2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3503-7770-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IROS58592.2024.10802816

Bipedal Safe Navigation over Uncertain Rough Terrain: Unifying Terrain Mapping and Locomotion Stability

Kasidit Muenprasitivej*[1], Jesse Jiang*[2], Abdulaziz Shamsah*[3,4], Samuel Coogan[2,5], and Ye Zhao[3]

***Abstract*— We study the problem of bipedal robot navigation in complex environments with uncertain and rough terrain. In particular, we consider a scenario in which the robot is expected to reach a desired goal location by traversing an environment with uncertain terrain elevation. Such terrain uncertainties induce not only untraversable regions but also robot motion perturbations. Thus, the problems of terrain mapping and locomotion stability are intertwined. We evaluate three different kernels for Gaussian process (GP) regression to learn the terrain elevation. We also learn the motion deviation resulting from both the terrain as well as the discrepancy between the reduced-order Prismatic Inverted Pendulum Model used for planning and the full-order locomotion dynamics. We propose a hierarchical locomotion-dynamics-aware sampling-based navigation planner. The global navigation planner plans a series of local waypoints to reach the desired goal locations while respecting locomotion stability constraints. Then, a local navigation planner is used to generate a sequence of dynamically feasible footsteps to reach local waypoints. We develop a novel trajectory evaluation metric to minimize motion deviation and maximize information gain of the terrain elevation map. We evaluate the efficacy of our planning framework on Digit bipedal robot simulation in MuJoCo.**[i]

I. Introduction

Legged robots show great promise for navigation tasks in environments with difficult-to-traverse or unknown terrain. As opposed to wheeled mobile robots, legged robots have the superior capability of traversing through irregular terrains by taking discrete footsteps [1]–[3]. However, highly varying and uncertain terrain profiles often induce tracking errors when executing bipedal motion plans or even pose a high risk in locomotion failures (*i.e.*, falling) [4]–[6]. Thus, navigation through complex and uncertain terrain requires collecting terrain data online to build a realistic terrain map and improve locomotion performance accordingly. On the other hand, the complex dynamics inherent to bipedal locomotion complicate the problem of designing navigation plans to sample the environment. Thus, the objectives of locomotion stability (*i.e.*, minimizing motion deviation from the desired stable trajectory in this study) and accurate environmental sampling are coupled, increasing the complexity of the entire navigation problem.

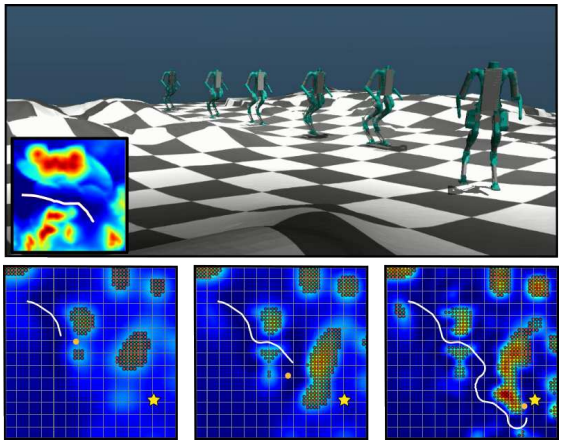


Fig. 1. (Top) The bipedal robot Digit navigates through an environment with rough terrain in our MuJoCo simulation. (Bottom) Snapshots of the trajectory of the bipedal robot at various time instants as it navigates towards the goal (yellow star). The white line depicts the traversed trajectory, and the orange dot is the current targeted local waypoint.

In this work, we propose a hierarchical planning strategy for bipedal robots which satisfies high-level global navigation objectives while maintaining dynamic feasibility of the generated trajectories in the local navigation planner. Additionally, we use Gaussian processes (GPs) with three different kernels to learn unknown terrain elevation. We also learn motion perturbation resulting from both terrain and model errors. Our planner is designed to incorporate the GP predictions in order to online improve the feasibility of reaching the desired goal. An example run of our planner is shown in Fig. 1.

A. Related Works

The RRT family of algorithms is commonly used in concert with GPs for robotic motion planning problems in uncertain environments. The study in [7] considers an aerial vehicle navigation problem and uses RRT to navigate around collision regions modeled using a GP. The work [8] uses RRT* to enable a mobile robot to avoid hazardous regions which are learned and updated online using a GP. For an

*Equally contributed authors.

This work was supported in part by the Office of Naval Research (ONR) Award N000142312223, National Science Foundation (NSF) grants IIS1924978, CMMI-2144309, FRR-2328254, USDA 2023-67021-41397, and NSF Graduate Research Fellowship under grant #DGE-2039655.

[1]Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: kmuenpra3@gatech.edu)

[2]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: jjiang@gatech.edu, sam.coogan@gatech.edu).

[3]George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA, 30332 USA (e-mail: ashamsah3@gatech.edu, ye.zhao@me.gatech.edu).

[4]Mechanical Engineering Department, College of Engineering and Petroleum, Kuwait University, PO Box 5969, Safat, 13060, Kuwait

[5]School of Civil and Environmental Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA

[i]Videos of the simulated experiments in Mujoco can be found at https://youtu.be/27hOcBNKAvU?si=dp3GSGHndtQ7ZMaC

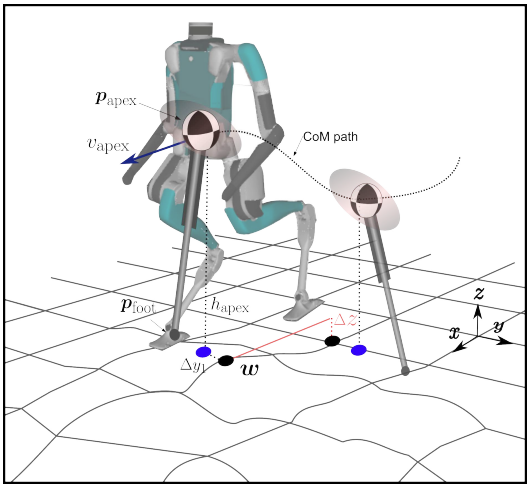


Fig. 2. Prismatic Inverted Pendulum Model (PIPM) model for our robot Digit for traversing over uncertain and uneven terrain.

information-gathering objective, the paper [9] learns optimal points to sample using a GP model of the environment and uses RRT* to plan information-gain-maximizing trajectories. Finally, the work [10] trains a GP model of terrain elevation using both external perception and proprioception sensors and then proposes an RRT* variant to plan a safe trajectory.

