Foothold-Based Planning for Legged Robot Autonomous Navigation Over Uneven Terrain

Jiyu Yu, Dongqi Wang, Zhenghan Chen, Ci Chen, Rong Xiong, *Senior Member, IEEE*, and Yue Wang, *Member, IEEE*

***Abstract*—Legged robots hold promise for traversing challenging terrains, which is crucial for tasks, such as emergency response and inspections. Although the locomotion capabilities of legged robots have greatly improved, navigating unknown and uneven environments in real-time applications remains a key challenge. This is because existing controllers typically have a short prediction horizon and often require manual remote control. In this article, we propose a fast, foothold-aware path planner that enables rapid planning while simultaneously taking into account the feasibility of footholds along the path, and an online foothold and trajectory planner that plans appropriate footholds and collision-free whole-body trajectories in real-time. This layered approach does not require the long offline computation time, allowing for both real-time and long-horizon planning. We then integrate these planners with existing controllers and the perception module to achieve fully autonomous navigation of legged robots over uneven and unknown terrains. Extensive simulations and real-world experiments validate the effectiveness of our approach, demonstrating enhanced performance in navigating highly challenging terrain scenarios.**

***Index Terms*—Legged robots, mobility and locomotion, motion control.**

I. Introduction

LEGGED robots offer significant advantages in real-world environments, making them ideal for tasks, such as emergency response and routine inspections [1], [2]. To accomplish these tasks, recent approaches, whether optimization-based [3], [4], [5] or learning-based [6], rely on manual teleoperation via a handheld controller, wherein a human operator selects appropriate paths or sends velocity commands to the robot. This dependence arises because these methods are designed as motion controllers rather than planners, with a focus on high-frequency execution to ensure robustness. They exhibit short prediction horizons and lack the ability to adapt to long-range obstacles and terrain variations. Therefore, there is a critical need for a high-level planner that can eliminate the reliance on manual teleoperation and guide existing low-level controllers to enable legged robots to autonomously navigate across complex and diverse terrains.

It is intuitive to treat a legged robot as a wheeled system and plan torso velocity for navigation. However, this simplification compromises its ability to traverse complex terrains, as rough surfaces are typically marked impassable for wheeled robots. Building on this idea, the authors in [7] and [8] assigned passability scores to account for terrain complexity, rejecting regions below a threshold. While effective in guiding the controller through moderately difficult terrains, such as ramps, these methods lack the capability for precise foothold planning. In more challenging settings, such as stepping stones, where accurate foothold placement is critical, such approach often fail due to the absence of modeling of the legged robot's discrete foothold capabilities. In these cases, foothold-aware planning is essential to guide the controller through complex terrains safely and reliably.

To simultaneously address precise foothold placement and long-horizon navigation, one viable approach is to formulate a trajectory optimization (TO) problem that accounts for all factors, such as the environment, robot dynamics, and kinematic constraints [9], [10], [11]. This approach then outputs the whole-body trajectory from the very beginning to the end of the motion, which includes the torso motion trajectory, footholds, leg swing trajectories, and other essential components. Despite its ability to generate precise footholds and long-horizon motions, TO is constrained by the capabilities of large-scale nonlinear solvers, when dealing with complex environments, the solver may fail to find a solution due to high nonlinearity. Moreover, the extended convergence time makes TO unsuitable for real-time applications, especially in scenarios where the robot must navigate unknown environments and build maps in real-time. In summary, achieving foothold-based autonomous navigation for legged robots over uneven and unknown terrains necessitates a solution that retains the benefits of TO, while ensuring real-time performance.

To address this, we propose a fast foothold-aware path planner and an online long-horizon foothold planner (see Fig. 1 top, highlight in red), which decomposes the aforementioned TO

Received 23 April 2025; accepted 26 June 2025. Date of publication 31 July 2025; date of current version 19 February 2026. Recommended by Technical Editor Y. Zhao and Senior Editor M. Indri. This work was supported in part by the National Nature Science Foundation of China under Grant 62373322, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LD25F030001, and in part by the Research and Development Project of Zhejiang Province under Grant 2025C01205(SD2). *(Corresponding author: Yue Wang.)*

Jiyu Yu, Zhenghan Chen, Ci Chen, Rong Xiong, and Yue Wang are with the State Key Laboratory of Industrial Control and Technology, and the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: wangyue@iipc.zju.edu.cn).

Dongqi Wang is with the Department of Engineering Mechanics, Zhejiang University, Hangzhou 310027, China.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TMECH.2025.3588355.

Digital Object Identifier 10.1109/TMECH.2025.3588355

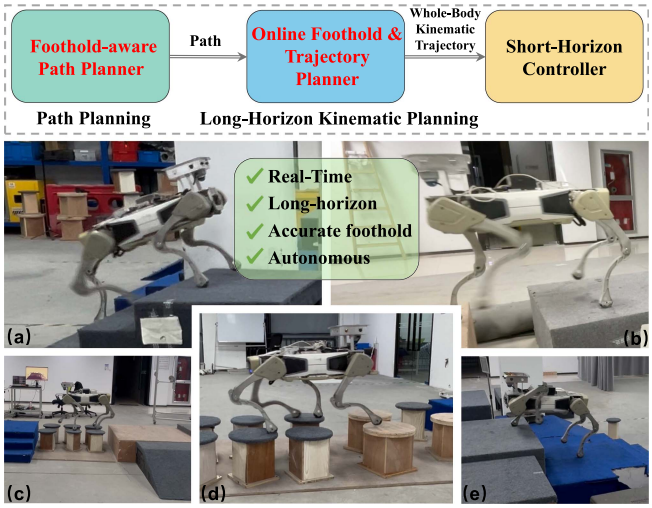


Fig. 1. Our framework and planner applied to various challenging terrains. (a) Staircase. (b) Large gap (35 cm). (c) Stepping stone 1. (d) Stepping stone 2 (maximum gap width: 40 cm). (e) Stepping up (height: 30 cm). The proposed planner is highlighted in red.

problem into layers. Specifically, during the upper layer path planning, we incorporate a sampling method. By generating $SE(2)$ sampled poses and mapping them to the robot's whole-body state, we can efficiently validate whether the state is feasible using a quasi-static model. This validation includes checking if the robot can place its feet, avoid collisions, and maintain balance. Consequently, we can rapidly generate a collision-free torso trajectory that accounts for feasible foothold placements. Building upon this torso trajectory, we proceed to long-horizon foothold planning and optimization of the whole-body kinematic trajectory. With the torso path resolved at the upper layer, we introduce mixed-integer programming (MIP) to precisely plan the kinematic footholds. Finally, by integrating these with dynamics optimization and the perception module, we validated the system in various simulated and real-world experiments, showing its efficiency and meets the requirements that we mentioned.

The contributions of this work are as follows. 1) A fast and foothold-aware path planner. 2) A planner capable of real-time generation of long-horizon accurate footholds and whole-body kinematic trajectory. 3) Integrating these with dynamics optimization and the perception module, we achieve fully autonomous legged robot navigation over uneven terrain in real-time.

II. Related Works

A. Navigation for Legged Robots

The TO method, as mentioned above, solves the navigation problem as a large-scale nonlinear programming problem to generate motion trajectories for the robot. Notable examples of this approach include the offline solvers TOWR [9], [12] and FROST [10], [11]. However, their solving times are often on the order of several minutes, especially in long-distance navigation tasks, which is detrimental to real-time applications. In recent years, the successful application of dynamic model predictive control (MPC) in legged robots has enabled robots to perform short-horizon planning based on the surrounding environment [3], [4], [5]. This shift allows researchers to focus on planning a traversable torso trajectory for the robot, with whole-body motion automatically generated by a local planner. Based on this, Brandao et al. [13] demonstrated the navigation of legged robots in large-scale environments, while Yang et al. [14] demonstrated global navigation in 3-D environments. These works significantly reduce planning time. The next challenge is how to evaluate terrain traversability and generate safe paths for low-level planners. Wermelinger et al. [7] introduced a framework to score traversability and plan routes for legged robots, focusing on terrain attributes, such as slope, roughness, and step height. Building on this, Norby and Johnson[15] developed a rapid global path planner using terrain data and a simplified robot model. However, as terrains become increasingly intricate, the effectiveness of simplistic terrain scoring approaches diminishes. To support fully autonomous navigation in legged robots, it is increasingly important to integrate kinematic constraints into path traversability evaluation—ensuring foothold safety, reachability, and stability, and enabling precise foothold planning. Jelavic et al. [16] inspired our work with a similar path planning framework, but their approach relies on predefined contact sequences for multiple robot types, whereas our quasi-static model dynamically checks contact feasibility along the path without gait restrictions. In addition, their system lacks local dynamic adjustments, which we address by integrating a kinematics-based planning module that enables obstacle avoidance along the global path.

B. Obstacle Avoidance and Foothold Planning

For legged robot obstacle avoidance, recent studies [17], [18], [19] have demonstrated that using obstacle representations, such as signed distance fields (SDF) or polytopes, legged and legged-manipulator robots can effectively integrate obstacle avoidance constraints into MPC, handling environmental collisions with minimal impact on solving time. They quickly obtain a corrected trajectory using gradient-based solvers, allowing the robot to navigate through very narrow spaces without collisions. However, treating legged robots as wheeled robots in this manner fails on discrete terrains, as legged robots, unlike wheeled robots, must account for both torso obstacle avoidance and foothold adjustment simultaneously.

