Path Planning for Inchworm-like Robot Moving in Narrow Space

Jiang Yong, Wang Hongguang, and Fang Lijin

Abstract—Considering the kinematic constraints and the path discreteness, a novel path planning method based on the feasible gait is proposed for an inchworm-like robot moving in a narrow space in this paper. Firstly, by analyzing the locomotion modes and the basic gaits of the robot, some arithmetic operators to describe the change of the position and orientation are defined, and a gait graph model is also presented. Secondly, making reference to the main idea of the partial dynamic window method, the path planning method is proposed, which include the environment modeling based on the dynamic partial grid map and the feasible gait searching algorithm based on the complete or incomplete traversal. Finally, some simulations and experiments are provided to validate the rationality and the validity of the path planning method.

I. INTRODUCTION

Some distinctive features of an inchworm-like robot are as follows: the simple mechanical structure, the small number of drivers, and the cyclical bionic movement [1]–[4]. These features make the robot easily achieve micro in size and move freely in a narrow space. The narrow space mentioned here means a two-dimensional space in which has an intensive obstacle distribution. Moreover, the robot body size is very close to the distance between the obstacles. In such space, the path planning methods for the inchworm-like robot have two evident characteristics, compared with the wheeled mobile robot. One is that the methods based on the configuration space (C-space), such as the potential field methods [5]–[7], are not efficient. Different from the wheeled mobile robot, which can be treated as a particle in C-space to build simple models, the kinematic models of the inchworm-like robot are very complex, because of its coupled freedoms in the mechanical structure and frequent switching between two standing feet. The other is that the results of path planning for the inchworm-like robot are a series of discrete footprints, unlike the continuous and smooth trajectories for the wheeled mobile robot. For this reason, the impacts of the obstacle distribution on the gait cycle of the inchworm-like robot must be considered. In addition, it is necessary for the planning methods to avoid the problem of ultra-high dimensions led by the intensive obstacle distribution in the narrow space.

In [8], after an introduction of a finite state machine (FSM) model, a fuzzy planning method is proposed which is based on a fuzzy multi-sensor data fusion scheme for an inchworm-like robot called CRAWLER. However, when this robot moving in a narrow space, the fuzzy planning method not only easily causes larger errors, but also has poor performance in real time, because of the large number of fuzzy planning rules to those complex gaits. In [9], a planning algorithm based on greedy strategy for the task-level skills that allow an inchworm-like climbing robot to walk vertically and horizontally, and to make transitions between surfaces is discussed. In [10], a path planning method which introduces a hybrid configuration space for the CRAWLER robot is developed. Generally the above two methods are applicable to the inchworm-like robot moving in the space with sparse obstacle distribution. But for the narrow space defined previously, the computations of those methods are very cumbersome and complicated, and can not guarantee that the gait process of the robot will always meet the restrictive conditions in the obstacle space.

Considering the kinematic constraints and the path discreteness of the inchworm-like robot, this paper makes reference to the main idea of the dynamic window approach described in [11]–[13], and proposes a novel path planning method which includes the environment modeling based on the dynamic partial grid map and the feasible gait searching algorithm based on the complete or incomplete traversal. Section II describes the mechanical structure and the locomotion modes of the inchworm-like robot. Section III analyzes the basic gaits of the robot. The novel path planning method is proposed in Section IV. Significant experimental results are shown in Section V. Finally, Section VI outlines the main conclusions in this work.

II. INCHWORM-LIKE ROBOT

The mechanical structure of the inchworm-like robot is designed as a bipedal robot with an under-actuated mechanism [14], which includes four revolute pairs and one prismatic pair. As shown in Fig. 1, two motors (motors 1 and 3) independently drive joints 1 and 5, respectively, thereby adjusting the tilt angle of the suction feet 1 and 2 so that the robot can grip the surface firmly. Motor 2 drive joint 3 to represents the prismatic motion of the legs that allows the robot to expand and contract its legs. Both joints 2 and 4 are passive revolute joints, which provide steering capability of the feet relative to the legs by coupled motion.
with joint 3.

The inchworm-like robot has three locomotion modes denoted by DL-Mode, LU-Mode and SU-Mode, respectively.

1) DL-Mode: Both joints 2 and 4 are locked and prevented from rotating. Thereby the torque output of motor 2 causes translation motion of legs 1 and 2. In this way the robot body can easily extend or contract.

2) LU-Mode: When the robot body extends beyond a certain range, joint 4 is unlocked while joint 2 is still locked. If motor 2 keep driving joint 3 to extend, joint 4 will generate a coupled motion with joint 3. In this way leg 2 can rotate clockwise relative to foot 2.

3) SU-Mode: When the robot body contracts beyond a certain range, joint 2 is unlocked while joint 4 is still locked. If motor 2 keep driving joint 3 to contract, joint 2 will generate a coupled motion with joint 3. In this way leg 1 can rotate counter-clockwise relative to foot 1.

