Improved hybrid A* path planning method for spherical mobile robot based on pendulum

Ziang Zhang®, Yixu Wan, You Wang, Xiaqing Guan, Wei Ren and Guang Li

Abstract
This article proposes a modification of hybrid A* method used for navigation of spherical mobile robots with the ability of limited partial lateral movement driven by pendulum. For pendulum-driven spherical robots with nonzero minimal turning radius, our modification helps to find a feasible and achievable path, which can be followed in line with the low time cost. Because of spherical shell shape, the robot is point contact with the ground, showing different kinematic model compared with common ground mobile robots such as differential robot and wheeled car-like robot. Therefore, this article analyzes the kinematic model of spherical robot and proposes a novel method to generate feasible and achievable paths conforming to kinematic constraints, which can be the initial value of future trajectory tracking control and further optimization. A concept of optimal robot’s minimum area for rotation is also proposed to improve search efficiency and ensure the ability of turning to any orientation by moving forward and backward in a finite number of times within limited areas.

Keywords
Spherical robot, navigation, nonholonomic robot, heuristic planning, path finding

Date received: 15 September 2020; accepted: 17 January 2021

Introduction
Spherical mobile robots based on pendulum are a particular kind driven by pendulum inside spherical shell, changing the center of gravity of the pendulum to produce eccentric moment for driving spherical shell movement. They are considered nonholonomic because of differential constraints that cannot be completely integrated, which means the robot’s configurations are not restricted to lower dimensional subspace of configuration space, especially referring to kinematic constraints of orientation change, and longitudinal and lateral motion. For a nonholonomic ground robot like wheeled robot, its configuration state can be defined and described by three degrees of freedom (3DOF): x, y referring to Cartesian coordinates and yaw angle θ referring to its orientation. Furthermore, there is another dimension for the state of spherical robot: roll angle φ referring to inclination of sphere shell for spherical robots, while wheeled robots usually stay parallel to the ground. Similar to wheeled car-like robot, spherical mobile robot cannot steer to turn around (changing their orientation θ) without longitudinal movement like moving

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
forward or backward with throttle (changing their coordinates \((x, y)\)). The difference is that wheeled car-like robots change orientation without lateral movement, whereas pendulum-driven spherical robot will do limited lateral excursion while pendulum sways, leading to inclination, which is a significant parameter for turning radius. In another word, though both nonholonomic, spherical robots show different mobile abilities and models when it comes to plane motion, both path planning method and path tracking need to be modified.

The development of spherical mobile robots has experienced a long time. Although extra tilt movement of spherical robots reduces motion stability, because of its advantages of safety and friendly interaction with environment for all electronic components that are enclosed in the spherical shell, it is easy to satisfy amphibious scenarios like factory patrol, security guard, fire protection, and so on. In the early stage, spherical robot is a spherical shell containing a universal steering wheel designed by Ferrière and Raucent. Halme et al. proposed a spherical robot with a single wheel resting on the bottom of a spherical shell. Bicchi et al. proposed a sphere containing a differential driven wheeled robot to indirectly drive spherical shell motion. Although designed in different ways, the main principle of spherical robot motion is to change the position of the center of gravity to produce eccentric moment. Joshi and Banavar proposed a design, where two DC motors associated with two flywheels are mounted inside the sphere. Xiao et al. proposed a spherical robot driven by pendulum with two inner motors and developed a series of spherical robots named BY-X for different uses, such as land and amphibious scenarios. Modeling analysis of spherical robot driven by pendulum with two motors has been decoupled by Kayacan et al. who discuss the relationship between tilt angle, lateral movement, and turning radius. Roozegar also proposed a model-free reinforcement learning algorithm based on extended classifier system to navigate a spherical robot, which is an end-to-end method of both plan and control for spherical robot.

The purpose of navigation for a robot is to find a physically achievable collision-free path to guide the robot to the target, avoiding all the obstacles at the same time, which guarantees that the generated path can be kept in line by the specific nonholonomic robot with a specific model, considering its kinematic constraints and dynamic constraints. A lot of research have been done on car-like robots. Fox et al. proposed dynamic window approach (DWA) algorithm, which samples in discrete control space. Rösmann et al. analyzed kinodynamic model of car-like robots and proposed time-band algorithm and motion planning control method to control robots. Kinematic model of more complicated car-like robot-like tractor with multiple trailers is analyzed by Xu and Li. By discussing the different kinematic models of different ground mobile robots like differential drive robot, wheeled robot, and spherical robot, the navigation methods need to be modified for adaption.

For a differential drive robot, if the left wheel and right wheel rotate at the same rate and in opposite directions, the robot would rotate its yaw direction with no translation. Balkcom and Mason proposed a method for differential drive vehicles that the optimal path can be described as a series of four basic action module: forward, backward, turning left and right, and only changing position when moving forward/backward. So, the path is indeed connected by several linear segments. For vehicles that have nonzero turning radius, Dubins proposed a well-known method as Dubins curves. The importance of the method is that with the constraints of curvature, Dubins proved that only considering moving forward, the minimal length of trajectory connecting initial and final poses can be described as a path specifically consisting of arcs and straight line segments, and the radius of all arcs can be defined as the same \( R_{\text{min}} \). Due to practical motion planning of ground robots, reversing also needs to be taken into consideration. So, Reeds and Shepp further proposed a more complex method connecting initial and final poses, considering moving forward and backward, making sure the whole generated path is collision-free.

Path planning techniques are generally classified into four groups: sampling, graph-search, interpolating, and numerical optimization. The widely used methods in robotics of sampling-based plan algorithms are probabilistic roadmap method, rapidly exploring random tree (RRT), and their modifications like RRT*, which initially cannot guarantee feasibility for nonholonomic robots, but constraints are taken into consideration by these improved methods, hence, RRT* methods have been widely used in motion plan of manipulators and trajectory finding in structured areas. Graph-search-based planning techniques are mainly based on grids, which can be obtained simultaneously from the different kinds of sensors of robots to estimate whether they are free or occupied and robot’s own position. To improve the search efficiency, Theta* algorithm was introduced by Daniel et al. to apply an any-angle search algorithm, which can connect distant grid cell with straight lines, making obtained path smoother and more efficient. Yap et al. proposed the Block A* method that utilizes a database of local distances between blocks, which contain more than eight nearest blocks, making it more efficient than A* or Theta*.

