393rd frame.



**FIGURE 10** Test result of planning algorithms in underground garage patrol mission. (a) Trajectories at the 45th second. (b) Trajectories at the 75th second.

section representing the safe corridor established by the planning algorithms.

Figure 10 shows that the reference trajectory was smooth, retained a set distance from obstacles and spherical robot could follow it well. However, due to its inherent motion characteristics, spherical robot was prone to shaking after a significant turn as a result of the roll angle returning to zero, resulting in some overshoot and lateral oscillations in the actual trajectory.

## 6.6 | Velocity control

The velocity controller proposed in Section 5.2 is tested on various textured ground surfaces. We made the robot start straight from rest at a low desired speed (0.5 m/s) and a high desired speed (1.5 m/s) on tiled and rubber floors, and drew the velocity and pitch angle change curve of spherical robot under the control of linear model predictive control (LMPC) controller and PID controller, as shown in Figures 11 and 12.

It can be clearly seen from Figure 11 that for tiled floor and rubber floor, rise time of the system under LMPC controller is shorter, and the rapidity is obviously better than that of PID controller. Figure 12 shows that the pitch angle of spherical robot controlled by LMPC controller varies more steadily, indicating a smoother acceleration process.

## 6.7 | Direction control

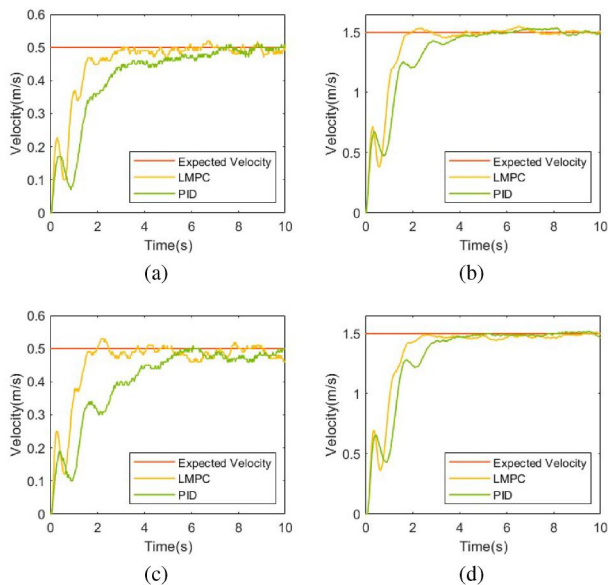
The direction controller proposed in Section 5.3 was tested on a tiled floor. When the robot was going straight at different velocities (including 0.5 m/s and 1 m/s), we gave turning instructions of varying sizes (including 0.12 and 0.19 rad), and set

the moment when the robot started turning as time 0. Comparing control effects of improved robust servo linear quadratic regulator (IRSLQR) controller and fuzzy PID controller proposed in Ref. [16], the curve of spherical robot's roll angle is drawn in Figure 13.

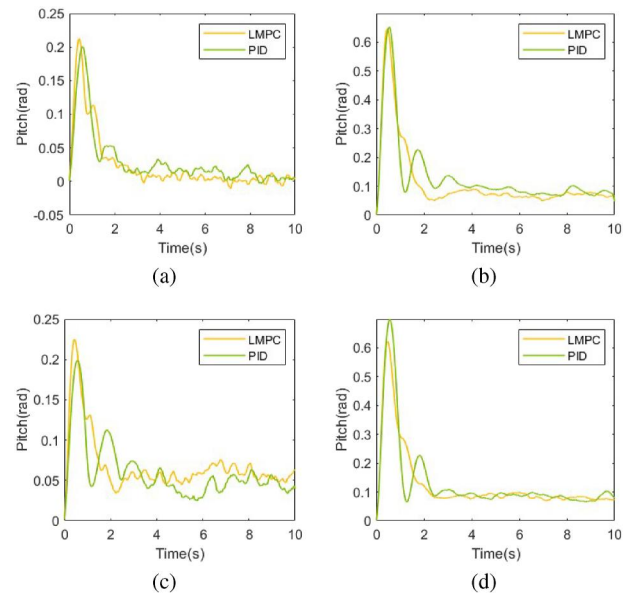
Figure 13 shows that at higher velocities, spherical robot has a more difficult time reaching larger turning angles. In general, when compared to the fuzzy PID-based direction controller, the IRSLQR-based direction controller has a shorter rise time, which allows the robot's roll angle to approach the desired turning angle faster.