In the process of extending, the robot can switch its locomotion modes from SU-Mode to DL-Mode or from DL-Mode to LU-Mode. Similarly, in the process of contracting, the robot can switch its locomotion modes from LU-Mode to SU-Mode or from DL-Mode to SU-Mode. However, it is impossible to switch directly between SU-Mode and LU-Mode.

III. GAIT ANALYSIS

A. Gait Cycle and Gait Operator

The inchworm-like robot has six kinds of basic gaits, which include moving forward in DL-Mode, moving backward in DL-Mode, turning left in SU-Mode, turning right in SU-Mode, turning left in LU-Mode and turning right in LU-Mode. As shown in Fig. 2, the gait cycle of moving forward in DL-Mode consists of six steps.

-- Step 1: Make foot 1 grip the surface, and rotate joint 1 to lift up foot 2 by \( \theta \) degree angle. (Fig. 2. a)

-- Step 2: Slide joint 3 to extend the robot body by \( d \) length. (Fig. 2. b)

-- Step 3: Rotate joint 1 in opposing direction to lower foot 2 by \( \theta \) degree angle. (Fig. 2. c)

-- Step 4: Make foot 2 grip the surface, and rotate joint 5 to lift up foot 1 by \( \theta \) degree angle. (Fig. 2. d)

-- Step 5: Slide joint 3 in opposing direction to contract the robot body by \( d \) length. (Fig. 2. e)

-- Step 6: Rotate joint 5 in opposing direction to lower foot 1 by \( \theta \) degree angle. (Fig. 2. f)

As a symmetrical arrangement of the mechanical structure, four kinds of turning gaits have the same process basically. For example, the gait cycle of turning left in SU-Mode consists of eight steps, as shown in Fig. 3.

-- Step 1: Make foot 1 grip the surface, and rotate joint 1 to lift up foot 2 by \( \theta \) degree angle. (Fig. 3. a)

-- Step 2: Slide joint 3 to contract the robot body until the locomotion mode is switched from DL-Mode to SU-Mode. (Fig. 3. b)

-- Step 3: Continue contracting the robot body to make joint 2 generate coupled motion with joint 3. In this case, the robot turns left by \( \alpha \) degree angle. (Fig. 3. c)

-- Step 4: Rotate joint 1 in opposing direction to lower foot 2 by \( \theta \) degree angle. (Fig. 3. d)

-- Step 5: Make foot 2 grip the surface, and rotate joint 5 to lift up foot 1 by \( \theta \) degree angle. (Fig. 3. e)

-- Step 6: Slide joint 3 to extend the robot body, and make joint 2 generate coupled motion with joint 3. In this case, foot 1 rotates counter-clockwise by \( \alpha \) degree angle. (Fig. 3. f)

-- Step 7: Continue extending the robot body until the locomotion mode is switched from SU-Mode to DL-Mode. (Fig. 3. g)

-- Step 8: Rotate joint 5 in opposing direction to lower foot 1 by \( \theta \) degree angle. (Fig. 3. h)

A gait operator denoted by \( c_{i+1} = T(c_i) \) describes a function relationship from an initial position and orientation \( c_i \) to a goal position and orientation \( c_{i+1} \) in the No. \( i+1 \) gait cycle of the robot. The position and orientation \( c_i \) is defined as:

\[
\begin{bmatrix}
 x_{x_{i1}} & x_{x_{i2}} & x_{x_{i3}} \\
 y_{x_{i1}} & y_{x_{i2}} & y_{x_{i3}} \\
 z_{x_{i1}} & z_{x_{i2}} & z_{x_{i3}}
\end{bmatrix}
\]

where \([x_{x_{i1}}, y_{x_{i1}}, z_{x_{i1}}]^T\) and \([x_{y_{i2}}, y_{y_{i2}}, z_{y_{i2}}]^T\) denote the coordinate vectors of the center support points on feet 1 and 2, respectively, \([x_{z_{i3}}, y_{z_{i3}}, z_{z_{i3}}]^T\) denotes the coordinate vector of the centroid point for the robot body.

Based on the kinematic analysis, six kinds of gait operators (corresponding to the basic gait cycles) denoted by \( T_{Df1}, T_{Df2}, T_{Sf1}, T_{Sr1}, T_{Lr1}, T_{Lr1} \), respectively, are defined as:

\[
c_{i+1} = T_{Df1}(c_i) = \begin{bmatrix}
 x_{x_{i1}} + d & x_{x_{i2}} + d & x_{x_{i3}} + d \\
 y_{x_{i1}} & y_{x_{i2}} & y_{x_{i3}} \\
 z_{x_{i1}} & z_{x_{i2}} & z_{x_{i3}}
\end{bmatrix}
\]