The problem of bipedal robot navigation in rough terrain has not yet been widely explored. The studies in [11], [12] propose methods for identifying stable footstep sequences for bipedal robots using sensor data to traverse over uneven terrain. The focus of these works is on finding stable local trajectories rather than long-run trajectories to reach global goals. The authors in [2] propose a terrain-adaptive bipedal locomotion controller that uses a piecewise linear terrain approximation for computing foot placements. The work in [3] proposes an omnidirectional control Lyapunov function (CLF) as a controller for a bipedal robot navigating on undulating terrain and integrates the CLF into a RRT* planner. In this work, an elevation map is constructed online using sensor data, but is not otherwise learned. Additionally, the omnidirectional nature of the planner relies on special behaviors such as turn-in-place and lateral stepping for bipedal locomotion feasibility.

In terrain mapping, GPs are employed to quantify uncertainties and learn complex terrain maps. The work in [13] introduce a novel nonstationary kernel that prioritizes exploration in regions with higher variation, effectively modeling rapidly changing terrain through a mixture model of base kernels. Another approach is the neural network kernel explained in [14], which designs a GP approximating a simple neural network while retaining the information-theoretic learning guarantees of Gaussian process theory. The nonstationary nature of this kernel is well-suited for learning discontinuous data. Additionally, KD-trees are used to manage dataset size, improving computational efficiency.

Leveraging GP approaches for terrain mapping has gained increasing attention in the locomotion community. In [15], a locally adaptive GP is implemented for terrain mapping in a legged navigation problem for the Boston Dynamics LittleDog quadruped, balancing GP model fidelity with computational tractability. In [16], GPs are used to evaluate candidate trajectories for a hopping robot locomotion planning problem. The work [17] learns a GP-based terrain map and uses the GP model to design foothold placements for the ETH ANYmal quadrupedal robot. However, learning terrain uncertainty via GP models for bipedal robot navigation has not been explored, to the best of the authors' knowledge. The inherent stability-critical, complex robot dynamics make the terrain learning and navigation problem more challenging.

B. Contributions

We propose a novel hierarchical planning framework for bipedal robot locomotion with high-level navigation tasks that generates dynamically feasible locomotion trajectories while simultaneously learning unknown terrain features. Our specific contributions are as follows.

- We propose a hierarchical locomotion-dynamics-aware planner based on RRT* which enables computationally efficient bipedal navigation while explicitly considering dynamical feasibility of the locomotion trajectories and learning uncertain rough terrain online. We construct both a footstep-by-footstep local navigation planner as well as a coarser global navigation planner which consider locomotion safety constraints.
- We develop the first ever planning framework that integrates Gaussian process models of unknown terrain elevation and motion perturbations for full-order bipedal locomotion. We propose a novel trajectory evaluation metric utilizing the GPs to minimize motion deviation and maximize information gain of the terrain estimation, thus increasing the feasibility of the navigation task. We benchmark the performance of multiple state-of-the-art GP terrain mapping methods to evaluate their relative advantages for the bipedal navigation task.
- We evaluate the proposed methodology on simulations of a Digit bipedal robot in MuJoCo [18], demonstrating the validity of the reduced-order trajectories generated by our planner when implemented on the simulator using full-order robot dynamics.

II. Preliminaries

A. Robot Model

We design our locomotion planner based on the Prismatic Inverted Pendulum Model (PIPM). PIPM has been proposed for agile, non-periodic locomotion over rough terrain [19] and integrated with Digit for navigation in partially observable environments and stair climbing tasks [20].

Here we reiterate for completeness the mathematical formulation of our ROM. As shown in Fig. 2, the CoM position $\boldsymbol{p}_{\text{com}} = (x_{\text{com}}, y_{\text{com}}, z_{\text{com}})^T$ is composed of the sagittal, lateral, and vertical positions in the global frame. We denote the apex CoM position as $\boldsymbol{p}_{\text{apex}} = (x_{\text{apex}}, y_{\text{apex}}, z_{\text{apex}})^T$, the foot placement as $\boldsymbol{p}_{\text{foot}} = (x_{\text{foot}}, y_{\text{foot}}, z_{\text{foot}})^T$, and h_{apex} is the relative apex CoM height with respect to the stance foot height. v_{apex} denotes the CoM velocity at $\boldsymbol{p}_{\text{apex}}$. We denote Δy_1 as the lateral distance between CoM and the high-level

waypoint at apex. We formulate the dynamics for the next walking step as a hybrid control system

$$\ddot{\boldsymbol{p}}_{\mathrm{com},n} = \begin{pmatrix} \omega_n^2(x_{\mathrm{com}} - x_{\mathrm{foot},n}) \\ \omega_n^2(y_{\mathrm{com}} - y_{\mathrm{foot},n}) \\ a_n\omega_n^2(x_{\mathrm{com}} - x_{\mathrm{foot},n}) + b_n\omega_n^2(y_{\mathrm{com}} - y_{\mathrm{foot},n}) \end{pmatrix}$$

where the asymptote slope $\omega_n = \sqrt{g/z_{\mathrm{apex},n}}$, and $z_{\mathrm{apex},n} = a_n x_{\mathrm{foot},n} + b_n y_{\mathrm{foot},n} + h_{\mathrm{apex}}$. The hybrid control input is $\boldsymbol{u}_n = (\omega_n, \boldsymbol{p}_{\mathrm{foot},n})$, with $\boldsymbol{p}_{\mathrm{foot},n}$ being a discontinuous input which creates a reset map.

B. Phase-space Planning

In phase-space planning (PSP), the sagittal CoM planning takes precedence over the lateral CoM planning. The decisions for the planning algorithm are primarily made in the sagittal phase-space, such as step length and CoM apex velocity, where we propagate the dynamics forward from the current apex state and backward from the next apex state until the two phase-space trajectories intersect. The intersection state defines the foot stance switching instant. On the other hand, the lateral phase-space parameters are searched for to adhere to the sagittal phase-space plan and have consistent timings between the sagittal and lateral plans. In this paper, we use the PSP method detailed in our previous work [20].

