Stable Trajectory Planning for Quadruped Robots Using Terrain Features at Feet End

Congfei Li, Shuyue Lin, *Member, IEEE*, Shenwei Qu, Zhuoyuan Liu, Qingjun Yang, Max Q.-H. Meng, *Fellow, IEEE*, and Yuxiang Sun, *Member, IEEE*

***Abstract*—Quadruped robots have received increasing attention in recent years. Most existing trajectory planning algorithms for quadruped robots focus on how to avoid obstacles and achieve shortest trajectory or time, which is similar to the planning algorithms for mobile robots. These algorithms could not take full advantage of the agility and flexibility of quadruped robots. This letter designs a trajectory planner by taking advantage of the agility and flexibility of quadruped robots. With our trajectories, quadruped robots could navigate through complex terrains with more stability (e.g., less momentum variations along Z-axis). To achieve this goal, we use ground features at the landing point of the feet end to construct objective function, rather than using the center point of the robot body. Current discrete map representations, such as grid map or cost map, are difficult for optimization algorithms to introduce environment constraints. So, we use the Sparse Variational Gaussian Process (SVGP) to predict terrain features with point-cloud data as input, so that the environment constraints can be introduced into the optimization problem. Experimental results in both simulation and real-world environments demonstrate the effectiveness of our method.**

***Index Terms*—Trajectory planning, quadruped robots, Gaussian process, stochastic optimization.**

I. Introduction

With the advancement of quadruped controllers (model-based controllers [1], [2], [3] and learning-based controllers [4], [5]), quadruped robots are now able to traverse highly irregular terrains. However, trajectory planning for quadruped robots is still largely inspired by wheeled-robot navigation and has not fully exploited the agility and flexibility of legged locomotion. Most existing planners focus on minimizing path length or travel time while avoiding obstacles, making them conceptually similar to mobile-robot planners. These methods typically rely on occupancy grid maps, where each cell is occupied, free, or unknown [6], and thus generate paths that are short in distance or time [7]. Such grid-based representations are suitable for wheeled robots [8], but less appropriate for quadruped robots because they cannot capture foothold-level terrain interactions.

Recent studies have begun to address this limitation by incorporating terrain or foothold information into planning and control. For instance, vision-based terrain-aware locomotion [9] jointly adjusts footholds and body pose based on perception; anisotropy-aware trajectory optimization [10] considers directional motion capability; and reinforcement-learning frameworks [11] adapt step height and gait frequency using terrain feedback. While these methods improve adaptability, they mainly operate at the joint or foothold level. By contrast, this work focuses on body-level trajectory planning that explicitly incorporates foot-end terrain features during planning. This higher-level formulation allows the planner to reason about local slopes and small-scale topography, enabling smoother and more agile motion on complex terrains. Therefore, unlike grid-based methods, such as A-star (A*), Rapidly-exploring Random Tree (RRT), or Rapidly-exploring Random Tree Star (RRT*), which only consider passable or blocked cells, and unlike joint-level optimizers that refine foothold control, our approach generates body trajectories informed by foot-end terrain geometry, offering a complementary and novel perspective on terrain-aware locomotion planning.

To provide a solution to above issues, we propose a trajectory planning method for quadruped robots based on continuous and differentiable Gaussian maps as well as the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [12]. The Sparse Variational Gaussian Process (SVGP) [13] is utilized to fit the terrain. We construct an optimization problem by taking into account the terrain features at feet end with a simplified kinematic model. This problem is a non-convex continuous problem filled with local minima. So, solving it with CMA-ES is suitable. Our method views the waypoints of a trajectory as decision variables of the problem, and models the environment for the problem as a constraint. Our code is open-sourced.[1] The contributions of this letter are summarized as follows.

1) We construct an optimization problem for quadruped trajectory planning, which takes into account the ground features at feet-end positions.

Received 4 July 2025; accepted 24 October 2025. Date of publication 17 December 2025; date of current version 9 January 2026. This article was recommended for publication by Associate Editor J. Luo and Editor A. Kheddar upon evaluation of the reviewers' comments. This work was supported by the City University of Hong Kong under Grant 9610675. *(Corresponding author: Yuxiang Sun.)*

Congfei Li is with the School of Mechatronics Engineering, Harbin Institute of Technology, Harbin 150001, China, and also with the Department of Mechanical Engineering, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong (e-mail: congfei.li@my.cityu.edu.hk).

Shuyue Lin and Yuxiang Sun are with the Department of Mechanical Engineering, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong (e-mail: shuyue.lin@cityu.edu.hk; yx.sun@cityu.edu.hk).

Shenwei Qu and Qingjun Yang are with the School of Mechatronics Engineering, Harbin Institute of Technology, Harbin 150001, China (e-mail: 23b908070@stu.hit.edu.cn; yqj@hit.edu.cn).

Zhuoyuan Liu is with the Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (e-mail: zhuo-yuan.liu@connect.polyu.hk).

Max Q.-H. Meng is with the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: max.meng@ieee.org).

Digital Object Identifier 10.1109/LRA.2025.3645657

[1]Code and video: https://github.com/lab-sun/Feetend-Planner

2) We employ SVGP to describe small-scale and static terrains, which introduces differentiable environment models into the optimization problem as a constraint.
3) We conduct both simulation and real-world experiments to verify the effectiveness of the proposed method.

II. Related Work

A. Trajectory Planning

Due to the simplicity of occupancy maps, many planners are designed with occupancy maps, such as the classic A* and RRT* algorithms, and many other improved versions [14], [15]. In addition to the sample-based and search-based methods, many methods have been proposed to endow robots with better mobility. Yu et al. [16] utilized mixed integer programming to plan footholds for navigating discrete terrains. Yang et al. [17] proposed a novel planner in multilayer structures based on tomographic understanding of the environment. Ortiz et al. [7] proposed a RRT-based algorithm combined motion primitives and optimization, 10 times faster than previous methods. Liu et al. [18] adopted a hierarchical architecture and decomposed the planning process into the search for front- and back-end optimization. With the development of machine learning, researchers have designed some learning-based methods to find a sensible path to guide their mobile robots. Liu et al. [19] used the diffusion model to derive a fast 2D path planner whose goal is to find a way through a maze. Comparing with these methods, our method could find a relatively easy path for controller to track, which takes full use of the non-linear model predictive control (NMPC). Several works have further introduced terrain features into the motion planning process of quadruped robots [9], [10], [11]. These methods mainly focus on improving foothold or contact-level reasoning, where local terrain cues help the robot determine stable foot placements. In contrast, our method leverages terrain cues at a body trajectory level, explicitly optimizing the robot's body motion to exploit terrain geometry for safer and smoother traversal.

B. Terrain Description

Environment modeling is essential for mobile robot trajectory planning, with four primary methods widely used: Grid Method (GM), Topological Method (TM), and Geometric Characteristic Method (GCM).

GM divides the workspace into equal-sized grids, marking free spaces as 0 and obstacles as 1. Smaller grids offer higher precision but require more computational resources, while larger grids improve efficiency at the cost of reduced accuracy. Recent advances, such as hexagonal grids, improve path planning safety and efficiency [20]. GM is simple to implement and extend to 3D environments, but faces scalability issues in large areas due to combinatorial explosion.

TM represents environments as graphs, using nodes for locations and edges for connections, often visualized through Voronoi diagrams. It reduces dimensionality, saving storage and computation time, making it suitable for large-scale environments [21]. However, TM is challenging to create and maintain, especially in environments with similar locations, and may produce suboptimal paths.