(2)
\[ c_{i+1} = T_D(c_i) = \begin{bmatrix} x_{i+1} - d \\ y_{i+1} \\ z_{i+1} \end{bmatrix} \]
\[ c_{i+1} = T_S(c_i) = \begin{bmatrix} x_{i+1} - P_i \\ y_{i+1} \\ z_{i+1} \end{bmatrix} + P_i \]
\[ c_{i+1} = T_E(c_i) = \begin{bmatrix} x_{i+1} \\ y_{i+1} \\ z_{i+1} \end{bmatrix} + P_{izi} \]
\[ c_{i+1} = T_T(c_i) = \begin{bmatrix} x_{i+1} + P_{izi} \\ y_{i+1} \\ z_{i+1} \end{bmatrix} + P_{izi} \]
\[ c_{i+1} = T_L(c_i) = \begin{bmatrix} x_{i+1} + P_{izi} \\ y_{i+1} \\ z_{i+1} \end{bmatrix} + P_{izi} \]
\[ c_{i+1} = T_G(c_i) = \begin{bmatrix} x_{i+1} + P_{izi} \\ y_{i+1} \\ z_{i+1} \end{bmatrix} + P_{izi} \]

Where
\[ P_i = (l_i - l_i) \cos \alpha \]
\[ P_i = (l_i - l_i) \sin \alpha \]
\[ P_i = (l_i - l_i) \cos \alpha - (l_i - l_i) \]
\[ P_i = l_i \sin \alpha \]
\[ P_i = l_i \sin \alpha \]

\[ B. \ FSM \ and \ Gait \ Graph \]

A lot of information, such as the switching between locomotion modes, the changing in sensor data and so on, are closely related to the gait cycles of the inchworm-like robot. In order to facilitate the gait analysis, a FSM model and a gait graph are introduced here.

The elements and the value rules of the state vector in the FSM model are as follows:

1) The state vector is composed of ten elements, i.e., \( LM, J_1, J_2, J_3, J_4, J_5, F_1, F_2, E, \) and \( GS. \) Each element is a binary value, as shown in Table I.

2) The meaning and the value rules to each element are defined as:

- \( LM \) means locomotion mode, and its value can be assigned with 00 (denotes DL-Mode), 01 (denotes LU-Mode) and 10 (denotes SU-Mode).
- \( J_1 \) means the state of joint 1, and its value can be assigned with 00 (denotes remain inactive), 01 (denotes rotate to lift up foot 2) and 10 (denotes rotate to lower foot 2).
- \( J_2 \) means the state of joint 2, and its value can be assigned with 00 (denotes remain inactive), 01 (denotes rotate clockwise) and 10 (denotes rotate counter-clockwise).
- \( J_3 \) means the state of joint 3, and its value can be assigned with 00 (denotes remain inactive), 01 (denotes slide to extend the robot body) and 10 (denotes slide to contract the robot body).
- \( J_4 \) means the state of joint 4, and its value can be assigned with 00 (denotes remain inactive), 01 (denotes rotate clockwise) and 10 (denotes rotate counter-clockwise).
- \( J_5 \) means the state of joint 5, and its value can be assigned with 00 (denotes remain inactive), 01 (denotes rotate to lift up foot 1) and 10 (denotes rotate to lower foot 1).
- \( F_1 \) means the state of foot 1, and its value can be assigned with 0 (denotes non-standing foot) and 1 (denotes standing foot).
- \( F_2 \) means the state of foot 2, and its value can be assigned with 0 (denotes non-standing foot) and 1 (denotes standing foot).
- \( E \) is an idle bit temporarily, and its value is assigned with 0.
- \( GS \) means the process of the basic gait cycle, and its value can be assigned with 0 (denotes the first half of the cycle) and 1 (denotes the second half of the cycle).

\[ \text{TABLE I} \]

<table>
<thead>
<tr>
<th>State Vector</th>
<th>( LM )</th>
<th>( J_1 )</th>
<th>( J_2 )</th>
<th>( J_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
</tbody>
</table>

A gait graph \( G \) of the inchworm-like robot is a simple digraph defined by:

\[ G = \{ V(G), E(G), \phi(G) \} \]

where \( V(G) = \{ v_1, v_2, \ldots, v_n \} \) denotes a node set of the gait graph \( G \) in which each node \( v_i \) (\( i = 1, 2, \ldots, n \)) corresponds to a state vector (see Table II). \( E(G) = \{ e_1, e_2, \ldots, e_{42} \} \) denotes an edge set of the gait graph \( G \), and each edge \( e_i \) (\( i = 1, 2, \ldots, 42 \)) represents a directional conversion between two correlative nodes \( v_j \) and \( v_k \), denoted as \( \phi(e_i) = \{ v_j, v_k \} \).

\[ \phi(G) : E \rightarrow V \times V \] is called a correlative function.