## 7 | CONCLUSIONS

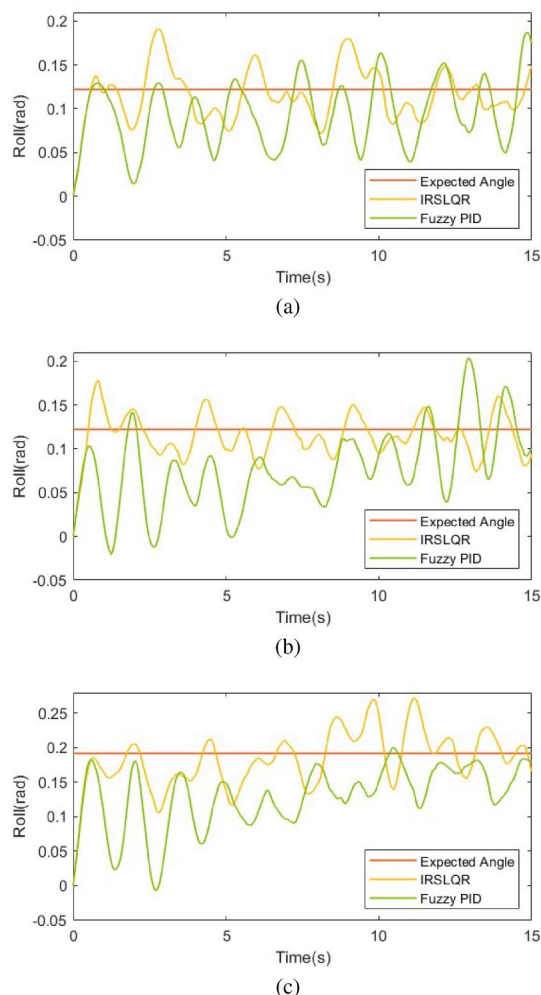
Aiming at exploration in harsh and unknown environments, this paper proposes a complete software and hardware framework of a novel spherical robot. In terms of hardware, the robot is driven by a heavy pendulum, with a fully enclosed spherical shell. These designs bring characteristics of strong protection, amphibious, anti-overturning and long-battery-life to the spherical robot. In terms of software, a comprehensive design has been created, including positioning and perception, planning, and motion control algorithms. As for positioning and perception, wheel odometry algorithm suitable for the spherical robot is designed based on the kinematic model, with GNSS data fused in Kalman filter to realise stable positioning results for spherical robots without cumulative errors. To detect obstacles, a stereo camera is used with SGBM algorithm, and the coordinates of those obstacles are transformed based on robot's present pose for planning module. In order to overcome the image shake introduced by spherical shell, we propose a set of real-time EIS algorithms for spherical robots based on IMU



**FIGURE 11** Curves of velocity change. (a) 0.5 m/s on tiled floor. (b) 1.5 m/s on tiled floor. (c) 0.5 m/s on rubber floor. (d) 1.5 m/s on rubber floor. LMPC, linear model predictive control; PID, proportional–integral–derivative.



**FIGURE 12** Curves of pitch change. (a) 0.5 m/s on tiled floor. (b) 1.5 m/s on tiled floor. (c) 0.5 m/s on rubber floor. (d) 1.5 m/s on rubber floor.



**FIGURE 13** Curves of roll angle change. (a) Velocity is 0.5 m/s and expected angle is 0.12 rad. (b) Velocity is 1.0 m/s and expected angle is 0.12 rad. (c) Velocity is 1.0 m/s and expected angle is 0.19 rad. IRSLQR, improved robust servo linear quadratic regulator.

data, sliding window filtering and perspective projection. As for planning, we improve the node expansion rules of Hybrid A\* algorithm based on spherical robot's kinematic model to ensure that the planned path is followable. We also design an instruction planning controller based on the kinematic model, which is used to tell the robot how to follow the reference path with minimum cost. As for motion control, we establish the dynamic model of spherical robot, then linearise it and design the velocity controller and direction controller. Among them, the velocity controller is designed based on linear MPC, and a Kalman filter is used to estimate system states and interference. Then, optimisation target function only related to system input is established to obtain optimal control instruction. The direction controller is designed based on RSLQR, and we introduce state compensation and velocity feedforward for better dynamic performance.