For foothold planning, Raibert's heuristic methods work well on flat surfaces by calculating foothold positions based on the robot's velocity [8]. In complex terrains, such as stairs, the methods in [5] select the closest terrain segment but often neglect the interaction between legs and torso, which is critical to planning multiple steps ahead [3]. Optimization-based approaches are increasingly popular for finding optimal footholds under specific criteria [3], with [20] using batch searches on *grid_map* [21]. Other works [4], [9], [22] treat footholds as continuous variables in TO using gradient-based methods. For fragmented terrains, foothold planning involves disjunctive constraints and is typically solved with MIP. While MIP is theoretically ideal for

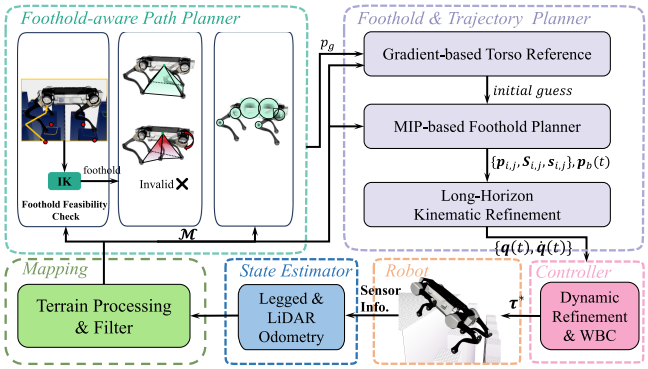


Fig. 2. Overview of the foothold-based planning architecture for autonomous navigation of legged robot. Our core contribution lies in the foothold-aware path planner and online foothold and trajectory planner. The foothold-aware path planner generates a feasible and foothold-considerate path $\boldsymbol{p}_g$ using a quasi-static model, which is passed to the online foothold and trajectory planner. A gradient-based solver assists the MIP-based foothold planner in efficiently computing accurate footholds, safe contact regions, and torso trajectories $(\{\boldsymbol{p}_{i,j}, \boldsymbol{S}_{i,j}, \boldsymbol{s}_{i,j}\}, \boldsymbol{p}_b(t))$. The kinematic refinement connects these discrete footholds and produces collision-free whole-body trajectories $\{\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)\}$, serving as reference for dynamic refinement. The dynamic refinement then outputs physically feasible joint trajectories and contact forces. The WBC computes the final torque command $\boldsymbol{\tau}^*$ by combining feedforward terms and potential difference feedback term. A state estimator provides torso pose and velocity estimates based on inertial measurement unit (IMU), legged odometry, lidar sensor, and joint encoder measurements. The mapping module processes point cloud and odometry to support planners with a terrain map.

multicontact problems, its practicality is limited by long computation times, although efforts, such as Tonneau et al. [23] simplified MIP into linear programming for faster results, sometimes at the cost of safety.

Considering both factors comprehensively, we integrate obstacle avoidance and footstep planning into a unified planning framework. In our approach, we leverage the fast convergence of gradient-based methods to generate heuristic initial values for the MIP solver, which subsequently adjusts the footstep positions to ensure that they remain within safe regions.

III. Planning Pipeline Overview

An overview of our system is presented in Fig. 2. The mapping module plays a crucial role by processing the raw elevation map into four informative layers, which serve as essential inputs for our planner. The fast foothold-aware planner (see Section IV) generates a path under a quasi-static assumption to ensure foothold traversability, which is passed to the online foothold and trajectory planner (see Section V). A gradient-based torso reference generator first produces an initial torso trajectory for obstacle avoidance. This trajectory is then used as a seed to guide the MIP-based planner, which efficiently determines accurate footholds and safe contact regions. Then, kinematic refinement connects these discrete footholds to generate whole-body collision-free trajectories, which are then provided to the dynamic controller (see Section VI). The dynamic refinement further optimizes these trajectories, ensuring dynamic feasibility and producing feedforward torque commands for the whole-body controller (WBC). Details on the implementation of the mapping, state estimation, and WBC modules, see the Supplementary Material.

IV. Fast Foothold-Aware Path Planner

With the robot model and terrain awareness (see the Supplementary Material), the fast path planner uses the rapidly exploring random tree (RRT) method to plan a path. The whole-body state of the robot is defined as the set $\mathcal{S}$

$$\mathcal{S} : \{\boldsymbol{q}_s = [\boldsymbol{q}_b, \boldsymbol{q}_j]^{\mathrm{T}} \in \mathbb{R}^{6+n_j}, \boldsymbol{q}_{j_{\min}} \leq \boldsymbol{q}_j \leq \boldsymbol{q}_{j_{\max}}\}$$

where $\boldsymbol{q}_b$ and $\boldsymbol{q}_j$ represent the base pose in the world frame (position and orientation) and the angles of n_j actuated joints, respectively. The joint angles are constrained by the limits of the joint space.

1) Sampling: The main challenge of sampling-based planning (SBP) methods is inefficient sampling in whole-body state spaces. To address this, robot models and terrain data are integrated, mapping the robot's whole-body state to its $SE(2)$ state. Sampling occurs in the robot's x–y position and yaw angle (x, y, γ), with the height determined by adding a predefined height to the *smooth layer*. The *smooth normal layer* provides pitch and roll for the floating base, ensuring that it aligns with the terrain. For the remaining joint states, $\boldsymbol{q}_j$, the configuration closest to the default foothold that matches the terrain is selected. Using the floating base's 6-D pose $\boldsymbol{q}_b$ and predefined default footholds ${}^{\mathcal{B}}\boldsymbol{p}_i$ for each leg, the positions of these footholds in the terrain ${}^{\mathcal{W}}\boldsymbol{p}_i$ are calculated. The nearest point ${}^{\mathcal{M}}\boldsymbol{p}_i$ on *Segmented terrain* to ${}^{\mathcal{W}}\boldsymbol{p}_i$ is selected as the foothold, generating the state of the entire body $\{\boldsymbol{q}_b, {}^{\mathcal{M}}\boldsymbol{p}_1, {}^{\mathcal{M}}\boldsymbol{p}_2, {}^{\mathcal{M}}\boldsymbol{p}_3, {}^{\mathcal{M}}\boldsymbol{p}_4\}$. This whole-body state is considered the most likely valid configuration as a tradeoff for efficiency, without checking other footholds.

2) Expansion and State Checker: During RRT expansion in $SE(2)$ space, the new state is connected via Reeds-Shepp curves, which are discretized and checked for validity by a state checker. This checker ensures that the path is executable, considering both collision-free navigation and foothold feasibility, and if states are valid, then the time interval between them is determined by dividing the distance between them by a default velocity. The state checker evaluates the following four key aspects.

a) *Foothold feasibility:* Inverse kinematics (IK) is used to verify if footholds in the state tuple $\{\boldsymbol{q}_b, {}^{\mathcal{W}}\boldsymbol{p}_1, {}^{\mathcal{W}}\boldsymbol{p}_2, {}^{\mathcal{W}}\boldsymbol{p}_3, {}^{\mathcal{W}}\boldsymbol{p}_4\}$ violate the joint limits.

b) *Convex gravito-inertia acceleration cone (GIAC):* The GIAC criterion is applied as a kinematics constraint for base position and footholds.

c) *Collision-free path:* The *elevation SDF* calculates the closest distance between the joints and obstacles, focusing on the hip and thigh joints.

d) *Terrain slope:* The base pitch and roll angles are constrained to prevent the robot from tipping over, based on the normal of the *smooth layer*.

During foothold feasibility check, each foothold ${}^{\mathcal{W}}\boldsymbol{p}_i$ is transformed into the hip frame, and IK is used to calculate joint angles. If any result exceeds joint limits $\{\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}\}$, then it is rejected. The convexity of the GIAC is verified using specific inequalities,

ensuring that footholds remain stable and lower than the base to prevent tipping. The convexity of the GIAC is verified by checking the following inequality. Note that the $^{\mathcal{W}}$ is omitted

$$\begin{aligned} &\boldsymbol{p}_{B1}\times\boldsymbol{p}_{B3}\cdot\boldsymbol{p}_{B2}\geq 0, \boldsymbol{p}_{B1}\times\boldsymbol{p}_{B3}\cdot\boldsymbol{p}_{B4}\geq 0\\ &\boldsymbol{p}_{B1}\times\boldsymbol{p}_{B2}\cdot\boldsymbol{p}_{B3}\leq 0, \boldsymbol{p}_{B1}\times\boldsymbol{p}_{B2}\cdot\boldsymbol{p}_{B4}\leq 0\\ &\forall i,\quad p_B^z > p_i^z \end{aligned} \tag{1}$$

where $\boldsymbol{p}_B$ denotes the base position, and $\boldsymbol{p}_{Bi}=\boldsymbol{p}_i-\boldsymbol{p}_B$ denotes the edge of the cone. p^z is coordinates in the z-axis. Collision checks are performed using spherical approximations of the hip and thigh joints, ensuring that these are outside of obstacles. The path planner ensures that each state along the path can support a full stance, allowing the local foothold planner to find solutions.