\[ \text{TABLE II} \]

| CORRESPONDING RELATIONSHIPS OF NODES AND STATE VECTORS |
|---------------------------------|-----------------|-----------------|-----------------|-----------------|
| Node   | State Vector | Node   | State Vector | Node   | State Vector |
| v_1    | 0x0008       | v_12   | 0x8608       | v_21   | 0x4188       |
| v_2    | 0x1008       | v_13   | 0xA008       | v_22   | 0x6008       |
| v_3    | 0x0108       | v_14   | 0x800C       | v_23   | 0x400C       |
| v_4    | 0x2008       | v_15   | 0x8004       | v_24   | 0x4004       |
| v_5    | 0x000C       | v_16   | 0x8014       | v_25   | 0x4014       |
| v_6    | 0x0004       | v_17   | 0x8904       | v_26   | 0x4024       |
| v_7    | 0x014       | v_18   | 0x8604       | v_27   | 0x4014       |
| v_8    | 0x0204       | v_19   | 0x8024       | v_28   | 0x4008       |
| v_9    | 0x0024       | v_20   | 0x8008       | v_29   | 0x4008       |
| v_{10} | 0x0104       | v_21   | 0x9008       | v_30   | 0x5008       |
| v_{11} | 0x0208       | v_22   | 0x8908       | v_31   | 0x4248       |

IV. PATH PLANNING METHOD

Considering the kinematic constraints and the path dispersion of the inchworm-like robot, and making
reference to the dynamic window approach in [11]–[13], a novel path planning method, which includes the environment modeling based on dynamic partial grid map and the feasible gait searching algorithm based on complete or incomplete traversal, is proposed.

A. Environment Modeling

Definition 1: An obstacle space denoted by \( C_{\text{obs}} \) is a set of all obstacles distributed randomly in the workspace of the inchworm-like robot. A free space denoted by \( C_{\text{fre}} \) is a set of the space that meets the kinematic constraints in each degree of freedom of the inchworm-like robot.

A \( C_{\text{fre}} \) projection of the robot is shown in Fig. 4. The fan-shaped regions \( ABC \) and \( DEF \), with the same radius \( l_{\text{min}} \), denote the free space projections for turning left and right in SU-Mode, respectively. Similarly, the fan-shaped regions \( GAH \) and \( LIK \), with the same radius \( l_{\text{max}} \), denote the free space projections for turning left and right in LU-Mode, respectively. The rectangular regions \( EI \) and \( JL \) denote the free space projections for moving forward and backward in DL-Mode, respectively.

![Fig. 4. Projection of free space (The minimum length of the robot is denoted as \( l_{\text{min}} \) and the maximum length is \( l_{\text{max}} \). The length in critical mode switch is denoted as \( l_{SD} \) (SU-Mode \( \rightarrow \) DL-Mode), \( l_{DL} \) (DL-Mode \( \rightarrow \) LU-Mode), respectively.](image)

Definition 2: The quantity of the unit cells in a grid map \( G_{rid} \) is called the grid resolution, which is denoted by \( G_{grid} : d_i \times d_j \),

\[
d_i = \frac{L_{\text{grid}}}{l_{ch}} \quad d_j = \frac{L_{GV}}{l_{cv}}
\]

where \( L_{\text{grid}} \) is the horizontal length of the grid map \( G_{grid} \), \( L_{GV} \) is the vertical length of the grid map \( G_{grid} \), \( l_{ch} \) is the horizontal length of the unit cell, and \( l_{cv} \) is the vertical length of the unit cell.

If the grid resolution of the grid map \( G_{rid} \) is \( G_{grid} : m \times n \), then the \( G_{rid} \) can be defined as a matrix \( n \times m \), which is called the grid matrix, denoted by \( M(G_{rid}) \).

\[
M(G_{rid}) = \begin{bmatrix}
m_{11} & m_{12} & \cdots & m_{1n} \\
m_{21} & m_{22} & \cdots & m_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
m_{m1} & m_{m2} & \cdots & m_{mn}
\end{bmatrix}
\]

Let any one of the unit cells in the grid map \( G_{rid} \) be represented by two opposite-angle points \( g_{ij} \) and \( g_{n+1-j,i} \), then the element \( m_{ij} \) of the grid matrix \( M(G_{rid}) \) can be put into a one-to-one correspondence with the unit cells (see Fig. 5):

\[
m_{ij} \leftrightarrow g_{ij} \leftrightarrow g_{n+1-j,i}
\]

where \( 1 \leq i \leq d_i \) and \( 1 \leq j \leq d_j \).

![Fig. 5. Corresponding relationship between grid map and matrix](image)

The necessary steps to build the dynamic partial grid map for the inchworm-like robot are as follows:

-- Step 1: Initialize the grid resolutions \( G_{grid} \) for the \( C_{\text{fre}} \) projections corresponding to six kinds of basic gaits of the robot (Let the \( C_{\text{fre}} \) be projected onto the plane ‘Z’O’X’ of the reference coordinates \( \{r\} \)).