The Standford University team proposed a developed algorithm named Hybrid A* and used in its nonholonomic autonomous driving car for DAPRA Urban Challenge, estimating heuristic cost in two different ways and adding Reeds–Shepp shot path to reduce search time cost. The algorithm adds direction of motion as the fourth dimension for robot’s state space: two Cartesian coordinates \((x, y)\), orientation \(\theta\), and forward/reversing direction \(d \in \{-1, 1\}\), and expands nodes by taking orientation change, direction change, and reversing movement into its
cost calculation consideration. Nemec et al. proposed an improved hybrid A* method for nonholonomic wheeled robot\textsuperscript{28} with parameter greed to change the weight of heuristic cost in total cost to increase search efficiency by changing the strategy of node expansion method.

Although Hybrid A* method takes nonholonomic constraints into consideration, its implementation needs modification for spherical robots with different kinematic models. Since the method is based on graph search in grid maps and there is up to one explored node in one grid of selected path, the prune of potential surrounding nodes leads to the decrease of probabilistic approximate completeness, affected by the size of grid and length of node expansion, which is also relevant to search success rate because of time and storage cost of node expansion. When in narrow space, the orientation of initial/goal affects the search cost and efficiency. A concept of minimal area rotation is proposed to help improve search success rate and efficiency.

**Kinematic model of spherical robot**

**Spherical robot**

Spherical robot moves by changing the position of its pendulum inside the spherical shell. The top view and front view of spherical robot motion are shown in Figure 1. The radius of spherical shell is $r$ and the position $P$ of robot is vertical projection of spherical center, with orientation indicated by $\theta$. The pendulum is driven by two motors with orthogonal direction: main motor rotates the pendulum about the main axis for moving forward/backward and secondary motor for limited rotation $\beta$ about the orientation direction of robot. With the torques of two motors, the component force of gravity of pendulum will drive the spherical shell moving around the ground by changing velocity $V$, orientation $\theta$, and shell’s tilt angle $\varphi$. Figure 2 shows two kinds of spherical robots designed and installed by us. Spherical robot in Figure 2(a) has a totally enclosed spherical shape, separating the internal components from outer space. The localization of robot is based on data fusion of inertial measurement unit and motor encoders. Cameras can be attached inside spherical shell for image transmission. Transparent part of spherical shell decreases the quality of cameras and 2D lidars if they are placed inside sphere, and the tilt of sphere affects the localization results. Therefore, two 2DOF pan-tilt platforms with cameras and 2D lidars are designed and attached on both sides of spherical robot for better positioning accuracy and detection of static/dynamic obstacles, as shown in Figure 2(b). GPS module and LED lights can also be attached for navigation of wild or dark scenarios. The kinematic model of spherical robot is shown as follows.

![Figure 1. Top and front view of spherical robot, motion: turning left. (a) Top view and (b) front view.](image1)

![Figure 2. Framework and hardware of our designed spherical robots: (a) Physical view of our enclosed robot, (b) physical view of our experiment robot, and (c) attached sensors of our experiment robot.](image2)
As shown in equation (1), the derivative of position and orientation is determined by parameters of sphere's velocity $v$, orientation $\theta$, tilt angle $\phi$, and its derivative $\dot{\phi}$. Meanwhile, the initial input variables are rotate speeds/torques of both motors, which have specific relevance with speed $v$ and tilt angle $\phi$, shown in the related paper of spherical robot control mentioned earlier. For a specific spherical mobile robot, the specific mechanical structure limits the tilting motion of pendulum, which leads to an extreme value: $b_{\text{max}}$, and theoretically determines the extreme value of parameter $\beta_{\text{max}}$, which leads to the maximum lateral movement distance $r_{\phi_{\text{max}}}$ and a minimal turning radius of spherical robot.

$$
\begin{align*}
\dot{x} &= v \cos \theta - r \dot{\phi} \sin \theta \\
\dot{y} &= v \sin \theta + r \dot{\phi} \cos \theta \\
\dot{\theta} &= \frac{v \tan \phi}{r}
\end{align*}
$$

(1)

With the determination of minimal turning radius, the rule of node expansion of spherical robot is also determined. Figure 3 shows a scenario of left-hand rotation: the spherical robot starts from rest and then tilts $\phi$ to left and moves to rotate a circle, poses are recorded with arrows. In this scenario, the control of two motors is decoupled. The motion trajectory begins from $(0, 0)$ toward the positive direction of $X$ axis, and the secondary motor rotates to make the sphere tilt left to $(0, r_{\phi_{\text{max}}})$, then the main motor rotates to make the sphere counterclockwise rotate a circle back to $(0, r_{\phi_{\text{max}}})$. Finally, with the rotation of the secondary motor, the sphere tilts right back to the initial position $(0, 0)$.

Summary and comparison

Common ground mobile robots like wheeled car-like robot and differential robot show different kinematic models compared with spherical robot, as shown in Figure 4. Both of them are stable on the ground and have no lateral movement ability under normal conditions, while spherical robot is one point contact and easy to tilt caused by internal (torque of motor) or external factors (wind and uneven ground), leading to lateral movement. Differential robot’s driving wheels are driven by two separate motors with different speeds, allowing it to move and even rotate in place, which means it has a minimal zero turning radius. Wheeled car-like robot’s turning radius depends on two parameters: the steering angle $\phi$ and the distance $L$ between front and back wheels: $R = L / \tan \beta$. For specific wheeled robot with specific $L$, there is a max steering angle $\phi_{\text{max}}$, leading to the minimal turning radius $R_{\text{min}}$ for wheeled robot. From this perspective, differential drive robot model is a degradation model of wheeled robot, whose front wheels and back wheels overlap when $L = 0$, and wheeled robot model is a degradation model of spherical robot with no tilt ability or lateral mobility. Monocycle robots show similar kinematic constraints compared with spherical robots, assuming no relative slip between the single wheel and ground. The analysis and control of monocycle robot are more complicated because it is not only one point contact with ground-like spherical robot but also tends to fall because of the shape of tyre. When the center gravity deviation of monocycle exceeds the tilt of wheel tyre, monocycle robot cannot keep it stable spontaneously. To some extent, spherical robot is a degenerated monocycle robot with a lower center of gravity and all components are in a spherical wheel tyre, more stable, and no risk of overturning.