V. Online Foothold and Trajectory Planner

A. Gradient-Based Torso Reference Generator

While reducing the dimensionality of the sampling space in path planning accelerates the process, the sampling-based planner still faces issues with uncertain planning times and slow environmental response. To handle sudden obstacles, such as pedestrians, and guide the MIP-based footholds planner, we designed a generator to solve the 6 DoF torso pose trajectory. This generator addresses the following discrete MPC problem:

$$\min_{\boldsymbol{p}_b[n],\boldsymbol{v}_b[n]} \sum_{n=0}^{N-1}\left(\|\boldsymbol{p}_b[n]-\boldsymbol{p}_g[n]\|^2_{\boldsymbol{Q}_b}+\|\boldsymbol{v}_b[n]\|^2_{\boldsymbol{R}_b}\right)\Delta t \tag{2a}$$

$$\text{s.t.}\quad \boldsymbol{p}_b[n+1]=\boldsymbol{p}_b[n]+\boldsymbol{f}_b(\boldsymbol{p}_b[n],\boldsymbol{v}_b[n])\Delta t,\quad n=0,\dots,N-1 \tag{2b}$$

$$\boldsymbol{p}_b[0]=\boldsymbol{p}_{t0} \tag{2c}$$

$$\mathcal{H}_{\text{sdf}}(\boldsymbol{F}(\boldsymbol{p}_b[n]))\geq \boldsymbol{r}_c,\quad n=1,\dots,N \tag{2d}$$

$$\boldsymbol{v}_{b_{\min}}\leq \boldsymbol{v}_b[n]\leq \boldsymbol{v}_{b_{\max}},\quad n=0,\dots,N-1 \tag{2e}$$

where Δt represents the discrete time step between the nth and $(n+1)$th points. $\boldsymbol{p}_b[n]$ represents the 6 DoF pose of the torso, while $\boldsymbol{p}_g[n]$ is the reference pose received from the foothold-aware path planner. The velocity is denoted by $\boldsymbol{v}_b[n]$, and $\boldsymbol{f}_b(\cdot)$ represents the dynamics of the system, constraining the robot's direction to align with its direction of velocity. Equation (2e) defines the velocity constraints, while (2a) is a quadratic cost function tracking the path $\boldsymbol{p}_g[n]$ and regularizing the velocity. For collision avoidance (2d), the robot's collision links are approximated by spheres, with centers computed via $\boldsymbol{F}(\boldsymbol{p}_b[n])$. The environment is modeled using an *elevation SDF* function, $\mathcal{H}_{\text{sdf}}(\cdot)$, which includes discrete precomputed derivatives, enabling rapid distance and derivative queries at spatial points. Since gradients of all terms are easily computed, a gradient-based solver is used for efficient optimization.

B. MIP-Based Footholds Planner

The safe region on the terrain general is not continuous, such as in stairs or stepping-stone scenes, where accessible regions are discretized into pieces, as shown in the right of Fig. 3. Each region is denoted by

$$\mathcal{A}_i:\{\boldsymbol{p}\in\mathbb{R}^3|\boldsymbol{S}_i\boldsymbol{p}\leq\boldsymbol{s}_i,\boldsymbol{S}_i\in\mathbb{R}^{n\times 3},\boldsymbol{s}_i\in\mathbb{R}^n\}. \tag{3}$$

Each foothold is subject to one and only one safe region constraint $\mathcal{A}^m_{i,j}$, derived from convex polygons within the *segmented terrain*. Specifically, this involves finding the foothold $\boldsymbol{p}_{i,j}$ (for leg i at step j) that satisfies one of a series of constraints, known as the constraint selection problem, which can be formulated as a problem with disjunctive constraints and solved using an MIP solver. By using Big-M formulation [3], the safe contact constraint can be rewritten as follows:

$$\forall i,j,m,\quad \boldsymbol{S}^m_{i,j}\boldsymbol{p}_{i,j}\leq \boldsymbol{s}^m_{i,j}+M(1-a^m_{i,j})\mathbf{1};\quad \sum_{m=0}^{n_s}a^m_{i,j}=1 \tag{4}$$

where $M\gg 0$ represents a significantly large number, n_s is the number of region, and the binary decision variable $a^m_{i,j}\in\{0,1\}$ indicates the selected region. For instance, if $a^1_{i,j}=1$, then $\boldsymbol{S}^1_{i,j}\boldsymbol{p}_{i,j}\leq\boldsymbol{s}^1_{i,j}$, and for other constraints, $\{S^m_{i,j},\boldsymbol{s}^m_{i,j}\}$ $(m\neq 1)$ are automatically satisfied since M is a very large number, and those constraints are inactive. The sum of $a^m_{i,j}$ equals 1 suggests that exactly one region is selected. In addition, the torso trajectory is incorporated into the planning process, with slight adjustments made to the torso's position using the approximate Zero Moment Point (ZMP). The torso position is parameterized as $\boldsymbol{c}(t)=\sum_{i=0}^{d}B_{i,d}(t)\boldsymbol{C}_i$, a B-spline curve of degree d based on the piecewise polynomial function $B_{i,d}(t)$, where $\boldsymbol{C}=[\boldsymbol{c}_0\quad \boldsymbol{c}_1\quad \dots]$ are the $d-1$ control points. Rotations are ignored to avoid unnecessary nonlinearity. The local footholds planning problem is then formulated as follows:

$$\min_{\boldsymbol{C},\boldsymbol{P},\boldsymbol{a}_{i,j}} \sum_{m=0}^{n_c}||\boldsymbol{c}(t_m)-\boldsymbol{c}^{\text{ref}}(t_m)||^2_{w_1}+||\boldsymbol{p}_{i,j}-\boldsymbol{p}^*_{i,j}||^2_{w_2}$$

$$\text{s.t.}\ \sum_{m=0}^{n}a^m_{i,j}=1,\quad a^m_{i,j}\in\{0,1\} \tag{5a}$$

$$\forall i,j,m,\quad \boldsymbol{S}^m_{i,j}\boldsymbol{p}_{i,j}\leq\boldsymbol{s}^m_{i,j}+M(1-a^m_{i,j})\mathbf{1} \tag{5b}$$

$$||\boldsymbol{c}(t_{i,j})-\boldsymbol{p}_{i,j}||^2\leq l_{\max} \tag{5c}$$

$$\forall i\neq o,\quad ||\boldsymbol{p}_{o,j}-\boldsymbol{p}_{i,j}||^2\geq l_{\min} \tag{5d}$$

$$\forall j>1,\quad ||\boldsymbol{p}_{i,j}-\boldsymbol{p}_{i,j-1}||^2\leq h_{\max} \tag{5e}$$

$$||\boldsymbol{c}^{x,y}(t_{i,j})-\sum_{i=1}^{4}\boldsymbol{p}^{x,y}_{i,j}/4||^2_{w_3}\leq\epsilon_{\max} \tag{5f}$$

where the cost consists of two terms: the first term tracks the trajectory generated by the torso reference generator, and the second term ensures that the footholds are as close as possible to the heuristic values. Equations (5c)–(5e) are kinematic constraints. These constraints ensure that the torso's position and the foot placement are not too far apart, prevent foot placements in the same step from being too close (which could result in self-collision), and limit the distance between two consecutive steps to avoid excessively large steps. Equation (5f) is the ZMP

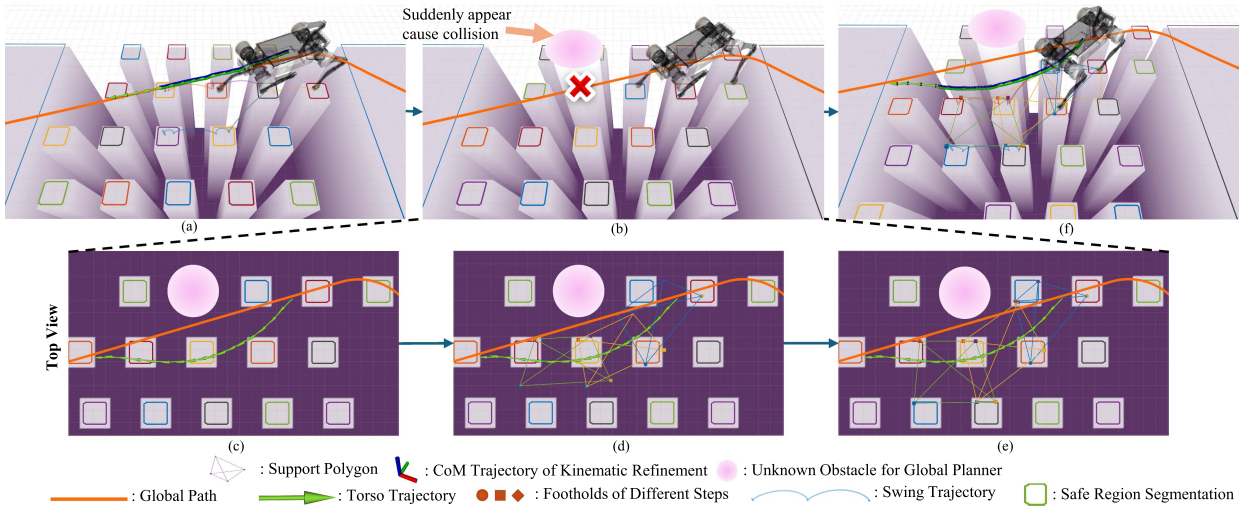


Fig. 3. Obstacle avoidance for legged robots in the stepping stone scenario. (a) Robot following the planned path toward the target point in the absence of obstacles. (b) Suddenly appearing obstacle causes the robot to face a collision risk if it continues to follow the path. (c), (d), and (e) Top views of the scenario, with the robot's visualization model omitted. (c) Planning result of the gradient-based torso reference generator. (d) Heuristic footholds and support polygons for steps 1, 3, and 5 based on this result. (e) MIP-optimized outcome, where all footholds are adjusted into safe areas, ensuring the robot does not misstep. (f) How the robot avoids obstacles while successfully navigating the challenging stepping stones.

constraint to ensure that the torso projection remains close to the center of the contact polygon.