-- Step 2: Calculate the coordinate values \( g_{ij} = (x_{ij}, z_{ij}) \) of the intersection points in the grid map for the \( C_{\text{fre}} \) projections. The calculation formulas to the basic gaits, i.e., moving forward in DL-Mode, moving backward in DL-Mode, turning left in SU-Mode, turning right in SU-Mode, turning left in LU-Mode and turning right in LU-Mode, are defined as:

\[
\begin{align*}
x_{ij} &= x_{si} + l_{SD} \cdot d_i + (l_{DA} - l_{SD}) \cdot (i - 1) \\
z_{ij} &= z_{si} - W \cdot d_i - 2W \cdot (j - 1) \\
x_{i+1,j} &= x_{si} + (l_{DA} - l_{SD}) \cdot (i - 1) \\
z_{i+1,j} &= z_{si} + W \cdot d_i - 2W \cdot (j - 1) \\
x_{ij} &= x_{si} + l_{min} \cdot (i - 1) \\
z_{ij} &= z_{si} + l_{min} \cdot (j - 1) \\
x_{i+1,j} &= x_{si} + d_i \cdot (l_{SD} - l_{min}) + l_{min} \cdot (i - 1) \\
z_{i+1,j} &= z_{si} - l_{min} \cdot (j - 1) \\
\end{align*}
\]
where \( i = 1, 2, \ldots, d_i + 1, j = 1, 2, \ldots, d_j + 1, \) and \( \text{W} \) denotes the width constant of the robot.

-- Step 3: Build the grid map by connecting all adjacent intersection points with straight lines, and compute the values of the elements in the grid matrix \( M(G_{id}) \) (see Definition 3).

-- Step 4: Determine whether the grid resolutions \( G_{id} \) need to be improved or not. If yes, return to the Step 2; On the contrary, end.

\[
\text{Definition 3: Let } C_{ob} = \{c_{ob1}, c_{ob2}, \ldots, c_{obk}\}, \text{ where } c_{ob} \quad (i = 1, 2, \ldots, k) \text{ is a separate and connected upset space in the } C_{ob}, \text{ and the grid map } G_{id} \text{ is built for the } C_{ob}. \text{ If the value of the element } m_{ij} \text{ in the grid matrix } M(G_{id}) \text{ is defined as}
\]

\[
m_{ij} = \begin{cases} 1 & \quad g_{n+1, i} \in G_{n+2-i, j} \cap c_{ob} \neq \emptyset \\ 0 & \quad g_{n+1, i} \in G_{n+2-i, j} \cap c_{ob} = \emptyset \\ 1 \leq i \leq n, 1 \leq j \leq m \end{cases}
\]

then the assigned grid matrix, which is a Boolean matrix and denoted by \( M(C_{ob}, C_{ob}) \), is called the correlation matrix between the \( C_{ob} \) and the \( C_{ob} \). Let the correlation matrixes between each \( c_{ob} \) \( (i = 1, 2, \ldots, k) \) in the \( C_{ob} \) and the \( C_{ob} \) be denoted by \( M(C_{ob}, C_{ob}) \), \( M(C_{ob}, C_{ob}) \), \( \ldots, M(C_{ob}, C_{ob}) \), respectively, then the Boolean matrix \( M(C_{ob}, C_{ob}) \), which is defined as

\[
M(C_{ob}, C_{ob}) = M(C_{ob}, C_{ob}) \lor \cdots \lor M(C_{ob}, C_{ob})
\]

(17)

is called the environment model based on the dynamic partial grid map.

B. Feasible Gait Searching

For the inchworm-like robot, any planning path from a starting point to a goal can be separated into a string of discrete gaits in order. Usually, there are many choices of the following gait for the robot which has just completed the current gait cycle, and all these choices satisfy the conditions of the obstacle avoidance and the kinematic constraints. However, only according to the current information, it is almost impossible to determine which gaits could eventually reach the goal and which is the best. To solve this question, the feasible gait searching algorithms based on complete and incomplete traversal are proposed.

The specific steps of the feasible gait searching algorithm based on complete traversal are as follows:

-- Step 1: Let \( n = 0 \), then initialize the position and orientation \( \theta_1 \) of the robot and the goal point \( P_{goal} \).

-- Step 2: Calculate \( M_i(C_{ob}, C_{ob}) \), \( i = 1, 2, \ldots, 6^n \) by looping \( 6^n \) times to model the current environment.

-- Step 3: Conforming to the rules (see Table III and Fig. 6), calculate the values of all joint parameters by looping \( 6^n \) times for six kinds of basic gaits, i.e., the stride length \( d_{BW} \) of moving forward in DL-Mode, the stride length \( d_{BW} \) of moving backward in DL-Mode, the steering angle \( r_{SUM} \) of turning left in SU-Mode, the steering angle \( r_{SUM} \) of turning right in SU-Mode, the steering angle \( r_{LUM} \) of turning left in LU-Mode and the steering angle \( r_{LUM} \) of turning right in LU-Mode.

-- Step 4: By looping \( 6^n \) times, carry out all of the basic gait cycles based on complete traversal, i.e., calculate the gait operators \( T_{BW}(c_i), T_{BW}(c_i), T_{BW}(c_i), T_{BW}(c_i), T_{BW}(c_i), T_{BW}(c_i) \) and \( T_{BW}(c_i) \).

