Front. Mech. Eng. 2023, 18(2): 26
https://doi.org/10.1007/s11465-022-0742-y

RESEARCH ARTICLE

Footholds optimization for legged robots walking on complex terrain

Yunpeng YIN[a], Yue ZHAO[b], Yuguang XIAO[a], Feng GAO (✉)[a]

[a] *State Key Laboratory of Mechanical System and Vibration, School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China*
[b] *AI Institute, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China*
✉ *Corresponding author. E-mail: gaofengsjtu@gmail.com (Feng GAO)*

ABSTRACT This paper proposes a novel continuous footholds optimization method for legged robots to expand their walking ability on complex terrains. The algorithm can efficiently run onboard and online by using terrain perception information to protect the robot against slipping or tripping on the edge of obstacles, and to improve its stability and safety when walking on complex terrain. By relying on the depth camera installed on the robot and obtaining the terrain heightmap, the algorithm converts the discrete grid heightmap into a continuous costmap. Then, it constructs an optimization function combined with the robot's state information to select the next footholds and generate the motion trajectory to control the robot's locomotion. Compared with most existing footholds selection algorithms that rely on discrete enumeration search, as far as we know, the proposed algorithm is the first to use a continuous optimization method. We successfully implemented the algorithm on a hexapod robot, and verified its feasibility in a walking experiment on a complex terrain.

KEYWORDS footholds optimization, legged robot, complex terrain adapting, hexapod robot, locomotion control

1 Introduction

Legged robots have more capacity in complex terrain than wheeled ones. The agile and dynamic control of legged robots in flat terrains has achieved great progress, but taking full advantage of the flexibility of legged robots in rough terrain remains a challenge. Because in the locomotion of a legged robot, its own motion balance and terrain perception should be considered at the same time, only then the corresponding footwork for adapting to complex terrain can be generated.

Heuristic walking gait [1,2] is a simple and effective blind locomotion strategy. Based on the model of spring-loaded inverted pendulum (SLIP), it considers the balance relationship between its own speed and foothold in the process of walking. However, in this strategy, the next foothold is related only to its body speed but not to the ground features. It is only suitable for flat road conditions and cannot effectively adapt to unstructured terrains. Moreover, there is an assumption that in the locomotion of legged robots, it has no relative sliding between the support legs and the ground. Nevertheless, the more uneven the ground is, the more difficult it is to satisfy this assumption. For example, when climbing stairs with a fixed gait, the robot often slips when stepping on the edge of a step, kicks to the eaves of a step, and even trips over.

If such a blind walking strategy is applied on complex terrains, the stability and safety of the robot cannot be guaranteed. Thus, it is necessary to introduce perceptive information of the terrain, to optimize the footholds selection, and avoid stepping on the edge of potholes or obstacles.

Researchers at LittleDog [3,4], HyQ [5,6], and ANYmal [7,8] proposed methods for footholds and trajectory optimization based on a one-time preprocessed costmap. This kind of methods is time consuming to plan the actions before the robot starts to move. After the action trajectories are determined, it runs open loop, which cannot be adjusted when environmental changes or error occurs. Then, onboard and online foothold optimization methods were proposed and applied in HyQ [9–11], ANYmal [12,13], Hexapod-III [14], Mini Cheetah [15], and Qingzhui [16–18]. They considered a similar approach (i.e., obtaining a grid heightmap of the

Received July 6, 2022; accepted November 7, 2022

terrain through a depth camera and generating a costmap), then conducted a random search in the batch area around the preplanned foothold. However, given that the costmap is discretized in grid cells, the optimization of the footholds can only enumerate a finite set of integers in the batch area, rather than a spatial continuous optimization. The distance of the searched footholds in the optimization process depends on the grid size. Hence, although the feasibility of a selected foothold can be guaranteed, it is not a global optimum. Later, HyQ researchers [19] utilized a convolutional neural network (CNN) to find the optimal foothold, whose input is also the batch area around the preplanned foothold. The CNN greatly improved the computational efficiency. ANYmal researchers [20,21] directly applied the reinforcement learning method to complete the autonomous movement of the robot in the field environment and offshore platform [22], which can be said to be a milestone achievement in the field of legged robots. However, these data-driven methods require considerable data for training, which increases the initial cost of algorithm deployment. In addition to such methods of constructing terrain maps, some researchers have extracted geometric features of known target structures through visual information [23–25], but these methods can only adapt to the terrain of preknown scenes.

Therefore, we need to develop a set of solutions that combine terrain awareness with the state constraints of the robot, do not rely on high-cost data training, and can operate efficiently onboard, to optimize the footholds, control the robot's actions in a spatially continuous way, and expand the adaptability of the legged robot in complex terrain.

The contributions of this paper are summarized as follows:

a) A continuous and derivable method for converting discrete grid maps is proposed. Through this method, the grid terrain heightmap is converted into a smooth terrain costmap.

b) A gradient-based continuous optimization method is constructed using the terrain costmap and the motion state of the robot to select the robot's footholds.

c) The body motion is corrected in accordance with the optimized footholds, and the swing leg trajectories are generated. Combined with the method of balance force control, an experiment is carried out on a hexapod robot passing through a complex terrain.

To our knowledge, this is the first footholds selection method for legged robots based on continuous optimization algorithm with no training data required, which can efficiently run online and onboard. We choose the continuous optimization method, for the reason that we want to realize optimal footholds in a continuous space. Given that it requires a continuous and derivable costmap and objective function, it can be efficiently deployed online. The remainder of this paper is structured as follows:

Section 2 provides the overview of the hexapod robot developed by Shanghai Jiao Tong University. Section 3 presents a novel and efficient method to make discrete heightmaps continuous and derivable, generalizing a continuous costmap from an elevation map obtained by a red, green, blue, depth (RGBD) camera. Section 4 formulates the cost function in consideration of the costmap above, along with the robot locomotion and kinematics, to instruct the footholds of the robot when walking. Section 5 explains how to use the optimized footholds on our robot to combine it with our control framework. Section 6 shows the results from the experiment of the hexapod walking on an irregular terrain with our method. Finally, conclusions are drawn in Section 7.

2 Robot overview

2.1 Hardware structure

LittleStrong is a bionic designed hexapod robot inspired by insects. Its six legs are symmetrically distributed on both sides of the body. To avoid legs' collision with each other when walking, the middle legs are installed slightly farther from the sagittal plane.

There are 3 degrees of freedom for each leg, serially constructed as hip, thigh, and shank joints. Thus, LittleStrong has 18 joints in total, as shown in Fig. 1. Each joint is driven by a brushless electric motor with a planetary gear set and an embedded encoder. The robot is about 0.4 m in height, 0.6 m in length, and 0.4 m in width when standing.

The coordinate systems of the robot are defined in Fig. 1. The world coordinate system (WCS) is fixed to the ground, and the position where the robot is turned on is regarded as the origin. The robot locomotion control is mostly completed in the WCS. The body coordinate system (BCS) is fixed at the center of its body, and all variables in BCS will be marked with a left superscript {B}. The six legs of the robot have their own leg coordinate system (LCS), which is fixed with the BCS but takes the leg's hip joint as the origin. All variables in LCS will be marked with a left superscript L, and the right subscript i represents the leg number.

The electrical system is shown in Fig. 2. There are two onboard computers that cooperate with each other. The locomotion control computer runs the GNU/Linux patched with Xenomai [26] real-time kernel. It communicates with 18 drivers and an inertial measurement unit (IMU) through EtherCAT protocol. The drivers provide power for 18 actuators (brushless motors), along with 18 encoders feeding back the angle and angular velocity of each motor. IMU is installed at the center of the robot body and feeds back attitude, angular velocity, and acceleration. The depth camera is fixed on the head of the

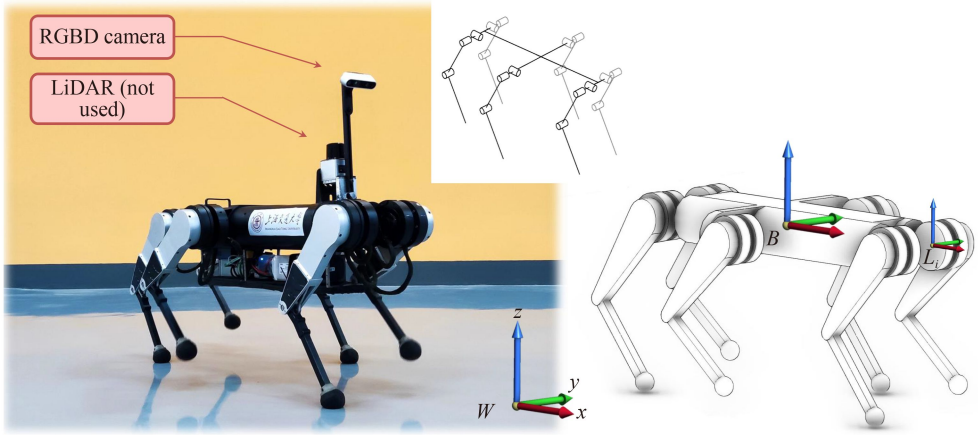

Fig. 1 Photo of the hexapod robot and its coordinate systems. RGBD: red, green, blue, depth.

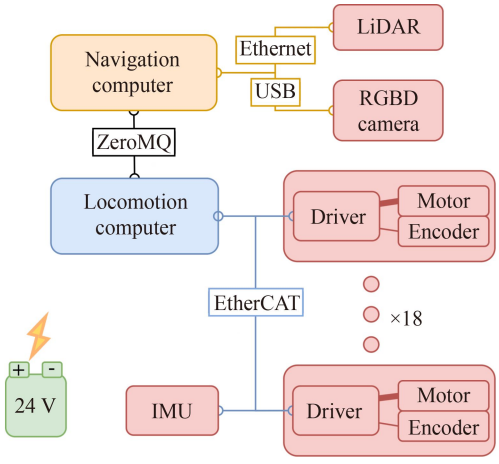

Fig. 2 Electrical system of the robot. IMU: inertial measurement unit, RGBD: red, green, blue, depth.

robot and connected to the navigation computer running the Robot Operating System. The navigation computer sends the locomotion commands to the locomotion control computer through ZeroMQ [27].

2.2 Control framework

The overall control framework of the robot is shown in Fig. 3. The orange blocks run on the navigation computer, which will be the main content of the footholds optimization in the later sections. The blue blocks run on the locomotion control computer. The locomotion control of the legged robot mainly includes two contents, namely, gait control and body balance control.

For gait control, the robot employs a gait scheduler to generate step cycles. A legged robot needs to periodically control the alternating lifting and landing of different legs to let the body move. For a hexapod, there are many optional gait combinations. Considering that a trot gait is commonly used for quadrupeds, a tripod-trot gait [28] is adopted for hexapods. Tripod-trot gait refers to that the six legs of the robot are divided into two groups. In each step cycle, there are three legs in swing phase and the other three legs in support phase. The gait scheduler is shown in Fig. 4, where the white areas represent the swing phase; the blue and red areas represent the support phase for different leg groups, corresponding to the lifting and landing of each leg, respectively.