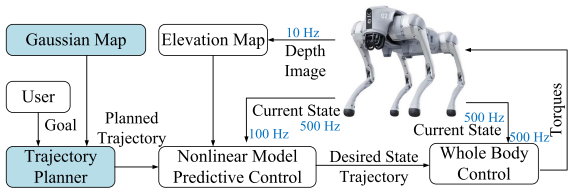

Fig. 1. The overview of our method. The combination of the middle-level NMPC controller and low-level whole-body controller are robust model-based controllers [24], [25]. With the perception information from the elevation map, the quadruped robot could select suitable foot end points and go through complex terrains. The main contributions in this letter are highlighted in blue.

GCM extracts geometric features such as lines and arcs to build compact environment models, ideal for structured environments [22]. However, extracting stable features from unstructured environments is difficult, limiting its applicability. Techniques such as obstacle shape-based path planning have been proposed to address these challenges [23].

C. Difference From Existing Methods

Our method differs from existing methods in the following two aspects: 1) we focus more on the terrain features at the feet end of quadruped robots to compute the cost of walking across the local area, which may take full advantage of quadruped controllers; 2) we model small-scale terrain areas with SVGP to incorporate the environment information into optimization as a constraint.

III. The Proposed Method

A. Method Overview

The overview of our proposed method is shown in Fig. 1. A human user sets a goal, and the planner finds the optimal solution according to the given Gaussian description of the current environment. Then, the trajectory of the quadruped robot body is sent to a NMPC controller to optimize the trajectories of all the joints of the robot. The optimized motion is passed to the low-level whole-body controller to generate torques for each actuated joint. Note that the robot provides depth information to the perception module and the state information to the controllers.

B. Terrain Modeling

We use SVGP to construct the terrain model. We first introduce the Gaussian process (GP) and then convert to SVGP. GP is a collection of random variables, for which any finite subset follows a Gaussian distribution. It is a powerful nonparametric Bayesian tool used for regression and classification in machine learning [26]. GP is fully specified by its mean function and covariance function (also known as the kernel). The mean function represents the expected value of the process, while the covariance function defines the covariance between different pairs of random variables, capturing the smoothness and structure of the functions that the GP models. GP is particularly useful for modeling complex, unknown functions, and providing uncertainty estimates along with predictions.

We define $\mathbf{X}$ as the vector of $\mathbf{x_i}$ and $\mathbf{h}$ is the vector of h_i:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n], \mathbf{h} = [h_1, h_2, \ldots, h_n]. \quad (1)$$

We train the Gaussian map on a given pre-built point-cloud map $\mathcal{P} = \{(\mathbf{x}_i, h_i)\}_{i=1}^{n}$, where $\mathbf{x_i} = (x_i, y_i)$ is the random variable that consists of the coordinate on the X-Y plane, and $h_i = z_i$ is the ground truth. GP is formally defined as a collection of random variables characterized by a mean function $\mu(\mathbf{x})$ and a kernel function $\kappa(\mathbf{x}, \mathbf{x}')$. Assume that the regressed GP model is $\Gamma(\mathbf{x})$ and $\Gamma(\mathbf{x}) \sim \mathcal{GP}[\mu(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')]$.

Define $\mathbf{\Gamma} = [\Gamma_1, \Gamma_2, \ldots, \Gamma_n]^T$ as the vector of the latent function value at the training point, where $\Gamma_i = \Gamma(\mathbf{x}_i)$. Given the dataset $\mathcal{P}$ with N observations $\mathbf{x}$ and scalar outputs h, GP regression assumes $h_i = \Gamma_i + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma_{\mathcal{P}}^2)$ is the noise term, reflecting the measurement noise. The posterior GP is defined by a posterior mean $\mu_{\mathcal{P}}(\mathbf{x})$ and covariance $\Sigma_{\mathcal{P}}(\mathbf{x}, \mathbf{x}')$. The predictions for a new input $\mathbf{x}^*$ follow:

$$p(h^*|\mathbf{h}) = \mathcal{N}\left[h^*|\mu_{\mathcal{P}}(\mathbf{x}^*), \Sigma_{\mathcal{P}}(\mathbf{x}^*, \mathbf{x}^*) + \sigma_{\mathcal{P}}^2\right], \tag{2}$$

$$\mu_{\mathcal{P}}(\mathbf{x}) = \mathbf{C}_{\mathbf{x}\mathcal{P}}(\sigma^2\mathbf{I} + \mathbf{C}_{\mathcal{P}\mathcal{P}})^{-1}\mathbf{h}, \tag{3}$$

$$\Sigma_{\mathcal{P}}(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x}, \mathbf{x}') - \mathbf{C}_{\mathbf{x}\mathcal{P}}(\sigma^2\mathbf{I} + \mathbf{C}_{\mathcal{P}\mathcal{P}})^{-1}\mathbf{C}_{\mathcal{P}\mathbf{x}'}, \tag{4}$$

where $\mathbf{C}_{\mathcal{P}\mathcal{P}} \in \mathbb{R}^{N\times N}$ is the covariance matrix of the training inputs, $\mathbf{C}_{\mathbf{x}\mathcal{P}} \in \mathbb{R}^{1\times N}$ is a row vector of kernel evaluations between $\mathbf{x}$ and training inputs, and $\mathbf{C}_{\mathcal{P}\mathbf{x}} = \mathbf{C}_{\mathbf{x}\mathcal{P}}^T$, and $\mathbf{h} = [h_1, h_2, \ldots, h_n]^T$ is the ground truth vector. The GP prediction is in the form of a probability distribution with the mean $\mu_{\mathcal{P}}(\mathbf{x}^*)$, which indicates the elevation at the point $\mathbf{x}^*$, and a variance $\Sigma_{\mathcal{P}}(\mathbf{x}^*, \mathbf{x}^*)$, indicating the uncertainty of the prediction at $\mathbf{x}^*$.

GP predictions depend on kernel hyperparameter $\mathbf{\Phi}$ and noise variance $\sigma_{\mathcal{P}}^2$. Hyperparameters $(\mathbf{\Phi}, \sigma_{\mathcal{P}}^2)$ are estimated by maximizing the logarithmic marginal likelihood, $\log[p(\mathbf{h})] = \log[\mathcal{N}(\mathbf{h}|\mathbf{0}, \sigma_{\mathcal{P}}^2\mathbf{I} + \mathbf{C}_{\mathcal{P}\mathcal{P}})]$. Once we have obtained the hyperparameters $(\mathbf{\Phi}, \sigma_{\mathcal{P}}^2)$, we can use these estimates in (2)–(4) to predict the regressed value of the unseen input point.

