Research Article

Hierarchically Planning Static Gait for Quadruped Robot Walking on Rough Terrain

Xingdong Li , Hewei Gao, Jian Li, Yangwei Wang, and Yanling Guo

College of Mechanical and Electrical Engineering, Northeast Forestry University, Harbin 150040, China

Correspondence should be addressed to Yanling Guo; guoyl_nefu@163.com

Received 11 January 2019; Revised 18 March 2019; Accepted 26 March 2019; Published 2 May 2019

Academic Editor: Gordon R. Pennock

Copyright © 2019 Xingdong Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Quadruped robot has great potential to walk on rough terrain, in which static gait is preferred. A hierarchical structure based controlling algorithm is proposed in this paper, in which trajectory of robot center is searched, and then static gaits are generated along such trajectory. Firstly, cost map is constructed by computing terrain features under robot body and cost of selecting footholds at default positions, and then the trajectory of robot center in 2D space is searched using heuristic A* algorithm. Secondly, robot state is defined from foothold and robot pose, and then state series are searched recursively along the trajectory of robot center to generate static gaits, where a tree-like structure is used to store such states. Lastly, a classical model for quadruped robot is designed, and then the controlling algorithm proposed in the paper is demonstrated on such robot model for both structured terrain and complex unstructured terrain in a simulation environment.

1. Introduction

Legged robot offers great potentials in some complex and dangerous environment, because it has more flexibility and adaptability than that of wheeled counterparts. This potential has sparked a great deal of research on legged robot in recent years, especially on quadruped robot [1–4]. Static gait [5–8] is mainly used to drive quadruped robot to walk on rough terrain, although there has also been work on adaptive gaits which make use of a biologically inspired CPG [9] to traverse rough terrain using both dynamic and static gait, where the terrain typically has only very small irregularities compared to the size of the robot. Static gait is that the quadruped robot maintains static stability, which keeps the center gravity of the robot inside supporting triangle composed of three standing feet. LittleDog [10] is a classical quadruped robot specifically designed to support the research applications as part of DARPA's Learning Locomotion program (DARPA BAA 05-25), and a great deal of good works [11, 12] developed for quadruped robot on rough terrain are validated by the platform.

This paper is focused on designing hierarchical controlling system for quadruped. Research on controlling algorithm

for quadruped robot has changed gradually to design static gait used on complex and dangerous terrain, such as excellent works [13, 14]. Bai S et al. [13] proposed concepts of main static gait and minor static gait; the main static gait is used for in general cases, and the minor static gait will be started when the main gait cannot control the robot. Space for searching possible gait is reduced greatly by employing such mixed gait, so it is very efficient to generate static gait for quadruped robot. Estremera J [14] introduced noncontinuous static gait for irregular terrain to increase efficiency of planning gait. S. Bai et al. proposed a controlling algorithm [15] for driving the quadruped robot to walk on structured terrain, where Best First Search (BFS) method was applied to search the valid walking path, but the searching space will be increased largely for some more rough terrain environment. Kolter et al. proposed a completed hierarchical frame [16] for controlling the little dog [10] to walk, in which three levels including high level planner, low level programming and low level controller are defined and realized. The high level planner extracts possible footholds which are ordered according to strategy of hierarchical apprenticeship learning (HAL) [17]; the low level planner generates trajectory of robot between two continuous states, and the low level controller is used to

drive the robot along the trajectory. Although the algorithm is hierarchical, robot center is determined by the footholds which are searched in the whole space, so the efficiency can be optimized. This paper will provide an algorithm to generate series of robot states hierarchically, where series of robot center are firstly searched, and then footholds are estimated from robot center and terrain feature. Annett et al. employed the similar idea [18] to drive hexapod robot, but the trajectory of robot center is searched without considering the complexity of the terrain beneath the foothold. In aspect of space complexity, most of the algorithms aforementioned store robot state by a queue structure; suppose there are n possible states in the next step, and series of robot state for d steps are computed; then there are $O(n^d)$ states at least in the queue, so the space complexity is very large, which will limit the scalability of the controlling algorithm. In this paper, the space complexity will be decreased by storing robot state in a tree-like structure.

The remainder of the paper is organized as follows. Section 2 provides algorithm for searching trajectory of robot body center; Section 3 describes algorithm for generating series of static gait along the trajectory of robot center; Section 4 defines a classical model of quadruped robot, and the whole algorithm proposed in this paper is validated for both structured terrain and complex unstructured terrain in simulation environment. Section 5 will conclude the whole paper.

2. Searching 2D Path of the Robot Center Heuristically

2.1. Constructing Cost Map of Terrain under Robot Body. The cost map of terrain under robot body is computed from the following 3 terrain features: terrain slope, step height, and terrain roughness. In order to estimate those features, as shown in Figure 1, a circle on the grid based map is made around the robot center, whose radius is set large enough to cover the whole robot body, and the features are estimated based on the grids inside the circle.