C. Gaussian Processes

In order to learn the uncertainties present in our bipedal system, we use Gaussian process (GP) regression:

Definition 1 (Gaussian Process Regression): Gaussian Process (GP) regression models a function $g_i : \mathbb{R}^n \to \mathbb{R}$ as a distribution with covariance $\kappa : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{>0}$. Assume a dataset of m samples $D = \{(\boldsymbol{\xi}^j, y_i^j)\}_{j\in\{1,\dots,m\}}$, where $\boldsymbol{\xi}^j \in \mathbb{R}^n$ is the input and y_i^j is an observation of $g_i(\boldsymbol{\xi}^j)$ under Gaussian noise with variance $\sigma_{\nu_i}^2$. Let $K \in \mathbb{R}^{m\times m}$ be a kernel matrix defined elementwise by $K_{j\ell} = \kappa(\boldsymbol{\xi}^j, \boldsymbol{\xi}^\ell)$ and for $\boldsymbol{\xi} \in \mathbb{R}^n$, let $k(\boldsymbol{\xi}) = [\kappa(\boldsymbol{\xi}, \boldsymbol{\xi}^1) \quad \kappa(\boldsymbol{\xi}, \boldsymbol{\xi}^2) \dots \kappa(\boldsymbol{\xi}, \boldsymbol{\xi}^m)]^T \in \mathbb{R}^m$. Then, the predictive distribution of g_i at a test point $\boldsymbol{\xi}$ is the conditional distribution of g_i given D, which is Gaussian with mean $\mu_{g_i,D}$ and variance $\sigma_{g_i,D}^2$ given by

$$\mu_{g_i,D}(\boldsymbol{\xi}) = k(\boldsymbol{\xi})^T(K + \sigma_{\nu_i}^2 I_m)^{-1}Y$$
$$\sigma_{g_i,D}^2(\boldsymbol{\xi}) = \kappa(\boldsymbol{\xi},\boldsymbol{\xi}) - k(\boldsymbol{\xi})^T(K + \sigma_{\nu_i}^2 I_m)^{-1}k(\boldsymbol{\xi}),$$

where I_m is the identity and $Y = \begin{bmatrix} y_i^1 & y_i^2 & \dots & y_i^m \end{bmatrix}^T$. In practice, we use a sparse Gaussian process regression approximation [21] to reduce computational complexity.

In this work, three different terrain mapping method using GP are benchmarked, namely a radial basis function (RBF) kernel, a Neural Network (NN) kernel with local approximation method [14], and an Attentive Kernel (AK) [13].

1) RBF kernel: The RBF kernel, a stationary kernel commonly used in GP regression, produces smooth predictions with uniform variability. The kernel is defined as

$$\kappa(\boldsymbol{\xi}^i, \boldsymbol{\xi}^j) = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{\xi}^i - \boldsymbol{\xi}^j\|^2}{2\ell^2}\right),$$

where σ_f^2 is signal variance and ℓ is a lengthscale.

2) NN kernel: The NN kernel is non-stationary and resembles a neural network with a single hidden layer of infinite nodes and a sigmoid activation function [22]. It models local correlation between data points based on their distance from the data origin until a saturation region is reached. The kernel is defined as

$$k(\boldsymbol{\xi}^i, \boldsymbol{\xi}^j) =$$
$$\sigma_f^2 \arcsin\left[\frac{\beta + 2\boldsymbol{\xi}^{iT}\Sigma\boldsymbol{\xi}^j}{\sqrt{(1+\beta+2\boldsymbol{\xi}^{iT}\Sigma\boldsymbol{\xi}^i)(1+\beta+2\boldsymbol{\xi}^{jT}\Sigma\boldsymbol{\xi}^j)}}\right]$$

where $\Sigma = \begin{bmatrix} \ell_x & 0 \\ 0 & \ell y \end{bmatrix}^{-2}$, β is a bias factor, and ℓ_x and ℓ_y are the lengthscales for input x and y, respectively.

The work by [14] applied the NN kernel to large-scale terrain reconstruction by introducing a local approximation method, facilitated by a KD-Tree algorithm for efficient nearest neighbor search. This approach enhances the accuracy of GP predictions in terrains with high variability by utilizing only the nearest training data around the query point.

However, this method introduces significant computational overhead due to the need for unique nearest neighbor search and separate predictions for each query point. To mitigate this, we propose a K-means clustering strategy to group all input locations into k clusters, where each cluster is defined by a center point c_i and a subset of training data $\{\boldsymbol{\xi}\}_i$. When a query point is provided, the nearest c_i is identified, and the GP prediction is computed using only the corresponding training data $\{\boldsymbol{\xi}\}_i$. This extended local approximation method reduces the computational runtime compared to the approach in [14].

3) Attentive Kernel: The AK is a nonstationary kernel that adapts to terrain variability by employing a neural network to determine the optimal weighted sum of multiple base kernels. Additionally, it assigns a membership vector to each input location through a secondary neural network. This approach allows the model to break correlations among training inputs within the same vicinity when abrupt changes occur in the training output, in contrast to the smooth behavior characteristic of the RBF kernel. The kernel is defined as

$$\kappa(\boldsymbol{\xi}^i, \boldsymbol{\xi}^j) = \alpha\bar{z}^T\bar{z}' + \sum_{m=1}^{M} \bar{w}_m\kappa_m(\boldsymbol{\xi}^i, \boldsymbol{\xi}^j)\bar{w}'_m$$

where α is a constant, $\bar{w}$ and $\bar{z}$ are the trained weight and membership vectors, respectively, and $\{\kappa(\boldsymbol{\xi}^i, \boldsymbol{\xi}^j)\}_{m=1}^M$ are base RBF kernels with different pre-defined lengthscales.