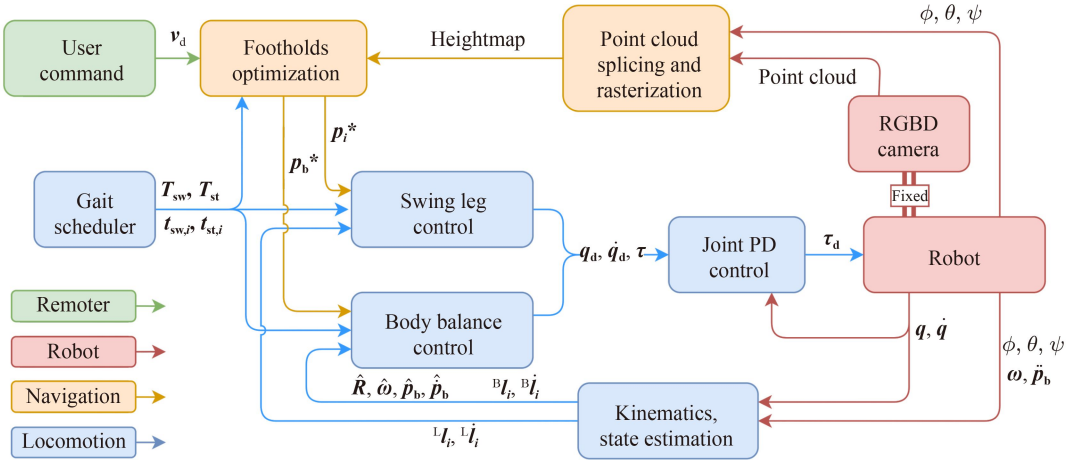

Fig. 3 Control framework of the robot for footholds optimization.

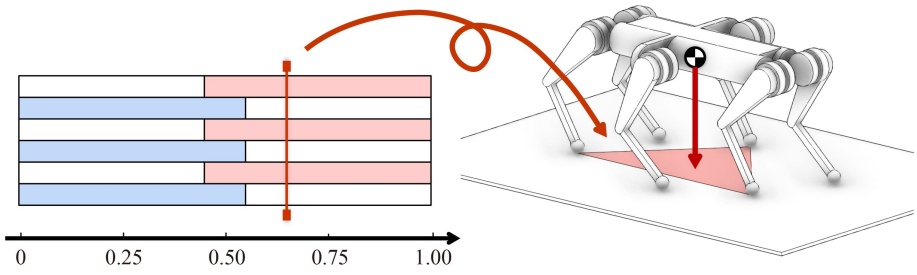

Fig. 4 Gait scheduler and tripod-trot gait for the robot.

The tripod-trot gait can furnish a triangular stable region underneath its supported legs in each gait cycle, as shown in Fig. 4. This can be considered a natural advantage of hexapods over quadrupeds, such that we can focus on the working task itself without wasting much attention on the balance control of the robot. For a swing leg, a suitable foothold must be selected to satisfy the speed and balance requirements of body movement when it becomes a supporting leg in the next step cycle. Additionally, a suitable swing leg trajectory is also needed to be designed. This will be further explained in Section 4 and Subsection 5.1.

The robot employs a centroid balance control (CBC) algorithm to generate support legs' control signals through the state feedback of the whole robot body. In the calculation process, we apply a simplified model of robot dynamics. The details are elucidated in Subsection 5.2.

The locomotion controller composed of the above two contents can satisfy the adaptive locomotion on general terrain. However, when in complex terrain, it can only pass through by blind walking. A gait optimizer should be introduced in combination with visual information to avoid pits and obstacles. The details will be elaborated below.

3 Costmap processing

The key idea for a legged robot adapting to complex terrains is to select suitable footholds for the next step. The selection of footholds integrates two parts: terrain perception and body state. This section focuses on terrain perception. To obtain spatial continuous optimal and apply continuous optimization algorithms, this section proposes a new method to transform the discrete grid heightmap into a continuous costmap. Subsection 3.1 proposes a new method to transform the discrete grid map into a second-order continuously derivable costmap. Note that this subsection is purely a theoretical approach to discrete grid continuation, and the examples used are only representative and simple grid maps. Subsection 3.2 introduces how to apply this method to convert the terrain point cloud obtained by the depth camera into a terrain costmap for robot locomotion optimization in the next section.

3.1 Grid map continuation

Convolution is a commonly used method in digital image processing [29]. In the process of image convolution, each element of the image is added to its local neighbors, weighted by the kernel. As shown in Fig. 5, the Hadamard product (i.e., multiplication of corresponding elements) is performed between the four pixels in the red box in the left original image and the yellow convolution kernel matrix. The elements of the generated intermediate matrix are summed to obtain the pixels in the red box in the processed image on the right. The lower part of the figure shows a 3D view of the process. Given that the convolution operation synthesizes the information between the pixels, the influence of image noise is well avoided. However, in this process, the convolution kernel operates on the image pixel by pixel and moves discretely. To obtain spatial continuously changing values, interpolation methods can be adopted.

In our grid map, each pixel (grid cell) represents the average height of the ground in a specific area (2 cm × 2 cm). To obtain the height between cells, continuous height values must be obtained through 2D interpolation. Considering that the direct interpolation will be affected by random noise, the convolution interpolation (CI) is used for continuous smoothing of the grid map on the basis of the image convolution technique. As shown in Fig. 6, in the CI process, different from the image convolution, the kernel (red box) is a mask-plate that moves smoothly on the grid map. The overlapping area (yellow box in dash line) between the mask-plate and the cell is defined as its weight. The sum of weighted cells is used as the center (point c) value of the mask-plate. In this way, a continuous heightmap can be obtained. Moreover, in the image convolution mentioned above, adjusting the size of its kernel, so called mask-matrix, can yield different filtering results. Similarly, in CI, the number of cells involved in convolution can be adjusted by altering the size of the mask-plate to obtain similar effects.

Although CI can obtain continuous costmaps, its results are not differentiable. As shown in Fig. 6, there will be obvious bending at the boundary of the grid, which will have a fatal impact on the optimization solution in the next section. To solve this problem, we propose a pyramid convolution interpolation (PCI) smoothing

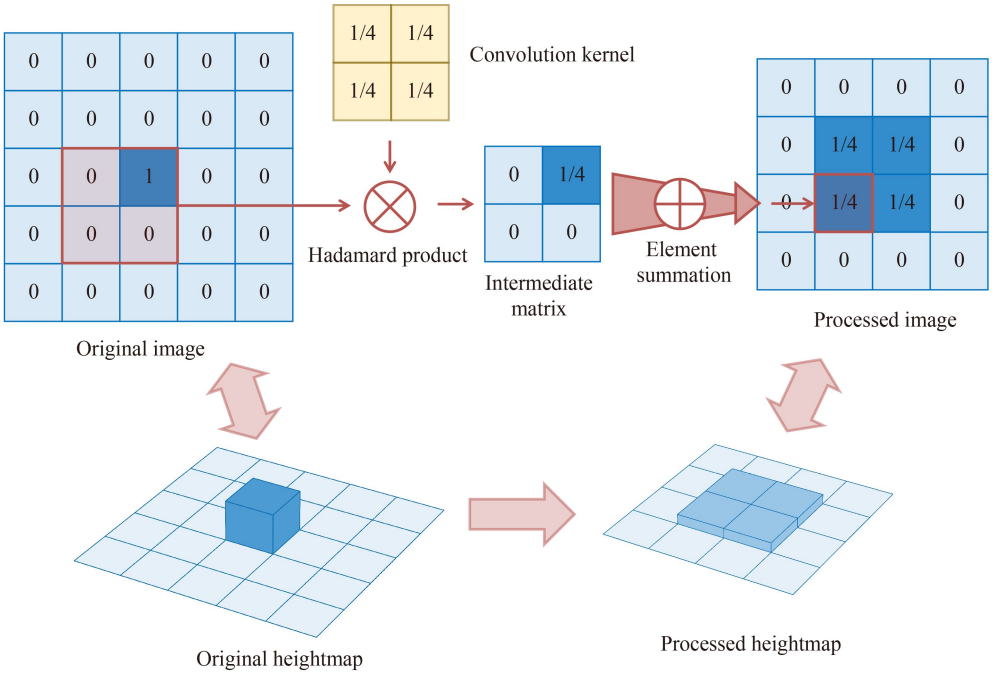

Fig. 5 Example of image convolution in digital image processing.

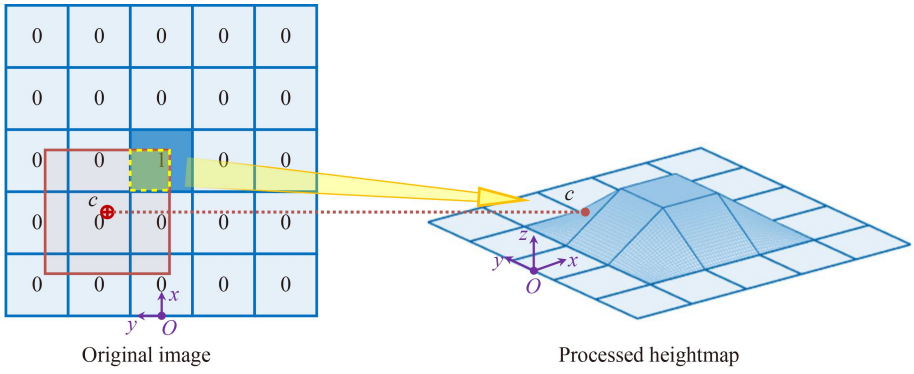

Fig. 6 Convolution interpolation for transforming a pixel into a continuous heightmap.

algorithm. In PCI, the kernel is no longer the mask-matrix used for image convolution nor the mask-plate mentioned in CI but a mask-pyramid (projected as red box) shown in Fig. 7. Similarly, the mask-pyramid moves smoothly on the grid map. The overlapping volume (projected as yellow box) above the cell, which is covered by the projection of the pyramid, is used as the weight of this cell. The sum of weighted cells is used as the center point value of the pyramid. From Fig. 7, the continuous heightmap obtained in this way is smoother and derivable than that in Fig. 6. The number of cells involved in convolution can also be adjusted by changing the size of the pyramid's bottom quadrilateral.

However, in the actual computing process, the overlapping volume of the pyramid and the cell is difficult to calculate; especially after the size of different bottom surfaces of the pyramid is adjusted, the volume shape of a block is more complex and difficult to calculate. This will have a significant impact on the calculation efficiency in the optimization iterative calculation process in the next section.