Future studies mainly focus on the following tasks. In the positioning and perception part, LiDAR will be added to form a fusion positioning and perception system covering indoors and outdoors together with cameras and other existing sensors. In

the planning part, more complex paths involving multiple forward/reversing switches will be investigated by applying more advanced local planning algorithms, and dynamic obstacles will be added. The greatest challenge in motion control is the contradiction between control accuracy and robustness. When terrain changes or random external disturbances occur, the robustness of the current controller will appear insufficient. In the future, we will try to apply robust control and random control methods or use learning strategies to adapt the spherical robot to more complex environments.

## ACKNOWLEDGEMENTS

This work was supported by the Fundamental Research Funds for the Central Universities (No. 226-2022-00086).

## CONFLICT OF INTEREST STATEMENT

The authors have no relevant financial or non-financial interests to disclose.


## DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

## ORCID

Wei Ren  <https://orcid.org/0009-0002-7300-5948>

Yixu Wang  <https://orcid.org/0000-0003-0665-0736>

Ziang Zhang  <https://orcid.org/0000-0002-4551-3489>

## REFERENCES

1. Bruzzone, L., Quaglia, G.: Review article: locomotion systems for ground mobile robots in unstructured environments. *Mech. Sci.* 3(2), 49–62 (2012). <https://doi.org/10.5194/ms-3-49-2012>
2. Yasin, J., et al.: Unmanned aerial vehicles (UAVs): collision avoidance systems and approaches. *IEEE Access* 8, 105139–105155 (2020). <https://doi.org/10.1109/access.2020.3000064>
3. Bicchi, A., et al.: Introducing the “Sphericle”: An Experimental Testbed for Research and Teaching in Nonholonomy. *International Conference on Robotics and Automation* (1997)
4. Ferrière, L., Raucourt, B.: ROLMOBS, a New Universal Wheel Concept, pp. 1877–1882. *International Conference on Robotics and Automation*, Leuven, Belgium (1998)
5. Zhan, Q., et al.: Mechanism design and motion analysis of a spherical mobile robot. *Chin. J. Mech. Eng.* 18(4), 542–545 (2005). <https://doi.org/10.3901/cjme.2005.04.542>
6. Liu, Z., Zhan, Q., Cai, Y.: Motion control of a spherical mobile robot for environment exploration. *Acta Aeronautica Astronautica Sinica* 29(6), 1673–1679 (2008)
7. DeJong, B., et al.: Design and analysis of a four-pendulum omnidirectional spherical robot. *J. Intell. Rob. Syst.* 86(1), 3–15 (2017). <https://doi.org/10.1007/s10846-016-0414-4>
8. Tomik, F., et al.: Design, fabrication and control of spherobot: a spherical mobile robot. *J. Intell. Rob. Syst.* 67(2), 117–131 (2012). <https://doi.org/10.1007/s10846-012-9652-2>
9. Muralidharan, V., Mahindrakar, A.: Geometric controllability and stabilization of spherical robot dynamics. *IEEE Trans. Automat. Control* 60(10), 2762–2767 (2015). <https://doi.org/10.1109/tac.2015.2404512>
10. Zhan, Q., Li, W.: Research progress and development trend of spherical mobile robots. *J. Mech. Eng.* 55(9), 1–17 (2019). <https://doi.org/10.3901/jme.2019.09.001>
11. Zhan, Q., Chi, X., Xi, X.: Linear motion control of an underactuated spherical mobile robot. *Appl. Mech. Mater.* 644, 351–355 (2014). <https://doi.org/10.4028/www.scientific.net/amm.644-650.351>