The computational complexity of standard GP is $\mathcal{O}(n^3)$, where n is the number of sample points. Many approximation approaches could reduce the computational complexity, such as SVGP. SVGP reduces the cost by replacing the full dataset with m points ($m < n$), resulting in a lower computation complexity of $\mathcal{O}(nm^2)$. The m points are called inducing points $\tilde{\mathcal{P}} = \{(\tilde{\mathbf{x}}_i, \tilde{h}_i)\}_{i=1}^{m}$ which is a subset of the dataset $\mathcal{P}$ and their corresponding regressed function values are $\tilde{\Gamma}_i = \Gamma(\tilde{\mathbf{x}}_i), i = 1, 2, \ldots, m$, called inducing variables $\tilde{\mathbf{\Gamma}} = [\tilde{\Gamma}_1, \tilde{\Gamma}_2, \ldots, \tilde{\Gamma}_m]^T$. The hyperparameters $(\mathbf{\Phi}, \sigma_{\mathcal{P}}^2)$ and the inducing points X_m are jointly estimated through a variational posterior distribution which approximates the exact GP posterior, $q(\mathbf{\Gamma}, \tilde{\mathbf{\Gamma}}) = p(\mathbf{\Gamma}|\tilde{\mathbf{\Gamma}})\psi(\tilde{\mathbf{\Gamma}})$, where $\psi(\tilde{\mathbf{\Gamma}})$ is an unconstrained variational distribution over $\tilde{\mathbf{\Gamma}}$ and $p((\mathbf{\Gamma}|\tilde{\mathbf{\Gamma}})$ is the conditional GP prior. We could obtain the estimation of inducing points $\tilde{\mathcal{P}}$ and hyperparameters $(\mathbf{\Phi}, \sigma_{\mathcal{P}}^2)$ by minimizing the Kullback-Leibler (KL) divergence between $q(\mathbf{\Gamma}, \tilde{\mathbf{\Gamma}})$ and $p(\mathbf{\Gamma}|\mathbf{h}, \mathbf{\Phi})$, which is equivalent to maximizing the evidence lower bound (ELBO):

$$\mathcal{L}_{\text{vof}}(\tilde{\mathcal{P}}) = \log\left[\mathcal{N}(\mathbf{h}|\mathbf{0}, \sigma_{\mathcal{P}}^2\mathbf{I} + \mathbf{P}_{nm})\right] - \frac{1}{2\sigma_{\mathcal{P}}^2}Tr(\mathbf{R}), \tag{5}$$

$$\mathbf{P}_{nm} = \mathbf{C}_{n\tilde{\mathcal{P}}}\mathbf{C}_{\tilde{\mathcal{P}}\tilde{\mathcal{P}}}^{-1}\mathbf{C}_{\tilde{\mathcal{P}}n}, \mathbf{R} = \mathbf{C}_{nn} - \mathbf{C}_{n\tilde{\mathcal{P}}}\mathbf{C}_{\tilde{\mathcal{P}}\tilde{\mathcal{P}}}^{-1}\mathbf{C}_{\tilde{\mathcal{P}}n}, \tag{6}$$

where $\mathcal{L}_{\text{vof}}(\tilde{\mathcal{P}})$ is the variational objective function, $Tr(\mathbf{R})$ is a regularization trace term, $\mathbf{C}_{\tilde{\mathcal{P}}\tilde{\mathcal{P}}}$ and $\mathbf{C}_{n\tilde{\mathcal{P}}}$ denote the induction covariance matrix and the cross-covariance between the training and induction inputs, respectively.

The rational quadratic (RQ) covariance function is chosen as the kernel function $\kappa_{RQ}(\mathbf{x}, \mathbf{x}')$ for the SVGP, because a Gaussian process prior with a rational quadratic kernel captures functions exhibiting variations at multiple length scales:

$$\kappa_{RQ}(\mathbf{x}, \mathbf{x}') = \sigma_k^2\left[1 + \frac{(\mathbf{x} - \mathbf{x}')^2}{2\alpha_k l^2}\right]^{-\alpha_k}, \tag{7}$$

where σ_k represents the signal variance, l is the length scale and α_k determines the relative weighting between large-scale and small-scale variations. Given the RQ kernel, the kernel parameters $\mathbf{\Phi}$ are defined as σ_k^2, l and α_k. We believe that the RQ kernel offers greater expressiveness in modeling the occupancy surface than the widely-used squared exponential (SE) kernel. This is because the RQ kernel effectively represents a scaled mixture of SE kernels with diverse characteristic length scales.

C. Trajectory Optimization

In this section, we propose an optimization problem involving environment as a constraint and solve the problem with stochastic optimization, that is, covariance matrix adaptation evolution strategy. We choose the set of position of the robot center as optimal variables ${}^{\mathbf{r}}\mathbf{x}$. Assuming that the quadruped robot walks with the trot gait, the quadruped robot steps ahead twice in a gait duration T. Because the NMPC controller would select the foot end points according to the elevation map, we just use the definition of the trot as the assumption of our planner. We define the decision variable as:

$${}^{\mathbf{r}}\mathbf{x} = [{}^{r}x_0, {}^{r}x_1, \ldots, {}^{r}x_N, {}^{r}y_0, {}^{r}y_1, \ldots, {}^{r}y_N]^T, \tag{8}$$

where N is the number of steps rather than the number of gait period which is twice that of steps. Thus, we can account for the influence of the environment on the mobility of robots more intricately. ${}^{r}x_i$ and ${}^{r}y_i$ respectively denote the X-Y coordinates of the robot position at each step in the world coordinate. As for the orientation of the robot, we add an additional part of objective function to smooth the trajectory, reducing the computational burden. Given the initial position $[{}^{r}x_{\text{initial}}, {}^{r}y_{\text{initial}}]^T$ and goal position $[{}^{r}x_{\text{goal}}, {}^{r}y_{\text{goal}}]^T$, the number of steps:

$$N = \max\left(\left\lceil\frac{{}^{r}x_{\text{goal}} - {}^{r}x_{\text{initial}}}{v_x T/2}\right\rceil, \left\lceil\frac{{}^{r}y_{\text{goal}} - {}^{r}y_{\text{initial}}}{v_y T/2}\right\rceil\right), \tag{9}$$

where T is the gait period and v_x and v_y are the forward and lateral velocity, respectively.

The main purpose of the optimization problem is to select a relatively even trajectory for the quadruped robot. Therefore, the main part of the objective function $\mathcal{L}$ is to minimize the height of environment on the trajectory $\mathcal{L}_H$. To enable the optimization solver to find the reasonable solution, there are also other parts of objective functions, that is, terminate objective $\mathcal{L}_T$, step length objective $\mathcal{L}_\Delta$ and orientation objective $\mathcal{L}_\theta$:

$$\mathcal{L}({}^{\mathbf{r}}\mathbf{x}) = \mathcal{L}_H({}^{\mathbf{r}}\mathbf{x}) + \mathcal{L}_T({}^{\mathbf{r}}\mathbf{x}) + \mathcal{L}_\Delta({}^{\mathbf{r}}\mathbf{x}) + \mathcal{L}_\theta({}^{\mathbf{r}}\mathbf{x}). \tag{10}$$

1) Height Objective: We sample the height of four points, all directly below four hips, and calculate the height of these four points on the trajectory as height objective:

$$\mathcal{L}_H({}^{\mathbf{r}}\mathbf{x}) = \mathbf{z}_b^T\mathbf{W}_{\mathbf{b}}\mathbf{z}_b + \mathbf{z}_h^T\mathbf{W}_{\mathbf{h}}\mathbf{z}_h, \tag{11}$$

where $\mathbf{z}_b = \Gamma({}^{\mathbf{r}}\mathbf{x})$, $\mathbf{z}_h = \Gamma[\Omega({}^{\mathbf{r}}\mathbf{x})]$, $\Omega({}^{\mathbf{r}}\mathbf{x}) : \mathbb{R}^{2N} \to \mathbb{R}^{8N}$. $\Omega({}^{\mathbf{r}}\mathbf{x})$ is a function that converts body coordinate to four hips' coordinate stack. $\mathbf{z}_b$ is the height at position ${}^{\mathbf{r}}\mathbf{x}$. $\mathbf{z}_h$ is the height stack at position of each hip. This is also a trick to speed up the optimization process. $\mathbf{W}_b \in \mathbb{R}^{2N\times 2N}$ and $\mathbf{W}_h \in \mathbb{R}^{8N\times 8N}$ are constant matrices p.s.d. weight matrices. This is the key objective to involve the environment into optimization problem. It should be noted that this term is not introduced as a direct stability metric; rather, it penalizes unnecessarily large hip elevations in order to reduce traversal difficulty, thereby promoting more feasible locomotion.