2.1.1. Estimating Slope of the Terrain. Let the grid (i, j) lie at center of the robot body, and radius of the circle is r_{cost} which is related to the scale of the robot, for each grid (c, r) inside the circle; there is a 3D point related to such grid. Suppose there are e grids totally inside the circle; all the 3D points are represented with an ordered set which is shown in

$$\begin{aligned} P_{ij}^C &= \{p_1^{Cij}(x_1^{Cij}, y_1^{Cij}, z_1^{Cij}), p_2^{Cij}(x_2^{Cij}, y_2^{Cij}, z_2^{Cij}), \dots, \\ & p_e^{Cij}(x_e^{Cij}, y_e^{Cij}, z_e^{Cij})\} = \{p_{cr}(x_{cr}, y_{cr}, z_{cr}) \mid \\ & \forall \text{grid}(c, r) \in \text{the circle centered at the grid}(i, j)\} \end{aligned} \quad (1)$$

Plane function is defined as (2), whose parameters are estimated from the set P_{ij}^C using the least square method.

$$Ax + By + Cz + D = 0 \quad (2)$$

FIGURE 1: 2D projection of grid based elevation map.

For optimizing the parameters of the plane handily, the triangle M is constructed as (3) and the vector b as (4).

$$M = \begin{bmatrix} \sum_{k=1}^e (x_k^{Cij})^2 & \sum_{k=1}^e x_k^{Cij} y_k^{Cij} & \sum_{k=1}^e x_k^{Cij} \\ \sum_{k=1}^e x_k^{Cij} y_k^{Cij} & \sum_{k=1}^e (y_k^{Cij})^2 & \sum_{k=1}^e y_k^{Cij} \\ \sum_{k=1}^e x_k^{Cij} & \sum_{k=1}^e y_k^{Cij} & e \end{bmatrix} \quad (3)$$

$$b = \left[\sum_{k=1}^e x_k^{Cij} z_k^{Cij}, \sum_{k=1}^e y_k^{Cij} z_k^{Cij}, \sum_{k=1}^e z_k^{Cij} \right]^T \quad (4)$$

Let the intermediate variables be computed from M and b as (5), and then the parameters of the plane are computed as (6).

$$[A', B', D']^T = M^{-1} * b \quad (5)$$

$$[A, B, C, D]^T = [-A', -B', 1, -D']^T \quad (6)$$

Slope of the terrain centered at the grid (i, j) is estimated using (7), where n_{plane} is the normal of the plane, and $n_z = [0, 0, 1]^T$.

$$S_{slope}(i, j) = \arccos \left(\frac{\text{dot}(n_{plane}, n_z)}{\|n_{plane}\| * \|n_z\|} \right) \quad (7)$$

2.1.2. Estimating Roughness of the Terrain. The elevation of the grid (c, r) is represented by $G(c, r)$ and the roughness of the terrain under the robot body is defined as the standard deviation of the elevations related to the grids inside the green circle in Figure 1, which is written formally as

$$R_{roughness}(i, j) = \text{StdVar}(\{G(c, r) \mid \forall \text{grid}(c, r) \in \text{the circle centered at the grid}(i, j)\}) \quad (8)$$

FIGURE 2: The processing diagram for computing the step height.

2.1.3. Estimating Step Height of the Terrain. For each grid (c, r) inside the circle centered at the grid (i, j) , maximum difference between the elevation of it and that of other grids around it. A sliding window is made to search adjacent grids of the grid (c, r) , which is shown as blue triangle in Figure 1. The greatest one is selected from all the maximum differences with respect to every grid (c, r) as the step height of the terrain centered at the grid (i, j) . (c_1, r_1) is the grid with respect to the maximum difference of height for the grid (c, r) , and $n_{overCrit}$ in Figure 2 is used to record the number of grids (c, r) satisfying that slope of the line between center point of (c, r) and that of (c_1, r_1) . n_{crit} is the empirical value of $n_{overCrit}$. The step height is computed by (9), which combines the slope value and the height value, so it considers the case that some grid in the sliding window may lie outside the circle.

$$s_{tep} \leftarrow \min \left\{ h_{tmp}, \frac{n_{overCrit}}{n_{crit}} \right\} \quad (9)$$

2.1.4. Computing Cost of Terrain Centered at Robot Body Center

(i) *Traversability Judgment.* The step height of the terrain under the robot body is used to judge the traversability of the robot; if the step height is larger than some threshold $\delta_{critSteHeit}$, then the robot is considered not to traverse such terrain, and the cost with respect to the central grid (i, j) is set to ∞ .

(ii) *Computing Cost of Grid (i, j) .* The traversability map justly indicates whether the robot can walk across the grid; in order to search optimized path, cost value must be assigned to each

grid with respect to the center of the robot body. Considering the traversability, cost value of the grid (i, j) can be computed from three features: slope, roughness, and step height using

$$C_{BodyCen}(i, j) = \begin{cases} \infty & \text{if } M_{travers}(i, j) = 0 \\ \lambda_s \frac{S(i, j)}{s_{max}} + \lambda_r \frac{R(i, j)}{r_{max}} + \lambda_h \frac{H(i, j)}{h_{max}} & \text{if } M_{travers}(i, j) = 1 \end{cases} \quad (10)$$

In (10), $M_{travers}$ is sign of the traversability. S, R, H represent slope, roughness, and step height, respectively. $\lambda_s, \lambda_r, \lambda_h$ are weight coefficients, which reflect contributions for the cost value, and quadratic sum of them is 1; i.e., $\sqrt{\lambda_s^2 + \lambda_r^2 + \lambda_h^2} = 1$. $s_{max}, r_{max}, h_{max}$ are maximum feature values which are used to normalize the cost value.