Therefore, this paper proposes a new convolution method, Normalized Derivable Convolution Interpolation (NDCI) algorithm. In PCI, the second-order continuity can be obtained because in the weighting process, with the smooth movement of the pyramid, the cell's weight gradually increases from zero, along with its second-order continuous. When the center of the convolution kernel coincides with the center of the grid, the weight value reaches its maximum. Then, to make the costmap differentiable, we only need to find the weight function with similar expression. Obviously, this characteristic can

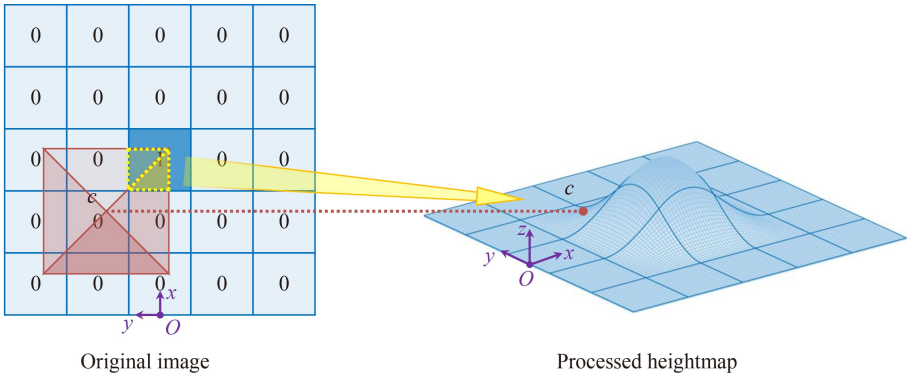

Fig. 7 Pyramid convolution interpolation for transforming a pixel into a continuous heightmap.

be obtained by a cosine function. The algorithm formula is as follows:

$$J(\boldsymbol{p}_{\mathrm{h}}) = \sum_{i=0}^{m}\sum_{j=0}^{n}\left(\cos\frac{\pi(x_{\mathrm{h}}-x_{i,j})}{2\sigma_x}\cos\frac{\pi(y_{\mathrm{h}}-y_{i,j})}{2\sigma_y}\right)^2, \quad (1)$$

where $\boldsymbol{p}_{\mathrm{h}} = [\,x_{\mathrm{h}} \quad y_{\mathrm{h}}\,]^{\mathrm{T}}$ is a horizontal coordinate, representing the position of the selected point on the whole map, x_{h} and y_{h} are x and y components of the horizontal coordinate for the position of the selected point on the heightmap, respectively, $\boldsymbol{c}_{i,j} = [\,x_{i,j} \quad y_{i,j}\,]^{\mathrm{T}}$ is the coordinate of the center of the pixel (i, j) on the map, $x_{i,j}$ and $y_{i,j}$ are x and y components of the horizontal coordinate for the position of the selected point on the heightmap, respectively, $\{m, n\}$ are the set numbers of pixels that need to be involved, similar to the size of the kernel in image convolution, $\{\sigma_x, \sigma_y\}$ are the smoothing factors in two directions, which can eliminate the local minimum to a certain extent and keep the continuity of the function. Briefly, they ensure that the variable value of the cosine function is kept between $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. Therefore, $\{\sigma_x, \sigma_y\}$ are related to $\{m, n\}$. In addition, on the basis of the chain rule, its partial derivative $\nabla J(\boldsymbol{p}_{\mathrm{h}})$ can also be easily obtained:

$$\nabla J(\boldsymbol{p}_{\mathrm{h}}) = \begin{bmatrix}\dfrac{\partial J(\boldsymbol{p}_{\mathrm{h}})}{\partial x_{\mathrm{h}}}\\ \dfrac{\partial J(\boldsymbol{p}_{\mathrm{h}})}{\partial y_{\mathrm{h}}}\end{bmatrix} = \begin{bmatrix}\displaystyle\sum_{i=0}^{m}\sum_{j=0}^{n}\left(-\frac{\pi}{\sigma_x}\cos\frac{\pi(x_{\mathrm{h}}-x_{i,j})}{2\sigma_x}\sin\frac{\pi(x_{\mathrm{h}}-x_{i,j})}{2\sigma_x}\cos^2\frac{\pi(y_{\mathrm{h}}-y_{i,j})}{2\sigma_y}\right)\\ \displaystyle\sum_{i=0}^{m}\sum_{j=0}^{n}\left(-\frac{\pi}{\sigma_y}\cos^2\frac{\pi(x_{\mathrm{h}}-x_{i,j})}{2\sigma_x}\cos\frac{\pi(y_{\mathrm{h}}-y_{i,j})}{2\sigma_y}\sin\frac{\pi(y_{\mathrm{h}}-y_{i,j})}{2\sigma_y}\right)\end{bmatrix}. \quad (2)$$

Figure 8 shows the outcome heightmap of NDCI, whose shape is similar to the PCI result in Fig. 7 but greatly reduces the difficulty of calculation. The lower part of Fig. 8 shows the gradient in the horizontal directions, respectively.

3.2 Costmap generation

In the previous subsection, we proposed a method for continuous smoothing of grid maps. This subsection will take the above approach and describes the process of generating a costmap from a complete heightmap obtained by the depth camera.

First, a discrete heightmap of the terrain around the robot is generated. The robot only has a depth camera on its head. To obtain complete terrain information around the robot, the point cloud data obtained by the depth camera during the movement of the robot should be recorded. Then, the point cloud data, combined with the robot's position and attitude information, are spliced and filtered. The spliced point cloud data are compressed, sampled, and rasterized, so we can obtain a heightmap of the terrain around the robot. The heightmap has (100 × 50) pixels, each pixel corresponds to an area (2 cm × 2 cm) projected on the horizontal plane, and each pixel's value (grayscale) represents a different height.

Secondly, the heightmap is transformed into a discrete height difference map. The places with large height difference are usually the edges of obstacles, which are the areas that the robot does not want to step on. The places with small height difference are flat areas, which can be safely stepped on with confidence. The edge detection algorithm commonly used in digital image processing is employed.

Lastly, the method mentioned in Subsection 3.1 is applied to convert the discrete height difference map into a continuous costmap. This costmap will be used as one of the items for optimization in the next section. Figure 9 shows our experimental environment, which will be further discussed in Section 6, as well as the processed discrete heightmap and continuous costmap.

The processing and splicing of point cloud data are

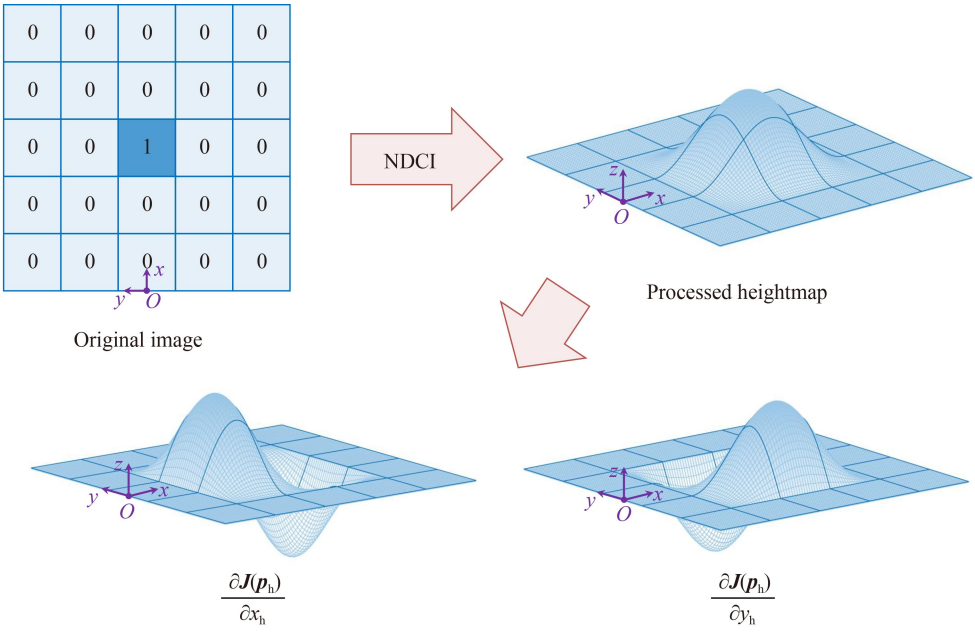

Fig. 8 NDCI for transforming a pixel into a continuous heightmap, with its gradients. NDCI: normalized derivable convolution interpolation.

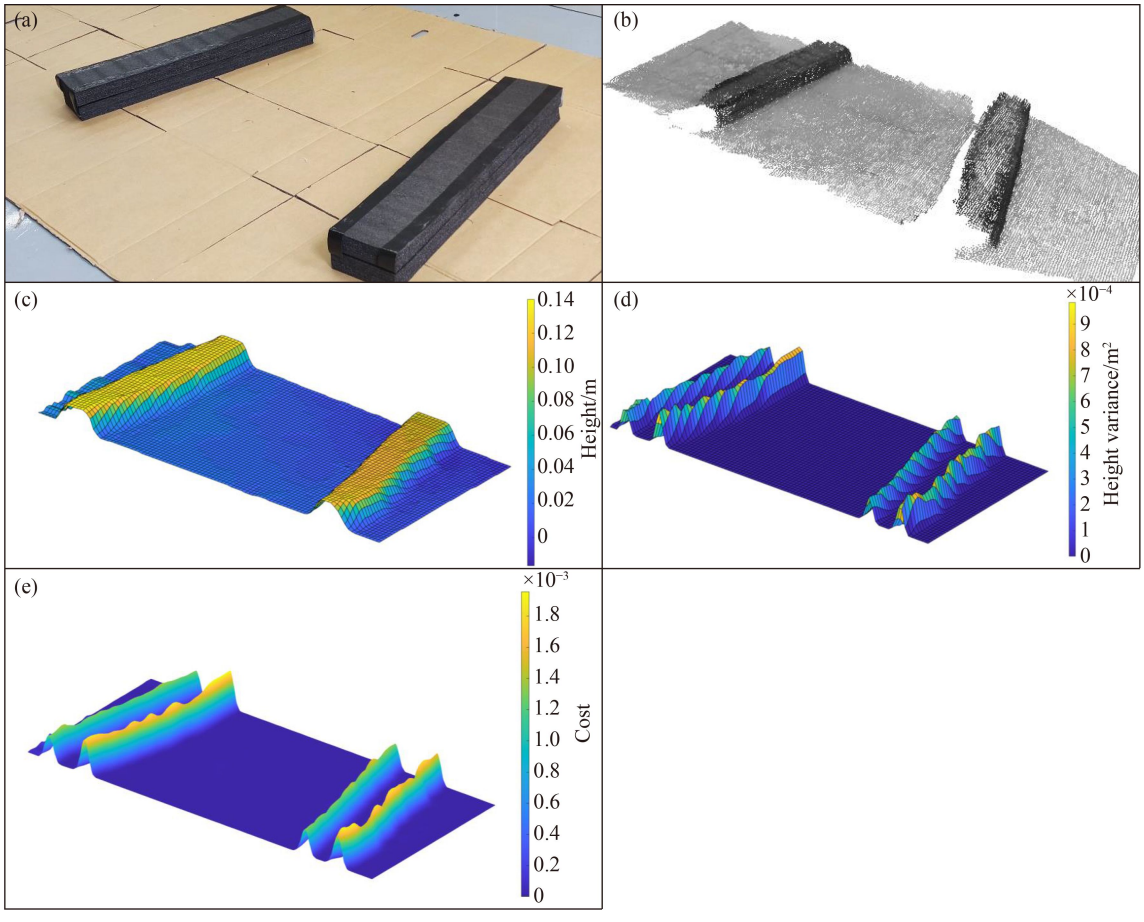

Fig. 9 Experimental environment and the postprocessing: (a) photograph, (b) point cloud, (c) heightmap, (d) height difference map, and (e) costmap.

updated in real time at the frequency of 30 Hz. The heightmap is sampled at once before each step cycle of the robot. After the discrete height difference map is obtained, the continuous costmap does not need to be calculated completely at one time, but is calculated iteratively in accordance with different points $\boldsymbol{p}_{\mathrm{h}}$ in the form of a cost function in the optimization process in the next section.