2) Terminate Objective: The terminate objective is:

$$\mathcal{L}_T({}^{\mathbf{r}}\mathbf{x}) = ({\mathbf{S}}^{\mathbf{r}}\mathbf{x})^T W_T({\mathbf{S}}^{\mathbf{r}}\mathbf{x}) \tag{12}$$

where $\mathbf{S} \in \mathbb{R}^{1\times 2N}$ is a selective matrix and $W_T \in \mathbb{R}$ is the weight factor.

3) Step Length Objective: In order to help solver find the optimal solution, we never allow our robot to step back. Therefore, we set ${}^{r}x_{i+1} - {}^{r}x_i \geq 0$. Meanwhile, The step length between two steps cannot exceed $\mathbf{v}T/2$, i.e. ${}^{\mathbf{r}}\mathbf{x}_{i+1} - {}^{\mathbf{r}}\mathbf{x}_i \leq \mathbf{v}T/2$, where ${}^{\mathbf{r}}\mathbf{x}_i = [{}^{r}x_i, {}^{r}y_i]^T$. This inequality constraint is handled through the relaxed barrier functions that convert inequality constraint to cost:

$$\mathrm{B}(l) = \begin{cases} -\mu\ln(l), & h \geq \delta \\ \frac{\mu}{2}\left[(\frac{l-2\delta}{\delta})^2 - 1\right] - \mu\ln(\delta), & h < \delta \end{cases} \tag{13}$$

Therefore, the step length objective is:

$$\mathcal{L}_\Delta({}^{\mathbf{r}}\mathbf{x}) = \mathrm{B}(\mathbf{l}_s) = \sum_i \mathrm{B}(\mathbf{l}_{s,i}). \tag{14}$$

where

$$\mathbf{l}_s = \begin{bmatrix}\mathbf{A_1}\\ \mathbf{A_2}\end{bmatrix} {}^{\mathbf{r}}\mathbf{x} + \begin{bmatrix}\mathbf{B_1}\\ \mathbf{B_2}\end{bmatrix} \geq \mathbf{0},$$

$$\mathbf{A}_1 = \begin{bmatrix}\mathbf{A}_s\\ \mathbf{0}\end{bmatrix}_{2N\times 2N}, \mathbf{A}_2 = \begin{bmatrix}-\mathbf{A}_s\\ -\mathbf{A}_s\end{bmatrix}_{2N\times 2N}, \mathbf{B_1} = \mathbf{0}_{2N\times 1},$$

$$\mathbf{B_2} = [v_xT/2\ \ldots v_xT/2\ v_yT/2\ \ldots\ v_yT/2]^T_{1\times 2N},$$

$$\mathbf{A}_s = \begin{bmatrix} -1 & 1 & 0 & \ldots & 0 & 0 \\ 0 & -1 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & -1 & 1 \end{bmatrix}_{N\times 2N}.$$

4) Orientation Objective: In order to smooth the trajectory curve, we penalize the local orientation and the difference of the adjacent orientation, respectively:

$$\mathcal{L}_{\boldsymbol{\theta}}({}^{\mathbf{r}}\mathbf{x}) = \boldsymbol{\theta}^T\mathbf{W}_{\boldsymbol{\theta}}\boldsymbol{\theta} + \Delta\boldsymbol{\theta}^T\mathbf{W}_{\Delta\boldsymbol{\theta}}\Delta\boldsymbol{\theta} \tag{15}$$

where $\boldsymbol{\theta}$ is the forward direction of the robot, $\Delta\boldsymbol{\theta}$ is the difference of the adjacent orientation. $\mathbf{W}_{\boldsymbol{\theta}}$ and $\mathbf{W}_{\Delta\boldsymbol{\theta}}$ are the weight matrices, respectively.

D. CMA-ES

CMA-ES is an evolutionary algorithm designed for nonlinear, non-convex black-box optimization problems in continuous domains. It has been widely used for continuous optimization. There are 4 main steps in the loop of CMA-ES:

Algorithm 1: CMA-ES Workflow.

Input: Initial distribution mean $\mathbf{m}^0$, Objective function $\mathcal{L}$

Output: Optimized solution ${}^{\mathbf{r}}\mathbf{x}$

Initialize: $\mathbf{p}_\mathbf{c}^\mathbf{0}$, $\mathbf{p}_\sigma^\mathbf{0}$, $\mathbf{C}^0$, σ^0, $g \leftarrow 0$
Set parameters: λ, c_m, c_c, c_1, c_σ, μ, w, and d_σ
while *termination criterion not met* **do**
  for $k = 1$ **to** λ **do**
    Sample new solutions according to (16);
  end
  Select the best solutions according to (10) ;
  Reorder the sampled solutions according to (18) ;
  Adapt the covariance matrix according to (19) ;
  Update the step size according to (21) ;
  $g \leftarrow g + 1$;
end

1) Sampling New Solutions: Sample the next generation with a multivariate normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{C})$:

$${}^{\mathbf{r}}\mathbf{x}_k^{g+1} \sim \mathbf{m}^g + \sigma^g\mathcal{N}(\mathbf{0}, \mathbf{C}^g),\ \ k = 1, \ldots, \lambda, \tag{16}$$

where ${}^{\mathbf{r}}\mathbf{x}_k^{g+1}$ is the k-th new solution at generation $g+1$, $\mathbf{m}^g$ is the mean value vector of the distribution at generation g, σ^g is the step size at generation g, $\mathbf{C}^g$ is the covariance matrix at generation g and λ is the sample size.

2) Reordering the Sampled Solutions: Select and recombine the top performing solutions of generation g. The new mean $\mathbf{m}^{g+1}$ is computed by recombining the current mean $\mathbf{m}^g$ with the hightest-fitness solutions selected from the current population based on their objective function values:

$$\mathbf{m}^{g+1} = \mathbf{m}^g + c_m\sum_{i=1}^{\mu} w_i({}^{\mathbf{r}}\mathbf{x}_{i:\lambda}^{g+1} - \mathbf{m}^g), \tag{17}$$

where ${}^{\mathbf{r}}\mathbf{x}_{i:\lambda}^{g+1}$ is the i-th best solution (ranked by $\mathcal{L}({}^{\mathbf{r}}\mathbf{x}_{1:\lambda}^{g+1}) \leq \mathcal{L}({}^{\mathbf{r}}\mathbf{x}_{2:\lambda}^{g+1}) \leq \cdots \leq \mathcal{L}({}^{\mathbf{r}}\mathbf{x}_{\lambda:\lambda}^{g+1})$), c_m is the learning rate, and μ is the selection size. w_i are recombination weights, satisfying that $\sum_{i=1}^{\mu} w_i = 1, w_1 \geq w_2 \geq \cdots \geq w_\mu > 0$.