In conclusion, the kinematic model of spherical robot is more complicated than wheeled robot and differential robot because of lateral movement. And path planning for spherical robots is less complicated than monocycle robots with less constraints on control space and action space because
there is no risk of overturning. For spherical robots, there are two pivotal parameters, in which traditional wheeled robots do not have tilt angle $\phi$ and lateral motion distance $r\phi$. The value of lateral motion not only affects the position of robot but also determines the corresponding turning radius $R$ under that situation.

**Improved search algorithm**

As discussed in the previous section, the kinematic model of spherical robots differs from wheeled robots, in which traditional Hybrid A* method is proposed for. So, modification is needed for adaption in spherical mobile robots. Moreover, hybrid A* method sacrifices complete theoretical guarantees of search completeness for the sake of real-time requirement of search and the dispersion of the continuous state space. There are several scenarios that may cause the failure of hybrid A* search: (1) search time is too long to guarantee real-time performance; (2) memory overflow caused by too many expanded nodes; and (3) search process ends before reaching the final state because of the empty of node’s open list. The first two reasons are caused by the parameter sets and the hardware situation (scenario 1 is more often), which can be improved by adjusting parameters, such as increasing the distance of expansion nodes, modifying the weight of heuristic value, and so on. The third reason mostly appeared during path search in narrow space, when narrow space is near the goal pose in particular case, the scenario becomes similar to issues of parallel parking of wheeled car-like robots. And if the basic grid size of map becomes smaller and node expansion distance is set shorter, the whole time and storage cost will increase correspondingly. The unique shape of spherical robots not only helps to decrease calculating costs when it comes to collision detection but also shows great potential to change orientation and find a feasible path in narrow space, avoiding getting stuck in the corner. With sensors attached to a robot, the robot has the ability to get the obstacle information nearby and calculate the minimal area of free space. Our method puts forward a concept called minimal rotation area method, which can help calculate a feasible path to rotate the robot and change its orientation toward a given yaw angle within this limited area. The framework of the traditional hybrid A* algorithm is based on implementations written by Kurzer.29,30

**Node expansion method**

Traditional hybrid A* method uses a kinematic model of car-like robot to expand the nodes, so the paths produced by hybrid A* method are guaranteed to be drivable for car-like robot. For spherical robot with a different model, improvements and modification in rules of node expansion are proposed. According to kinematic model of spherical robot, we have the parameter $R_{\text{min}}$ referring to minimal turning radius of spherical robot, and relationship between its tilt angle, lateral movement, and corresponding turning radius. For the cost of calculation, the potential expansion of chosen node is discretized and pruned into six subnodes.

Figure 5 shows examples of node expansion, illustrating the difference between wheeled and spherical robot. For a wheeled robot, given the parent node $N_0$ and its six potential expansion ways, the subnode $N_1$ inherits from the left front expansion of parent node $N_0$ and can iteratively expand its subnodes. However, for parent node $N_i$ of spherical robot, the six potential expanded nodes inherit $N_i$’s tilt angle $N_i,\phi$, which determines the lateral movement between parent node and subnode. For node expansion of spherical robot, the subnode $N_{i+1}$ is the result of a left front expansion from its parent node $N_i$, which means the original tilt angle $\phi$ of $N_{i+1}$ is left-leaning. Specifically, the lateral movement only occurs when the parent and subnode share different tilt angle $\phi$. Since the node expansion process is iterative, its state can be represented by the structure Node:

```plaintext
Node:
structure Node
[x, y, $\theta$]—representing pose: position and orientation
[d]—representing the direction of movement: 1 means forward and $-1$ means reversing
[$\phi$]—representing tilt (roll) angle
parent_node—recording its predecessor node
$g$—spent cost moving here from start node
$h_1$—heuristic cost of holonomic A* path cost from current node to goal position
$h_2$—heuristic cost of nonholonomic Dubins path ignoring obstacles from current node to goal pose
$h$—estimated heuristic cost from current node to goal pose
f—total cost of current node
end
```

As shown in equations (3) to (5), there are different lateral movement scenarios: parameters $i,j \in [1,2,3]$ mean the turning condition of parent node and current node, respectively ($1,2,3$ indicate left, vertical, and right leaning, respectively). For example, $i = 1, j = 3$ indicate that the parent node is left-leaning with tilt angle $\phi$ and the current node is.
right-leaning with tilt angle $-\varphi$. Hence, the parameter $d_i[j]$ is a 2D vector containing lateral movement distance of nine different arrangements of tilt angle $\varphi$ of parent node and current node. The parameter $d\theta[j]$ is a vector of orientation change of each node expansion and $\varphi[j]$ records tilt angle of the current node, and the parameter $d$ is the direction of main motor motion: forward or reverse, and the change of motor direction is also considered into total cost

$$
d_i[j] = \begin{bmatrix}
0 & R\varphi & 2R\varphi \\
-R\varphi & 0 & R\varphi \\
-2R\varphi & -R\varphi & 0
\end{bmatrix}
$$

$$
d\theta[j] = \begin{bmatrix}
d\theta \\
0 \\
-d\theta
\end{bmatrix}
$$

$$
\varphi[j] = \begin{bmatrix}
\varphi \\
0 \\
-\varphi
\end{bmatrix}
$$

The whole node state update function is shown in the following equation (6), where parameters $dx$ and $dy$ are position change of arc decomposed to $X$ and $Y$ axes, the same method like position drift in traditional hybrid A* method, and lateral movement $d_i[j]$ and change of $\varphi$ are taken into consideration

$$
x = \text{parent_node}.x + d_i[j]\cos\theta + dx \\
y = \text{parent_node}.y + d_i[j]\sin\theta + dy \\
\theta = \text{parent_node}.\theta + d\theta[j] \\
d = \begin{cases}
1, \text{ forward} \\
-1, \text{ reversing}
\end{cases} \\
\varphi = \varphi[j]
$$