4 Locomotion optimization

As mentioned earlier, footholds optimization includes two parts: terrain perception and the robot's own state. In the previous section, the cost function based on terrain perception was generated. In consideration of the robot state, this section will build a continuous and derivable objective function to obtain optimized footholds. Afterward, the body trajectory is corrected in accordance with the selected foothold, and the motion control in Section 5 can be performed.

4.1 Foothold preplanning

The robot adopts the stable walking mode of tripod-trot gait. As described in Subsection 2.2, during the alternating swing of the two leg groups, when the ith leg is in the swing phase, according to the heuristic algorithm [2] inspired by the SLIP model, the preplanning of the target foothold is as follows:

$$\begin{aligned}&\boldsymbol{p}_{\mathrm{pre},i}=\boldsymbol{p}_{\mathrm{b}}+\boldsymbol{R}\cdot{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{hip},i}+\frac{T_{\mathrm{st}}}{2}\boldsymbol{v}+(T_{\mathrm{sw}}-t_{\mathrm{sw},i})\boldsymbol{v}_{\mathrm{d}},\\&\boldsymbol{R}=\boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi),\end{aligned}\tag{3}$$

where $\boldsymbol{p}_{\mathrm{pre},i}$ is the preplanning of the target foothold, $\boldsymbol{p}_{\mathrm{b}}$ is the current position of the body, $\boldsymbol{R}\in SO(3)$ is the current attitude matrix, calculated from the Euler angles, roll (ϕ), pitch (θ), and yaw (ψ) measured by the IMU, ${}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{hip},i}$ is the position of the ith leg's hip joint in the BCS, $\boldsymbol{v}$ is the current velocity of the body, $\boldsymbol{v}_{\mathrm{d}}$ is the desired velocity of the body, T_{st} is the duration of the standing phase, T_{sw} is the duration of the swing phase, and $t_{\mathrm{sw},i}$ is the time when the ith leg enters the swing phase.

After the preplanned foothold is obtained, in consideration of the terrain information, calculating an adjustment $\Delta\boldsymbol{p}_i$ relative to the preplanned foothold is necessary. The addition of them is the optimized foothold position.

$$\boldsymbol{p}_i=\boldsymbol{p}_{\mathrm{pre},i}+\Delta\boldsymbol{p}_i.\tag{4}$$

The next subsection will find the optimized foothold $\boldsymbol{p}_i$.

4.2 Footholds optimization

With the tripod-trot gait, there are only three legs in swing phase at every footstep. Therefore, the legs can be divided into two groups (leg groups A and B) in accordance with which are in swing phase, and only one group needs to be optimized at each footstep. Only the horizontal components of the footholds are considered because the height component is directly obtained from the heightmap in accordance with the coordinates of the horizontal components.

The horizontal components of the foothold point $\boldsymbol{p}_i\in\mathbb{R}^3$ are extracted and reorganized into $\boldsymbol{h}_i\in\mathbb{R}^2$ as

$$\begin{cases}\boldsymbol{p}_i=\begin{bmatrix}p_{i,x} & p_{i,y} & p_{i,z}\end{bmatrix}^{\mathrm{T}},\\ \boldsymbol{h}_i=\begin{bmatrix}p_{i,x} & p_{i,y}\end{bmatrix}^{\mathrm{T}}.\end{cases}\tag{5}$$

Then, the grouped footholds are merged into one vector $\boldsymbol{h}$ as

$$\boldsymbol{h}=\begin{cases}\begin{bmatrix}\boldsymbol{h}_0^{\mathrm{T}} & \boldsymbol{h}_2^{\mathrm{T}} & \boldsymbol{h}_4^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}, \text{ leg group A},\\ \begin{bmatrix}\boldsymbol{h}_1^{\mathrm{T}} & \boldsymbol{h}_3^{\mathrm{T}} & \boldsymbol{h}_5^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}, \text{ leg group B}.\end{cases}\tag{6}$$

Similarly, the preplanned foothold $\boldsymbol{p}_{\mathrm{pre},i}$ is extracted as $\boldsymbol{h}_{\mathrm{pre},i}$, and the grouped preplanned footholds are merged into a vector $\boldsymbol{h}_{\mathrm{pre}}$,

$$\boldsymbol{h}_{\mathrm{pre},i}=\begin{bmatrix}p_{\mathrm{pre},i,x} & p_{\mathrm{pre},i,y}\end{bmatrix}^{\mathrm{T}},\tag{7}$$

$$\boldsymbol{h}_{\mathrm{pre}}=\begin{cases}\begin{bmatrix}\boldsymbol{h}_{\mathrm{pre},0}^{\mathrm{T}} & \boldsymbol{h}_{\mathrm{pre},2}^{\mathrm{T}} & \boldsymbol{h}_{\mathrm{pre},4}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}, \text{ leg group A},\\ \begin{bmatrix}\boldsymbol{h}_{\mathrm{pre},1}^{\mathrm{T}} & \boldsymbol{h}_{\mathrm{pre},3}^{\mathrm{T}} & \boldsymbol{h}_{\mathrm{pre},5}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}, \text{ leg group B}.\end{cases}\tag{8}$$

Finally, the footholds optimization equation is established as

$$\begin{aligned}&\boldsymbol{h}^*=\arg\min_{\boldsymbol{h}\in\mathbb{R}^6}\left(\alpha\sum_i J(\boldsymbol{h}_i)+(\boldsymbol{h}-\boldsymbol{h}_{\mathrm{pre}})^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{h}-\boldsymbol{h}_{\mathrm{pre}})\right),\\&i\in\begin{cases}\begin{bmatrix}0 & 2 & 4\end{bmatrix}^{\mathrm{T}}, \text{ leg group A},\\ \begin{bmatrix}1 & 3 & 5\end{bmatrix}^{\mathrm{T}}, \text{ leg group B},\end{cases}\\&\text{s.t.}\quad -\Delta\boldsymbol{h}_{\lim}\leqslant\boldsymbol{h}-\boldsymbol{h}_{\mathrm{pre}}\leqslant\Delta\boldsymbol{h}_{\lim},\ \Delta\boldsymbol{h}_{\lim}>0,\end{aligned}\tag{9}$$

where $\boldsymbol{h}^*$ is optimized horizontal coordinate of the foothold.

The first item of the optimization objective $J(\boldsymbol{h}_i)$ is the terrain cost shown in Section 3, as to keep away from the edge of potholes and obstacles, to prevent footsteps from slipping, and to maintain the predictive balance of the robot. α is the weighting factor of this cost. The second item is the balance cost of the robot state. The weighting matrix $\boldsymbol{W}=\beta\boldsymbol{D}+\gamma\boldsymbol{Q}$ is a symmetric matrix, as shown below, where the diagonal matrix $\boldsymbol{D}$ indicates that the optimized footholds cannot be too far away from the preplanned footholds $\boldsymbol{h}_{\mathrm{pre}}$, and the symmetrical matrix $\boldsymbol{Q}$ indicates that the grouped footholds can be adjusted in the same direction. $\{\beta, \gamma\}$ are their weights. The constraint $\Delta\boldsymbol{h}_{\lim}$ represents the maximum amount of adjustment $\Delta\boldsymbol{p}_i$ allowed to prevent exceeding the leg workspace.

$$\boldsymbol{W}=\beta\underbrace{\begin{bmatrix}1&&&&&\\&1&&&&\\&&1&&&\\&&&1&&\\&&&&1&\\&&&&&1\end{bmatrix}}_{\boldsymbol{D}}+\gamma\underbrace{\begin{bmatrix}2&0&-1&0&-1&0\\0&2&0&-1&0&-1\\-1&0&2&0&-1&0\\0&-1&0&2&0&-1\\-1&0&-1&0&2&0\\0&-1&0&-1&0&2\end{bmatrix}}_{\boldsymbol{Q}}. \tag{10}$$

The designation of the symmetric matrix $\boldsymbol{Q}$ is intuitive. Let $\Delta\boldsymbol{h}=\boldsymbol{h}-\boldsymbol{h}_{\text{pre}}$, which is an expansion of the following quadratic formula:

$$\begin{aligned}\Delta\boldsymbol{h}^{\mathrm{T}}\boldsymbol{Q}\Delta\boldsymbol{h}=&({}_0\Delta\boldsymbol{h}-{}_2\Delta\boldsymbol{h})^2+({}_0\Delta\boldsymbol{h}-{}_4\Delta\boldsymbol{h})^2\\&+({}_2\Delta\boldsymbol{h}-{}_4\Delta\boldsymbol{h})^2+({}_1\Delta\boldsymbol{h}-{}_3\Delta\boldsymbol{h})^2\\&+({}_1\Delta\boldsymbol{h}-{}_5\Delta\boldsymbol{h})^2+({}_3\Delta\boldsymbol{h}-{}_5\Delta\boldsymbol{h})^2.\end{aligned} \tag{11}$$

To distinguish from the right subscript of $\boldsymbol{h}_i$ representing different legs, the left subscript of ${}_n\Delta\boldsymbol{h}$ represents the nth element of $\Delta\boldsymbol{h}$. When it is used as a cost function, it indicates that the adjustments of the footholds of different legs are encouraged to be biased in the same direction, to prevent different legs from moving away from or approaching each other, which would impair the body balance.

In the objective function Eq. (9), the gradient ∇J of the first term $J(\boldsymbol{h}_i)$ is given by Eq. (2). The second term is a quadratic type, whose gradient can be easily obtained. In this way, the objective function can be solved via the Levenberg–Marquardt method [30].

After the optimized horizontal coordinate $\boldsymbol{h}_i^*\in\mathbb{R}^2$ of the foothold is obtained, it is easy to find the height components $\boldsymbol{p}_{i,z}^*$ at the corresponding position on the heightmap. Then, they are combined into a 3D point $\boldsymbol{p}_i^*=\begin{bmatrix}h_{i,x}^* & h_{i,y}^* & p_{i,z}^*\end{bmatrix}^{\mathrm{T}}\in\mathbb{R}^3$, which is the final optimized position of the foothold for i-leg.