3) Adapting the Covariance Matrix: The covariance matrix adaptation employs a dual update mechanism, including a rank-one correction guided by the evolution path and a rank-μ correction:

$$\mathbf{C}^{g+1} = \mathbf{C}^g + c_1\mathbf{p}_c^{g+1}(\mathbf{p}_c^{g+1})^T + c_\mu\sum_{i=1}^{\lambda} w_i\mathbf{y}_{i:\lambda}^{g+1}(\mathbf{y}_{i:\lambda}^{g+1})^T, \tag{18}$$

$$\mathbf{p}_c^{g+1} = (1-c_c)\mathbf{p}_c^g + \sqrt{c_c(2-c_c)\mu_{eff}}\frac{\mathbf{m}^{g+1} - \mathbf{m}^g}{c_m\sigma^g}, \tag{19}$$

where $\mathbf{y}_{i:\lambda}^{(g+1)} = (\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)})/\sigma^{(g)}$, $\mu_{\text{eff}} = 1/\sum_{i=1}^{\mu} w_i^2$, and $\mathbf{p}_c^g \in \mathbb{R}^n$ is the evolution path. μ_{eff} is the effective sample size, c_1 is the learning rate for the rank-one update of the covariance matrix update, c_μ is the learning rate for the rank-μ update of the covariance matrix update, and c_c is learning rates for evolution path. c_1, c_μ, and c_c are all less than 1.

4) Updating the Step Size: We update the step size σ^{g+1} at generation $g+1$:

$$\sigma^{g+1} = \sigma^{g} \exp\left[\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma^{g+1}\|}{E\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right], d_\sigma \approx 1 \quad (20)$$

$$\mathbf{p}_\sigma^{g+1} = (1-c_\sigma)\mathbf{p}_\sigma + \sqrt{c_\sigma(1-2c_\sigma)\mu_{eff}}(\mathbf{C}^g)^{-\frac{1}{2}}\frac{\mathbf{m}^{g+1}-\mathbf{m}^g}{c_m\sigma^g}, \quad (21)$$

where $\mathbf{p}_\sigma^g \in \mathbb{R}^n$ is the conjugate evolution path at generation g, d_σ is the damping parameter, and $1/c_\sigma$ denotes the backward time horizon of the evolution path $\mathbf{p}_\sigma^g$. Algorithm 1 illustrates the more detailed workflow of the CMA-ES.

IV. Experimental Results and Discussions

A. Experiment Setup

We conduct comparative experiments across three simulated scenarios to validate the effectiveness of our approach. A real-world test is also conducted to further illustrate the capabilities of our planner. A unitree GO2 quadruped robot has been used in both the simulations and real-world tests.

In scenario-I ($\mathcal{S}_\mathrm{I}$), the quadruped robot navigates through a constrained corridor densely populated with small obstacles without collisions with the robot. The height of these obstacles are low enough for the quadruped robot to step on and the width and length of them are small enough for the quadruped robot to straddle. In scenario-II ($\mathcal{S}_\mathrm{II}$), the quadruped robot is required to traverse several negative obstacles (i.e., gaps) during navigation, where taking a direct trajectory toward the target by crossing a gap could potentially cause failures of the NMPC controller. In scenario-III ($\mathcal{S}_\mathrm{III}$), the quadruped robot is required to traverse irregularly arranged stepping stones along its trajectory, while maintaining dynamic stability.

We compare the proposed method with three classical path planners, A* [27], TA-RRT* [28], and Dijkstra [29], and a baseline. The baseline is defined as a direct linear connection between the start and target positions (i.e., the Euclidean shortest trajectory). They are variants specifically designed to be extendable to complex and uneven terrains. In [27], the cost function is augmented with ground traversability models derived from elevation maps. [28] introduces adaptive sampling and step size adjustment based on terrain complexity analysis. Dijkstra in [29] improved terrain adaptation capabilities by custom cost function.

B. Evaluation Metrics

The evaluation metrics used in traditional planning algorithms are typically planning time, the length of the planned trajectory, the number of iteration steps, and the value of the cost function in optimize-based algorithms. Since our method mainly focuses on the stability of the robot through complex scenarios to take full advantage of modern controllers, we adopt the length of the planned trajectory L_p in the X-Y plane, the length of the simulated trajectory L_s considering the motion along the Z-axis, the mean planned yaw in local coordinate θ_p, the mean simulated yaw in local coordinate θ_s, and the mean momentum along the Z-axis of the center of mass (CoM) M_z. Due to the stochastic nature of our method and some comparative methods, we compute each trajectory 1,000 times and report the mean and standard deviation of the performance metrics. The main metric is M_z because it indicates stability of the robot during the navigation. The smaller of M_z, the more stable the robot is.

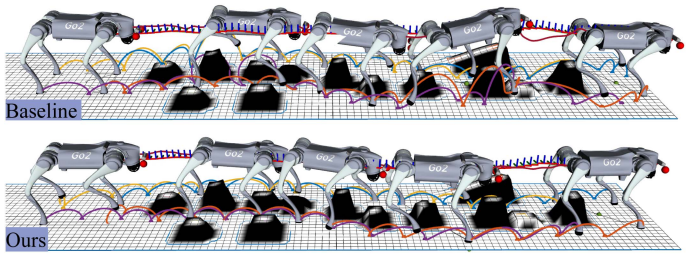

Fig. 2. The snapshots of the quadruped robot navigating through $\mathcal{S}_\mathrm{I}$ respectively guided by our trajectory and the baseline trajectory. The four lines in yellow, purple, orange and green are the foot-end swing trajectories. The line formed by coordinates is the trajectory produced by the NMPC controller, and the red line around the NMPC-produced trajectory is the CoM trajectory. The grids are the elevation map used for NMPC. The black areas denote the edges of obstacles. The total size of this scenario is 0.6 m × 4 m. The quadruped robot starts at (0.0 m, 0.0 m) and the goal is (3.5 m, 0.0 m). For further clarity, please watch our demonstration video.

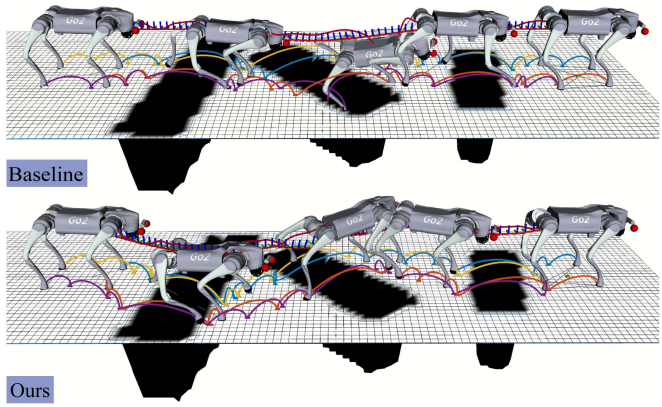

Fig. 3. The snapshots of the quadruped robot navigating through $\mathcal{S}_\mathrm{II}$ respectively guided by our trajectory and the baseline trajectory. Please refer to the caption of Fig. 2 for the denotations. The total size of this scenario is 1.0 m × 4.0 m. The quadruped robot starts at (0.0 m, 0.0 m) and its goal is at (3.0 m, 0.0 m). For further clarity, please watch our demonstration video.

TABLE I
Comparative Results in $\mathcal{S}_\mathrm{I}$. Different From the Deterministic Methods, the Stochastic Methods (i.e., TA-RRT* and Ours) Present Statistical Results (Mean± Std) Over 1,000 Times of Tests, Where Std Represents the Standard Deviation. The Best Results are Highlighted in Bold.