III. Problem Statement

We now formally define the problem we study in this work. Consider an environment in which the terrain elevation is uncertain, creating multiple challenges for bipedal locomotion. First, regions with high terrain elevation may be untraversable and therefore become obstacles. Second, the terrain elevation is an input to the PSP model, so inaccurate terrain estimations increase deviation and create instability in planned footstep trajectories. The primary objective is for the

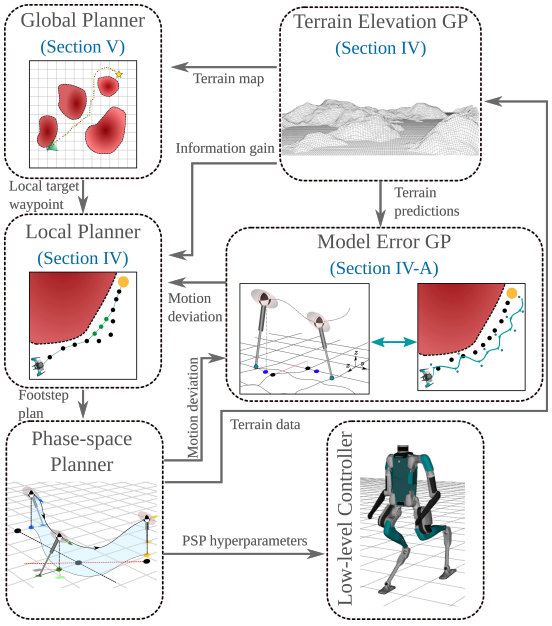


Fig. 3. Overall block diagram of the proposed global-local planning framework for bipedal navigation over rough terrain.

robot to reach a desired location in such uncertain environment. Thus, in addition to the reach-avoid objective, the robot must learn an accurate representation of the environment to improve the dynamical feasibility of planned trajectories. Additionally, there also exists motion perturbations resulting from the model error between the PIPM used for planning and the full-order dynamics, which increases the complexity of the overall uncertainty learning problem.

Problem Statement: Design a hierarchical planning framework for a bipedal robot which generates dynamically feasible trajectories to reach a desired goal location in an environment with unknown terrain features. Learn online the terrain elevation and the resulting motion perturbations in order to avoid untraversable regions and minimize the error between the desired motion plans and the measured trajectories from a full-body robot dynamic simulation.

Overall Framework: Our approach to this problem is as follows. We first initialize the terrain GP with an *a priori* dataset of terrain elevations. The model error GP is trained offline, as detailed in Section IV-A, to characterize the terrain's impact on robot motion. The local planner, LDA-L-RRT*, introduced in Section IV-B, generates dynamically feasible waypoints that avoid untraversable regions while balancing exploration and minimizing motion perturbation. The global planner, LDA-G-RRT*, outlined in Section V, provides near-horizon targets for the local planner, guiding the robot towards the global goal. As the robot reaches each local waypoint, the terrain GP is updated with new data, and the global planner is re-executed. This process repeats until the robot reaches the global target. The framework is depicted in Fig. 3 and summarized in Algorithm 1.

Algorithm 1: Global-Local Planning Framework

Input: Start waypoint $\boldsymbol{w}_0$, target waypoint $\boldsymbol{w}_t$
Initialize *Terrain GP* $\hat{z}(x, y)$;
Initialize *Model Error GP* $\Delta\hat{y}_1$;
Initialize *Current position* $\boldsymbol{w}_c = \boldsymbol{w}_0$;
while $\boldsymbol{w}_c \neq \boldsymbol{w}_t$ **do**
  Run LDA-G-RRT* algorithm with target x_t and obtain local target waypoint $\boldsymbol{w}_\ell$;
  Run LDA-L-RRT* with target $\boldsymbol{w}_\ell$ and obtain footstep plan;
  Execute footstep plan and collect terrain data $\{(x_i, y_i), z_i\}$ along the trajectory;
  Update current waypoint $\boldsymbol{w}_c$;
  Retrain terrain GP $\hat{z}$ on collected data;
end

IV. Local Navigation Planner

In this section, we propose a local navigation planner to generate a footstep-by-footstep motion plan. We first define local navigation trajectories.

Definition 2 (Local Navigation Trajectory): A local navigation trajectory $\mathcal{A}_{\boldsymbol{w}\to\boldsymbol{w}'}$ from a start waypoint $\boldsymbol{w} = (x, y, \theta)$ to an end waypoint $\boldsymbol{w}'$ is an n-step sequence $\{a_{HL,0}, \cdots, a_{HL,n-1}\}$ of high level actions $a_{HL,i} = (d_i, \Delta\theta_i, \Delta z_i)$, where the parameters $d, \Delta\theta, \Delta z$ represent the distance, heading angle change, and terrain elevation change, respectively, between two adjacent waypoints. The sequence $\mathcal{A}_{\boldsymbol{w}\to\boldsymbol{w}'}$ induces a set of apex CoM waypoints $\mathcal{W}(\mathcal{A}_{\boldsymbol{w}\to\boldsymbol{w}'}) = \{\boldsymbol{w}_0, \cdots, \boldsymbol{w}_{n-1}\}$ such that $\boldsymbol{w}_0 = \boldsymbol{w}$, $\boldsymbol{w}_{n-1} = \boldsymbol{w}'$, and $\boldsymbol{w}_{i+1} = \boldsymbol{w}_i + \left[d_i \cos(\sum_{j=0}^{i} \Delta\theta_j + \theta_0), d_i \sin(\sum_{j=0}^{i} \Delta\theta_j + \theta_0), \Delta\theta_i\right]$, $\forall i \in \{0, \cdots, n-2\}$. The local navigation trajectory parameters are illustrated in Fig. 4(c).

A. Gaussian Process Learning of Terrain and Model Errors

We first detail the two GP structures we use to learn the unknown terrain elevation and the motion perturbations resulting from both terrain and model errors. This structure builds on our previous GP modeling work in [23].

We use a terrain GP $\hat{z}(x, y)$ with sensor noise $\epsilon \sim \mathcal{N}(0, \sigma_\nu^2)$ which takes a global location (x, y) as input and predicts terrain height at that location, z. The three terrain GP approaches discussed in Section II-C are evaluated independently for their respective accuracy and complexity. At runtime, the GP is updated with data collected as the robot traverses through the environment. The mean prediction of the terrain GP is used in the PSP controller to generate feasible footstep trajectories and to construct global and local elevation maps, with the latter being a subset of the former. While the local map could be constructed independently without using a GP, we opt for using a subset of the global map as the GP prediction is generally more reliable with denser data points at the local level. The variance of the terrain GP is used to determine the information gain along each local trajectory as discussed later in Section IV-B.