4.3 Body trajectory correction

The preplanned footholds are obtained on the basis of the heuristic algorithm, which conform to the basic SLIP balance model. After new optimized footholds are generated, this balance may be broken to some extent. Given that the optimized footholds include the costs of adapting to the terrain, the body trajectory should be corrected in accordance with the optimized footholds.

$$\begin{cases}\Delta\boldsymbol{p}_{\text{b}}^*=\sum_i\left(\boldsymbol{p}_i^*-\boldsymbol{p}_{\text{pre},i}\right),\ i\in\begin{cases}\begin{bmatrix}0&2&4\end{bmatrix}^{\mathrm{T}},\text{ leg group A},\\\begin{bmatrix}1&3&5\end{bmatrix}^{\mathrm{T}},\text{ leg group B},\end{cases}\\\Delta\boldsymbol{v}_{\text{b}}^*=0,\end{cases} \tag{12}$$

where $\Delta\boldsymbol{p}_{\text{b}}^*$ is the correction amount on the original body trajectory. The speed correction $\Delta\boldsymbol{v}_{\text{b}}^*$ is zero. Then, according to the gait cycle, within the current swing phase T_{sw}, which refers to the position and velocity startpoint and endpoint conditions, the body trajectory correction is subjected to cubic interpolation.

To further improve the accuracy, there is a discussion that is it possible to run the footholds optimization repeatedly in accordance with the corrected body trajectory? In fact, it is not necessary. The reason is that, in Eq. (10), the balance cost of the second term, the quadratic matrix $\boldsymbol{Q}$ already contains the inspiration of the grouped legs adjusting to the same direction. In the experiment, we have tried to iterate the optimization many times but found little improvement.

5 Control strategy

Subsection 2.1 described the basic gait scheduler that directs the lift-up and touch-down of each leg. Section 4 introduced the optimization method for the footholds and the correction of the body trajectory. However, these two parts are not enough to control a robot.

For a legged robot, animal-like walking is realized by swinging and touching-down its legs alternately. Therefore, the legs are divided into two groups, namely, swing and support. During the walking process, the swing trajectories of swing legs should be controlled to step on the desired foothold point of each leg. The support legs should also be controlled to maintain body stability and its movement. These actions rely on the cooperation of each joint of the robot, which means that the motors need to provide the corresponding torque. This section will go through the remaining part of the control framework shown in Fig. 3.

5.1 Swing leg control

In the WCS, assuming there is no relative sliding between the supporting legs and the ground, the current tip of the supporting leg is fixed to the ground. The optimal footholds are also calculated in the WCS, which will be the support points for the next gait cycle of each leg. Therefore, when the leg enters the swing phase, the current support point and the optimized foothold must be connected through a smooth curve in combination with the gait scheduler. This curve is the leg swing trajectory.

The current tip position of the leg is obtained by forward kinematics and state estimation [31]. This study

adopts a spatial composite trajectory based on cycloid. The expression is as follows:

$$\boldsymbol{p}_{\xi_i}(t_{\mathrm{sw},i}) = \begin{bmatrix} p_{x,\xi_i} \\ p_{y,\xi_i} \\ p_{z,\xi_i} \end{bmatrix} = \frac{\boldsymbol{p}_i^* - \boldsymbol{p}_{\xi_i=0}}{2\pi}(\xi_i - \sin(\xi_i)) + \begin{bmatrix} 0 \\ 0 \\ h_z \\ \hline 2 \end{bmatrix}(1-\cos(\xi_i)),$$
$$\xi_i = 2\pi\frac{t_{\mathrm{sw},i}}{T_{\mathrm{sw}}} \in [0,\ 2\pi]. \tag{13}$$

The trajectory $\boldsymbol{p}_{\xi_i} \in \mathbb{R}^3$ is a function of the ith leg's swing time $t_{\mathrm{sw},i}$, with ξ_i for the swing phase. $\boldsymbol{p}_i^*$ is the target foothold optimized in Section 4, $\boldsymbol{p}_{\xi_i=0}$ is the tip position before leg lifting (when $t_{\mathrm{sw},i} = 0$), h_z is the step height that needs to be raised in the middle of the swing process. This trajectory is continuous and derivable, and the derivative of the trajectory is the velocity of the tip. Figure 10 shows a step-walk sequence. The red solid line represents the swing leg, and the blue solid line is the support leg. The red dotted line is the swing trajectory, and the yellow dotted line is the swing trajectory when the height component of $\boldsymbol{p}_i^*$ is equal to 0.

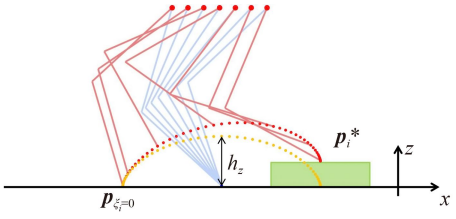

Fig. 10 Swing trajectory based on cycloid spatial composite trajectory.

After the swing trajectory is determined, the target position of the tip in the LCS ${}^{\mathrm{L}}\boldsymbol{l}_i^*$ is obtained as follows:

$${}^{\mathrm{L}}\boldsymbol{l}_i^* = \boldsymbol{R}^{\mathrm{T}}\left(\boldsymbol{p}_{\xi_i} - \boldsymbol{p}_{\mathrm{b}}\right) - {}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{hip},i}. \tag{14}$$

Then, the virtual force ${}^{\mathrm{L}}\boldsymbol{f}_{\mathrm{sw},i}^*$ of the swing leg in LCS is generated by tracking the swing trajectory through the virtual model impedance control. The virtual model is shown in Fig. 11.

$${}^{\mathrm{L}}\boldsymbol{f}_{\mathrm{sw},i}^* = {}^{\mathrm{L}}\boldsymbol{K}_{\mathrm{P}}\left({}^{\mathrm{L}}\boldsymbol{l}_i^* - {}^{\mathrm{L}}\boldsymbol{l}_i\right) + {}^{\mathrm{L}}\boldsymbol{K}_{\mathrm{D}}\left({}^{\mathrm{L}}\dot{\boldsymbol{l}}_i^* - {}^{\mathrm{L}}\dot{\boldsymbol{l}}_i\right). \tag{15}$$

The inertia term is not included in the impedance model formulation because we assume that the leg has a negligibly small mass. The notations with a subscript *, i.e., ${}^{\mathrm{L}}\boldsymbol{l}_i^*$ and its derivative ${}^{\mathrm{L}}\dot{\boldsymbol{l}}_i^*$, are the target position and velocity of the ith leg in the LCS, respectively, ${}^{\mathrm{L}}\boldsymbol{l}_i$ and its derivative ${}^{\mathrm{L}}\dot{\boldsymbol{l}}_i$ are the position and velocity of the ith leg obtained by the forward kinematics, respectively. Diagonal matrices $\{{}^{\mathrm{L}}\boldsymbol{K}_{\mathrm{P}}, {}^{\mathrm{L}}\boldsymbol{K}_{\mathrm{D}}\} \in \mathbb{R}^{3\times3}$ represent the three-dimensional stiffness and damping of the leg, respectively. Through this virtual model, the feedforward force ${}^{\mathrm{L}}\boldsymbol{f}_{\mathrm{sw},i}^*$ of the swing leg can be obtained.

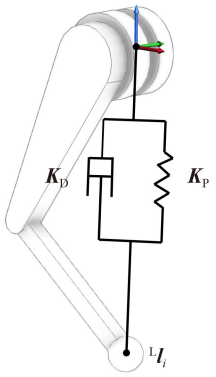

Fig. 11 Leg virtual model of impedance control.

5.2 Body balance control

A CBC is used to generate the virtual force (or called ground reaction force, GRF [2,32]) of the supporting legs. The states that need to be controlled for body balance include the desired position $\boldsymbol{p}_{\mathrm{b}}^* \in \mathbb{R}^3$ and velocity $\dot{\boldsymbol{p}}_{\mathrm{b}}^* \in \mathbb{R}^3$ of the center of mass (CoM), along with the attitude $\boldsymbol{R}^* \in SO(3)$ and angular velocity $\boldsymbol{\omega}_{\mathrm{b}}^* \in \mathbb{R}^3$ of the body. The feedback control regulation can be described by the following formula:

$$\ddot{\boldsymbol{p}}_{\mathrm{b}}^* = \boldsymbol{K}_{\mathrm{P,p}}\left(\boldsymbol{p}_{\mathrm{b}}^* - \hat{\boldsymbol{p}}_{\mathrm{b}}\right) + \boldsymbol{K}_{\mathrm{D,p}}\left(\dot{\boldsymbol{p}}_{\mathrm{b}}^* - \hat{\dot{\boldsymbol{p}}}_{\mathrm{b}}\right),$$
$$\dot{\boldsymbol{\omega}}_{\mathrm{b}}^* = \boldsymbol{K}_{\mathrm{P,a}}\left(\log\left(\boldsymbol{R}^*\hat{\boldsymbol{R}}^{\mathrm{T}}\right)\right) + \boldsymbol{K}_{\mathrm{D,a}}\left(\boldsymbol{\omega}_{\mathrm{b}}^* - \hat{\boldsymbol{\omega}}_{\mathrm{b}}\right), \tag{16}$$

where $\left\{\hat{\boldsymbol{p}}_{\mathrm{b}}, \hat{\dot{\boldsymbol{p}}}_{\mathrm{b}}, \hat{\boldsymbol{\omega}}_{\mathrm{b}}\right\} \in \mathbb{R}^3$ and $\hat{\boldsymbol{R}} \in SO(3)$ are the estimated states of the current position, velocity, angular velocity, and attitude of the robot. The diagonal matrices $\{\boldsymbol{K}_{\mathrm{P,p}}, \boldsymbol{K}_{\mathrm{D,p}}, \boldsymbol{K}_{\mathrm{P,a}}, \boldsymbol{K}_{\mathrm{D,a}}\} \in \mathbb{R}^{3\times3}$ are the stiffness (subscript with P) and damping (subscript with D) of position (subscript with ,p) and attitude (subscript with ,a). log(*) is the mapping from a rotation matrix ($SO(3)$) to a rotation vector ($\mathbb{R}^3$). After the above PD control regulation, the control values of the CoM's acceleration $\ddot{\boldsymbol{p}}_{\mathrm{b}}^* \in \mathbb{R}^3$ and angular acceleration $\dot{\boldsymbol{\omega}}_{\mathrm{b}}^* \in \mathbb{R}^3$ are obtained.