-- Step 5: Loop \( 6^n \) times to judge whether the planning path has reached the goal point. If yes, end; On the contrary, go on the next step.

-- Step 6: Determine whether the number of the searching loop has reached the limit. If yes, end; On the contrary, return to the Step 2.

Although the above algorithm is a global optimal path planning, with the increase in the number of search loop, the storage space used in the algorithm will grow exponentially, and its efficiency of real-time calculation will be poorer. Therefore the feasible gait searching algorithm based on incomplete traversal is proposed, in which the evaluation function \( f(c_i) \) and the searching parameter \( K \) are introduced.

The specific steps of the feasible gait searching algorithm based on incomplete traversal are as follows:

-- Step 1: Let \( k = 1 \), then initialize the \( c_1 \) and the \( P_{goal} \).

-- Step 2: Calculate \( M_i(C_{ob}, C_{ob}) \), \( i = 1, 2, \ldots, k \) by looping \( k \) times to model the current environment.

-- Step 3: Conforming to the rules (see Table III and Fig. 6), calculate the values of all joint parameters by looping \( k \) times for six kinds of basic gaits, i.e., \( d_{BW}, d_{BW}, r_{SUM}, r_{SUM}, r_{LUM}, r_{LUM} \) and \( r_{LUM} \).

-- Step 4: By looping \( k \) times, carry out all of the basic gait cycles based on incomplete traversal, i.e., calculate the gait operators \( T_{BW}(c_i), T_{BW}(c_i), T_{BW}(c_i), T_{BW}(c_i), T_{BW}(c_i), T_{BW}(c_i) \) and \( T_{BW}(c_i) \).

-- Step 5: Loop \( 6k \) times to judge whether the planning path has reached the goal point. If yes, end; On the contrary, go on the next step.

-- Step 6: Determine whether the number of the searching loop has reached the limit. If yes, end; On the
contrary, go on the next step.

Step 7: Calculate $f(c_i)$, $i=1, 2, \ldots, 6k$ by looping 6k times to evaluate the planning progresses for the different path branches, then select $K$ optimal branches to continue the next search loop. Let $K = K$ and return to the Step 2.

The evaluation function $f(c_i)$ is defined as

$$f(c_i) = \lambda_1 \cdot D_{RR}(c_i) + \lambda_2 \cdot \theta_y(c_i) + \lambda_3 \cdot T(c_i)$$

where $D_{RR}(c_i)$ denotes the distance between the robot and the goal. $\theta_y(c_i)$ denotes the yawing angle of the robot relative to the goal. $T(c_i)$ denotes the consumptive time when the robot arrives at the position and orientation $c_i$. $\lambda_j$ is the weighting coefficient, with $j=1, 2, 3$.

In Table III, $D_{max}_{fw}$, $D_{max}_{bw}$, $r_{max_{st}}$, $r_{max_{sr}}$, $r_{max_{ll}}$, and $r_{max_{lr}}$ denote the maximum values of the joint parameters $d_{fw}$, $d_{bw}$, $r_{SUM_{st}}$, $r_{SUM_{sr}}$, $r_{SUM_{ll}}$, and $r_{SUM_{lr}}$ in the space $C_{max}$, respectively. Let