MIP problems are typically solved using the branch-and-bound algorithm, with the time complexity of this method being highly sensitive to both the problem's initial values and its scale. To manage this, at step j, the default footholds for leg i are first generated based on the torso trajectory (with the subscript j omitted)

$$\boldsymbol{p}_i^* = \boldsymbol{p}_{i,\text{nom}} + \sqrt{h_r/g}(\boldsymbol{v}_{b,\text{meas}} - \boldsymbol{v}_b) \tag{6}$$

where h_r is the height of torso, and g is gravitational acceleration constant. $\boldsymbol{p}_i^*$ denotes the heuristic foothold, $\boldsymbol{p}_{i,\text{nom}}$ represents the nominal foothold located directly beneath the hip, and $\boldsymbol{v}_{b,\text{meas}}$ is measured velocity. Next, the set $\mathcal{A}_i$ is sorted in ascending order based on the distance to the heuristic foothold, and the closest m candidates are selected. Any candidates within this set that exceed the robot's motion limits are discarded. By following these steps, the scale of the MIP problem is effectively constrained. Finally, the binary variable a_i^{closest}, corresponding to the element in set $\mathcal{A}_i^{\text{closest}}$ closest to $\boldsymbol{p}_i^*$, is set to 1, with $\boldsymbol{p}_i^*$ serving as the initial guess for the MIP problem.

C. Long-Horizon Kinematic Refinement

In *kinematic refinement* stage, all the generalized coordinates of robot is chosen as state, and the control input is the generalized velocity

$$\boldsymbol{x}_k \triangleq [\boldsymbol{\theta}_k^\top, \boldsymbol{p}_{k,b}^\top, \boldsymbol{q}_{k,j}^\top]^\top, \boldsymbol{u}_k \triangleq [{}^{\mathcal{B}}\boldsymbol{\omega}_k^\top, {}^{\mathcal{B}}\boldsymbol{v}_{k,b}^\top, \dot{\boldsymbol{q}}_{k,j}^\top]^\top \in \mathbb{R}^{6+n_j} \tag{7}$$

where the subscript k here represents the variables used in the *kinematic refinement* layer. $\boldsymbol{\theta}_k$ and $\boldsymbol{p}_{k,b}$ are the torso XYZ-Euler angles and position in the world frame $\mathcal{W}$, and ${}^{\mathcal{B}}\boldsymbol{\omega}_k$ and ${}^{\mathcal{B}}\boldsymbol{v}_{k,b}$ are the angular velocity and linear velocity in the base frame $\mathcal{B}$, respectively. $\boldsymbol{q}_{k,j}$ and $\dot{\boldsymbol{q}}_{k,j}$ are angles and velocity of actuated joints.

1) System Modeling: The whole-body kinematic model is used as the system dynamics in the MPC problem

$$\frac{d}{dt}\begin{bmatrix}\boldsymbol{\theta}_k\\ \boldsymbol{p}_{k,b}\\ \boldsymbol{q}_{k,j}\end{bmatrix} = \begin{bmatrix}\boldsymbol{T}(\boldsymbol{\theta}_k){}^{\mathcal{B}}\boldsymbol{\omega}_k\\ \boldsymbol{R}_{\mathcal{B}}(\boldsymbol{\theta}_k){}^{\mathcal{B}}\boldsymbol{v}_{k,b}\\ \dot{\boldsymbol{q}}_{k,j}\end{bmatrix} \tag{8}$$

where $\boldsymbol{T}(\cdot) : \mathbb{R}^3 \to \mathbb{R}^{3\times3}$ maps torso angular velocity to XYZ-Euler angle derivatives, and $\boldsymbol{R}_{\mathcal{B}}(\cdot) : \mathbb{R}^3 \to \mathbb{R}^{3\times3}$ is the base to world rotation matrix.

2) Reference: The reference of *kinematic refinement* layer is the result of footholds planner, which gives the footholds $\boldsymbol{p}_{i,j}$ of leg i in step j, the corresponding safe region selected by binary variable $a_{i,j}$ in (5b) $\{\boldsymbol{S}_{i,j}, \boldsymbol{s}_{i,j}\}$, and the trajectory of the torso position $\{\boldsymbol{p}_b^{\text{ref}}, \boldsymbol{v}_b^{\text{ref}}\}$. The swing trajectories between two adjacent footholds are obtained using two quintic splines with a specified height, and their position and velocity represented by $\{\boldsymbol{p}_i^{\text{ref}}, \boldsymbol{v}_i^{\text{ref}}\}$ [5]. The torso rotation is determined by projecting the hips position onto the *smooth layer* and fitting them points to a plane, and the torso should remain parallel to this plane, and this reference for rotation is represented by a rotation matrix $\boldsymbol{R}_{k,\text{ref}}$.

3) Cost and Soft Constraints: Since the soft constraints are using penalty method and will finally be added to the cost function, the intermediate cost is built up by two components, the quadratic motion tracking cost $L_{k,Q}$ and the penalty cost $L_{k,P}$. The motion tracking cost is used to follow the reference generated by the Section V-C2

$$\begin{aligned} L_{k,Q} = &||\boldsymbol{p}_{k,b} - \boldsymbol{p}_b^{\text{ref}}||^2_{\boldsymbol{Q}_{k,b}} + ||\boldsymbol{v}_{k,b} - \boldsymbol{v}_b^{\text{ref}}||^2_{\boldsymbol{Q}_{k,v}} \\ &+ \left\|\log\left(\boldsymbol{R}_{k,b}\boldsymbol{R}_{k,\text{ref}}^\top\right)^\vee\right\|^2_{\boldsymbol{Q}_{k,r}} + ||\boldsymbol{\omega}_{k,b}||^2_{\boldsymbol{Q}_{k,\omega}} \\ &+ ||\boldsymbol{p}_i - \boldsymbol{p}_i^{\text{ref}}||^2_{\boldsymbol{Q}_{k,\text{foot,p}}} + ||\boldsymbol{v}_i - \boldsymbol{v}_i^{\text{ref}}||^2_{\boldsymbol{Q}_{k,\text{foot,v}}} \end{aligned}$$

$$+ ||\boldsymbol{q}_{k,j} - \boldsymbol{q}_{\text{nom},j}||^2_{\boldsymbol{Q}_{k,j}} + ||\dot{\boldsymbol{q}}_{k,j}||^2_{\boldsymbol{Q}_{k,dj}} \tag{9}$$

where $\log(\boldsymbol{R}_{k,b}\boldsymbol{R}_{k,\text{ref}}^\top)^\vee$ is the logarithmic map of the orientation error, $\boldsymbol{q}_{\text{nom},j}$ is the nominal joint position defined by stance state, and the $\boldsymbol{Q}_{k,\cdot}$ are the diagonal positive semi-definite matrices. This cost term defines the torso pose and velocity tracking task, and the feet end-effector position and velocity tracking task. $||\boldsymbol{q}_{k,j} - \boldsymbol{q}_{\text{nom},j}||^2_{\boldsymbol{Q}_{k,j}}, ||\dot{\boldsymbol{q}}_{k,j}||^2_{\boldsymbol{Q}_{k,dj}}, ||\boldsymbol{\omega}_{k,b}||^2_{\boldsymbol{Q}_{k,\omega}}$ serve as regularization terms.

The soft constraints terms here are safe contact constraints, collision avoidance constraints, and joint limits constraints. For safe contact constraints, it adds limits to the footholds position when the leg is in contact with the ground using the results from footholds planner

$$\boldsymbol{h}_k^{\text{foot}} = -\boldsymbol{S}_{i,j}\boldsymbol{p}_{i,j} + \boldsymbol{s}_{i,j} \geq \boldsymbol{0}, \quad \text{if } i \in \mathcal{C} \tag{10}$$

where $i \in \mathcal{C}$ means leg i is in contact of terrain, j represents the variable at step j, and $\{\boldsymbol{S}_{i,j}, \boldsymbol{s}_{i,j}\}$ are safe regions selected by binary variable $a_{i,j}$ in (5b). For self-collision avoidance, the distance between two links is queried from flexible collision library, represented by $d^c(\cdot) : \mathbb{R}^{n_j} \to \mathbb{R}$. The collision avoidance constraints finally can be written as

$$h^c_{k,n} = d^c(\boldsymbol{q}_{k,j}) - \epsilon_n \geq 0, \text{for self-collision} \tag{11}$$

where ϵ_n denotes the minimum allowed distance threshold for each collision. The joint limits constraints include the positions and velocities box constraints. All soft constraints are handled using the barrier function and add the cost [24].

4) Hard Equality Constraints: For each foot in contact, the velocity of the foot is restricted to zero so that it can stay still during the contact phase

$$\begin{cases} \boldsymbol{v}_i = \boldsymbol{0}, & \text{if } i \in \mathcal{C} \\ \boldsymbol{n}^\top \boldsymbol{v}_i = \boldsymbol{n}^\top \boldsymbol{v}_i^{\text{ref}} + k_p(\boldsymbol{p}_i^{\text{ref}} - \boldsymbol{p}_i), & \text{if } i \notin \mathcal{C} \end{cases} \tag{12}$$

where $\boldsymbol{n}$ is the surface normal from the *smooth normal* layer. k_p is the gain in position error of the end-effector feet.

In this layer, constraints related to footholds, collision, and joint kinematics are extracted and refined at this level. Due to the absence of the underactuated robot's dynamic processes in the system model, the problem's nonlinearity is reduced, so we can extend the prediction horizon of the *kinematic refinement* module, together with MIP-based foothold planner for the next few gait cycles, achieving long-horizon planning up to 4–5 s.