2.2. Computing Average Terrain Cost Beneath 4 Feet

2.2.1. Computing Cost for Selecting Foothold. There are 4 features of the terrain beneath foothold:

- (i) f_1 : curvature of the terrain around the candidate foothold p_c , it is used to recognize whether the terrain is flat or not, which is considered for less sliding of the foot.
- (ii) f_2 : slope of the terrain around the candidate foothold p_c , it is computed as the angle between the normal at p_c and the horizontal surface.
- (iii) f_3 : the feature for illustrating concave terrain or convex one, let $p_1, p_2, \dots, p_8$ be 8 points with uniform distribution around the candidate foothold p_c , and v_i is the directed vector from p_c to p_i where $i \in [1, 8]$; the intersection angle between v_i and the normal of terrain at point p_c is computed, and then all the 8 angles are averaged as the feature value. The concave terrain is better than the convex one, as long as the foot can not be stuck.
- (iv) f_4 : the feature for determining uphill terrain or downhill one, the vertical plane P is constructed from the motion direction of the robot d and the vertical upward axis Z , suppose n' is the vector which is projected onto the plane P from the terrain normal n at the candidate foothold, the rotational angle from d to n' is computed as the feature value. It is very important to judge that the terrain is uphill or downhill, because the front foot of the quadruped robot prefers uphill terrain, while the back foot prefers downhill terrain.

Both the normal and curvature are estimated from 3D points inside the sphere with a radius, so the terrain features are related to the radius. In order to get terrain information with different scales, two radiuses are employed, 5mm and 10mm, respectively. As shown in (11), ξ is feature vector of the terrain beneath foothold, and $\lambda_1 - \lambda_4$ and $\lambda_5 - \lambda_8$ in the

feature vector ξ are terrain features $f_1 - f_4$ computed with the radius $5mm$ and $10mm$, respectively.

$$\xi = [\lambda_1, \lambda_2, \dots, \lambda_j, \dots, \lambda_8]^T \quad (11)$$

Now the cost function is defined as the weighted sum of feature values, which is represented formally as

$$C_{foot}(\xi) = W^T \xi = \sum_{k=1}^8 \omega_k \lambda_k \quad (12)$$

W is weight vector as (13), which can be optimized using expert learning [19].

$$W = [\omega_1, \omega_2, \dots, \omega_j, \dots, \omega_8]^T \quad (13)$$

2.2.2. Computing Cost of Terrain around Default Foothold. Default foothold is defined as the projection point of shoulder of the quadruped robot; a triangle is made around the default foothold which lies at center of it. There are 4 triangles related to default footholds, respectively. Average cost for each default foothold is computed from all the grids in the triangle. Equation (14) shows 4 average costs, where n_{lf} is the number of footholds inside the triangle, ξ_k^{lf} is the feature vector of the k th foothold, and W_{lf}^T is the weight vector with respect to left front foot.

$$\begin{aligned} C_{lf} &= \frac{\sum_{k=1}^{n_{lf}} W_{lf}^T \xi_k^{lf}}{n_{lf}} \\ C_{rf} &= \frac{\sum_{k=1}^{n_{rf}} W_{rf}^T \xi_k^{rf}}{n_{rf}} \\ C_{lb} &= \frac{\sum_{k=1}^{n_{lb}} W_{lb}^T \xi_k^{lb}}{n_{lb}} \\ C_{rb} &= \frac{\sum_{k=1}^{n_{rb}} W_{rb}^T \xi_k^{rb}}{n_{rb}} \end{aligned} \quad (14)$$

Similarly, let the center of the robot body lie in the grid (i, j) , and the cost related to foot is computed by averaging 4 costs in (14), which is shown in

$$C_{AvgFeet}(i, j) = \frac{C_{lf} + C_{rf} + C_{lb} + C_{rb}}{4} \quad (15)$$

2.3. Searching 2D Path Based on Cost Map. Differing from the case of using traditional A* algorithm, cost map here is updated in real-time, because the cost map consists of terrain cost related both robot body and foot, and default foothold varies with the moving direction of the robot. Let the robot center be at the grid (i, j) in the grid based elevation map; then the whole cost value related to (i, j) is computed using (16), where $C_{BodyCen}(i, j)$ and $C_{AvgFeet}(i, j)$ are estimated by (10) and (15), respectively. $\lambda_{BodyCen}$, $\lambda_{AvgFeet}$ are weight coefficients for two kinds of cost, and $\sqrt{\lambda_{BodyCen}^2 + \lambda_{AvgFeet}^2} = 1$,

which can guarantee the last cost value belongs to the range $[0, 1]$.

$$\begin{aligned} C(i, j) &= \lambda_{BodyCen} C_{BodyCen}(i, j) \\ &+ \lambda_{AvgFeet} C_{AvgFeet}(i, j) \end{aligned} \quad (16)$$