The calculation method of spent cost $g$, heuristic cost $h$, and total cost $f$ is similar to traditional hybrid A* method, as shown in equation (7). Spent cost $g$ records the total cost from start pose to current node, summing its parent node’s spent cost parent.$g$ and cost of current expansion: path length cost $g_l$, orientation change cost $g_o$, and cost of forward/reversing switch $g_s$ by comparing the parameter $d$ of parent node and subnode. Cost-to-go value $h$ represents the final estimated cost that will be taken from the current node to the final pose by recording the larger cost of holonomic path with obstacles $h_1$ and nonholonomic path ignoring obstacles $h_2$: cost-to-go $h_1$ represents the guiding heuristic A* path, which shows the direction of further node expansion. $h_1$ is the length of traditional A* path connecting current node with goal node, avoiding obstacles, which do not satisfy kinematic constraints like turning radius and moving orientation. As shown in Figure 6, the start pose, goal pose, and path are shown with red, green, and blue arrows, respectively, with obstacles in darker green cells. Each grid expanded by A* method is recorded with grayscale and the darkness illustrates heuristic distance value to goal, for example, the start cell is the lightest grid. Therefore, each expanded node can request potential heuristic map for the value of $h_1$, avoiding a lot of repeated calculations. For unexplored grid, an extra A* search method is needed and the value will be added to the potential map. Since the calculation of $h_1$ does not take kinematic constraints into consideration, nonholonomic cost $h_2$ records the cost of Dubins path connecting current node and goal pose, considering orientation switch and turning radius, ignoring obstacles or collision-detection. Because no physical map information is needed, the value of $h_2$ can be easy to get by searching in a Dubins path lookup table, which is established before node expansion, saving operation time. The total cost $f$ of node is the sum of spent cost $g$ and heuristic cost $h$, and nodes in open list $O$ will be sorted by value of $f$. The parameter $\text{greed}$ changes the weight of $g$ and $h$, making the expansion more/less aggressive

$$
g = \text{parent_node}.g + g_i + g_s + g_t \\
h_1 = A*\text{method}(\text{current_node}, \text{goal}, \text{map}).\text{cost} \\
h_2 = \text{Dubinspath}(\text{current_node}, \text{goal}).\text{cost} \\
h = \max(h_1, h_2) \\
f = g + h * (1 + \text{greed})
$$

In conclusion, compared with the traditional hybrid A* method of wheeled robots, expanded nodes of our improved method contain information about not only position, orientation, forward or backward motion direction but also tilt angle $\varphi$. Tilt angle illustrates lateral movement, which needs to be considered if parent node and subnode share different tilt angle, in that case, the orientation sub_node.\theta of expanded subnode is reserved, while translation of lateral movement will be added into position state (sub_node.x, sub_node.y). The process of node expansion is depicted in Algorithm 1: every time given a parent node $N_{\text{parent}}$, six subnodes will be initialized and calculated. If the subnode has been expanded and stored in closed list $C$, the node with lower cost $g$ will be kept and parent node of $N_{\text{parent}}$ will be updated correspondingly. If the subnode has
Algorithm 1. Node expansion method.

```
Input: parent node $N_{\text{parent}}$, open set $O$, closed set $C$, potential heuristic map

1 count ← 0;
2 while count < 6 do
3     initial node $N_{\text{current}}$;
4     update current node's state value: $x, y, \theta, d, \phi$;
5     update current node's cost value: $g, h, f$;
6     if $N_{\text{current}} \in C$ then
7         // retain the node with smaller cost value $g$
8         $N_{\text{current}}.g \leftarrow g_{\text{lower}}$;
9         update $N_{\text{current}}$-parent node;
10     else
11         if $N_{\text{current}} \in O$ then
12             update cost values $g$ and $h$;
13         else
14             $O$.push($N_{\text{current}}$);
15     end
16     count ← count + 1;
17 end
```

been on the open list $O$ and $N_{\text{parent}}$ has lower cost $g$, the cost values of $g$ and $h$ will both be updated. New expanded nodes will be pushed into priority queue $O$ and sorted according to the value of $f$.

Collision detection

For robots with rectangular chassis like car, there are several ways for collision detection, as shown in Figure 7: rigorous numerical calculation for area sum of these four triangles consists of each side of the car and obstacle point $P/Q$ compared with the area of robot. If the obstacle is not in contact with the car, the area sum will be greater than the area of the car itself or vice versa. In high speed or relatively less rigorous scenarios, to reduce calculation cost, the car can be covered with one or two circles by turning the problem into detecting the collision between obstacles and circles, which reduces the accuracy and searching completeness.

In comparison, the collision detection of spherical robot is relatively easier for calculation, because any part of spherical shell shares the same distance to the center of the ball (equal to the radius of the robot $R_{\text{body}}$) like circular sweeping robot to a certain extent. Therefore, the collision detection problem is simplified as calculating whether the distance between obstacles and center of robot is greater than the radius of spherical shell $R_{\text{body}}$. In practice, as shown in Figure 8, since the radius of sphere is $r$, the obstacles will inflate and the shape robot model is degenerated as a point $P$. Therefore, there is a hypothetical premise in the following discussion that the inflated layer has been added in the grid map, and robots are simplified as a point with a certain direction (like a virtual arrow), bringing convenience in calculation for planning.

Minimal area for rotation

Compared with the traditional hybrid A* algorithm, our proposed improved hybrid A* search algorithm is appropriate for spherical mobile robots by changing node expansion strategy preserving kinematic constraints of spherical robot, whereas hybrid A* method cannot promise search completeness. So, although the improved node expansion method ensures that the generated path is kinematically feasible and pruned search strategy will improve the search cost, hybrid A* search will still face some scenarios that the whole search method fails because the open list is empty before reaching the final goal pose, which usually appears in narrow space surrounded with obstacles that the expanded nodes of the path coincide with each other. Another kind of scenario is that the robot recovers from distraction and needs to renavigate the surrounding areas, where a minimal area for rotation is needed to fully know about the surrounding environment with different kinds of attached sensors. As mentioned above in the section of collision-detection, compared with car-like robot, spherical robot has advantages that the orientation of spherical robot

Figure 7. (a, b) Different collision detection methods of wheeled robot.