Method	L_p (m)	L_s (m)	θ_p (deg)	θ_s (deg)	M_z (kg·m/s)
Baseline	**3.50**	3.81	**0.00**	**0.07**	1.03
A*	3.52	3.84	0.32	0.09	0.98
TA-RRT*	3.98±0.25	4.24±0.21	2.35±1.68	0.42±0.21	1.19±0.17
Dijkstra	3.72	**3.48**	0.49	0.12	0.73
Ours	3.53±0.02	3.60±0.01	1.13±0.13	0.10±0.01	**0.63±0.02**
Ours$^-$	3.51±0.03	3.78±0.01	0.12±0.03	0.43±0.09	1.08±0.06
Ours$^\sim$	3.53±0.01	3.60±0.01	1.13±0.08	0.10±0.03	0.72±0.02

C. Comparative Experiments and Ablation Study

The method comparison in these three scenarios are shown in Table I, Table II, and Table III. From these tables, we can see that our method could produce a shorter and smoother

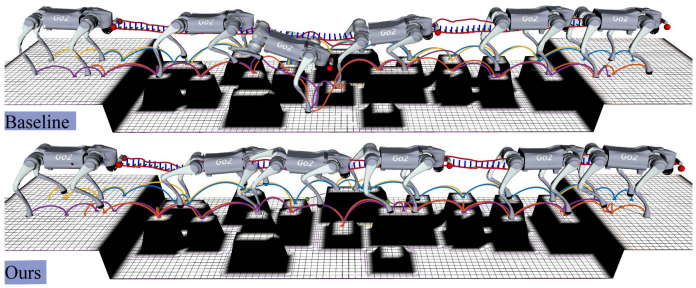

Fig. 4. The snapshots of the quadruped robot navigating through $\mathcal{S}_{\text{III}}$ respectively guided by our trajectory and the baseline trajectory. Please refer to the caption of Fig. 2 for the denotations. The total size of this scenario is 1 m × 5 m. The quadruped robot starts at (0.0 m, 0.0 m) and its goal is at (4.0 m, 0.0 m). For further clarity, please watch our demonstration video.

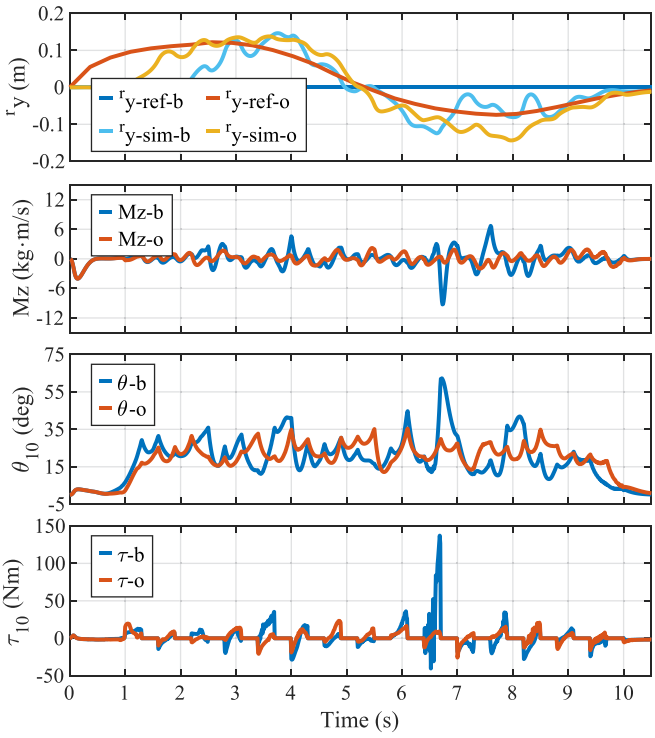

Fig. 5. The plots of the quadruped robot states when navigating through $\mathcal{S}_{\text{I}}$. ${}^{r}y$ denotes the coordinate of the CoM along Y-axis, "ref" denotes the trajectory from our method or baseline, "b" denotes baseline, and "o" denotes our method. M_z is the CoM moment along Z-axis, θ_i is the i-th joint angle in degree, and τ_i is the i-th joint toque.

TABLE II
COMPARATIVE RESULTS IN $\mathcal{S}_{\text{II}}$. DIFFERENT FROM THE DETERMINISTIC METHODS, THE STOCHASTIC METHODS (I.E., TA-RRT* AND OURS) PRESENT STATISTICAL RESULTS (MEAN± STD) OVER 1,000 TIMES OF TESTS, WHERE STD REPRESENTS THE STANDARD DEVIATION. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

Method	L_p (m)	L_s (m)	θ_p (deg)	θ_s (deg)	M_z (kg·m/s)
Baseline	**3.00**	3.79	**0.00**	**0.05**	1.82
A*	3.24	3.61	0.37	0.15	1.35
TA-RRT*	3.45±0.23	3.75±0.42	6.57±1.94	0.64±0.21	1.58±0.52
Dijkstra	3.34	3.72	0.15	0.08	1.49
Ours	3.37±0.03	3.48±0.11	2.35±0.26	0.21±0.02	**1.10±0.03**
Ours^{-}	3.08±0.02	3.68±0.03	0.08±0.01	0.08±0.03	1.79±0.05
Ours$^{\sim}$	3.33±0.03	**3.46±0.03**	2.27±0.07	0.20±0.00	1.13±0.02

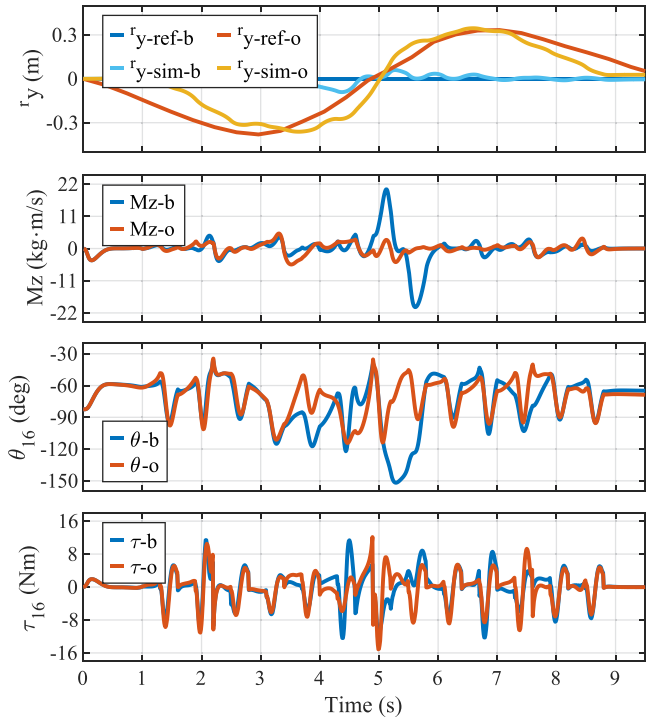

Fig. 6. The plots of the quadruped robot states when navigating through $\mathcal{S}_{\text{II}}$. ${}^{r}y$ denotes the coordinate of the CoM along Y-axis, "ref" denotes the trajectory from our method or baseline, "b" denotes baseline, and "o" denotes our method. M_z is the CoM moment along Z-axis, θ_i is the i-th joint angle in degree, and τ_i is the i-th joint toque.