$C(i, j)$ in (16) represents the cost for robot body center moving across the grid (i, j) along some direction, in order to search optimal path base on the cost map C using A* algorithm, the grid based map should be organized into a graph structure, where node is represented by the grid, and edge is represented by the moving cost from the grid to an adjacent grid. The edge of the graph is computed as (17), and it is the real cost value for search 2D path of robot body center.

$$\begin{aligned} C_E(i, j, i', j') &= \beta_d \sqrt{reso_y^2 (i - i')^2 + reso_x^2 (j - j')^2} \\ &+ \beta_c C(i', j') \end{aligned} \quad (17)$$

$(i, j), (i', j')$ are current and adjacent grids and $reso_x, reso_y$ are resolutions of grids in elevation map, β_d, β_c are weight coefficients which satisfy the constraints: $\beta_c > \beta_d \geq 1$, and the larger the β_c is, the greater the tendency is taking a detour to find a safe path.

The movement direction $D_{(i', j')}$ for computing the whole cost of grid (i', j') is determined by the relative positions between grid (i, j) and grid (i', j') , which is represented formally using

$$D_{(i', j')} = [(j' - j) reso_x, (i' - i) reso_y]^T \quad (18)$$

Designing heuristic function is a key step to search optimal path using A* algorithm; in our case, the heuristic function is listed as (19), where (i_t, j_t) is the terminal grid, and the heuristic function must satisfy the constraint shown in (20) for searching optimal path.

$$h(i, j, i_t, j_t) = \sqrt{reso_y^2 (i - i_t)^2 + reso_x^2 (j - j_t)^2} \quad (19)$$

$$h(i, j, i_t, j_t) \leq C_E(i, j, i', j') + h(i', j', i_t, j_t) \quad (20)$$

According to the theory of triangle inequality, the heuristic function here satisfies the constraint. The searched 2D path of robot body center includes both coordinates of the points and movement direction on the path.

3. Searching Series of Static Gaits

3.1. Planning Standard Static Gait of Quadruped Robot. In terms of energy savings and robot stability, standard order of switching flying foot is left front foot(LF) $\rightarrow$ right back foot(RB) $\rightarrow$ right front foot(RF) $\rightarrow$ left back foot(LB). Figure 3 shows one circle of standard gaits, black solid dots are supporting footholds, solid circles are feet before flying, and dotted circles are targets of flying feet. Projection of robot center gravity must fall into the supporting triangle which is determined by the supporting feet.

FIGURE 3: Nominal static gait pattern of one circle.

FIGURE 4: Yaw angle computation.

The star in Figure 3 represents the projection of robot center, which lies in central line of interactional triangle of two supporting triangles. Let $V(v_x, v_y)$ be the vertex of the interactional triangle and $P_c(p_x, p_y)$ be mid-point of the faced edge, projected point of robot body gravity P_G is computed using (21), where α is coefficient of proportionality with range $[0, 05]$.

$$P_G = \alpha (P_c - V) + V \quad (21)$$

In addition to selecting proper foothold, adjusting pose of the robot body is an important step to drive the quadruped robot to walk. The pose is generally determined by the parameters: height of robot body center, angle of yaw, angle of roll, and angle of pitch. The following part will briefly introduce how to get such parameters.

(a) *Height of Robot Body Center.* Three supporting feet are used to fit a plane P_{lane} , and a vertical line across the robot center intersects with the plane P_{lane} at point p_{int} ; the distance between p_{int} and the robot center is computed as the height of robot body center.

(b) *Angle of Yaw.* The yaw is determined by 3 supporting feet and target of the flying foot, which is shown in Figure 4. Taking the case that left front foot is flying as an example; foot is firstly projected to the horizontal plane, and P_B is center of two projections with respect to back feet; P_F is center with respect to front feet. The yaw angle is computed from the directed vector $D_{Yaw} = \overrightarrow{P_B P_F}$.

(c) *Angle of Raw and Pitch.* The interaction angle between robot body and horizontal plane is varied with both raw and pitch angle of the robot; in terms of robot stabilization, robot body should be parallel with the terrain plane in which 3 supporting feet lie.

To sum up, there are 5 steps to compute the pose of robot body, given 3 supporting feet and the target of the flying foot.

- (i) 3D point P_G^{3D} is found from the plane in which there are 3 supporting points, and projection of P_G^{3D} onto the horizontal plane is P_G .
- (ii) P_G^{3D} is translated upward by height of the robot body center, which is used as the coordinate of robot body center.
- (iii) Rotational transformation R_{Yaw} is computed from yaw angle.
- (iv) Another rotational transformation $R_{RollPitch}$ is computed from roll and pitch angle.
- (v) Rotational transformation of robot body is computed by $R_{robot} = R_{Yaw} * R_{RollPitch}$.

3.2. *Collision Detection.* There are two cases that may produce collision between robot (body or leg) and terrain: one is flying foot from original point to target point, the other is adjusting pose of robot body.

3.2.1. *Movement of Foot.* Foot is planned to avoid the collision between leg and terrain; a simple method is employed to generate foot trajectory; foot is moving along rectangular box. Take right front foot as an example, which is shown in Figure 5.

The robot is projected into 2D plane $Y = 0$, and it walks along X positive direction, X coordinate of start point is $0.1m$, and that of end point is $0.17m$, rectangle between the two points represents obstacle. If no collision is produced at any key positions of foot, then the process of flying foot is safe. The key position is shown in Table 1.