VI. Controller

A. Dynamic Refinement

The state vector and control input vectors defined as follow:

$$\begin{aligned} \boldsymbol{x}_d &\triangleq [\boldsymbol{\theta}_d^\top, \boldsymbol{p}_{d,b}^\top, {}^{\mathcal{B}}\boldsymbol{\omega}_d^\top, {}^{\mathcal{B}}\boldsymbol{v}_{d,b}^\top, \boldsymbol{q}_{d,j}^\top]^\top \in \mathbb{R}^{12+n_j} \\ \boldsymbol{u}_d &\triangleq [\boldsymbol{\lambda}_i^\top, \dot{\boldsymbol{q}}_{d,j}^\top]^\top \in \mathbb{R}^{3n_c+n_j} \end{aligned} \tag{13}$$

where the subscript d here represents the variables used in the *dynamic refinement* layer (to distinguish them from *kinematic refinement* layer).

1) System Modeling: The dynamics of the system is a so-called kino-dynamics [5]. It establishes the relationship between the contact force and the base acceleration

$$\frac{d}{dt}\begin{bmatrix} \boldsymbol{\theta}_d \\ \boldsymbol{p}_{d,b} \\ {}^{\mathcal{B}}\boldsymbol{\omega}_d \\ {}^{\mathcal{B}}\boldsymbol{v}_{d,b} \\ \boldsymbol{q}_{d,j} \end{bmatrix} = \begin{bmatrix} \boldsymbol{T}(\boldsymbol{\theta}_d)^{\mathcal{B}}\boldsymbol{\omega}_d \\ \boldsymbol{R}_{\mathcal{B}}(\boldsymbol{\theta}_d)^{\mathcal{B}}\boldsymbol{v}_{d,b} \\ \boldsymbol{M}_B^{-1}\left(-\boldsymbol{h}_B + \sum_{i\in\mathcal{C}} \boldsymbol{J}_{B,i}^\top \boldsymbol{\lambda}_i\right) \\ \dot{\boldsymbol{q}}_{d,j} \end{bmatrix} \tag{14}$$

where $\boldsymbol{M} \in \mathbb{R}^{(6+n_j)\times(6+n_j)}$ is inertia matrix, $\boldsymbol{h} \in \mathbb{R}^{(6+n_j)}$ is the vector of Coriolis, centrifugal, and gravity terms, $\boldsymbol{J}$ is contact Jacobians, and subscript B corresponds to the top six rows of the them.

2) Transition between Dynamic and Kinematic Refinement: Once the *kinematic refinement* solves the problem, the solution will be immediately sent to the *dynamic refinement*. The solution is denoted by $\{\boldsymbol{x}_k^{\text{sol}}(t), \boldsymbol{u}_k^{\text{sol}}(t)\}$, which contains the joint angles and angular velocities of the whole-body for the upcoming T, as well as the torso position and velocity. The solver will query and automatically interpolate the *kinematic refinement* solution based on time t, and fill the corresponding values in $\boldsymbol{x}_d(t), \boldsymbol{u}_d(t)$, which will serve as the initial values for *dynamic refinement*. For the contact force $\boldsymbol{\lambda}_i$, the initial values are robot total gravity divided by the number of contacts. The details of the controller we used see the Supplementary Material.

Overall, the role of *dynamic refinement* is to generate the contact forces based on the kinematic solution $\{\boldsymbol{x}_k^{\text{sol}}(t), \boldsymbol{u}_k^{\text{sol}}(t)\}$, while ensuring that the final trajectory is consistent with the dynamics. Unlike [3], [4], and [5], a key distinction is that the hard equality constraints related to the swing phase and the designated footholds are removed from this layer. Incorporating these terms into the hard constraints in dynamic layer can introduce errors from mapping, state estimation, and sensor calibration, potentially reducing the robustness of the system.

VII. Experiments

All the modules in the proposed framework are implemented in C++ with robot operating system (ROS) [25] as a communication middleware. The path planner is implemented and tested based on open motion planning library (OMPL) [26], which offers a convenient interface to benchmark our algorithm. The MIP problem is solved by Gurobi [27]. The MPC is executed via the OCS2 library [28], and the dynamics of the robot is calculated by the Pinocchio library [29].

A. Fast Foothold-Aware Path Planner Evaluation

1) Success Rate on Different Terrains: The path planner is evaluated in four different terrains with three difficulty levels.

a) *Maze:* $9\,\text{m} \times 3\,\text{m}$ in size, start and goal are fixed in $SE(2)$ states (0,0,0) and $(5, 0, 0)$, respectively. Obstacles are randomly distributed on the map. As the difficulty increases, the obstacles become larger and denser.

b) *Staircase:* $9\,\text{m} \times 9\,\text{m}$ in size, start and goal are (0,0,0) and $(6, 6, \frac{\pi}{2})$, respectively. There are three staircases with different step heights. As the difficulty increases, the staircases become higher.

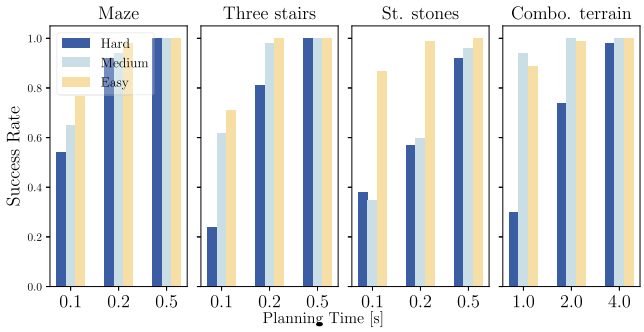


Fig. 4. Foothold-aware path planner success rate on different terrain. For all difficulty levels and planning times, we run the planner 100 times.

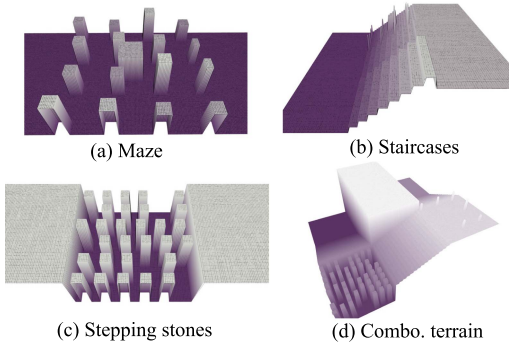


Fig. 5. Different terrain we used in the experiments.

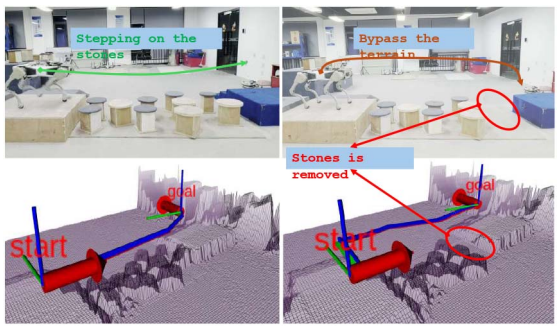


Fig. 6. Legged robot planning acorss the stepping stones. If the last two stones in the left image in front of us are removed, then the path planner will search for alternative paths, as shown in the right image.

c) *Stepping stone:* $9\,\mathrm{m} \times 3\,\mathrm{m}$ in size, start and goal are (0,0,0) and $(5, y, 0)$, respectively, and $y \in [-1.3\mathrm{m}, 1.3\mathrm{m}]$ is randomly sampled. As the difficulty increases, the steps will be removed at random.

d) *Combined terrain:* $9\,\mathrm{m} \times 9\,\mathrm{m}$ in size, start and goal are (0,0,0) and $(6, 6, \pi)$, respectively. Combining the three terrains above, the robot should cross all of them.

We first run the planner for a sufficient duration to ensure the existence of a solution. The statistics result is shown in Fig. 4, for all difficulty levels and planning times, we run the test for 100 times. For single kind terrain, the planner takes at most $1\,\mathrm{s}$ to reach the goal, and for combination terrain, it spends more time since the path is longer than others. The terrains are shown in Fig. 5. In the case of stairs, the path chooses the appropriate stair height and avoids high steps. In the case of the stepping stone, the path bypasses the missing stone.

2) Comparative Study: We also compare our planner with the open-source *Quad-SDK* planner [8], and the result is shown in the Table I. On the single-terrain scenario, our planner takes at most $1\,\mathrm{s}$ to reach the goal and on the combination terrain takes near $4\,\mathrm{s}$ to reach high success rate. To better analyze the detailed performance of our planner, we choose planning time [0.1, 0.2, 0.5] s for single-terrain scenario and [1.0, 2.0, 4.0] s for combined terrain. All tested for 100 times. Quad-SDK is ultrafast on flat terrain ("maze" scenario), since it simply uses a traversability score as a criterion. However, it is hard to find solutions on "staircase" and lost the opportunity on more challenging terrain. Some planning results are shown in Fig. 9(a) and (b).

B. Foothold and Trajectory Planner Evaluation

1) MIP-Based Foothold Planning Time Performance: We evaluate the computational efficiency of the proposed foothold planner. Without an initial guess from the torso trajectories, the MIP solver must explore all possible combinations, often leading to unnecessary delays and an increase in computation time. To assess the planner's real-time capability, we focus on the average, the slowest 10%, and the maximum of planning times, as presented in Table II. We used a laptop with an AMD R7-5800H CPU (8-Core, 3.2 GHz) and 32 GB of RAM for the test. The results show that with the initial guess, the planner consistently achieves average planning times below 70 ms, even in challenging terrains. While planning stepping stone terrain with obstacles, the absence of an initial guess proved to be critical, as the solver failed to find a solution within 4000 ms. During the planning process, our method's maximum planning time did not exceed 200 ms, indicating its suitability for real-time applications.