Figure 8. Collision detection method of spherical robot.
does not affect the collision-detection result if the center of robot is unchanged. Differential robot like sweeping robot can turn around in place with zero turning radius, and for spherical robot which has lateral movement while tilting and nonzero turning radius, in order to change orientation in a limited area, we proposed a concept of minimal rotation area for spherical robot according to the characteristics of its spherical shell.

As the hybrid A* search fails because of an empty open list, there is a certain expanded node \( N_l \) with the lowest heuristic cost \( h \) between the start and goal pose. Since there is a generated 2D A* path from \( N_l \) to goal position (heuristic holonomic path with obstacles for calculation of \( N_l, h_l \)), there might be an orientation difference between the orientations of A* path and final expanded node. If the robot can rotate itself with a certain degree to coincidence, the hybrid A* search continues iteratively till goal pose is finally reached, making the whole trajectory a series of feasible curves for spherical robot connected with several joints. However, the spherical robot has nonzero turning radius and lateral movement while tilting and turning around, which is why the minimal area for rotation is proposed.

The proposed minimal rotation area for spherical robot is an area that ensures the robot to freely change its orientation toward any degree by rotation while keeping its center within this limited area. The distance of surrounding obstacles can be detected by sensors attached in robots like radars, lidars, or cameras when in narrow space. So, the robot can get real information around itself and calculate a free circle area, like a bubble, making sure that the robot can move collision-free by keeping the center within this area. As shown in Figure 9, there are four actions of node expansion: combination of left/right leaning and forward/reversing. The initial pose is in \((0, 0, 0)\), and the radius of robot is set to 0.3 m, tilt angle is set to \(\pi/6\) and the step rad of each expansion is set to \(\pi/16\), which means there need to be at least 32 times of node expansions for the robot to rotate a circle. All the blue arrows are orientations of expanded nodes and all the tilt-changing point is in an envelope circle, illustrating the area where the spherical center moves. The minimal area is a circle with a radius of 0.16 m, equal to the envelope circle in this scenario, showing that the robot can turn to any angle on the premise that the spherical center does not leave this area. Because spherical robot has a special phenomenon that it has lateral movement when turning around, the lateral motion is proportional to the tilt angle and the turning radius is relevantly proportional to the tilt angle, which makes tilt angle a significant parameter to be further studied.

In general, if the lateral movement and expanded distance are shorter, the radius of minimal rotation area can be smaller, but to satisfy omnidirectional demand, the number of expanded nodes for a total rotation will increase and the cost time for changing the direction of both motors is to be considered. Meanwhile, the maximum of lateral movement \( r_\varphi \) is a constant for a specific sphere, so there can be an optimization for parameters of lateral movement/tilt angle and node expansion rad/distance. The aim is to make the rad of each expansion as large as possible to decrease the expanded numbers for a complete circle rotation. Once the free circle area is decided, the sphere can move forward or backward directly to make sure the left leaning can reach the edge of free space and move left forward a certain distance. As shown in Figure 10, a left forward motion is illustrated, and relevant parameter is shown in equation (8).

The equation for \( f(\varphi) \) is shown in equation (9), and this equation shares the same monotonicity with the function \( y(\varphi) = \theta \) when \( \varphi \in [0, \varphi_{\text{max}}] \).
\[ L = r \varphi < R_u \]
\[ R = \frac{r}{\tan \varphi} \]
\[ \frac{\theta}{2} = \frac{d}{R + L} \]
\[ d^2 = R_u^2 - L^2 \]
\[ \varphi_{\text{max}} = \frac{\pi}{6} \]
\[ f(\varphi) = \tan^2 \frac{\theta}{2} = \frac{(R_u^2 - r^2 \varphi^2) \tan^2 \varphi}{(1 + \varphi \tan \varphi)^2} \]

From Figure 11, we can find that the value and derivative of \( f(\varphi) \) keep both positive in domain \([0, \varphi_{\text{max}}]\), so \( f(\varphi) \) increases monotonically in domain \([0, \varphi_{\text{max}}]\), which means tilt angle should be set to \( \varphi_{\text{max}} \) in this scenario. And under circumstance of any detected radius of free area \( R_u \), an optimal tilt angle \( \varphi \) is calculated based on the derivative of equation along with the maximum rad \( \theta \) per expansion. In practical algorithm, the optimal strategy and policy of node expansion in the given minimal area for rotation are determined with max tilt angle \( \varphi_{\text{max}} \) and discrete interpolation between nodes. Algorithm 2 illustrates the process of minimal rotation method: given legal initial node \( N_i \) and target node \( N_g \) nodes will be expanded with calculated tilt angle \( \varphi \) and step expanded rad \( \theta \) within observed free space (minimal rotation area) until target node’s orientation is reached.

**Algorithm 2. Minimal rotation method.**

**Input:** map, initial node \( N_i \), target node \( N_g \)

1. if \( N_i \) or \( N_g \) is not legal then
2. \( \text{return false;} \)
3. end
4. calculate radius \( R \) of minimal rotation area;
5. calculate tilt angle \( \varphi \) and expanded step rad \( \theta \);
6. \( i \leftarrow 0; \)
7. initial \( N_i = N_i; \)
8. while \( N_i, \theta \neq N_g, \theta \) do
9. \( i \leftarrow i + 1; \)
10. initial and expand new node \( N_i \) with \( \varphi \) and \( \theta \);
11. \( N_i, \text{parent node} \leftarrow N_i-1; \)
12. end
13. \( \text{return true;} \)