3.2.2. *Adjusting Pose of Robot Body.* Before flying foot, gravity of robot body should be adjusted into the supporting triangle, which may produce collision between leg and terrain. Figure 6 shows the process of adjusting robot body, given that the foot is fixed (taking right front foot as an example).

There are 5 key poses of robot; if none of these poses produce collision, then it is safe to adjust pose of robot body when fixing standing feet. Similarly, after projecting the robot onto 2D plane $Y = 0$, key poses are listed in Table 2

TABLE 1: Key points in the foot trajectory.

	1	2	3	4	5	6	7
X	0.100	0.100	0.100	0.135	0.170	0.170	0.170
Z	-0.270	-0.235	-0.200	-0.200	-0.200	-0.235	-0.270

TABLE 2: Key poses in the body adjusting trajectory.

	1	2	3	4	5
X	0	0.025	0.05	0.05	0.05
α_y	0	0	0	-7.5°	-15°

FIGURE 5: Foot Trajectory.

FIGURE 6: Moving trajectory of the robot body.

3.3. Searching Series of Robot States

3.3.1. Defining Robot State. State after taking next step of robot is defined from 4 elements: foothold, height of center of robot body, pitch angle, and roll angle. It is formally represented as

$$\mathbf{s} = [\mathbf{f}_{oot}, r_{oll}, p_{itch}, h_{eight}] \quad (22)$$

Now the problem is changed into how to select such 4 elements.

(a) Set of Footholds. A rectangle is made to be centered at default target of foothold; several candidate footholds are included in the rectangle, which are ordered by the cost value computed using (12).

$$\mathbf{F}_{candi} = \{\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_{n_{foot}}\} \quad (23)$$

In (23), there are n_{foot} candidates in ordered set $\mathbf{F}_{candi}$; cost value of $\mathbf{f}_1$ is smallest and that of $\mathbf{f}_{n_{foot}}$ is largest.

(b) Set of Roll Angles. Ordered set of roll angles is shown in (24), where n_{roll} is the number of candidate roll angles, default roll angle is r , and step length for adjusting roll angle is Δ_r .

$$\begin{aligned} \mathbf{R}_{candi} &= \{r_1, r_2, \dots, r_{n_{roll}}\} \\ &= \{r, r + \Delta_r, r - \Delta_r, r + 2\Delta_r, r - 2\Delta_r, \dots\} \end{aligned} \quad (24)$$

(c) Set of Pitch Angles. Similar to the set of roll angles, that of pitch angles is shown in (25), where n_{pitch} is the number of candidate roll angles, default pitch angle is p , and step length for adjusting pitch angle is Δ_p .

$$\begin{aligned} \mathbf{P}_{candi} &= \{p_1, p_2, \dots, p_{n_{pitch}}\} \\ &= \{p, p + \Delta_p, p - \Delta_p, p + 2\Delta_p, p - 2\Delta_p, \dots\} \end{aligned} \quad (25)$$

(d) Set of Heights. Set of vertical distance between robot body center and terrain is shown in (26), where n_{height} is the

number of candidate heights, default height is h , and step length for adjusting h is Δ_h .

$$\begin{aligned} \mathbf{H}_{candi} &= \{h_1, h_2, \dots, h_{n_{height}}\} \\ &= \{h, h + \Delta_h, h - \Delta_h, h + 2\Delta_h, h - 2\Delta_h, \dots\} \end{aligned} \quad (26)$$

Let the number of states for next step be n , then it can be computed using

$$n = n_{foot} \times n_{roll} \times n_{pitch} \times n_{height} \quad (27)$$

Set of n states $\mathbf{S}_{candi}$ is ordered in terms of priority as (28); the first element in the set has the highest priority.

$$\begin{aligned} \mathbf{S}_{candi} = \{s_1, s_2, \dots, s_n\} &= \{f_1 r_1 p_1 h_1, f_1 r_1 p_1 h_2, \dots, \\ &f_1 r_1 p_2 h_1, \dots, f_1 r_2 p_1 h_1, \dots, f_2 r_1 p_1 h_1, \dots\} \end{aligned} \quad (28)$$

According to the order in set $\mathbf{S}_{candi}$, priority for adjusting 4 parameters is

$$\begin{aligned} \text{Height of robot center} &> \text{Pitch angle} > \text{Roll angle} \\ &> \text{Foothold} \end{aligned} \quad (29)$$

Terrain complexity has a great influence on the stability of robot, so foothold should be selected with the lowest cost as much as possible. Comparing to translation, the rotation has greater impact on the stability of robot, so the height parameter is firstly adjusted. According to the model of quadruped robot, it is relatively safe to adjust pitch angle firstly.