Given the GP model of terrain elevation, we characterize the model error between the referenced waypoint and the actual CoM position at apex in the lateral direc-

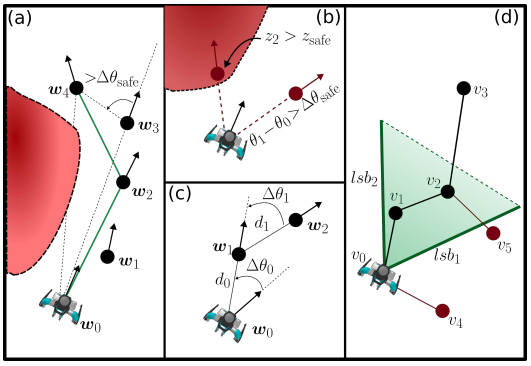


Fig. 4. (a) Illustration of the smoothing algorithm aims to find the smoothest path from start $\boldsymbol{w}_0$ to target $\boldsymbol{w}_4$. A direct connection between $\boldsymbol{w}_0$ and $\boldsymbol{w}_4$ is invalid due to an obstacle, and connecting to $\boldsymbol{w}_3$ is infeasible because the heading angle exceeds $\Delta\theta_{\text{safe}}$. Thus, the algorithm connects $\boldsymbol{w}_0$ to $\boldsymbol{w}_2$ and then to $\boldsymbol{w}_4$. (b) Illustration of LDA-L-RRT* vertex selection criteria. The left red vertex is invalid because it lies in an obstacle, and the right red vertex is impossible because the heading angle change exceeds $\Delta\theta_{\text{safe}}$. (c) Illustration of the local navigation trajectory parameters. (d) Illustration of the LDA-G-RRT* safety criteria: Solid green lines represent locomotion safety barriers, and the shaded green area is their convex hull. Connecting v_0 to v_4 is invalid as v_4 lies outside the safety barriers. The connection from v_2 to v_5 is also invalid due to crossing a safety barrier. However, the connection from v_2 to v_3 is valid, as it does not intersect any safety barrier.

tion (*i.e.*, $\Delta y_1 = y - y_{com}$) at each step using the GP $\Delta\hat{y}_1(d_c, \Delta\theta_c, \Delta z_c, d_n, \Delta\theta_n, \Delta z_n)$. We focus on modeling lateral deviation because the PSP controller is designed to prioritize achieving the desired sagittal distance of the waypoint with lateral error minimization as a secondary objective. The model error GP takes high-level actions from the last two walking steps as input. Parameters with subscript c denote values measured between the previous and current waypoints, while subscript n denote values measured between the current and next waypoints. The model error GP is trained offline on a dataset generated by simulating steps over the entire range of input parameters using the low-level controller in Section VI and measuring the resulting motion perturbation. In practice, the robot's stance affects motion perturbation, with left-foot steps causing leftward deviation and right-foot steps causing rightward deviation. For efficient learning, we train a single model error GP using the absolute value of lateral deviations from both foot stances. Given GP predictions, deviations are then assigned positive values for left-foot steps and negative values for right-foot steps.

The local navigation planner requires predictions of the expected model error for proposed waypoint sequences. To obtain these predictions, we call the model error GP $\Delta\hat{y}_1$ for each step of the sequence. While the input d and $\Delta\theta$ can be directly extracted from the waypoint sequence, Δz depends on the unknown terrain elevation. We approximate Δz using the mean $\mu_{\hat{z}}$ of the terrain GP predictions at the waypoints. Since each output $\Delta\hat{y}_1$ is in the local frame w.r.t. a specific waypoint, we apply coordinate transforms on the outputs to place them in the global frame.

B. Locomotion-dynamics-aware Local RRT*

We now define the locomotion-dynamics-aware RRT* algorithm we use for local planning.

Definition 3 (Locomotion-dynamics-aware Local RRT):* The LDA-L-RRT* algorithm modifies the standard RRT* algorithm by placing additional constraints on new vertices in the search as follows. First, the configuration of a vertex is $\boldsymbol{w} = (x_{\text{apex}}, y_{\text{apex}}, \theta)$, where $(x_{\text{apex}}, y_{\text{apex}})$ is the planar apex position and θ is the heading angle. Then, consider one step of the standard RRT* algorithm in which a random point in the environment $(x_{\text{rand}}, y_{\text{rand}})$ is selected, for which the nearest vertex is $\boldsymbol{w}_1 = (x_{\text{apex},1}, y_{\text{apex},1}, \theta_1)$. A candidate vertex $\boldsymbol{w}' = (x'_{\text{apex}}, y'_{\text{apex}}, \theta')$ is calculated as

$$\begin{bmatrix} x'_{\text{apex}} & y'_{\text{apex}} \end{bmatrix} = d_{\text{safe}} \frac{\begin{bmatrix} x_{\text{rand}} & y_{\text{rand}} \end{bmatrix} - \begin{bmatrix} x_{\text{apex},1} & y_{\text{apex},1} \end{bmatrix}}{\left\| \begin{bmatrix} x_{\text{rand}} & y_{\text{rand}} \end{bmatrix} - \begin{bmatrix} x_{\text{apex},1} & y_{\text{apex},1} \end{bmatrix} \right\|_2},$$
$$\theta' = \arctan\left(\frac{y_{\text{rand}} - y_{\text{apex},1}}{x_{\text{rand}} - x_{\text{apex},1}}\right),$$

where d_{safe} is a safe step distance determined as in [20, Theorem IV.1]. Then, the candidate $\boldsymbol{w}'$ is added to the graph if and only if it satisfies the following conditions.

1) The heading angle change between connected vertices is less than a dynamically feasible limit $\Delta\theta_{\text{safe}}$ calculated as in [20, Theorem IV.2]:
$$|\theta' - \theta_1| \leq \Delta\theta_{\text{safe}}.$$
2) The GP predicted terrain elevation at $\boldsymbol{w}'$ is smaller than a dynamically feasible limit z_{safe}:
$$\mu_{\hat{z}}(x'_{\text{apex}}, y'_{\text{apex}}) \leq z_{\text{safe}},$$
where z_{safe} is the maximum height the robot's passivity-based controller can stabilize stepping motion despite terrain variations. This limits our navigation to relatively flat terrains, avoiding areas with significant elevation changes.

If $\boldsymbol{w}'$ does not satisfy the conditions above, a new candidate $\boldsymbol{w}'$ is calculated by performing the above procedure with the same random point $(x_{\text{rand}}, y_{\text{rand}})$ and the next closest vertex in the graph $\boldsymbol{w}_2$. The process continues through all of the vertices in the current graph until a candidate vertex is successfully added to the graph. If there is no node that satisfies both conditions, the nearest vertex $\boldsymbol{w}_{\text{near}}$ to $(x_{\text{rand}}, y_{\text{rand}})$ that does not have a child node is identified. A final candidate vertex $\boldsymbol{w}''$ is proposed with values

$$\begin{bmatrix} x''_{\text{apex}} \\ y''_{\text{apex}} \end{bmatrix} = \begin{bmatrix} x_{\text{apex,near}} \\ y_{\text{apex,near}} \end{bmatrix} + d_{\text{safe}} \begin{bmatrix} \cos(\theta_{\text{near}} + \Delta\theta_{\text{safe}}) \\ \sin(\theta_{\text{near}} + \Delta\theta_{\text{safe}}) \end{bmatrix},$$
$$\theta'' = \theta_{\text{near}} + \Delta\theta_{\text{safe}}.$$

This candidate vertex always satisfies condition 1), so it is added to the graph if it also satisfies condition 2). If not, then no vertex is added to the graph in the current step.