The position acceleration and angular acceleration need to be provided by the GRFs of the support legs. Here, the mass of legs is assumed to be negligible compared with the mass of body, and the body mass is concentrated on the CoM, ignoring the influence of the Coriolis force and centrifugal force. Then, the simplified dynamics of the robot satisfies

$$\begin{cases} m(\ddot{\boldsymbol{p}}_{\mathrm{b}}^* + \boldsymbol{g}) = \sum_1^{\mu} \boldsymbol{f}_i, \\ \hat{\boldsymbol{R}} \cdot {}^{\mathrm{B}}\boldsymbol{I} \cdot \hat{\boldsymbol{R}}^{\mathrm{T}}\dot{\boldsymbol{\omega}}_{\mathrm{b}}^* \approx \sum_1^{\mu} \hat{\boldsymbol{R}} \cdot {}^{\mathrm{B}}\boldsymbol{l}_6 \times \boldsymbol{f}_i, \end{cases} \tag{17}$$

where m is the mass of CoM, $\boldsymbol{g} \in \mathbb{R}^3$ is the local gravitational acceleration, and ${}^{\mathrm{B}}\boldsymbol{I} \in \mathbb{R}^{3\times3}$ is the inertia matrix of the body in the BCS. μ means the leg number in contact, ${}^{\mathrm{B}}\boldsymbol{l}_i$ is the tip position in the BCS, and $\boldsymbol{f}_i$ is the

GRF of the ith leg. Equation (17) is rewritten in matrix form as

$$\underbrace{\begin{bmatrix} \boldsymbol{E} & \cdots & \boldsymbol{E} \\ \left[\hat{\boldsymbol{R}}\cdot{}^{\mathrm{B}}\boldsymbol{l}_1\right]\times & \cdots & \left[\hat{\boldsymbol{R}}\cdot{}^{\mathrm{B}}\boldsymbol{l}_6\right]\times \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_6 \end{bmatrix}}_{\boldsymbol{f}} = \underbrace{\begin{bmatrix} m(\ddot{\boldsymbol{p}}_{\mathrm{b}}^{*}+\boldsymbol{g}) \\ \hat{\boldsymbol{R}}\cdot{}^{\mathrm{B}}\boldsymbol{I}\cdot\hat{\boldsymbol{R}}^{\mathrm{T}}\dot{\boldsymbol{\omega}}_{\mathrm{b}}^{*} \end{bmatrix}}_{\boldsymbol{b}}, \tag{18}$$

where $\boldsymbol{E}\in\mathbb{R}^{3\times3}$ is the identity matrix. The symbol $[*]\times$ maps from a vector ($\mathbb{R}^3$) to an antisymmetric matrix ($\mathbb{R}^{3\times3}$). $\boldsymbol{A}$, $\boldsymbol{f}$, and $\boldsymbol{b}$ are the merged matrix and vectors defined in Eq. (18). The legs in the support phase form an over-restraint system for the body. Therefore, the virtual force of each support leg should be obtained through the following optimization equation:

$$\begin{aligned} \boldsymbol{f}_{\mathrm{st}}^{*} &= \arg\min_{\boldsymbol{f}\in\mathbb{R}^{18}}\left((\boldsymbol{A}\boldsymbol{f}-\boldsymbol{b})^{\mathrm{T}}\boldsymbol{S}(\boldsymbol{A}\boldsymbol{f}-\boldsymbol{b})+\delta\boldsymbol{f}^{2}\right), \\ &\text{s.t.}\quad \boldsymbol{C}\boldsymbol{f}<\boldsymbol{f}_{\mathrm{lim}}. \end{aligned} \tag{19}$$

The diagonal matrix $\boldsymbol{S}\in\mathbb{R}^{18\times18}$ is the selection matrix, and its elements will be zeros if the corresponding leg is in the swing phase, δ is the weighting factor for CBC, Matrix $\boldsymbol{C}$ and vector $\boldsymbol{f}_{\mathrm{lim}}$ ensure that the optimized GRFs satisfy the friction cone and upper and lower bound constraints, $\boldsymbol{f}_{\mathrm{st}}^{*}$ is the virtual force of the supporting legs and $\boldsymbol{f}_{\mathrm{st},i}^{*}$ is the virtual force of the supporting legs. The virtual force of the supporting leg can be optimized by the balance optimizer model above:

$$\boldsymbol{f}_{\mathrm{st}}^{*}=\begin{bmatrix}\boldsymbol{f}_{\mathrm{st},1}^{*} & \cdots & \boldsymbol{f}_{\mathrm{st},6}^{*}\end{bmatrix}^{\mathrm{T}}. \tag{20}$$

Then, the feedforward force ${}^{\mathrm{L}}\boldsymbol{f}_{\mathrm{st,i}}^{*}$ in the LCS is obtained as follows:

$${}^{\mathrm{L}}\boldsymbol{f}_{\mathrm{st},i}^{*}=\hat{\boldsymbol{R}}^{\mathrm{T}}\boldsymbol{f}_{\mathrm{st},i}^{*}. \tag{21}$$

5.3 Joint PD control

The virtual force obtained above is transformed into joint space as feedforward torque, and the final control torque of the actuator is generated through joint feedback control.

$${}^{\mathrm{L}}\boldsymbol{f}_{i}^{*}=\begin{cases}{}^{\mathrm{L}}\boldsymbol{f}_{\mathrm{sw},i}^{*}, & \text{swing leg},\\ {}^{\mathrm{L}}\boldsymbol{f}_{\mathrm{st},i}^{*}, & \text{support leg},\end{cases} \tag{22}$$

$$\boldsymbol{\tau}_{i}^{*}=\boldsymbol{J}_{i}^{\mathrm{T}}\cdot{}^{\mathrm{L}}\boldsymbol{f}_{i}^{*}+\boldsymbol{K}_{\mathrm{P,j}}(\boldsymbol{q}_{i}^{*}-\boldsymbol{q}_{i})+\boldsymbol{K}_{\mathrm{D,j}}(\dot{\boldsymbol{q}}_{i}^{*}-\dot{\boldsymbol{q}}_{i}), \tag{23}$$

where ${}^{\mathrm{L}}\boldsymbol{f}_{i}^{*}$ is the virtual feedforward force of the ith leg in the LCS, $\boldsymbol{J}_i\in\mathbb{R}^{3\times3}$ is the Jacobian matrix of the ith leg, and the diagonal matrices $\{\boldsymbol{K}_{\mathrm{P,j}},\boldsymbol{K}_{\mathrm{D,j}}\}\in\mathbb{R}^{3\times3}$ are the joint stiffness and damping coefficients of each leg. $\{\boldsymbol{q}_i^{*},\dot{\boldsymbol{q}}_i^{*}\}\in\mathbb{R}^3$ are the target angle and angular velocity for each joint of the leg, respectively, which are obtained by the inverse kinematic solution. $\{\boldsymbol{q}_i,\dot{\boldsymbol{q}}_i\}\in\mathbb{R}^3$ are the current angle and angular velocity read by the encoder installed on each joint. Finally, the control torque $\boldsymbol{\tau}_i^{*}$ for the joint of each leg is obtained and sent to the motor driver.

So far, the entire robot control framework is accomplished.

6 Contrast experiments and analysis

The approach mentioned above is tested on a hexapod robot, LittleStrong. We conduct experiments on a terrain composed of irregularly placed strips. As shown in Fig. 12, each strip is about 0.2 m wide and 0.12 m high. To compare our foothold optimization algorithm, we set up a control group, comparing the results of the hexapod passing through the terrain without and with using the foothold optimization method proposed in Subsection 6.1. The comparison of the experiments is analyzed in Subsection 6.2.

6.1 Contrast experiments

(1) Without the presented foothold optimization algorithm

First, the robot walks blindly, adopting a heuristic method with fixed gait cycle and step length. In the experiment, the robot slips when stepping on the edge of the strip. The process is demonstrated in the attached video; a screenshot of the process is shown in Fig. 13.

When there is a sliding step, the robot exhibits

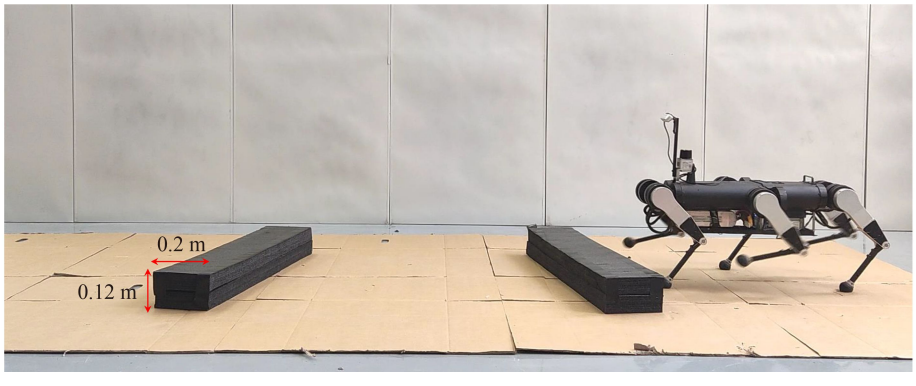

Fig. 12 Experiment setup for footholds optimization with the hexapod robot.

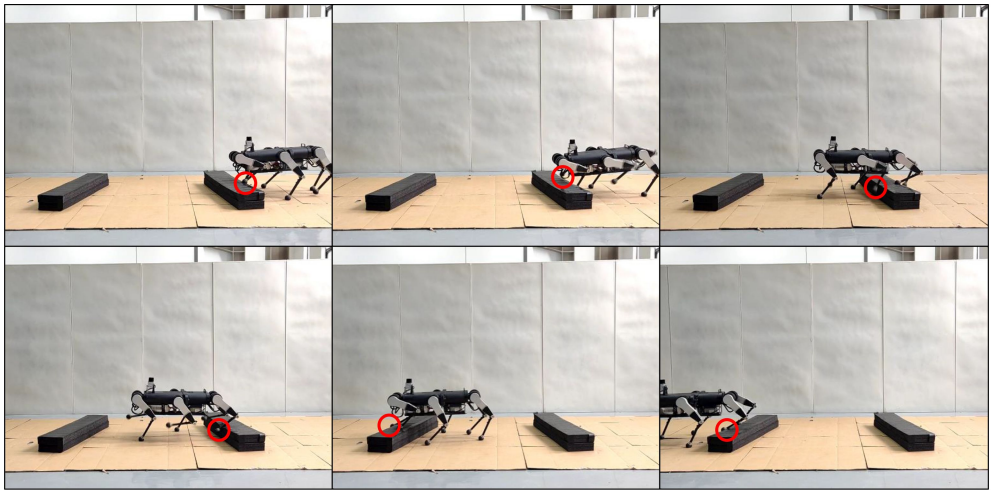

Fig. 13 Screenshots of blind walking. Slips are circled in red.

instantaneous shaking. Figure 14 shows the graph of the Euler angles of robot attitude, which indicates that when slips occur, the robot loses its balance, and the Euler angles have large peaks. Hence, in the case of irregular terrain, the balance of the robot, when facing obstacles, is difficult to ensure when a blind walking gait is adopted.

(2) With the presented footholds optimization algorithm

After our online foothold optimization method is deployed, experiments are carried out under the same scenario mentioned above. The process is also shown in the attached video; Fig. 15 shows the screenshots of it. The robot adopts a variable step length, and each step is stepped on a position that is relatively safe from the edge of the strip, and no sliding occurs. Figure 16 shows the point cloud data processing, and the outstanding points in it are the swing leg tips before touching down.

Figure 17 depicts the footholds optimization results from the corresponding experiment. The six thinner curves are the step length of each leg. In the tripod-trot gait, the back-right, middle-left, and front-right legs are in one group; the back-left, front-left, and middle-right legs are in another group. Accordingly, the bold blue curve is the average step length of the leg group coming into the standing phase. From the attached video, a smaller step length is carried out before the strip to avoid stepping on the edge of it. Meanwhile, a larger step length is used when leaving the strip.