3.3.2. Searching Gaits Recursively. When the quadruped robot explores one step forwardly, there are n candidate states, and they are selected in turn by the element order in the set $\mathbf{S}_{candi}$ in (28). Several steps can be searched one time; robot state of k th step is $s_{i_1 i_2 \dots i_k}$, which indicates that the i_k th state is selected from n candidates for k th step. Let ordered set of candidate robot state for $k + 1$ th step be $\mathbf{S}_{candi}^{k+1}$ as shown in

$$\mathbf{S}_{candi}^{k+1} = \{s_1^{k+1}, s_2^{k+1}, \dots, s_n^{k+1}\} \quad (30)$$

Let s_j^{k+1} be the first state from the set $\mathbf{S}_{candi}^{k+1}$ that satisfies the following conditions:

- (i) Robot state s_j^{k+1} will not violate kinematic constraint of both robot body and leg.
- (ii) No collision will occur when the robot is moving from state $s_{i_1 i_2 \dots i_k}$ to state s_j^{k+1} .

If s_j^{k+1} is found from $\mathbf{S}_{candi}^{k+1}$, then the robot state for $k + 1$ th step is $s_{i_1 i_2 \dots i_k i_{k+1}} = s_j^{k+1}$; else no possible robot state can be continuously searched from state $s_{i_1 i_2 \dots i_k}$; now backtracking must be carried out. Searching process restarts from the $i_k + 1$ th candidate state of k th step and state for next step $k + 1$ is searched until reaching to the target searching depth.

The searching process with recursive backtracking is represented by a recursive tree shown in Figure 7. For convenience, suppose there are only 2 candidate states for each searching step, gaits of 4 steps are illustrated by green line with arrow.

None of the searching paths on left and below of green line can reach the 4th step, because some switches of robot states would make collision. States on right and above green line have not been searched yet. In this searching process, only 5 states in the green line are restored. All of the states below the green line may be constructed in the worst case, but if there is no state transformation from s_{12} to s_{121} , then the children states s_{1211} and s_{1212} will not need to be considered.

The following heuristic rules can be used to prune some unnecessary branches.

- (i) If the kinematic constraint can not be satisfied for current height value, then the larger height value needs not be considered.
- (ii) If collision happens between rear leg of robot and terrain for current pitch angle, then the smaller pitch angle needs not be considered.
- (iii) If collision happens between left leg of robot and terrain for current roll angle, then the larger roll angle needs not be considered.

The more steps the robot searches forwardly, the more optimized the searched series of static gaits.

4. Simulation Result and Discussion

4.1. Model of the Quadruped Robot. The model of the quadruped robot in this paper is simplified as long as there are three degrees in one leg. A simple model with DH parameters is shown in Figure 8, and the foot can be reached anywhere within its motion space.

In Figure 8, a is the half length of robot body, b is the half width of robot body, c is the width of shoulder, d is the length of thigh, and e is the length of shank. Taking right back leg as an example, the red arrow line represents rotation axis (Z), and the green arrow line represents common vertical line between two rotational axes, whose direction is equal to that of the axis X . Z_0 and Z_4 are two virtual axes introduced for convenient representation, but there is no rotational degree of freedom actually. Z_0 lies at the center of robot body, and it is supposed that X_0 and X_1 share the same line.

Variable DH parameters of the model above are shown in Tables 3 and 4. θ s in the tables above are related to the joint angles directly, whose ranges are shown in Table 5.

Constant parameters are shown in Table 6.

4.2. Walking on Structured Terrain. In order to validate the effectiveness of the algorithm for planning standard static gait, a simple simulation is carried out. The quadruped robot is driven along some straight line, and the footholds are divided into two categories: valid foothold at flat terrain and invalid foothold at terrain with edge.

Structured terrain model is generated in OpenRAVE [20] simulation environment, which is composed of several

FIGURE 7: Tree-style structure for recursively searching series of static gaits.

FIGURE 8: DH parameters based model of the quadruped robot.

TABLE 3: The DH parameters for the left legs.

i	LF				LB			
	a_i	d_i	θ_i	α_i	a_i	d_i	θ_i	α_i
1	b	0	0	90°	b	0	0	90°
2	0	a	θ_2^{LF}	90°	0	$-a$	θ_2^{LB}	90°
3	$-d$	c	θ_3^{LF}	0	$-d$	c	θ_3^{LB}	0
4	$-e$	0	θ_4^{LF}	0	$-e$	0	θ_4^{LB}	0

FIGURE 9: Series of standard static gaits for two circles on structured terrain.

FIGURE 10: Different searching results by looking one and four steps ahead, respectively.

cuboids with length of $0.4m$ and width of $0.04m$; height of the cuboid is randomized in the range $[0m, 0.06m]$. The points at the edge are invalid foothold, and, in terms of priority, all the valid footholds around the default footholds are ordered according to the distance between default foothold and candidate foothold. The distance is closer; then the priority of the foothold is higher. In order to compute the pose of the robot body, parameters in the equation (24)-(27) are set by the value in Table 7.

Standard static gaits for two circles are shown in Figure 9. All the candidate footholds belong to either set of valid footholds or set of invalid footholds and only the point at the edge of the terrain is invalid foothold. All of the valid footholds are ordered in terms of the distance to the default foothold. Priority of the foothold nearer to the default foothold is high.

The strategy with recursive backtracking is used to search gait series for the quadruped robot, recursive depth which is the gait circles for searching forwardly once and has great impact to the searching result, and the depth is 2 for the searching process shown in Figure 9. The recursive depth has great impact on the gait series searched, which is demonstrated by the simulation. In general, when the depth is not deep, the searching efficiency is satisfied, but the searching process may fall into local minima; then the

result will not be optimized, even though there is no valid result.