With these conditions, the waypoint sequences generated by the LDA-L-RRT* algorithm are guaranteed to be dynamically feasible with respect to the PIPM safety conditions proposed in [20] and will avoid untraversable regions with excessive terrain elevation. However, the resulting trajectories can change heading angle rapidly at each step, directly

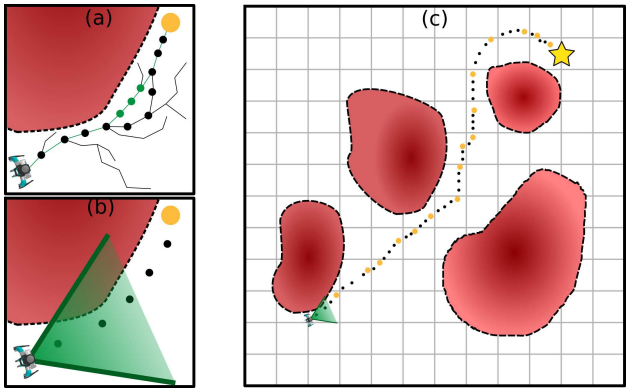


Fig. 5. (a) Trajectory generated by the LDA-L-RRT* algorithm for reaching a local waypoint. The green dots show the trajectory modifications made by the proposed smoothing algorithm. (b) Illustration of the locomotion safety barrier region (green triangle) constraining vertices near the start point in the global navigation planner. (c) Trajectory generated by the LDA-G-RRT* algorithm. The black and orange dots combined represent the planned trajectory, from which the orange dots are selected as local waypoints.

increasing motion errors. Thus, we propose a trajectory-smoothing algorithm to smooth the LDA-L-RRT* trajectories to improve trajectory tracking performance. The idea of the smoothing algorithm is to replace the "zigzag"-prone trajectories typical of RRT-generated trajectories with straight lines more amenable to bipedal locomotion. The algorithm begins at the starting waypoint $\boldsymbol{w}_0 = (x_{\text{apex},0}, y_{\text{apex},0}, \theta_0)$ of a LDA-L-RRT* motion plan and finds the furthest waypoint $\boldsymbol{w}_i = (x_{\text{apex,i}}, y_{\text{apex,i}}, \theta_i)$ along the trajectory which satisfies the following conditions.

1) There is no untraversable terrain along the line connecting $\boldsymbol{w}_0$ and $\boldsymbol{w}_i$:

$$\mu_{\hat{z}}(x'_{\text{apex}}, y'_{\text{apex}}) \leq z_{\text{safe}}$$
$$\forall (x'_{\text{apex}}, y'_{\text{apex}}) \in conv(\boldsymbol{w}_0, \boldsymbol{w}_i),$$

where $conv()$ is the convex hull, i.e., minimal convex set containing the two points.

2) The heading angle change between $\boldsymbol{w}_0$ and $\boldsymbol{w}_i$ and between $\boldsymbol{w}_i$ and $\boldsymbol{w}_{i+1}$ is valid:

$$|\theta_0 - \theta_i| \leq \Delta\theta_{\text{safe}}, |\theta_i - \theta_{i+1}| \leq \Delta\theta_{\text{safe}}$$

Once $\boldsymbol{w}_i$ is identified, a new sequence of waypoints with appropriate step lengths is generated on the line between $\boldsymbol{w}_0$ and $\boldsymbol{w}_i$, replacing the waypoints $\{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_{i-1}\}$. Then, the smoothing algorithm continues from $\boldsymbol{w}_i$, repeating until the target waypoint $\boldsymbol{w}_\ell$ is reached. Fig. 4(a) illustrates the smoothing algorithm, Fig. 4(b) shows the vertex safety constraints, and Fig. 5(a) shows a conceptual example of the waypoint sequence generated by LDA-L-RRT*.

For each local target, we run the LDA-L-RRT* algorithm m times to generate candidate trajectories $\{\mathcal{A}_{\boldsymbol{w}\to\boldsymbol{w}',j}\}, j \in \{1, \cdots, m\}$. We then select an optimal trajectory $\mathcal{A}^*_{\boldsymbol{w}\to\boldsymbol{w}'}$ using the formula

$$j^* = \arg\max_j [-\alpha(\text{error}(\mathcal{A}_{\boldsymbol{w}\to\boldsymbol{w}',j})) + \beta(\text{info}(\mathcal{A}_{\boldsymbol{w}\to\boldsymbol{w}',j}))],$$

$$\text{error}(\mathcal{A}_{\boldsymbol{w}\to\boldsymbol{w}',j}) = \sum_{a_{HL}\in\mathcal{A}_{\boldsymbol{w}\to\boldsymbol{w}',j}} T(\hat{y}_1(d, \Delta\theta, \Delta z))$$

$$\text{info}(\mathcal{A}_{\boldsymbol{w}\to\boldsymbol{w}',j}) = \sum_{\boldsymbol{w}_i\in\mathcal{W}(\mathcal{A}_{\boldsymbol{w}\to\boldsymbol{w}',j})} \frac{1}{2} log(2\pi\sigma^2_{\hat{z}}(\boldsymbol{w}_i)) + \frac{1}{2},$$

where $\alpha, \beta \in \mathbb{Z}_{\geq 0}$ and T is the transform from a local waypoint frame to the global frame. The optimal solution is $\mathcal{A}^*_{\boldsymbol{w}\to\boldsymbol{w}'} = \mathcal{A}_{\boldsymbol{w}\to\boldsymbol{w}',j^*}$. Intuitively, the optimal path minimizes the robot's CoM lateral deviation, $error$, predicted by the model error GP $\hat{y}_1$ over the waypoint sequence. The optimal path also maximizes the information gain, $info$, rewarding traversal of areas which have high uncertainty in the terrain elevation GP. The parameters α, β tune the importance of these two objectives.