**Algorithm 3. Pathfinding algorithm.**

**Input:** map, start pose, goal pose, potential heuristic map

1. initial \( O, C \in \emptyset \);
2. initial \( N_s, N_g \);
3. if \( N_s \) or \( N_g \) is not legal then
4. \( \text{return null;} \)
5. end
6. initial potential heuristic map \( P \);
7. \( O.push(N_s); \)
8. while \( O \neq \emptyset \) do
9. \( N_l \leftarrow O.popMin(); \)
10. \( C.push(N_l); \)
11. if \( N_l.pose = N_g.pose \) then
12. \( \text{return } N_l; \)
13. else
14. if \( \text{minimal_rotation_method}(N_l, N_g) \) then
15. \( \text{return } N_g; \)
16. end
17. else
18. if \( \text{Dubins_shot}(N_l, N_g) \) then
19. \( \text{return } N_l; \)
20. else
21. \( \text{expansion_method}(N_l); \)
22. end
23. end
24. end
25. \( \text{return null;} \)

**Pathfinding algorithm**

The basic model of our pathfinding algorithm is based on the traditional hybrid A* method, which can be summarized as maintaining a priority queue named Open list \( O \). The whole pathfinding algorithm is illustrated in Algorithm 3. Given
legal start and goal pose, the potential heuristic map \( P \) is initialized with an \( A^* \) search from start grid to goal grid, recording holonomic estimated cost of every expanded grids. \( N_s \) is pushed into open list \( O \), then the algorithm gets into while loop until \( O \) is empty or the goal is reached, as mentioned in the previous section: the node with minimal total cost \( f \) will pop from \( O \) and be pushed into close list \( C \), then its potential legal (not expanded and collision-free) subnodes will be initialized and pushed into open list \( O \) iteratively, sorted by their total cost \( f \) from small to large. If potential subnode is already in \( O \) or \( C \), its cost value \( g \) and \( h \) will be updated to retain more optimal node with lower cost value. Minimal rotation area method is applied when goal position is reached but with different orientation, \( N_l \) and \( N_g \) will be sent to Algorithm 2 and return expanded node if rotation method succeeds. Besides, there is a probability to generate a Dubins path connecting current expanded node with goal pose, and if the whole Dubins path is collision-free (which means we have already founded a path connecting start and goal pose) or the goal pose is already expanded, the hybrid \( A^* \) search method ends with success, and finally, the goal node will trace back its parent node iteratively back to start node, generating the feasible path connecting start and goal pose.

As mentioned above, the pathfinding method needs to ensure real-time requirement. Meanwhile, the potential heuristic map ensures that there is a holonomic \( A^* \) path for further node expansion. Since the improved hybrid \( A^* \) search needs to record every expanded node, the iteration times of node expansion and the number of expanded nodes determine the time/storage cost. When it comes to open space, the hybrid \( A^* \) search is relatively fast to expand nodes toward goal pose, whereas a lot of node expansion occurs in narrow space with reversing and forward/backward switch because these actions cause high cost \( g \) but low reduction of heuristic cost \( h \). Although both nonholonomic with nonzero turning radius, compared with wheeled robot, spherical robot has some better characteristics: spherical robot has relatively smaller turning radius compared with wheeled robot with similar size and spherical shell which do not affect collision detection result for robot’s poses with different orientation.

**Experimental results**

The proposed system has many adjustable parameters. Simulation and physical experiments are both completed. The parameters of robot are as follows: the minimal radius for expansion \( R_{\text{min}} = 2 \text{ m} \) and the radius of spherical shell is \( 0.3 \text{ m} \). The size of grid cell is \( 1 \times 1 \text{ m}^2 \); yaw angle of spherical robot is discretized into 72 directions so the resolution is \( 5 (360/72) \). The algorithm is wrapped as a plug-in global planner algorithm in ROS(robot operation system) and was evaluated on a PC with Intel i5 2.90 GHz. The local plan algorithm is DWA\(^{31} \), which is also modified for trajectory tracking of spherical robot. All experiments are conducted in known environments with static obstacles. Since the purpose of our proposed path-finding algorithm is to find a feasible path from initial to goal pose, path regeneration and dynamic obstacles are not taken into consideration. The attached sensors like cameras and lidars are used for localization.

**Simulation experiments**

Simulation experiments are conducted in three different scenarios. As shown in Figure 12 of scenario 1, there are three paths generated by our improved hybrid \( A^* \) plan algorithm, feasible for our spherical robot, with the same start pose and same goal position but different goal orientation. All these three generated paths are feasible for spherical robot, whereas the smoothness of paths is different because of different goal orientation. Dubins path search succeeds in path(1), which helps save the search time and iteration times. The search time and iteration times for each path are shown in Figure 13, it is obvious that path (1) cost lowest time and iteration times, reaching 10 Hz pub frequency. Although Dubins shot fails in path (2) and (3),

![Figure 12. Scenario 1: three generated paths (1), (2), and (3) with the same start node and different goal orientation: (a) Path (1), (b) path (2), and (c) path (3).](image-url)
reversing paths are generated and time cost raises to about 800 ms, and it still meets the real-time requirement of global planning in low-speed scenarios.

The three paths of scenario 2 in Figure 14 illustrate a typical example of different generated paths with/without applying minimal rotation area method. Path (4) illustrates a simple path with forward goal orientation, whereas path (5) and path (6) illustrate the path with the same goal position but reverse orientation compared to path (4). As shown, without applying minimal rotation area in path (5), although goal position is reached during node expansion, the orientation difference in path (5) makes search algorithm continue and finally turn around in outer free space, generating a long path. For path (6), when expanded node reached the goal position with different orientation, minimal rotation area method is applied, a fast search is performed, and a guide path with several continuous zigzags of left/right leaning and forward/backward switch is generated. As shown in Figure 15, the time cost and final iteration times of path (6) have been significantly reduced compared with path (5), showing similar performance with path (4), proving the superiority of searching speed and efficiency.

Experiments results in maze scenario with larger scale and complicated obstacles distribution are shown in Figure 16. Table 1 records four metrics of simulation experiments for measuring performance. The value of path length is the sum of the distances of all adjacent nodes in the generated path, referring to a total distance of the generated path. The total degree of orientation change records the amount of rotational movement from the initial to goal node, measuring the smoothness of path. Time for path calculation is also recorded in parameter: cost time, which is crucial as the optimal algorithm must construct paths efficiently. Parameter direction switch shows the times of forwarding/reversing of the path. Since the velocity will change the sign in each switching point, the parameter illustrates the smoothness of motion speed.