$$C_{rea} = C_{fire} - C_{fire} \cap C_{obs}$$

TABLE III

| IF $P_{goal}$ $\in$ $C_{obs}$ | THEN the values of the joint parameters $d_{fw}$, $d_{bw}$, $r_{SUM_{st}}$, $r_{SUM_{sr}}$, $r_{SUM_{ll}}$, and $r_{SUM_{lr}}$ are as follows, respectively.
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Area I</td>
<td>$D_{max}<em>{fw}$, $D</em>{max}<em>{bw}$, $\min(\angle P</em>{AP}, \angle r_{max_{st}})$, $\min(\angle P_{AP}, \angle r_{max_{sr}})$, $\min(\angle P_{AP}, \angle r_{max_{ll}})$, $\min(\angle P_{AP}, \angle r_{max_{lr}})$</td>
</tr>
<tr>
<td>Area II</td>
<td>$D_{max}<em>{fw}$, $D</em>{max}<em>{bw}$, $\min(\angle P</em>{AP}, \angle r_{max_{st}})$, $\min(\angle P_{AP}, \angle r_{max_{sr}})$, $\min(\angle P_{AP}, \angle r_{max_{ll}})$, $\min(\angle P_{AP}, \angle r_{max_{lr}})$</td>
</tr>
<tr>
<td>Area III</td>
<td>$D_{max}<em>{fw}$, $D</em>{max}<em>{bw}$, $\min(\angle P</em>{AP}, \angle r_{max_{st}})$, $\min(\angle P_{AP}, \angle r_{max_{sr}})$, $\min(\angle P_{AP}, \angle r_{max_{ll}})$, $\min(\angle P_{AP}, \angle r_{max_{lr}})$</td>
</tr>
<tr>
<td>Area IV</td>
<td>$D_{max}<em>{fw}$, $D</em>{max}<em>{bw}$, $\min(\angle P</em>{BP}, \angle r_{max_{st}})$, $\min(\angle P_{BP}, \angle r_{max_{sr}})$, $\min(\angle P_{BP}, \angle r_{max_{ll}})$, $\min(\angle P_{BP}, \angle r_{max_{lr}})$</td>
</tr>
<tr>
<td>Area V</td>
<td>$D_{max}<em>{fw}$, $D</em>{max}<em>{bw}$, $\min(\angle P</em>{BP}, \angle r_{max_{st}})$, $\min(\angle P_{BP}, \angle r_{max_{sr}})$, $\min(\angle P_{BP}, \angle r_{max_{ll}})$, $\min(\angle P_{BP}, \angle r_{max_{lr}})$</td>
</tr>
<tr>
<td>Area VI</td>
<td>$D_{max}<em>{fw}$, $D</em>{max}<em>{bw}$, $\min(\angle P</em>{BP}, \angle r_{max_{st}})$, $\min(\angle P_{BP}, \angle r_{max_{sr}})$, $\min(\angle P_{BP}, \angle r_{max_{ll}})$, $\min(\angle P_{BP}, \angle r_{max_{lr}})$</td>
</tr>
<tr>
<td>Area VII</td>
<td>$D_{max}<em>{fw}$, $D</em>{max}<em>{bw}$, $\min(\angle P</em>{BP}, \angle r_{max_{st}})$, $\min(\angle P_{BP}, \angle r_{max_{sr}})$, $\min(\angle P_{BP}, \angle r_{max_{ll}})$, $\min(\angle P_{BP}, \angle r_{max_{lr}})$</td>
</tr>
<tr>
<td>Area VIII</td>
<td>$D_{max}<em>{fw}$, $D</em>{max}<em>{bw}$, $\min(\angle P</em>{BP}, \angle r_{max_{st}})$, $\min(\angle P_{BP}, \angle r_{max_{sr}})$, $\min(\angle P_{BP}, \angle r_{max_{ll}})$, $\min(\angle P_{BP}, \angle r_{max_{lr}})$</td>
</tr>
<tr>
<td>Area IX</td>
<td>$D_{max}<em>{fw}$, $D</em>{max}<em>{bw}$, $\min(\angle P</em>{BP}, \angle r_{max_{st}})$, $\min(\angle P_{BP}, \angle r_{max_{sr}})$, $\min(\angle P_{BP}, \angle r_{max_{ll}})$, $\min(\angle P_{BP}, \angle r_{max_{lr}})$</td>
</tr>
<tr>
<td>Area X</td>
<td>$D_{max}<em>{fw}$, $D</em>{max}<em>{bw}$, $\min(\angle P</em>{BP}, \angle r_{max_{st}})$, $\min(\angle P_{BP}, \angle r_{max_{sr}})$, $\min(\angle P_{BP}, \angle r_{max_{ll}})$, $\min(\angle P_{BP}, \angle r_{max_{lr}})$</td>
</tr>
<tr>
<td>Area XI</td>
<td>$D_{max}<em>{fw}$, $D</em>{max}_{bw}$, $0, 0, 0, 0$</td>
</tr>
<tr>
<td>Area XII</td>
<td>$D_{max}<em>{fw}$, $D</em>{max}_{bw}$, $0, 0, 0, 0$</td>
</tr>
</tbody>
</table>

V. SIMULATION AND EXPERIMENT

A. Simulation

The given initial position and orientation $c_1$, the given goal point $P_{goal}$, and the given obstacle space $C_{obs}$ for the robot are shown in Fig. 8 (a). The grid resolutions corresponding to the basic gaits are defined as $G_{4x4}$ by 4 turning left or right in SU-Mode, $G_{7x7}$ by 7 turning left or right in SU-Mode and $G_{1x2}$ by moving forward or backward in DL-Mode, respectively. Let $K = 15$, and $\lambda_1 = 0.8$, $\lambda_2 = 0.2$, $\lambda_3 = 0$. The simulation results (Fig. 7 (e)) indicate that after eight gait cycles, there are nine paths eventually reaching the goal point $P_{goal}$, namely:

$$tr_1 : c_{goal} \bullet T_{\text{fw}}(T_{\text{fw}}(T_{\text{fw}}(T_{\text{fw}}(T_{\text{fw}}(T_{\text{fw}}(T_{\text{fw}}(T_{\text{fw}}(c_1))))))))$$

$$tr_2 : c_{goal} \bullet T_{\text{fw}}(T_{\text{fw}}(T_{\text{fw}}(T_{\text{fw}}(T_{\text{fw}}(T_{\text{fw}}(T_{\text{fw}}(T_{\text{fw}}(T_{\text{fw}}(c_1))))))))$$

$$tr_3 : c_{goal} \bullet T_{\text{bw}}(T_{\text{bw}}(T_{\text{fw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(c_1))))))))$$