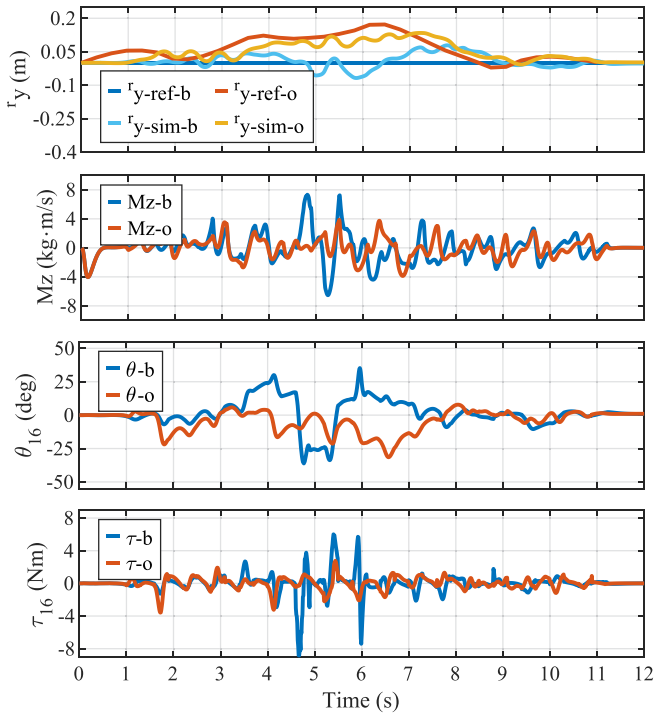

Fig. 7. The plots of the quadruped robot states when navigating through $\mathcal{S}_{\text{III}}$. ${}^{r}y$ denotes the coordinate of the CoM along Y-axis, "ref" denotes the trajectory from our method or baseline, "b" denotes baseline, and "o" denotes our method. M_z is the CoM moment along Z-axis, θ_i is the i-th joint angle in degree, and τ_i is the i-th joint toque.

(a) The real-world scenario 1. (b) Experiment snapshots of scenario 1. (c) The real-world scenario 2. (d) Experiment snapshots of scenario 2.

Fig. 8. (a) The scene setup of the real-world scenario 1. The size of this scene is 1.0 m × 5.0 m. The quadruped robot starts at (0.0 m, 0.0 m) and its goal is (5.0 m, −0.4 m). The obstacles are outlined by red. The computer in the upper left corner is used to generate trajectories and collect data. (b) The snapshots of the quadruped robot navigating through the real-world scenario 1 guided by the trajectory from our method. The red dashed line is the real trajectory. (c) The scene setup of the real-world scenario 2. The size of this scene is 0.8 m × 2.5 m. The quadruped robot starts at (0.0 m, 0.0 m) and its goal is (2.5 m, 0.05 m). The scene is constructed by assembling wooden boxes of different heights to resemble $\mathcal{S}_{\text{II}}$. The region highlighted by the red box below shows the side view of the area marked by the red box above. The boxes range in height from 0.1 m to 0.2 m. (d) The snapshots of the quadruped robot navigating through the real-world scenario 2 guided by the trajectory from our method. For further clarity, please watch our demonstration video.

TABLE III
COMPARATIVE RESULTS IN $\mathcal{S}_{\text{III}}$. DIFFERENT FROM THE DETERMINISTIC METHODS, THE STOCHASTIC METHODS (I.E., TA-RRT* AND OURS) PRESENT STATISTICAL RESULTS (MEAN± STD) OVER 1,000 TIMES OF TESTS, WHERE STD REPRESENTS THE STANDARD DEVIATION. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

Method	L_p (m)	L_s (m)	θ_p (deg)	θ_s (deg)	M_z (kg·m/s)
Baseline	**4.00**	4.48	**0.00**	0.07	1.26
A*	4.13	4.35	0.17	0.11	1.21
TA-RRT*	4.47±0.16	4.92±0.46	5.32±3.01	0.42±0.21	1.53±0.24
Dijkstra	4.12	4.41	0.18	0.09	1.21
Ours	4.01±0.01	**4.30**±0.14	1.34±0.24	0.11±0.02	**1.02±0.17**
Ours^{-}	4.05±0.04	4.32±0.02	0.08±0.01	0.11±0.04	1.17±0.05
Ours$^{\sim}$	4.01±0.02	4.31±0.38	0.10±0.12	**0.03±0.04**	1.05±0.17

trajectory. Most importantly, our method presents the lowest M_z among the compared methods, indicating that the trajectory from our method allows the robot moving stably when navigating $\mathcal{S}_{\text{I}}$. Besides, our method gets rather small standard deviation, meaning that the computational stability (i.e., consistency of the results under randomness) of our method is higher. Moreover, the quantitative comparison between the planned trajectories and their simulated executions reveals that our method achieves significantly smaller discrepancies in both trajectory length and mean yaw angle. These results demonstrate that the proposed method enables more effective utilization of the NMPC controller.

Meanwhile, we conducted an ablation study by removing the *Height Objective* defined in (14). The corresponding results are denoted as Ours^{-} in the tables. As shown, Ours^{-} produces results comparable to the baseline, since without the height-related penalty, the remaining objectives mainly generate short and smooth trajectories without considering the terrain.

In addition, we conducted a sensitivity analysis on the main weight matrices W_b and W_h to evaluate the robustness of our method. Specifically, W_b and W_h are randomly perturbed within $\pm 25\%$ of their nominal values, and the corresponding results are denoted as Ours$^{\sim}$. As shown in the tables, although the performance slightly decreases, all key metrics remain within 5% of the optimal results, demonstrating that the proposed algorithm is robust and insensitive to parameter variations.

1) Scenario-I: In $\mathcal{S}_{\text{I}}$, a number of small obstacles are placed on the flatten narrow ground. Within the given map, there is no enough space for the quadruped robot to fully avoid these obstacles. As shown in Fig. 2, the baseline trajectory could also lead the quadruped robot to the goal by the NMPC controller. Our method could pass through the scene by straddling the obstacles, neither stepping on the obstacles nor taking detours. We could see that the trajectory with our method could actively adjust the heading direction before encountering obstacles in around 2s ∼ 3s and 6s ∼ 8s in the first sub-figure of Fig. 5.

The fluctuation of M_z in the same time segment benefits from this active adjustment, which could be seen in the second sub-figure of Fig. 5. Around 6.5s, the swing angle of the the quadruped robot's 10-th joint changes from 15° to 55° in approximately 0.3s, resulting in intense vibration of the torque when the robot is guided by the baseline, shown in the third and forth sub-figure of Fig. 5. Due to the advance adjustment along the Y-axis, the quadruped robot is able to pass through $\mathcal{S}_{\text{I}}$ smoothly without drastic changes in M_z, θ_{10}, and τ_{10}.

2) Scenarios-II: In $\mathcal{S}_{\text{II}}$, there are three gaps that are not perpendicular to the heading direction, which makes the gaps wider along the X-axis. If the quadruped robot steps over these gaps along the X-axis directly, it may touch the bottom of the gap in simulation, as shown in Fig. 3. We could see the sharp change of M_z and θ_{16} in around 5s ∼ 5.8 s in Fig. 6. If the quadruped robot is guided by the trajectory from our method, the heading direction could be approximately perpendicular to the gap, allowing the robot span a shorter distance.

3) Scenario-III: $\mathcal{S}_{\text{III}}$ consists of two platforms and some irregular steeping stones with different heights between these two platforms, as shown in Fig. 4. From the first sub-figure of Fig. 7, we could see the terrain adaptation along Y-axis of the trajectory from our method. The large height difference between two adjacent foot-end point of the same foot makes large undulation of the robot's CoM, which creates instability for the robot. It can be seen from the second sub-figure of Fig. 7 that M_z moves from 7 Nm down to −7 Nm in about 0.5 s. Finally, the dramatic change puts an extra burden for the actuators of the robot, which could be observed in the third and forth sub-figures of Fig. 7. Guided by the trajectory from our method, the quadruped robot navigates through a relatively even terrain with smaller height difference of foot-end. Therefore, the fluctuations in M_z, θ_{16} and τ_{16} with the trajectory from our method are smaller than these with baseline trajectory.