V. Global Navigation Planner

In large, global environments, footstep-by-footstep local planning to reach a global goal is computationally expensive. Therefore, we propose a coarse global planner to generate waypoints as inputs for the LDA-L-RRT* algorithm (Section IV). This planner guides the robot toward the global goal, completing the local-global planning framework.

We now define the locomotion-dynamics-aware RRT* algorithm we use for global planning.

Definition 4 (Locomotion-dynamics-aware Global RRT):* The locomotion-dynamics-aware global RRT* (LDA-G-RRT*) algorithm modifies the standard RRT* algorithm by placing additional constraints on new vertices in the search as follows. First, we partition the global environment into hyper-rectangular regions $\{W_q\}_{q\in Q}$:

$$W_q = \{(x_{\text{com}}, y_{\text{com}}) \mid \underline{x}_q \leq x_{\text{com}} \leq \overline{x}_q, \underline{y}_q \leq y_{\text{com}} \leq \overline{y}_q\},$$

where the inequality is taken elementwise for lower and upper bounds $\underline{\cdot}_q, \overline{\cdot}_q \in \mathbb{R}$ and Q is a finite index set of the regions. The configuration of a vertex is $v = (x_{\text{com}}, y_{\text{com}})$. Additionally, for the starting vertex $v_0 = (x_0, y_0)$ we know the heading angle θ_0 from the robot's current state. We create locomotion safety barriers around the start vertex, defined as

$$lsb_{1,2} = \left\{ \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + 2d_{\text{step}}\gamma \begin{bmatrix} \cos(\theta_0 \pm \Delta\theta_{\text{safe}}) \\ \sin(\theta_0 \pm \Delta\theta_{\text{safe}}) \end{bmatrix} \right\}, \gamma \in [0, 1],$$

where d_{step} is a desired distance between vertices.

Then, consider one step of the standard RRT* algorithm in which a random point in the environment $(x_{\text{rand}}, y_{\text{rand}})$ is selected, for which the nearest vertex is v. A candidate vertex $v' = (x', y')$ is calculated as

$$\begin{bmatrix} x' & y' \end{bmatrix} = d_{\text{step}} \frac{\begin{bmatrix} x_{\text{rand}} & y_{\text{rand}} \end{bmatrix} - \begin{bmatrix} x & y \end{bmatrix}}{\left\| \begin{bmatrix} x_{\text{rand}} & y_{\text{rand}} \end{bmatrix} - \begin{bmatrix} x & y \end{bmatrix} \right\|_2}$$

Then, the candidate v' is added to the graph if and only if it satisfies the following conditions.

1) Any vertex connected to the starting vertex v_0 must be in the convex hull $conv\{lsb_1, lsb_2\}$.
2) Any connection between vertices in the graph must not intersect a locomotion safety barrier.
3) The GP predicted terrain elevation at v' is smaller than a dynamically feasible limit z_{safe}:

$$\mu_{\hat{z}}(x', y') \leq z_{\text{safe}}.$$

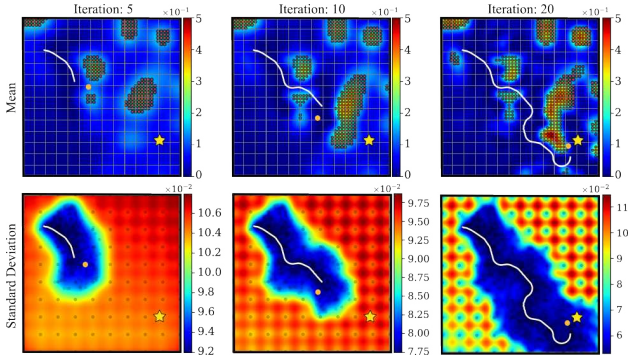


Fig. 6. Snapshots of the system during a sample run. In each plot, the white line shows the CoM trajectory, the orange dot marks the local target waypoint, and the yellow star indicates the global goal. The left plot shows early terrain characterization near the start. The middle plot depicts the robot navigating initial obstacles and mapping the obstacle near the goal. The final plot shows that the robot learned the terrain sufficiently to reach the goal.

The locomotion safety barriers, illustrated in Fig. 4(d), serve as warm starts for the LDA-L-RRT* by ensuring that the initial steps of the robot towards the nearest local waypoint from LDA-G-RRT* are feasible, taking into account of the heading angle of the starting vertex v_0.

In practice, we select the step size d_{step} for the LDA-G-RRT* larger than d_{safe} for the LDA-L-RRT*. Additionally, the locomotion safety barrier constraints on vertex selection for LDA-G-RRT* only hold within a small radius of the starting vertex v_0, shown in Fig. 5(b),(c), whereas LDA-L-RRT* constrains every vertex. These two features result in the computational efficiency of LDA-G-RRT*.

Once a sequence of vertices $\{v_0, v_1, \cdots, v_n\}$ from v_0 to the desired target position v_n has been found, we generate a sequence of corresponding local waypoints recursively as follows. We start at v_0 and identify the largest index i such that the vertices $\{v_0, v_1, \cdots, v_i\}$ are all elements of the same hyper-rectangular region W_0. The vertex v_i is the first local waypoint. Then, we move to vertex v_{i+1} and find the largest index j such that the vertices $\{v_{i+1}, v_1, \cdots, v_j\}$ are all elements of the same hyper-rectangular region W_i, adding v_j to the sequence of local waypoints. We repeat this process until the target v_n has been added to the sequence of local waypoints and return the complete sequence.

VI. Results

We evaluate our framework on simulations of a Digit bipedal robot navigating three environments with varying terrain features (including N37W112, N24W102, and N17E10) retrieved from elevation dataset by Shuttle Radar Topography Mission. Each environment is re-scaled to 20×20 meters in size, with terrain elevation varying between 0 and 0.5 meters. The terrain GP $\hat{z}$ model is initialized with 100 evenly-spaced points across the terrain and updated online with new data as the robot progresses. The sensor collects 10 sample points within a 3-meter radius at each step. To facilitate GP sampling, we assume no sensor occlusion, making all data points accessible for collection. The model error GP $\Delta\hat{y}_1$, trained offline with 2,000 data points, yielded an average motion perturbation of 1.75e-2 meters per step and an average prediction error of 2.06e-4 meters per step, demonstrating its accuracy in evaluating motion perturbations. Finally, for the LDA-L-RRT* local navigation planner, we evaluate three candidate trajectories for each local target.