2) Prediction Horizon are Planning Frequency: Table III shows the relationship between the *dynamic refinement* prediction time and planning time. When the planning horizon is forcibly extended, the planning frequency decreases, and the resulting trajectory becomes difficult to converge, which negatively impacts the system's robustness. Therefore, we have placed *kinematic refinement* as the upper layer of *dynamic refinement*. *Kinematic refinement* does not need to consider complex dynamics, so even with a 4 s planning horizon, the planning time remains around 10 ms. Meanwhile, *dynamic refinement* can operate at a high frequency (greater than 100 Hz). This way, our system balances both long-term planning capabilities and rapid replanning capabilities.

3) Visualization of Planning Process: Fig. 3 illustrates the local kinematic planner's process in the presence of obstacles. A cylinder simulates a sudden obstacle (e.g., a pedestrian). When an obstacle appears, the gradient-based torso reference generator quickly computes an obstacle avoidance trajectory. This trajectory helps generate heuristic footholds and support polygons, which the MIP planner refines to find safe regions and solve for footholds that meet kinematic constraints. Each foothold is confined to a safe area. *Kinematic refinement* then

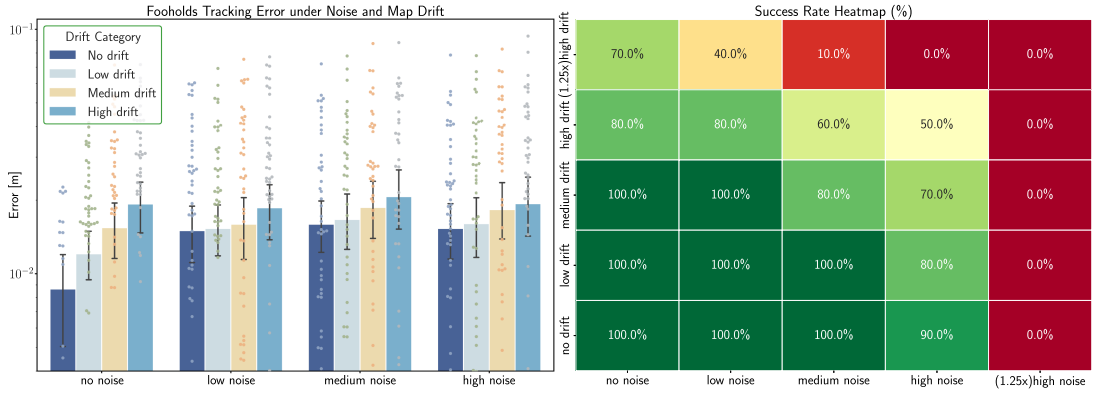


Fig. 7. Left: The footholds tracking error during robot cross stepping stone (level: easy) under different combinations of noise and map drift. The scatter points in the figure represent the distribution of error. Right: The success rate heatmap across under different combinations of noise and map drift.

TABLE I
COMPARISON WITH QUAD-SDK

		Maze			Staircase			St. stones			Comb. terrain		
Planning time [s]		0.1	0.2	0.5	0.1	0.2	0.5	0.1	0.2	0.5	1.0	2.0	4.0
Success rate	Quad-SDK [8]	**0.90**	**1.0**	**1.0**	0.03	0.07	0.16	-	-	-	-	-	-
	Ours	0.77	0.98	**1.0**	**0.62**	**0.71**	**1.0**	**0.38**	**0.57**	**0.92**	**0.30**	**0.74**	**0.98**

'-' means it failed to reach the goal in 5s.
Bold values mean the best value in the table for the ablation/comparative experiments.

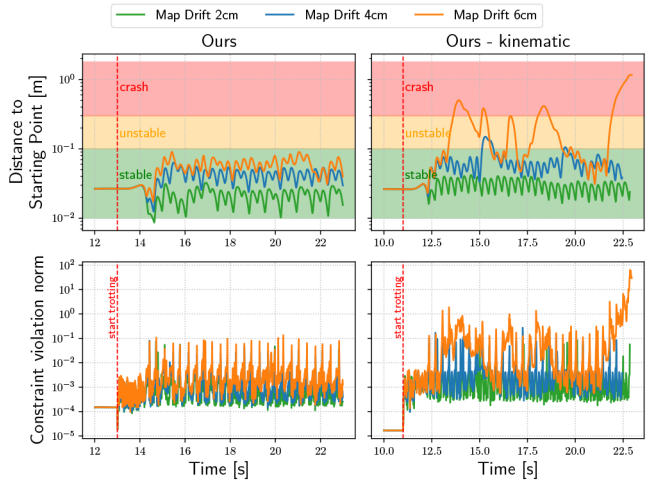


Fig. 8. Robustness test against map drift. On the left is our method, and on the right is the ours w/o kinematic refinement. The results show that under the same map drift, our method maintains more stable trotting at the designated point.

TABLE II
MIP PLANNING TIME EVALUATION

Terrain	w/o initial (ms)			**w/ initial** (ms)		
	Avg.	10% slowest	Max.	Avg.	10% slowest	Max.
Maze	156.82	238.03	281.87	61.46	100.74	**134.25**
Staircasers	85.53	369.34	369.35	47.88	85.04	**106.37**
Comb. terrain	103.48	306.64	1640.42	54.47	73.70	**123.03**
St. stone w/ Ob.	-	-	>4000	70.23	121.27	**144.24**

Bold values mean the best value in the table for the ablation/comparative experiments.

TABLE III
PLANNING TIME OF DYNAMIC REFINEMENT UNDER DIFFERENT PLANNING HORIZON (TESTED ON I7-1185G7E 4-CORE, 1.80 GHZ CPU)

Planning Horizon	1 s	2 s	3 s	4 s
Mean [ms]	5.52	12.18	17.79	21.57
Max [ms]	14.92	29.25	43.55	49.78

adjusts these footholds, producing whole-body joints' angles and velocities that satisfy kinematic constraints.

C. Closed-Loop Experiments in Physical Simulation

1) Robustness Against Perception Errors: We simulate perception noise as a linear combination of Gaussian and uniform noise: $\boldsymbol{\xi} \sim k_n(U(-\boldsymbol{I}, \boldsymbol{I}) + N(\boldsymbol{0}, \boldsymbol{I}))$, where k_n represents noise intensity. The noise is categorized into three levels: low ($k_n = 0.01$ m), medium ($k_n = 0.02$ m), and high ($k_n = 0.03$ m). Map drift is also categorized into low (2 cm), medium (4 cm), and high (6 cm) levels. These levels are based on real-world sensor noise, where the maximum noise does not exceed the low level. We test foothold tracking error and success rate while crossing stepping stones under different noise and map drift conditions, as shown in Fig. 7. First, noise does affect accuracy, but the difference between medium and high noise is small This is because terrain filtering and segmentation were applied, which can mitigate the impact of noise to a certain degree. However, when the noise intensity increases to $k_n = 0.04$ m, the terrain has completely deformed, and the terrain processing module fails to provide viable foothold regions, resulting in task failure. In real-world experiments, the maps have much smaller noise and drift, staying within the system's tolerance. Second, foothold accuracy is more sensitive to map drift, as it causes early or delayed foot

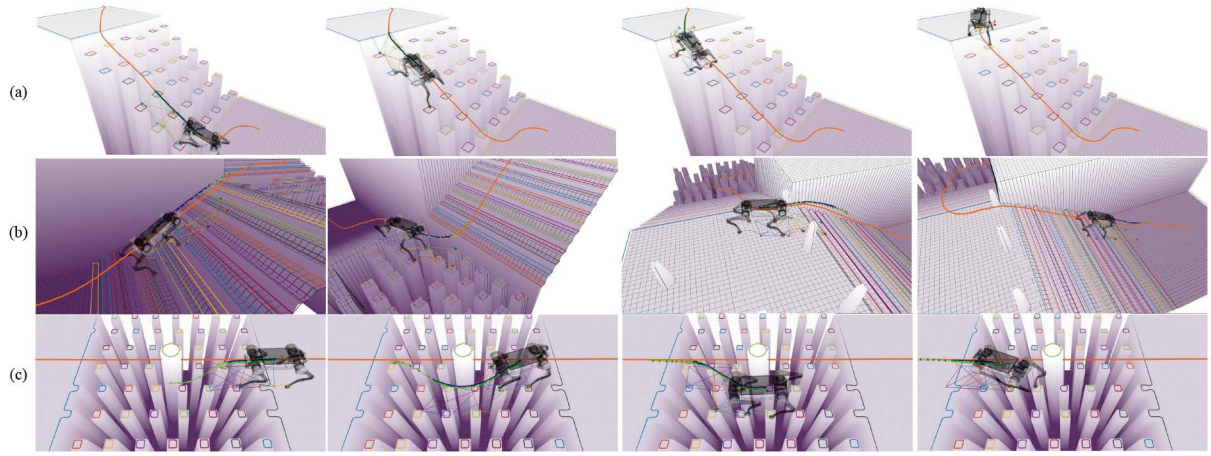


Fig. 9. Screenshots of different terrain experiments. (a) and (b) Robot following the planned path while traversing various terrains. (a) Legged robot climbs the steep stepping stones, with the path avoiding the steepest part. (b) In one experiment, the robot ascended stairs, crossed stepping stones, navigated through obstacle terrain, descended stairs, and then reached its destination. The entire path was approximately 15 m long. (c) Robot deviating from the path and executing obstacle avoidance planning when obstacles appear in 'hard' scenarios.

touch-down, leading to instability in legged systems. This is due to hard foothold constraints in the existing framework, which cause violations. Our approach places foothold constraints in the kinematic layer, improving solution quality by avoiding these violations. We tested our method against an ablation method under map drift during in-place trotting (see Fig. 8). Our approach shows better robustness to perception errors and smaller constraint violations compared to the ablation method.