In scenario 1, the distance and orientation difference of initial and goal node are 9.55 m and 90°. In scenario 2, the distance and orientation difference of initial and goal node are 6 m and 180°. Generated paths of path (1) to (6) have larger path length and orientation change, because of limitation of both obstacles and kinematic constraints. It is obvious that the distance and orientation change of path (6) is significantly lower than (5), because...
of minimal rotation method. However, there are five times of direction switch in path(6). In free space, frequent direction switch motion will limit the speed of robot, increasing total operation time.

**Physical experiments**

The physical experiments are conducted on flat ground surface environment. The general speed of the whole

---

**Table 1.** Results of simulation experiments.

| Scenarios | Path Distance (m) | Orientation Change (°) | Cost ΔT (ms) | Direction Switch |
|-----------|-------------------|-------------------------|--------------|-----------------|
| Path(1)   | 26.26             | 213.27                  | 123          | 0               |
| Path(2)   | 30.40             | 242.44                  | 737          | 4               |
| Path(3)   | 27.57             | 297.88                  | 813          | 1               |
| Path(4)   | 5.94              | 0                       | 47           | 0               |
| Path(5)   | 37.79             | 454.48                  | 206          | 0               |
| Path(6)   | 15.97             | 344.78                  | 101          | 5               |
| Path(7)   | 19.49             | 309.30                  | 473          | 2               |
| Path(8)   | 19.27             | 148.74                  | 158          | 0               |
| Path(9)   | 24.39             | 374.64                  | 197          | 0               |
motion is set 0.6 m/s, and the scenario is similar to scenario of path (1). Figure 17 shows one of paths of our spherical robot moving in physical experiments. The motion scenario is similar to Figure 12 (a), moving from the start pose in open space to goal pose in narrow space. Our spherical robot reached goal pose successfully, following the green global path generated by our proposed path planning algorithm. Red arrows in Figure 17 (a) are recorded poses of spherical robot with position and orientation. Black curves in Figure 17 (b) are local path generated by our improved DWA local planner algorithm. In low-speed forward motion condition with repeated trajectory generation and control calculation, the generated path of our proposed algorithm is followed well in line with our improved control method.

Another motion scenario is depicted in Figure 18, similar to scenario in Figure 13(b). Physical trajectory of Figure 18(a) consists of a series of red points showing location for better display and comparison between generated path and physical trajectory. The reason why the physical trajectory does not follow the green global guide path in narrow space is that local plan algorithm takes the distance of obstacles into consideration, shown in black curves and purple layer of Figure 18 (b). Because of repulsive force from obstacles, despite no risk of collision, the robot avoids being too close to obstacles of both sides. More complicated path containing direction switch is not evaluated in physical experiments because of the current limited local plan algorithm, which needs to be further studied.

Table 2 records parameters of both planned paths and actual trajectories in physical experiments mentioned above. Because of inevitable tilt during the whole motion, the path length and orientation change of actual trajectory are greater than the planned path. The cost time of path generation meets the real-time requirement, and the average velocity of our robot is slightly lower than the set speed (0.6 m/s). Recorded velocity data from motor encoders show that the actual speed will decrease while turning a corner. According to experimental data and comparison of planned paths and actual trajectories, the given paths generated by our proposed algorithm can be well tracked and can be the baseline of our future study.

### Table 2. Results of physical experiments.

| Scenarios  | Path length (m) | Orientation change (°) | Cost time | Average velocity (m/s) |
|-----------|----------------|------------------------|-----------|------------------------|
| Experiment 1| Planned path  | 13.74                  | 159.35    | 103 ms                 |
|            | Actual trajectory | 14.32                  | 183.20    | 24.2 s                 | 0.59 |
| Experiment 2| Planned path  | 28.86                  | 421.47    | 238 ms                 |
|            | Actual trajectory | 30.22                  | 503.22    | 55.9 s                 | 0.54 |

Conclusion
An improved hybrid A* search algorithm is proposed for path plan of spherical robot. Firstly, we analyze the kinematic model of spherical robot and modify node expansion method by taking lateral movement and tilt angle switch into consideration. Secondly, inspired by differential robot with zero turning radius, minimal rotation area method is proposed for spherical robot to be used in scenarios that goal position is expanded but with different orientation, helping reach expected orientation with less expanded nodes and lower time cost. The proposed path-finding algorithm takes not only the length of trajectory into consideration but also reversing condition, tilt angle changing, and forward/backward switch. Finally, simulation and physical experiments are conducted for evaluation of the algorithm. The search frequency of our proposed algorithm is guaranteed. Meanwhile, the generated path can iteratively be updated and published for path tracking of physical spherical robot.

Future studies mainly focus on three tasks: (1) Minimal rotation area method can also be applied in other scenarios to generate feasible path-like tight bend and small areas to adjust orientation. (2) Velocity optimization can be further taken into consideration along with path optimization, smoothing curves in open space, and adding speed limit to the entire path, considering curvature and change of motion direction. (3) Dynamic obstacles are not considered in our experiments. Meanwhile, physical experiments of more complicated paths containing multiple forward/reversing switch will be conducted in our future work list by applying more advanced local plan algorithm and control method in our spherical robot.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed the receipt of following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Key Research and Development Program of China [No. SQ2019YFB130016] and the Fundamental Research Funds for the Central Universities.

ORCID iD
Ziang Zhang 
https://orcid.org/0000-0002-4551-3489

References
1. LaValle S. Planning algorithms. Cambridge: Cambridge University Press, 2006.
2. Kayacan E, Bayraktaroglu ZY, and Saey W. Modeling and control of a spherical rolling robot: a decoupled dynamics approach. *Robotica* 2011; 30: 671–680.