D. Real-World Tests

In the real-world scenario 1, we place some obstacles on a narrow path. The path was 1.0 m wide and 5.0 m long with

some bushes on either side, as shown in Fig. 8(a). There are in total six obstacles in the sense, where the obstacles of number 2, 3, 5 are too high for the quadruped robot to step on or over and the obstacles of number 1, 4, 6 are thin enough for the quadruped robot to straddle. The snapshots of the experiment is shown in Fig. 8(b). We can see that the quadruped robot could exactly avoid the high and wide obstacles that would be difficult for the quadruped robot to cross. For the lower and blocking obstacles, the robot could easily straddle them following the generated trajectory.

In the real-world scenario 2, we use wooden boxes of varying heights to simulate scattered stones, as illustrated in Fig. 8(c), which resembles $\mathcal{S}_{\text{II}}$ used in the simulation analysis. The boxes are designed with three height levels: 0.10 m, 0.15 m, and 0.20 m. The trajectory generated by the proposed method guides the robot to traverse paths with smaller elevation differences, thereby enhancing the safety and stability of the navigation. In this scenario, we use a motion capture system to correct the robot body positions, which prevents state estimation failures caused by foot slippage and falls. Fig. 8(d) shows snapshots of the quadruped robot following the computed trajectory. With the perceptive locomotion based on the NMPC controller, the robot can step across the wooden blocks with different heights.

V. Conclusions and Future Work

We proposed here a novel trajectory planning algorithm, with the aim of taking full advantage of the capability of the NMPC controller. The trajectories from our method guide the quadruped robot cross complex terrains with more stability. The algorithm is formed as an optimization problem, which considers the geometry terrain features at the feet-end points. Thus, we employ SVGP to fit the point cloud of the terrain and get a precise terrain model. The optimization problem is non-convex, so we select CMA-ES as the solver. Since the objective function focuses on features at feet end, our algorithm cannot handle thin and tall obstacles. We may extend our planning algorithm to more challenging scenarios in the future.

References

[1] H. Li et al., "PINN-based predictive control combined with unknown payload identification for robots with prismatic quasi-direct-drives," *IEEE Robot. Autom. Lett.*, vol. 10, no. 11, pp. 11275–11282, Nov. 2025.

[2] 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.

[3] Y. Liu et al., "Adaptive MPC-based multi-terrain trajectory tracking framework for mobile spherical robots," *IEEE/ASME Trans. Mechatron.*, early aceess, Feb. 13, 2025, doi: 10.1109/TMECH.2025.3528106.

[4] Z. Zhuang et al., "Robot parkour learning," in *Proc. 7th Conf. Robot Learn.*, in Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., Nov. 2023, vol. 229, pp. 73–92.

[5] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal Parkour: Learning agile navigation for quadrupedal robots," *Sci. Robot.*, vol. 9, no. 88, 2024, Art. no. eadi7566.

[6] Y. Ren, Y. Cai, F. Zhu, S. Liang, and F. Zhang, "ROG-Map: An efficient robocentric occupancy grid map for large-scene and high-resolution lidar-based motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 8119–8125.

[7] J. Ortiz-Haro, W. Hönig, V. N. Hartmann, M. Toussaint, and L. Righetti, "iDb-RRT: Sampling-based kinodynamic motion planning with motion primitives and trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 10702–10709.

[8] M. Cho, Y. Lee, and K.-S. Kim, "Model predictive control of autonomous vehicles with integrated barriers using occupancy grid maps," *IEEE Robot. Autom. Lett.*, vol. 8, no. 4, pp. 2006–2013, Apr. 2023.

[9] S. Fahmi, V. Barasuol, D. Esteban, O. Villarreal, and C. Semini, "ViTAL: Vision-based terrain-aware locomotion for legged robots," *IEEE Trans. Robot.*, vol. 39, no. 2, pp. 885–904, Apr. 2023.

[10] W. Zhang, S. Xu, P. Cai, and L. Zhu, "Agile and safe trajectory planning for quadruped navigation with motion anisotropy awareness," in *Proc. 2024 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 8839–8846.

[11] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "RLOC: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 2908–2927, Oct. 2022.

[12] N. Hansen, "CMA-ES," in *Proc. Parallel Problem Solving Nature—PPSN XVIII: 18th Int. Conf.*, 2024. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-99253-2_1

[13] Y. Ma, B. Zhang, K. Huang, and L. Wang, "Probabilistic prediction and early warning for bridge bearing displacement using sparse variational Gaussian process regression," *Struct. Saf.*, vol. 114, 2025, Art. no. 102564.

[14] 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.

[15] Z. Wei, S. Liu, and R. Song, "A path planning method for quadruped robot based on 3 d elevation diagram," in *Proc. 7th Int. Symp. Auton. Syst.*, 2024, pp. 1–6.

[16] J. Yu et al., "A fast motion and foothold planning framework for legged robots on discrete terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 10678–10685.

[17] 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. 2025.

[18] H. Liu and Q. Yuan, "Safe and robust motion planning for autonomous navigation of quadruped robots in cluttered environments," *IEEE Access*, vol. 12, pp. 69728–69737, 2024.

[19] J. Liu, M. Stamatopoulou, and D. Kanoulas, "DiPPeR: Diffusion-based 2D path planner applied on legged robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024, pp. 9264–9270.

[20] X. Shao et al., "Path planning of mobile robot based on improved ant colony algorithm based on honeycomb grid," in *Proc. IEEE 5th Adv. Inf. Technol., Electron. Automat. Control Conf.*, 2021, vol. 5, pp. 1358–1362.

[21] D. An et al., "ETPNav: Evolving topological planning for vision-language navigation in continuous environments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 7, pp. 5130–5145, Jul. 2025.

[22] J. Kim, S. Park, and M. Kim, "Safety map: Disaster management road network for urban resilience," *Sustain. Cities Soc.*, vol. 96, 2023, Art. no. 104650.

[23] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Syst. Appl.*, vol. 227, 2023, Art. no. 120254.

[24] M. Elobaid et al., "Adaptive non-linear centroidal MPC with stability guarantees for robust locomotion of legged robots," *IEEE Robot. Autom. Lett.*, vol. 10, no. 3, pp. 2806–2813, Mar. 2025.

[25] L. Amatucci, G. Turrisi, A. Bratta, V. Barasuol, and C. Semini, "Accelerating model predictive control for legged robots through distributed optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 12734–12741.

[26] B. Jin and X. Xu, "Forecasting wholesale prices of yellow corn through the Gaussian process regression," *Neural Comput. Appl.*, vol. 36, no. 15, pp. 8693–8710, 2024.

[27] Z. Liu, S. Guo, F. Yu, J. Hao, and P. Zhang, "Improved a* algorithm for mobile robots under rough terrain based on ground trafficability model and ground ruggedness model," *Sensors*, vol. 24, no. 15, 2024, Art. no. 4884. [Online]. Available: https://www.mdpi.com/1424-8220/24/15/4884

[28] T. Oh, Y.-J. Won, and S. Lee, "TA-RRT*: Adaptive sampling-based path planning using terrain analysis," *Appl. Sci.*, vol. 15, no. 5, 2025, Art. no. 2287. [Online]. Available: https://www.mdpi.com/2076-3417/15/5/2287

[29] M. A. Arain, I. Havoutis, C. Semini, J. Buchli, and D. G. Caldwell, "A comparison of search-based planners for a legged robot," in *Proc. 9th Int. Workshop Robot Motion Control*, 2013, pp. 104–109.