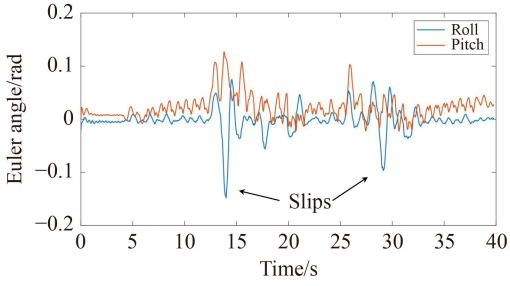

Fig. 14 Euler angles during blind walk.

With the elimination of the interference of sliding and

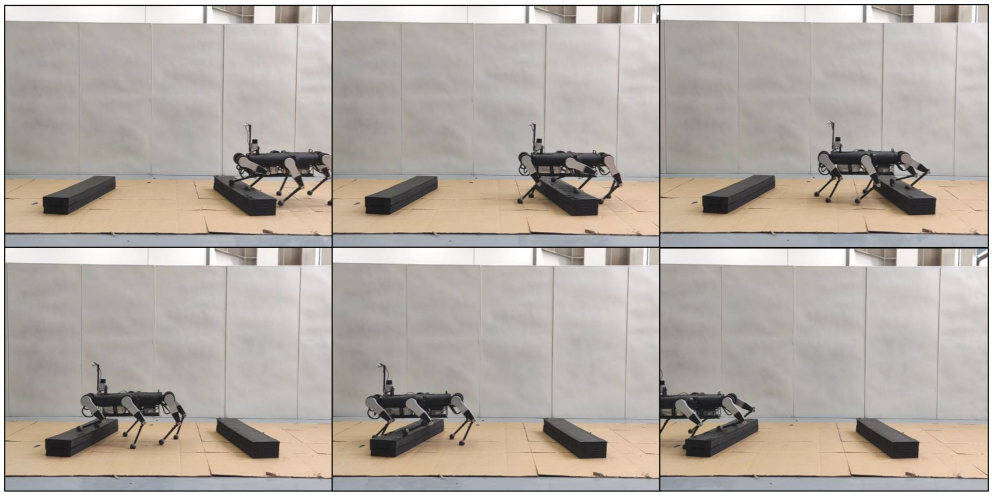

Fig. 15 Screenshots of footholds optimization.

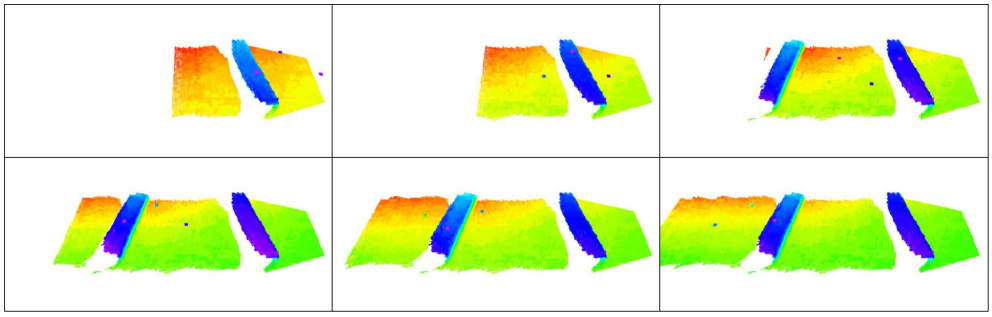

Fig. 16 Screenshots of point cloud processing.

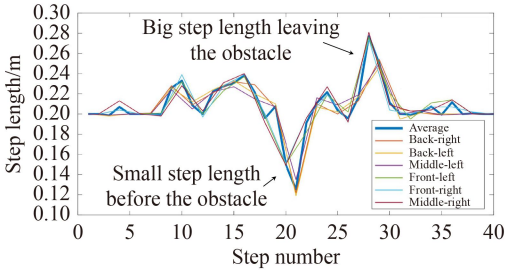

Fig. 17 Step length with footholds optimization.

falling under the feet, the balance of the robot has been greatly improved. As shown in Fig. 18, compared with Fig. 14, the Euler angles are well smoothed, and no peaks occur.

6.2 Analysis of the contrast experiments

We have conducted 10 comparative experiments, as shown above. From Table 1, when the blind walking algorithm is used, the foot slides for a total of 47 times, and the Euler angle fluctuation variance of the body is 5.71×10^{-4} rad^2 for roll and 5.40×10^{-4} rad^2 for pitch. After the foothold optimization algorithm is applied, sliding occurs 2 times (caused by the failure of point cloud splicing in one experiment, which is beyond the discussion of this paper). The fluctuation variance of the body Euler angle is reduced to 2.72×10^{-4} rad^2 for roll and 3.21×10^{-4} rad^2 for pitch, indicating that our algorithm significantly improves the stability of the robot.

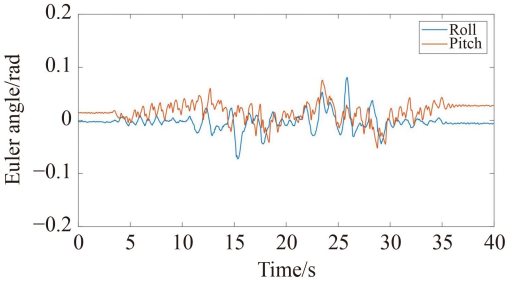

Fig. 18 Euler angles when using foothold optimization.

Table 1 Contrast between blind walking and footholds optimized walking

Experiment	Slip number	Variance of Euler angle/(10^{-4} rad^2)	
		Roll	Pitch
Blind walking	47	5.71	5.40
Footholds optimized	2	2.72	3.21

7 Conclusions

In this paper, we propose a novel smooth and derivable method for generating a costmap from a terrain grid map. Moreover, we propose an algorithm for online optimization of footholds, combined with the robot's heuristic gait. The algorithm is fast and stable, and has a clear physical meaning and optimization objective, which can be directly deployed on the onboard computer. We verify this method on a hexapod robot. On an irregular terrain, the robot using blind walking will stumble. Now, with the proposed algorithm, combined with real-time visual information, the robot can pass through the terrain gracefully. The real-time property and feasibility of the algorithm are proved.

Nomenclature

Abbreviations

BCS	Body coordinate system
CBC	Centroid balance control
CI	Convolution interpolation
CNN	Convolutional neural network
CoM	Center of mass
GRF	Ground reaction force
IMU	Inertial measurement unit
LCS	Leg coordinate system
NDCI	Normalized derivable convolution interpolation
PCI	Pyramid convolution interpolation
RGBD	Red, green, blue, depth

SLIP	Spring-loaded inverted pendulum
WCS	World coordinate system

Variables

$\boldsymbol{A}$	Merged matrix for CBC
$\boldsymbol{b}, \boldsymbol{f}$	Merged vectors for CBC
$\boldsymbol{c}_{i,j}$	Center of the pixel (i, j) on the map
$\boldsymbol{C}$	Matrix of friction cone and upper and lower bound constraints for GRF
$\boldsymbol{E}$	Identity matrix
$\boldsymbol{f}_i$	GRF of the ith leg
${}^{\mathrm{L}}\boldsymbol{f}_i^*$	Virtual feedforward force of the ith leg in the LCS
$\boldsymbol{f}_{\mathrm{lim}}$	Friction cone and upper and lower bound constraints for GRF
$\boldsymbol{f}_{\mathrm{st}}^*$	Virtual force of the supporting legs
$\boldsymbol{f}_{\mathrm{st},i}^*$	Virtual force of the ith supporting leg
${}^{\mathrm{L}}\boldsymbol{f}_{\mathrm{st},i}^*$, ${}^{\mathrm{L}}\boldsymbol{f}_{\mathrm{sw},i}^*$	Virtual feedforward forces of the ith supporting leg and swing leg in the LCS, respectivley
$\boldsymbol{g}$	Local gravitational acceleration
h_z	Step height that needs to be raised in the middle of the swing process
$\boldsymbol{h}$	Merged vector of $\boldsymbol{h}_i$
$\boldsymbol{h}^*$	Horizontal component of optimized footholds
$\boldsymbol{h}_i$	Horizontal component of $\boldsymbol{p}_i$
$\boldsymbol{h}_i^*$	Optimized horizontal coordinate of the foothold
$\boldsymbol{h}_{\mathrm{pre}}$	Merged vector of $\boldsymbol{h}_{\mathrm{pre},i}$
$\boldsymbol{h}_{\mathrm{pre},i}$	Horizontal component of preplanned foothold
$\Delta\boldsymbol{h}$	Intermediate variable of the difference between the optimized footholds and the preplanned footholds
$\Delta\boldsymbol{h}_{\mathrm{lim}}$	Maximum amount of adjustment $\Delta\boldsymbol{p}_i$ allowed
${}_n\Delta\boldsymbol{h}$	nth element of $\Delta\boldsymbol{h}$
i	Index for the item, such as the ith leg, or the (i, j) pixel
${}^{\mathrm{B}}\boldsymbol{I}$	Inertia matrix of the body in the BCS
j	Index for the item, such as the (i, j) pixel
$\boldsymbol{J}_i$	Jacobian matrix of the ith leg
$J(\boldsymbol{h}_i)$	Terrain cost at the position of $\boldsymbol{h}_i$
$J(\boldsymbol{p}_{\mathrm{h}})$	Cost function for NDCI
$\nabla J(\boldsymbol{p}_{\mathrm{h}})$	Partial derivative of the cost function for NDCI
$\boldsymbol{K}_{\mathrm{D,j}}$, $\boldsymbol{K}_{\mathrm{P,j}}$	Joint damping and stiffness matrices, respectively
$\boldsymbol{K}_{\mathrm{D,p}}$, $\boldsymbol{K}_{\mathrm{P,p}}$	Damping and stiffness of body position, respectively
$\boldsymbol{K}_{\mathrm{D,a}}$, $\boldsymbol{K}_{\mathrm{P,a}}$	Damping and stiffness of body attitude, respectively
${}^{\mathrm{L}}\boldsymbol{K}_{\mathrm{D}}$, ${}^{\mathrm{L}}\boldsymbol{K}_{\mathrm{P}}$	Damping and stiffness of the leg, respectively
${}^{\mathrm{B}}\boldsymbol{l}_i$	Tip position in the BCS
${}^{\mathrm{L}}\boldsymbol{l}_i$, ${}^{\mathrm{L}}\dot{\boldsymbol{l}}_i$	Position and velocity of the ith leg in the LCS, respectively
${}^{\mathrm{L}}\boldsymbol{l}_i^*$, ${}^{\mathrm{L}}\dot{\boldsymbol{l}}_i^*$	Target position and velocity of the ith leg in the LCS, respectively
m, n	Set numbers of pixels that need to be involved
$\boldsymbol{p}_{\mathrm{b}}$	Position of the robot body
$\boldsymbol{p}_{\mathrm{b}}^*$, $\dot{\boldsymbol{p}}_{\mathrm{b}}^*$, $\ddot{\boldsymbol{p}}_{\mathrm{b}}^*$	Target position, velocity, and acceleration of the robot's CoM, respectively
$\boldsymbol{p}_{\mathrm{h}}$	Horizontal coordinate of the position of the selected point on the whole map
$\boldsymbol{p}_i$	Foothold point of the ith leg
$p_{i,x}$, $p_{i,y}$, $p_{i,z}$	x, y, and z components of the foothold $\boldsymbol{p}_i$, respectively
$\boldsymbol{p}_i^*$	Optimized footholds
$\boldsymbol{p}_{i,z}^*$	Height components at the position $\boldsymbol{p}_i^*$ on the heightmap
$\boldsymbol{p}_{\mathrm{pre},i}$	Preplanning of the target foothold
$\boldsymbol{p}_{\xi_i}$	Swing trajectory of the ith leg
$\hat{\boldsymbol{p}}_{\mathrm{b}}$, $\hat{\dot{\boldsymbol{p}}}_{\mathrm{b}}$	Estimated position and velocity of the robot, respectively
$\Delta\boldsymbol{p}_{\mathrm{b}}^*$	Correction amount on the original body trajectory
$\Delta\boldsymbol{p}_i$	Foothold adjustment for the ith leg
${}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{hip},i}$	Position of the ith leg's hip joint in the BCS
$\boldsymbol{q}_i$, $\dot{\boldsymbol{q}}_i$	Current angle and angular velocity read by the ith encoder, respectively
$\boldsymbol{q}_i^*$, $\dot{\boldsymbol{q}}_i^*$	Target angle and angular velocity of the ith joint, respectively
$\boldsymbol{R}$	Current attitude matrix
$\boldsymbol{R}^*$	Desired attitude of the body
$\hat{\boldsymbol{R}}$	Estimated current attitude of the robot
$\mathbb{R}$	Set of real number
$\boldsymbol{S}$	Selection matrix for CBC
$t_{\mathrm{sw},i}$	Time when the ith leg enters the swing phase
T_{st}, T_{sw}	Duration of the standing and swing phase, respectively
$\boldsymbol{v}$	Velocity of the body
$\boldsymbol{v}_{\mathrm{d}}$	Desired velocity of the body
$\Delta\boldsymbol{v}_{\mathrm{b}}^*$	Speed correction
x_{h}	x component of the horizontal coordinate of the position of the selected point on the heightmap
$x_{i,j}$	x component of the center of the pixel (i, j) on the map
y_{h}	y component of the horizontal coordinate of the position of the selected point on the heightmap
$y_{i,j}$	y component of the center of the pixel (i, j) on the map
$\boldsymbol{W}$, $\boldsymbol{D}$, $\boldsymbol{Q}$	Weighting matrices for footholds optimization
$\alpha, \beta, \gamma,$	Weighting factors for footholds optimization
δ	Weighting factor for CBC
σ_x, σ_y	Smoothing factor in x and y directions, respectively
ϕ, θ, ψ	Euler angle roll, pitch, and yaw measured by the IMU
μ	Number of legs in contact
$\boldsymbol{\tau}_i^*$	Control torque of the ith joint
ξ_i	Swing phase
$\hat{\boldsymbol{\omega}}_{\mathrm{b}}$	Angular velocity of the body
$\boldsymbol{\omega}_{\mathrm{b}}^*$, $\dot{\boldsymbol{\omega}}_{\mathrm{b}}^*$	Target angular velocity and acceleration of the robot, respectively