12. Song, Z., Wu, B., Zhou, T.: Trajectory Tracking Control of a Spherical Robot Based on Adaptive PID Algorithm. 2019 Chinese Control and Decision Conference, pp. 5171–5175 (2019)
13. Kayacan, E., et al.: Adaptive neuro-fuzzy control of a spherical rolling robot using sliding-mode-control-theory-based online learning algorithm. *IEEE Trans. Cybern.* 43(1), 170–179 (2012). <https://doi.org/10.1109/tsmcb.2012.2202900>
14. Ayati, M., Zarei, S.: Fault detection algorithm based on sliding-mode method for spherical rolling robots. In: *RSI International Conference on Robotics and Mechatronics (ICRoM)*, pp. 334–339 (2017)
15. Liu, D., Sun, H., Jia, Q.: A family of spherical mobile robot: driving ahead motion control by feedback linearization. In: *International Symposium on Systems and Control in Aerospace and Astronautics*, pp. 1–6 (2008)
16. Wang, Y., et al.: Fuzzy PID controller based on yaw angle prediction of a spherical robot. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3242–3247 (2021)
17. Hirschmuller, H.: Stereo processing by semiglobal matching and mutual information. *IEEE Trans. Pattern Anal. Mach. Intell.* 30(2), 328–341 (2007). <https://doi.org/10.1109/tpami.2007.1166>
18. Ren, W., et al.: Real-time Electronic Image Stabilization for Spherical Robot Based on Inertial Measurement Unit and Gray Projection Method, pp. 2122–2127. *China Automation Congress (CAC)* (2021)
19. Dolgov, D., et al.: Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Robot Res.* 29(5), 485–501 (2010). <https://doi.org/10.1177/0278364909359210>
20. Zhang, Z., et al.: Improved hybrid A\* path planning method for spherical mobile robot based on pendulum. *Int. J. Adv. Rob. Syst.* 18(1), 172988142199295 (2021). <https://doi.org/10.1177/1729881421992958>
21. Li, B., Shao, Z.: A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles. *Knowl. Base Syst.* 86, 11–20 (2015). <https://doi.org/10.1016/j.knosys.2015.04.016>
22. Campos-Macías, L., et al.: A hybrid method for online trajectory planning of mobile robots in cluttered environments. *IEEE Rob. Autom. Lett.* 2(2), 935–942 (2017). <https://doi.org/10.1109/lra.2017.2655145>
23. Liu, Y., et al.: Direction and trajectory tracking control for nonholonomic spherical robot by combining sliding mode controller and model prediction controller. *IEEE Rob. Autom. Lett.* 7(4), 11617–11624 (2022). <https://doi.org/10.1109/lra.2022.3203224>
24. Andersson, J., et al.: CasADi: a software framework for nonlinear optimization and optimal control. *Math. Program. Comput.* 11, 1–36 (2019). <https://doi.org/10.1007/s12532-018-0139-4>
25. Liu, Y., et al.: New Hierarchical Sliding Mode Control Method for Velocity Tracking of the Spherical Robot, pp. 1455–1460. *China Automation Congress (CAC)* (2021)
26. Liu, Y., et al.: Multi-terrain velocity control of the spherical robot by online obtaining the uncertainties in the dynamics. *IEEE Rob. Autom. Lett.* 7(2), 2732–2739 (2022). <https://doi.org/10.1109/lra.2022.3141210>
27. Hu, T., et al.: Optimal velocity control of spherical robots based on offset-free linear model predictive control. In: *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 363–368 (2022)
28. Wang, Y., et al.: Robust servo linear quadratic regulator controller based on state compensation and velocity feedforward of the spherical robot: theory and experimental verification. *Int. J. Adv. Rob. Syst.* 20(2), 17298806231153229 (2023)
29. Wise, K.: Bank-to-turn missile autopilot design using loop transfer recovery. *J. Guid. Control Dynam.* 13(1), 145–152 (1990). <https://doi.org/10.2514/3.20528>

**How to cite this article:** Ren, W., et al.: Spherical robot: a novel robot for exploration in harsh unknown environments. *IET Cyber-Syst. Robot.* e12099 (2023). <https://doi.org/10.1049/csy2.12099>