$$tr_4 : c_{goal} \bullet T_{\text{bw}}(T_{\text{bw}}(T_{\text{fw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(c_1))))))))$$

$$tr_5 : c_{goal} \bullet T_{\text{lw}}(T_{\text{lw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(c_1))))))))$$

$$tr_6 : c_{goal} \bullet T_{\text{lw}}(T_{\text{lw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(c_1))))))))$$

$$tr_7 : c_{goal} \bullet T_{\text{lr}}(T_{\text{lr}}(T_{\text{lr}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(c_1))))))))$$

$$tr_8 : c_{goal} \bullet T_{\text{lr}}(T_{\text{lr}}(T_{\text{lr}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(c_1))))))))$$

$$tr_9 : c_{goal} \bullet T_{\text{lr}}(T_{\text{lr}}(T_{\text{lr}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(c_1))))))))$$

In order to get the optimal path in time, three weighting coefficients are readjusted as $\lambda_1 = 0, \lambda_2 = 0, \lambda_3 = 1$. Then the comparison concludes that the optimal path in time is

$$tr_8 : c_{goal} \bullet T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(T_{\text{bw}}(c_1))))))))$$

which includes eight basic gait paths in the order of their occurrence (see Fig. 7 (f), i.e., turning left in SU-Mode, moving forward in DL-Mode, turning left in SU-Mode, moving forward in DL-Mode, moving forward in DL-Mode, turning left in LU-Mode, turning right in SU-Mode, and moving forward in DL-Mode.
Let the initial values of the $c_1$, $P_{\text{goal}}$, $C_{\text{obs}}$, $G_{\text{dif}}$ and $\lambda_j$ (j=1, 2, 3) be same as the former simulation. When let $K=6$, there are six paths (i.e., $tr_1$, $tr_2$, $tr_3$, $tr_5$, $tr_7$ and $tr_8$) eventually reaching the $P_{\text{goal}}$ after eight gait cycles (Fig. 8.a). When let $K=4$, there are four paths (i.e., $tr_1$, $tr_3$, $tr_7$ and $tr_8$) eventually reaching the $P_{\text{goal}}$ after eight gait cycles (Fig. 8.b). When let $K=1$, the robot eventually reaches the $P_{\text{goal}}$ after eleven gait cycles (Fig. 8.c), and the only path is denoted as

$$c_{12} = T_{Dl}(T_{Dh}(T_{Dh}(T_{Dh}(T_{Sb}(T_{Ld}(T_{Dl}(T_{Sb}(c_1))))))))$$

which includes eleven basic gaits in the order of their occurrence, i.e., turning left in SU-Mode, moving forward in DL-Mode, turning right in LU-Mode, turning right in SU-Mode, moving backward in DL-Mode, turning right in SU-Mode, moving backward in DL-Mode, moving backward in DL-Mode, moving backward in DL-Mode, turning right in LU-Mode, and moving backward in DL-Mode.

![Fig. 8. Simulations of path planning with different $K$](image)

### B. Experiment

Now we can only make the experiments of offline path planning on the real inchworm-like robot because of the poor performance of the sensors. Let $K=20$, and $\lambda_1=0.8$, $\lambda_2=0.2$, $\lambda_3=0$. The initial values of the $c_1$, $P_{\text{goal}}$, $C_{\text{obs}}$ and $G_{\text{dif}}$ are known in advance for the robot. The experimental results indicate that after nine basic gait cycles, the robot eventually reaches the goal without any contact with the obstacles (Fig. 9).

$$c_{10} = T_{Dl}(T_{Dh}(T_{Dh}(T_{Dh}(T_{Sb}(T_{Ld}(T_{Dl}(T_{Sb}(c_1)))))))$$

Let $K=1$, and other parameters remain unchanged. In the same experimental circumstances, the robot successfully reaches the goal without any contact with the obstacles after fourteen basic gait cycles, as shown in Fig. 10.

![Fig. 9. Experiment of path planning ($K=20$)](image)

### VI. CONCLUSION

Aiming at the problem of path planning for the inchworm-like robot moving in a narrow space, a novel path planning method which includes the environment modeling based on dynamic partial grid map and the feasible gait searching algorithm based on complete or incomplete traversal is proposed in this paper.

By the simulation analysis and the experiment verification, we can deduce three main conclusions. First, for the inchworm-like robot moving in a narrow space, this path planning method is feasible and reasonable. Second, the values of three weighting coefficients in the evaluation function have a direct impact on the uniqueness of the path planning results. If not unique, we can get an optimal path by readjusting the weighting coefficients for re-evaluation or re-planning. Third, the value of the searching parameter $K$ has a direct impact on the optimality of the path planning results. The greater the value, the more likely we get a global optimal path. But equally we have to increase the computation and the storage. In the future work, we will continue to study how to improve the efficiency of the searching algorithm and the adaptability of the planning method.

### REFERENCES