* Symbol maps from a vector ($\mathbb{R}^3$) to an antisymmetric matrix ($\mathbb{R}^{3\times3}$)

Acknowledgements This work was supported by the National Key R&D Program of China (Grant No. 2021YFF0306202).

References

1. Raibert M H, Tello E R. Legged robots that balance. IEEE Expert, 1986, 1(4): 89
2. Bledt G, Powell M J, Katz B, Di Carlo J, Wensing P M, Kim S. MIT Cheetah 3: design and control of a robust, dynamic quadruped robot. In: Proceedings of 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. Madrid: IEEE, 2018
3. Vernaza P, Likhachev M, Bhattacharya S, Chitta S, Kushleyev A, Lee D D. Search-based planning for a legged robot over rough terrain. In: Proceedings of 2009 IEEE International Conference on Robotics & Automation. Kobe: IEEE, 2009, 2380–2387
4. Kalakrishnan M, Buchli J, Pastor P, Mistry M, Schaal S. Fast, robust quadruped locomotion over challenging terrain. In: Proceedings of 2010 IEEE International Conference on Robotics & Automation. Anchorage: IEEE, 2010, 2665–2670
5. Semini C, Tsagarakis N G, Guglielmino E, Focchi M, Cannella F, Caldwell D G. Design of HyQ—a hydraulically and electrically actuated quadruped robot. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 2011, 225(6): 831–849
6. Winkler A, Havoutis I, Bazeille S, Ortiz J, Focchi M, Dillmann R, Caldwell D, Semini C. Path planning with force-based foothold adaptation and virtual model control for torque controlled quadruped robots. In: Proceedings of 2014 IEEE International Conference on Robotics & Automation (ICRA). Hong Kong: IEEE, 2014, 6476–6482
7. Hutter M, Gehring C, Jud D, Lauber A, Bellicoso C D, Tsounis V, Hwangbo J, Bodie K, Fankhauser P, Bloesch M, Diethelm R, Bachmann S, Melzer A, Hoepflinger M. ANYmal—a highly mobile and dynamic quadrupedal robot. In: Proceedings of 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Daejeon: IEEE, 2016, 38–44
8. Winkler A W, Bellicoso C D, Hutter M, Buchli J. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. IEEE Robotics and Automation Letters, 2018, 3(3): 1560–1567
9. Mastalli C, Focchi M, Havoutis I, Radulescu A, Calinon S, Buchli J, Caldwell D G, Semini C. Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion. In: Proceedings of 2017 IEEE International Conference on Robotics & Automation (ICRA). Singapore: IEEE, 2017, 1096–1103
10. Mastalli C, Havoutis I, Focchi M, Caldwell D G, Semini C. Motion planning for quadrupedal locomotion: coupled planning, terrain mapping and whole-body control. IEEE Transactions on Robotics, 2020, 36(6): 1635–1648
11. Mastalli C, Havoutis I, Winkler A W, Caldwell D G, Semini C. On-line and on-board planning and perception for quadrupedal locomotion. In: Proceedings of 2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA). Woburn: IEEE, 2019, 1–7
12. Fankhauser P, Bjelonic M, Bellicoso C D, Miki T, Hutter M. Robust rough-terrain locomotion with a quadrupedal robot. In: Proceedings of 2018 IEEE International Conference on Robotics and Automation (ICRA). Brisbane: IEEE, 2018, 5761–5768
13. Jenelten F, Miki T, Vijayan A E, Bjelonic M, Hutter M. Perceptive locomotion in rough terrain—online foothold optimization. IEEE Robotics and Automation Letters, 2020, 5(4): 5370–5376
14. Chai X, Gao F, Xu Y L. Perception-based gait planning for a hexapod robot walking on typical structured terrain. In: Zhang X M, Wang N F, Huang Y J, eds. Mechanism and Machine Science. Singapore: Springer, 2017, 169–181
15. Kim D, Carballo D, Di Carlo J, Katz B, Bledt G, Lim B, Kim S. Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot. In: Proceedings of 2020 IEEE International Conference on Robotics and Automation (ICRA). Paris: IEEE, 2020, 2464–2470
16. Mao L H, Tian Y, Gao F, Zhao Y. Novel method of gait switching in six-legged robot walking on continuous-nondifferentiable terrain by utilizing stability and interference criteria. Science China Technological Sciences, 2020, 63(12): 2527–2540
17. Mao L H, Gao F, Tian Y, Zhao Y. Novel method for preventing shin-collisions in six-legged robots by utilising a robot–terrain interference model. Mechanism and Machine Theory, 2020, 151: 103897
18. Zhao Y, Gao F, Yin Y P. Obstacle avoidance and terrain identification for a hexapod robot. Research Square, 2020, preprint
19. Magaña O A V, Barasuol V, Camurri M, Franceschi L, Focchi M, Pontil M, Caldwell D G, Semini C. Fast and continuous foothold adaptation for dynamic locomotion through CNNs. IEEE Robotics and Automation Letters, 2019, 4(2): 2140–2147
20. Miki T, Lee J, Hwangbo J, Wellhausen L, Koltun V, Hutter M. Learning robust perceptive locomotion for quadrupedal robots in the wild. Science Robotics, 2022, 7(62): eabk2822
21. Tsounis V, Alge M, Lee J, Farshidian F, Hutter M. DeepGait: planning and control of quadrupedal gaits using deep reinforcement learning. IEEE Robotics and Automation Letters, 2020, 5(2): 3699–3706
22. Gehring C, Fankhauser P, Isler L, Diethelm R, Bachmann S, Potz M, Gerstenberg L, Hutter M. ANYmal in the field: solving industrial inspection of an offshore HVDC platform with a quadrupedal robot. In: Ishigami G, Yoshida K, eds. Field and Service Robotics. Singapore: Springer, 2021, 247–260
23. Liu W, Gao Y, Gao F, Li S Y. Trajectory adaptation and safety control via control barrier functions for legged robots. In: Proceedings of 2021 China Automation Congress (CAC). Beijing: IEEE, 2021, 5571–5576
24. Park H W, Wensing P M, Kim S. Jumping over obstacles with MIT Cheetah 2. Robotics and Autonomous Systems, 2021, 136: 103703
25. Qi S H, Lin W C, Hong Z J, Chen H, Zhang W. Perceptive autonomous stair climbing for quadrupedal robots. In: Proceedings of 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Prague: IEEE, 2021, 2313–2320
26. Gerum P. Xenomai—Implementing a RTOS emulation framework

on GNU/Linux. White Paper, 2004, 81
27. Hintjens P. ZeroMQ: Messaging for Many Applications. O'Reilly Media, 2013
28. Sun Q, Gao F, Chen X B. Towards dynamic alternating tripod trotting of a pony-sized hexapod robot for disaster rescuing based on multi-modal impedance control. Robotica, 2018, 36(7): 1048–1076
29. Ekstrom M P. Digital Image Processing Techniques. 2nd ed. Academic Press, 2012
30. Ranganathan, A. The Levenberg–Marquardt algorithm. Tutoral on LM Algorithm, 2004, 11(1): 101–110
31. Blösch M, Hutter M, Höpflinger M A, Leutenegger S, Gehring C, Remy C D, Siegwart R Y. State estimation for legged robots—consistent fusion of leg kinematics and IMU. In: Proceedings of Robotics: Science and Systems Conference (RSS 2012). Sidney: Robotics: Science and Systems Conference, 2012
32. Focchi M, Del Prete A, Havoutis I, Featherstone R, Caldwell D G, Semini C. High-slope terrain locomotion for torque-controlled quadruped robots. Autonomous Robots, 2017, 41(1): 259–272