2) Closed-Loop Experiment on Different Terrain: A set of experiments is run to demonstrate the robot's ability to traverse various terrains using our framework. The tests are designed to assess the robot's robustness and adaptability in different challenging terrains, as shown in Fig. 6.

3) Terrain Adaptability and Obstacles Avoidance: We evaluate the terrain adaptability of the ablation and comparison methods, as well as the success rate of the obstacle avoidance task. These methods are listed as follows.

a) *Ours - dynamic* is our framework *without* dynamic refinement.
b) *Ours - foothold* is our framework *without* foothold planning.
c) *Ours - kinematic* is our framework *without* kinematic refinement.
d) *Our Planner + RL* [6] builds upon the reinforcement learning (RL) framework with terrain perception introduced in [6], using the open-source implementation from [30]. We provide the RL controller with the path generated by our foothold-aware planner as input for navigation.
e) *Quad-SDK* [8] is the open-source quadruped motion framework.

Terrain adaptability is evaluated by testing the robot on various terrains to see if it remains stable or loses control. Each experiment is repeated three times. For obstacle avoidance, three difficulty levels are used: the "easy" level tests basic terrain traversal with sudden obstacles on stairs, the "medium" level involves steeper stairs to test planning and execution, and the "hard" level involves stepping stones to test both obstacle avoidance and accurate foothold placement.

The results are summarized in Table IV. First, without dynamic refinement (*ours-dynamic*), performance is poor due to the inability to optimize contact forces, causing instability even on flat terrain. Second, removing kinematic refinement (*ours-kinematic*) still allows terrain adaptation via the foothold planner, but the lack of long-horizon planning causes mismatches between foothold and torso trajectories, leading to failures in "hard" scenarios. Without the foothold planner (*our-foothold*), the system can only passively adapt and struggle on complex terrain. Third, the RL-based framework (*ours planner + RL*) demonstrates adaptability to various rough terrains. However, it fails on terrains requiring precise footholds, such as stepping stones and narrow bridges, due to footholds deviation. Although the RL controller is given both perception information and guidance from our foothold-aware planner, it struggles to effectively leverage this information for accurate foothold selection. A possible reason is that the RL framework treats the task as a control problem rather than a planning problem, making it difficult for the reactive controller to achieve precise end-effector placement and long-horizon planning. Finally, the open-source framework *Quad-SDK* fails because it only uses traversability scores, missing solutions for more challenging terrains.

D. Real World Experiments

1) Real-World Path Planning: When the stepping stones are normally placed, a traversable path over them can be planned. However, when the last two stones are removed, creating a gap that the robot cannot cross, the path planner will reroute the robot, as illustrated in the Fig. 6.

2) Planning Across Diverse Terrains: The different types of terrain in hardware test can be found in Figs. 1 and 10. The local map of the environment is built using point clouds generated by a Livox Mid-360 LiDAR sensor mounted on the robot's head. Our method was tested on terrains that can only be traversed by legged robots, such as a large gap and stepping stone. We

TABLE IV
TERRAIN ADAPTABILITY AND OBSTACLE AVOIDANCE SUCCESS RATE

Method	Terrain Adaptability						Obstacle Avoidance		
	Slope	Stair	Tubular	Rubble	St. stone	Narrow Br.	Easy	Medium	Hard
Ours - dynamic	×	×	×	×	×	×	0/10	0/10	0/10
Ours - foothold	✓	✓	×	✓	×	×	7/10	2/10	0/10
Ours - kinematic	✓	✓	✓	✓	✓	✓	**10/10**	5/10	1/10
Our planner+RL [6]	✓	✓	✓	✓	×	×	**10/10**	7/10	0/10
Quad-SDK [8]	✓	✓	×	✓	×	×	6/10	0/10	0/10
Ours	✓	✓	✓	✓	✓	✓	**10/10**	**10/10**	**7/10**

Bold values mean the best value in the table for the ablation/comparative experiments.

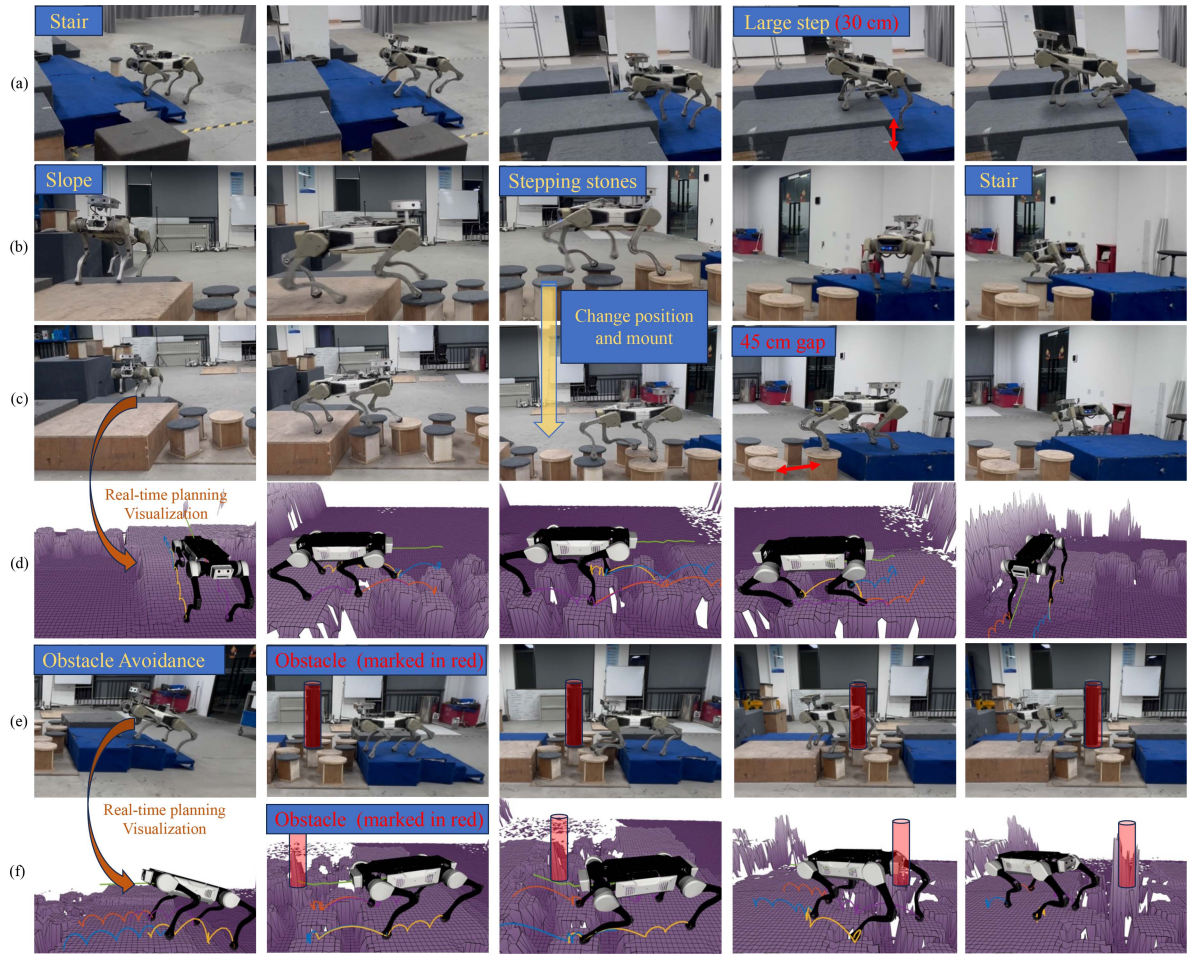


Fig. 10. Screenshots of different terrain experiments. (a),(b), and (c) Real robot across the various types of terrain. (d) Real-time planning visualization in Rviz. (e) and (f) Legged robot avoid the unknown obstacle on the stepping stone and its planning.

have incorporated five metrics that reflect the system's performance under various real-world conditions. These metrics are as follows:

a) Tracking root-mean-square error (RMSE) of foot swing trajectory, denoted as E_{sw} (cm).
b) RMSE of foothold placement, denoted as $E_{\text{rmse-f}}$ (cm).
c) Maximum foothold placement error, denoted as $E_{\text{max-f}}$ (cm).
d) Tracking RMSE of torso trajectory, denoted as E_{t} (cm).
e) Tracking RMSE of torso rotation, denoted as E_{ori} (rad).

Table V reports the statistical results of four experimental trials of each terrain, evaluating navigation accuracy and foot trajectory tracking error metrics across different terrains. Due to the high leg swing speed during locomotion, the RMSE

TABLE V
METRICS OF REAL-WORLD EXPERIMENT ACROSS DIFFERENT TERRAINS

Terrain	Foot Tracking Metrics			Torso Tracking Metrics	
	E_{sw}[cm]↓	$E_{\text{rmse-f}}$[cm]↓	$E_{\text{max-f}}$[cm]↓	E_{t}[cm]↓	E_{ori}[rad]↓
Flat	3.09 ± 0.39	1.06 ± 0.18	1.82 ± 0.47	1.58 ± 0.38	0.03 ± 0.00
Slope	3.20 ± 0.31	1.19 ± 0.37	1.96 ± 0.75	1.84 ± 0.30	0.08 ± 0.01
Stair	3.59 ± 0.82	1.24 ± 0.16	1.96 ± 0.17	2.41 ± 0.43	0.16 ± 0.06
St. Stone	4.90 ± 0.39	1.41 ± 0.17	3.60 ± 1.01	3.56 ± 0.49	0.04 ± 0.01

TABLE VI
COMPUTATION TIME IN REAL-WORLD EXPERIMENTS

Name	Mean [ms]	Max [ms]
Lidar Odom.	14.16	34.93
Mapping	35.18	108.74
MIP Footholds Planner	90.78	167.92
Torso Ref. Gen.	0.93	1.94
Kinematic refinement	8.25	24.47
Dynamic refinement	5.52	14.92
WBC & legged Odom.	0.52	0.89

of the swing leg trajectory tends to be relatively large. However, as the leg decelerates at the end of the swing phase and prepares to make contact with the terrain, our kinematic constraints and optimization effectively guide the robot to accurately track the planned foothold. At the same time, the robot's torso demonstrates strong tracking precision while traversing various terrains. Notably, on the most challenging terrains stepping stones, our system achieves a foothold tracking accuracy within 4 cm, which is smaller than the radius of the smallest stone (8 cm), thereby enabling successful traversal, demonstrate the real-world applicability of the proposed system.