Two cases are shown in Figure 10; the first case is that the algorithm searches forwardly only one step, and gait series after the first step are not considered, so collision may occur by some step as shown in Figure 10(a), in which the left back leg will be touched with the terrain when preparing for the next gait step. The collision mentioned above would not occur if the feasibility of such gait step is considered as shown in Figure 10(b), in which the recursive depth is 4.

4.3. Walking on Complex Unstructured Terrain. Simulation in Section 4.2 demonstrates the effectiveness of the algorithm for searching gait series on the structured terrain, which also validates the logistic correctness of the controlling algorithm, and in this section the complexity of the terrain will be increased, where algorithms for both searching 2D path of body center and searching gait series are validated comprehensively.

Another terrain model is generated in simulation environment, whose size is $1.4m \times 1.2m$, and 2.5D elevation map based on grid with respect to such terrain model is shown in Figure 11(a).

The radius for computing terrain features (roughness, slope, and step height) is $r_{cost} = 0.1m$; maps of their feature

FIGURE 11: Geometric features of 3D terrain under the robot body.

TABLE 4: The DH parameters for the right legs.

i	RF				RB			
	a_i	d_i	θ_i	α_i	a_i	d_i	θ_i	α_i
1	$-b$	0	0	90°	$-b$	0	0	90°
2	0	a	θ_2^{RF}	90°	0	$-a$	θ_2^{RB}	90°
3	$-d$	$-c$	θ_3^{RF}	0	$-d$	$-c$	θ_3^{RB}	0
4	$-e$	0	θ_4^{RF}	0	$-e$	0	θ_4^{RB}	0

TABLE 5: Range of the joint angle.

$\theta_2^{LF}, \theta_2^{RF}, \theta_2^{LB}, \theta_2^{RB}$	$\theta_3^{LF}, \theta_3^{RF}, \theta_3^{LB}, \theta_3^{RB}$	$\theta_4^{LF}, \theta_4^{RF}$	$\theta_4^{LB}, \theta_4^{RB}$
$[75^\circ, 105^\circ]$	$[-45^\circ, 45^\circ]$	$[-180^\circ, 0^\circ]$	$[0^\circ, 180^\circ]$

TABLE 6: Constant parameters of the model.

a	b	c	d	e
0.15m	0.07m	0.02m	0.09m	0.1m

TABLE 7: Parameters for generating candidate body pose.

r	Δ_r	p	Δ_p	h	Δ_h	n_{roll}	n_{pitch}	n_{height}
0°	5°	0°	5°	0.14m	0.015m	5	5	5

FIGURE 12: Cost map of the 3D terrain.

FIGURE 13: Trajectory of 2D body center.

values are shown in Figures 11(b), 11(c), and 11(d), respectively, where color represents feature value.

Cost map is constructed from three feature maps using (10), where parameters are set as Table 8.

The larger some feature weight is, the greater its contribution to the cost function is. Taking slope feature as an example, if λ_s is larger, then the robot will walk along a path far from the point with deep slope, although that path may be longer than that crossing such point. In our simulation environment, step height is relatively low, and the robot can walk across nearly all the steps, so λ_h is set lower in this paper. The proportion of these 3 parameters is

$$\lambda_h : \lambda_r : \lambda_s = 1 : 2 : 2 \quad (31)$$

The computed cost map is shown in Figure 12, where the cost value is represented by the color. According to simulation result in the figure, the cost value computed from the approach proposed in this paper can measure the complexity of the terrain accurately.

The cost map in Figure 12 is projected into 2D space, in which 2D path of the robot body center is heuristically searched using the updated A* algorithm discussed in Section 2.3. β_d, β_c in (17) are parameters for weighing moving

FIGURE 14: Motion trajectory of the quadruped.

distance and walking safety. Figure 13 shows path of body center searched with two kinds of parameters, $\beta_d = 1, \beta_c = 5$ and $\beta_d = 1, \beta_c = 10$, respectively, where color stands for the cost value of the terrain.

Coordinate of the start point is $(-0.05m, 0.05m)$ or $[1, 71]$ in grid unit, and coordinate of the end point is $(0.65m, -0.075m)$ or $[141, 46]$ in grid unit. Resolution of grids in both X and Y direction is $5mm$. Point in the search path is so dense that only one point among three points is shown in Figure 13. According to the simulation result in the figure, if the parameter β_c is set larger, then the robot will avoid the region with higher cost terrain, though it may walk for a longer distance.

The 2D path of body center with parameter $\beta_d = 1, \beta_c = 5$ is selected, and the quadruped robot is planned to walk along such 2D path using the algorithm proposed in the paper; the real walking trajectory is shown in Figure 14. For showing clearly, 2.5D elevation map is projected into 2D space, where the elevation value is represented by the color. According to the simulation result in the figure, the projected point of robot body center moves along the searched 2D path, and none of the feet touches the terrain with higher cost, which can demonstrate the effectiveness of the proposed planning algorithm for driving the quadruped robot walking on complex terrain.