3. Ferri`ere L and Raucent B. ROLLMOBS: a new universal wheel concept. In: *IEEE international conference on robotics & automation*, Leuven, Belgium, 20–20 May 1998, Vol. 3, pp. 1877–1882.

4. Halme A, Schonberg T, and Wang Y. Motion control of a spherical mobile robot. In: *Proceedings of 4th IEEE international workshop on advanced motion control*, Mie, Japan, 18–21 March 1996, Vol. 1, pp. 259–264. IEEE.

5. Bicchi A, Balluchi A, Prattichizzo D, et al. Introducing the “SPHERICLE”: an experimental testbed for research and teaching in nonholonomy. In: *Proceedings of international conference on robotics & automation*, Albuquerque, NM, USA, 25–25 April 1997, Vol. 3, pp. 2620–2625. IEEE.

6. Joshi VA and Banavar RN. Motion analysis of a spherical mobile robot. *Robotica* 2009; 27: 333–353.

7. Xiao AP, Sun HX, Lao QZ, et al. The design and analysis of a spherical mobile robot. *Dev Innov Mach Electr Product* 2004; 17(1): 14–17.

8. Roozegar M, Mahjoob MJ, Esfandyari MJ, et al. XCS-based reinforcement learning algorithm for motion planning of a spherical mobile robot. *Appl Intell* 2016; 45: 736–746.

9. Fox D, Burgard W, and Thrun S. Controlling synchro-drive robots with the dynamic window approach to collision avoidance. In: *Proceedings of IEEE/RSJ international conference on intelligent robots and systems*, Vol. 3, IROS ’96, Osaka, Japan, 8 November 1996, pp. 1280–1287. IEEE.

10. Rösmann C, Hoffmann F, and Bertram T. Kinodynamic trajectory optimization and control for car-like robots. In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Vancouver, BC, Canada, 24–28 September 2017, pp. 5681–5686. IEEE.

11. Rösmann C, Krämer M, Makarov A, et al. Exploiting sparse structures in nonlinear model predictive control with hypergraphs. In: *2018 IEEE/ASME international conference on advanced intelligent mechatronics (AIM)*, Auckland, New Zealand, 9–12 July 2018, pp. 1332–1337. IEEE.

12. Xu Y and Liu R. Path planning for mobile articulated robots based on the improved A* algorithm. *Int J Adv Robot Syst* 2017; 14(4): 1–10.

13. Li B, Zhang Y, Acarma T, et al. Trajectory planning for a tractor with multiple trailers in extremely narrow environments: a unified approach*. In: *2019 international conference on robotics and automation (ICRA)*, Montreal, QC, Canada, 20–24 May 2019, pp. 8557–8562. IEEE.

14. Chitsaz H, LaValle SM, Balkcom DJ, et al. Minimum wheel-rotation paths for differential-drive mobile robots. In: *Proceedings of 2006 IEEE international conference on robotics and automation*, Roma, Italy, 10–14 April 2007, pp. 2718–2723. IEEE.

15. Balkcom DJ and Mason MT. Time optimal trajectories for bounded velocity differential drive vehicles. *Int J Robot Res* 2002; 21: 199–217.

16. Dubins LE. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *Am J Math* 1957; 79: 497–516.

17. Reeds JA and Shepp LA. Optimal paths for a car that goes both forwards and backwards. *Pacific J Math* 1990; 145: 367–393.

18. González D, Pérez J, Miliánés V, et al. A review of motion planning techniques for automated vehicles. *IEEE Trans Intell Transp Syst* 2016; 17(4): 1135–1145.

19. Kavraki LE, Svestka P, Latombe J, et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans Robot Autom* 1996; 12: 566–580.

20. LaValle SM. *Rapidly-exploring random trees: a new tool for path planning*. 1998. Citeseer.

21. Akine M, Bekris KE, Chen BY, et al. Probabilistic roadmaps of trees for parallel computation of multiple query roadmaps. In: Dario P and Chatila R (eds) *Robotics research the eleventh international symposium*. Berlin, Heidelberg: Springer, 2005, pp. 80–89.

22. LaValle SM and Kuffner JJ Jr. Randomized kinodynamic planning. *Int J Robot Res* 2001; 20: 378–400.

23. Correa DSO, Sciotti DF, Prado MG, et al. Mobile robots navigation in indoor environments using kinect sensor. In: *2012 2nd brazilian conference on critical embedded systems*, Campinas, Brazil, 20–25 May 2012, pp. 36–41. Campinas, Brazil: IEEE.

24. Gomez C, Hernandez AC, Crespo J, et al. A topological navigation system for indoor environments based on perception events. *Int J Adv Robot Syst* 2016; 14: 1729881416678134. DOI: 10.1177/1729881416678134.

25. Daniel K, Nash A, Koenig S, et al. Theta*: any-angle path planning on grids. *J Artif Intell Res* 2010; 39: 533–579.

26. Yap P, Burch N, Holte RC, et al. Block A*: database-driven search with applications in any-angle path-planning. In: *AAAI conference on artificial intelligence*, San Francisco, California, USA, 7–11 August 2011. IEEE.

27. Dolgov D, Thrun S, Montemerlo M, et al. Path planning for autonomous vehicles in unknown semi-structured environments. *Int J Robot Res* 2010; 29: 485–501.

28. Nemec D, Gregor M, Bubenková E, et al. Improving the hybrid A* method for a non-holonomic wheeled robot. *Int J Adv Robot Syst* 2019; 16(1): 1729881419826857.

29. Kurzer K. Hybrid A*: path planner for the KTH research concept vehicle, 2015 https:github.com/karlkurzer/path_plan ner (accessed 10 August 2020). Github.

30. Kurzer K. Path planning in unstructured environments: a real-time Hybrid A* implementation for fast and deterministic path generation for the KTH research concept vehicle. School of Industrial Engineering and Management (ITM), 2016, p. 63. http://www.diva-portal.org/smash/record.jsf?pid=diva2%3 A1057261&dswid=-1770 (accessed 27 May 2018).

31. Fox D, Burgard W, and Thrun S. The dynamic window approach to collision avoidance. *IEEE Robot Autom Mag* 1997; 4(1): 23–33.