3) Computation Performance: In the real-world experiment, we have two onboard computers. One is the NVIDIA Jetson AGX Orin 64 GB, which runs the perception part of the system, including Lidar Odometry and mapping. The other is a microcomputer with Intel i7-1185G7E CPU (4-Core, 1.80 GHz) and 32 GB of RAM, which runs the refinement parts, the WBC, and legged odometry (in one thread), and handles communication with the motor driver. Table VI presents the computation time statistics for each module of the pipeline during the real-world experiment. The foothold-aware path planner is a simple-based planner, and is set to search and plan for at most 2 s during each planning step. It can be seen that each part of the system can run in real-time.

4) Real World Obstacles Avoidance: We conducted real-world obstacle avoidance experiments, which serve as the physical counterpart to the simulation experiments described in Section VII-C3. As shown in Fig. 10(e) and (f), the robot ascended from the bottom of a staircase, detected obstacles, and successfully avoided it.

VIII. CONCLUSION

In this work, we propose foothold-based planning for legged robot autonomous navigation. The system aims to achieve: real-time, long-horizon foothold and trajectory planning and autonomous navigation over uneven terrain. Simulation and real-world experiments demonstrate the effectiveness of these aspects. The primary limitations of the system are: 1) its dependence on the quality of terrain mapping, as unclear map edges may result in missed steps; 2) the lack of full kinematic considerations in foothold planning. Although simplified kinematics was used due to real-time constraints, this approach still carries the risks of exceeding the robot's motion capabilities. Future work will focus on improving the stability and robustness of the planning and motion control.

REFERENCES

[1] C. D. Bellicoso et al., "Advances in real-world applications for legged robots," *J. Field Robot.*, vol. 35, no. 8, pp. 1311–1326, 2018.

[2] J. Hooks et al., "ALPHRED: A multi-modal operations quadruped robot for package delivery applications," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5409–5416, Oct. 2020.

[3] T. Corbères et al., "Perceptive locomotion through whole-body MPC and optimal region selection," *IEEE Access*, vol. 13, pp. 69062–69080, 2025.

[4] F. Jenelten, R. Grandia, F. Farshidian, and M. Hutter, "TAMOLS: Terrain-aware motion optimization for legged systems," *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3395–3413, Dec. 2022.

[5] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model-predictive control," *IEEE Trans. Robot.*, vol. 39, no. 5, pp. 3402–3421, Oct. 2023.

[6] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Sci. Robot.*, vol. 7, no. 62, 2022, Art. no. eabk2822.

[7] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krüsi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *Proc. 2016 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 1184–1189.

[8] J. Norby et al., "Quad-SDK: Full stack software framework for agile quadrupedal locomotion," in *Proc. ICRA Workshop Legged Robots*, May 2022, pp. 1–5.

[9] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1560–1567, Jul. 2018.

[10] A. Hereid and A. D. Ames, "FROST: Fast robot optimization and simulation toolkit," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 719–726.

[11] A. Hereid, O. Harib, R. Hartley, Y. Gong, and J. W. Grizzle, "Rapid trajectory optimization using C-FROST with illustration on a Cassie-series dynamic walking biped," in *Proc. 2019 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4722–4729.

[12] V. S. Medeiros, E. Jelavic, M. Bjelonic, R. Siegwart, M. A. Meggiolaro, and M. Hutter, "Trajectory optimization for wheeled-legged quadrupedal robots driving in challenging terrain," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4172–4179, Jul. 2020.

[13] M. Brandao, O. B. Aladag, and I. Havoutis, "GaitMesh: Controller-aware navigation meshes for long-range legged locomotion planning in multi-layered environments," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3596–3603, Apr. 2020.

[14] B. Yang, J. Cheng, B. Xue, J. Jiao, and M. Liu, "Efficient global navigational planning in 3-D structures based on point cloud tomography," *IEEE/ASME Trans. Mechatron.*, vol. 30, no. 1, pp. 321–332, Feb. 2024.

[15] J. Norby and A. M. Johnson, "Fast global motion planning for dynamic legged robots," in *Proc. 2020 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 3829–3836.

[16] E. Jelavic, K. Qu, F. Farshidian, and M. Hutter, "LSTP: Long short-term motion planning for legged and legged-wheeled systems," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4190–4210, Dec. 2023.

[17] M. Gaertner, M. Bjelonic, F. Farshidian, and M. Hutter, "Collision-free MPC for legged robots in static and dynamic scenes," in *Proc. 2021 IEEE Int. Conf. Robot. Autom.*, 2021, pp. 8266–8272.

[18] J.-R. Chiu, J.-P. Sleiman, M. Mittal, F. Farshidian, and M. Hutter, "A collision-free MPC for whole-body dynamic locomotion and manipulation," in *Proc. 2022 Int. Conf. Robot. Autom.*, 2022, pp. 4686–4693.

[19] Q. Liao, Z. Li, A. Thirugnanam, J. Zeng, and K. Sreenath, "Walking in narrow spaces: Safety-critical locomotion control for quadrupedal robots with duality-based optimization," in *Proc. 2023 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 2723–2730.

[20] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, "Perceptive locomotion in rough terrain–online foothold optimization," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5370–5376, Oct. 2020.

[21] P. Fankhauser and M. Hutter, "A universal grid map library: Implementation and use case for rough terrain navigation," in *Robot Operating System (ROS)–The Complete Reference (Volume 1)*, A. Koubaa, Ed. Berlin, Germany: Springer, 2016, ch. 5. [Online]. Available: http://www.springer.com/de/book/9783319260525

[22] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping, and whole-body control," *IEEE Trans. Robot.*, vol. 36, no. 6, pp. 1635–1648, Dec. 2020.

[23] S. Tonneau, D. Song, P. Fernbach, N. Mansard, M. Taïx, and A. Del Prete, "SL1M: Sparse l1-norm minimization for contact planning on uneven terrain," in *Proc. 2020 IEEE Int. Conf. Robot. Autom.*, 2020, pp. 6604–6610.
[24] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback MPC for torque-controlled legged robots," in *Proc. 2019 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4730–4737.
[25] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system," (n.d.). [Online]. Available: https://www.ros.org
[26] I. A. ucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.
[27] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2023. [Online]. Available: https://www.gurobi.com
[28] F. Farshidian et al., "OCS2: An open source library for optimal control of switched systems," (n.d.). [Online]. Available: https://github.com/leggedrobotics/ocs2
[29] J. Carpentier, N. Mansard, F. Valenza, J. Mirabel, G. Saurel, and R. Budhiraja, "Pinocchio-efficient and versatile rigid body dynamics algorithms," 2021. [Online]. Available: https://github.com/stack-of-tasks/pinocchio
[30] lnotspotl, "Tbai: Towards better athletic intelligence," 2024, gitHub repository. [Online]. Available: https://github.com/lnotspotl/tbai

Jiyu Yu received the bachelor's degree in electrical engineering and automation from Xi'an Jiaotong University, Xi'an, China, in 2021. He is currently working toward the Ph.D. degree in control science and engineering with the Institute of Cyber-Systems and Control, Department of Control Science and Engineering, Zhejiang University, Hangzhou, China.

His research interests include optimal control, motion planning, and whole-body control for legged systems.

Dongqi Wang received the bachelor's degree in vehicle engineering from Beijing Jiaotong University, Beijing, China, in 2019. He is currently working toward the Ph.D. degree in solids mechanics with Department of Engineering Mechanics, Zhejiang University, Hangzhou, 310027, China.

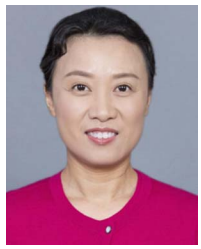
His current research interests include optimal control, motion planning, and whole-body control for legged systems

Zhenghan Chen received the B.E. degree in automation from the Harbin Institute of Technology, Shenzhen, China, in 2023. He is currently working toward the M.S. degree in control science and engineering with the Institute of Cyber-Systems and Control, Department of Control Science and Engineering, Zhejiang University, Hangzhou, China.

His research interests include humanoid robots, control, and reinforcement learning.

Ci Chen received the B.S. degree in agricultural mechanization and automation from Northeast Agricultural University, Harbin, China, in 2018, and the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2024. She is currently a postdoctoral researcher with the School of Automation and Intelligent Sensing, Shanghai Jiao Tong University.

Her research interests include deep reinforcement learning and legged robots.



Rong Xiong (Senior Member, IEEE) received the Ph.D. degree in control science and engineering from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2009.

She is currently a Professor with the Department of Control Science and Engineering, Zhejiang University. Her current research interests include motion planning, SLAM, and legged robot.

Yue Wang (Member, IEEE) received the Ph.D. degree in control science and engineering from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2016.

He is currently a Professor with the Department of Control Science and Engineering, Zhejiang University. His current research interests include mobile robotics and robot perception.