5. Conclusions

In this paper, a hierarchical algorithm for planning quadruped robot walking on rough terrain is proposed. There are two steps to execute the algorithm; the first step is searching the trajectory of robot center using heuristic algorithm A*, in which the terrain feature beneath the foothold is also considered; the second step is searching series of static gait along the trajectory of robot center, and the searching space is shrunk greatly, so the time complexity of the algorithm is reduced clearly. In aspect of space complexity, robot state is introduced for representing node of static gait, and tree-like structure is constructed to store such states for saving memory space overhead. Lastly, a classical model of quadruped robot is defined, and then the whole algorithm is validated in simulation environment.

TABLE 8: Parameters for computing cost value.

λ_s	λ_r	λ_h	s_{max}	r_{max}	h_{max}
0.667	0.667	0.333	1.5708Rad	0.02m	0.035m

Data Availability

The datasets generated during and/or analysed during the current study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work has been supported by Fundamental Research Funds for the Central Universities (Grant no. 2572017BB12); the State Key Laboratory of Robotics and System (HIT) Open Fund (Grant no. SKLRS-2017-KF-06); the National Natural Science Foundation of China (Grant no. 16051575097); the National Science Foundation for Postdoctoral Scientists of China (Grant nos. 2015M571381 and 2017T100217); and the Natural Science Foundation of Heilongjiang Province (Grant no. JJ2019LH2071).

References

- [1] M. Focchi, A. del Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, and C. Semini, "High-slope terrain locomotion for torque-controlled quadruped robots," *Autonomous Robots*, vol. 41, no. 1, pp. 259–272, 2017.
- [2] C. Gehring, S. Coros, M. Hutler et al., "An optimization-based approach to controlling agile motions for a quadruped robot," *IEEE Robotics and Automation Magazine*, vol. 23, pp. 34–43, 2016.
- [3] X. Kong, B. Yu, L. Quan, K. Ba, and M. Li, "Characteristic parameters sensitivity of position servo control for hydraulic drive unit of a quadruped robot in trotting gait," *Robot*, vol. 37, no. 1, pp. 63–73, 2015.
- [4] J. Degraeve, M. Burm, P.-J. Kindermans, J. Dambre, and F. Wyffels, "Transfer learning of gaits on a quadrupedal robot," *Adaptive Behavior*, vol. 23, no. 2, pp. 69–82, 2015.
- [5] R. B. McGhee, "Finite state control of quadruped locomotion," *Simulation*, vol. 9, no. 3, pp. 135–140, 2016.
- [6] R. B. McGhee, "Some finite state aspects of legged locomotion," *Mathematical Biosciences*, vol. 2, no. 1-2, pp. 67–84, 1968.
- [7] S. Ma, T. Tomiyama, and H. Wada, "Omnidirectional static walking of a quadruped robot," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 152–161, 2005.
- [8] J. Estremera and P. G. de Santos, "Generating continuous free crab gaits for quadruped robots on irregular terrain," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1067–1076, 2005.
- [9] H. Kimura, Y. Fukuoka, and A. H. Cohen, "Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts," *International Journal of Robotics Research*, vol. 26, no. 5, pp. 475–490, 2007.
- [10] M. P. Murphy, A. Saunders, C. Moreira, A. A. Rizzi, and M. Raibert, "The littledog robot," *International Journal of Robotics Research*, vol. 30, no. 2, pp. 145–149, 2011.
- [11] J. R. Reubla, P. D. Neuhaus, B. V. Bonnlander, M. J. Johnson, and J. E. Pratt, "A controller for the Little Dog quadruped walking on rough terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1050–1079, Roma, Italy, April 2007.
- [12] J. Zico Kolter and A. Y. Ng, "The stanford littledog: a learning and rapid replanning approach to quadruped locomotion," *International Journal of Robotics Research*, vol. 30, no. 2, pp. 150–174, 2011.
- [13] S. Bai, K. Low, G. Seet et al., "A new free gait generation for quadrupeds based on primary/secondary gait," in *Proceedings of the International Conference on Robotics and Automation*, pp. 1371–1376, Detroit, MI, USA, May 1999.
- [14] J. Estremera and P. G. De Santos, "Free gaits for quadruped robots over irregular terrain," *International Journal of Robotics Research*, vol. 21, no. 2, pp. 115–130, 2002.
- [15] S. Bai, K. Low, and M. Teo, "Path generation of walking machines in 3D terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2216–2221, Washington, DC, USA, May 2002.
- [16] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 811–818, Pasadena, Calif, USA, May 2008.
- [17] J. Kolter Z, P. Abbeel, and A. Y. Ng, "Hierarchical apprenticeship learning, with application to quadruped locomotion," in *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 769–776, Vancouver, Canada, December 2007.
- [18] A. Stelzer, H. Hirschmüller, and M. Görner, "Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain," *International Journal of Robotics Research*, vol. 31, no. 4, pp. 381–402, 2012.
- [19] X. Li, J. Li, and Y. Guo, "Foothold selection for quadruped robot based on learning from expert," in *Proceedings of the IEEE International Conference on Advanced Robotics and Mechatronics*, pp. 223–228, Hefei, China, August 2017.
- [20] R. Diankov, *Automated Construction of Robotic Manipulation Programs*, Carnegie Mellon University, Robotics Institute, 2010.