TABLE I

Benchmarking for Evaluated GPs

Metric	AK	RBF	NN
Avg. Error (Path) [m]	**2.04e-4**	1.10e-3	2.83e-4
Avg. Std. Dev. (Path) [m]	**2.58e-2**	2.67e-2	5.19e-2
Avg. Error (Env.) [m]	52.60	49.61	**49.02**
Avg. Std. Dev. (Env.) [m]	**3.13e-2**	3.93e-2	5.48e-2
Avg. Total Steps	638	**617**	667
Retrain every # steps	**20**	14	19.33

Fig. 6 shows snapshots of a sample run with the AK-based terrain GP, illustrating the robot's navigation and terrain estimation improvements. Fig. 7 presents the final results, highlighting the accuracy of the learned terrain model. All simulations were run on a laptop with an Intel i7 CPU and 16 GB of RAM.

We also implement the simulation using full-order dynamics in MuJoCo, as depicted in Fig. 1. We use a variation of the angular momentum LIP planner [24] to track the PSP plans as introduced in [20]. PSP hyperparameters (e.g., $\dot{x}_{apex}$, and $\Delta\theta$) are used to design full-body joint trajectories through geometric inverse kinematics. A passivity-based controller [25] is used for full-body trajectory tracking. The supplemental video for this work shows Digit navigating through multiple environments using our framework.

In Table I, we benchmark the efficacy of the AK, RBF, and NN kernel. For each of the three environments, experiments were conducted with each kernel until three successful runs were achieved. Success was achieved by incrementally increasing the GP retraining frequency (measured in walking steps) until the robot reached the goal without encountering impassable terrain heights. The AK demonstrated the lowest prediction error along the robot's traversed trajectory (Path) and the lowest standard deviation in both path and global map (Env.). The NN kernel excelled in minimizing accumulated errors across the global map, despite higher uncertainty. Although RBF kernel needed fewer steps to reach the goal, it required more frequent GP retraining, whereas AK and NN kernel permit longer intervals between retraining, highlighting their adaptability to non-stationary terrain.

Fig. 8 shows a computational time analysis on terrain N37W112. Initially, the NN kernel incurs higher prediction times due to its local approximation method, but it maintains a consistent time regardless of training data size by using a fixed number of nearest neighbors. Conversely, the AK and RBF start with faster prediction times but slow as training data grows, eventually surpassing the NN kernel. The AK also requires extra training time to optimize kernel weights and membership vectors. Thus, the NN kernel is preferable for long-horizon planning due to its consistent computational time, while the AK is better suited for near-term reach-avoid navigation, offering accuracy and low uncertainty.

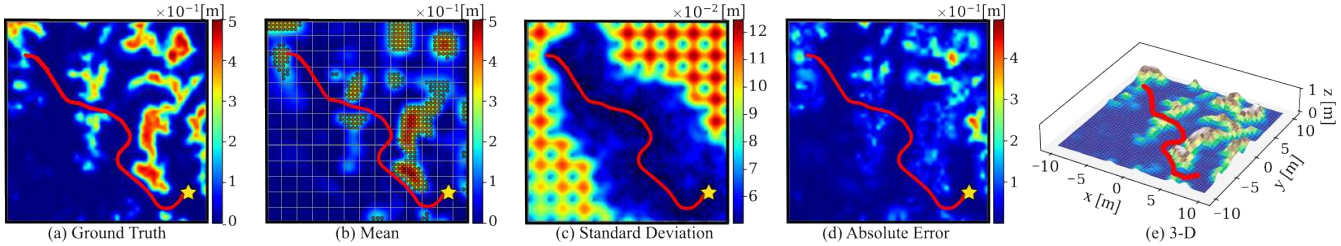
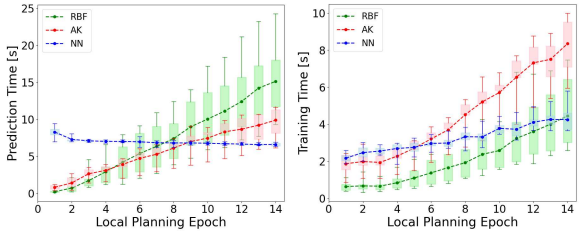


Fig. 7. A sample run of our framework using the AK-based GP. In each graph, the red line denotes the CoM trajectory and the yellow star marks the global goal. (a) Ground truth elevation map with a heatmap; regions with elevation above 0.15 meters are untraversable. (b) Final GP terrain estimation for the AK method. (c) Standard deviation of the terrain GP, indicating higher uncertainty with increasing distance from the trajectory. (d) Absolute error of GP estimation compared to ground truth, showing high accuracy near the trajectory. (e) 3D view of the Digit robot's traversed path.



(a) (b)

Fig. 8. Analysis of the GPs' computational runtime averaging over 10 successful runs per kernels: (a) Prediction times (b) Training times

VII. Conclusion

We propose a hierarchical planning framework for bipedal navigation in rough and uncertain terrain using GP-based uncertainty learning. Future work will involve hardware experiments with Digit navigating outdoor fields.

Acknowledgment

The authors thank Hyunyoung Jung for assistance with the Digit simulation in MuJoCo and Weizhe Chen for visualization of the GP terrain elevation map.

References

[1] A. Torres-Pardo, D. Pinto-Fernández, M. Garabini, F. Angelini, D. Rodriguez-Cianca, S. Massardi, J. Tornero, J. C. Moreno, and D. Torricelli, "Legged locomotion over irregular terrains: State of the art of human and robot performance," *Bioinspiration & Biomimetics*, vol. 17, no. 6, p. 061002, 2022.

[2] G. Gibson, O. Dosunmu-Ogunbi, Y. Gong, and J. Grizzle, "Terrain-adaptive, alip-based bipedal locomotion controller via model predictive control and virtual constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2022, pp. 6724–6731.

[3] J.-K. Huang and J. W. Grizzle, "Efficient anytime clf reactive planning system for a bipedal robot on undulating terrain," *IEEE Transactions on Robotics*, 2023.

[4] M. Dai, X. Xiong, and A. D. Ames, "Data-driven step-to-step dynamics based adaptive control for robust and versatile underactuated bipedal robotic walking," 2022.

[5] L. Krishna, G. A. Castillo, U. A. Mishra, A. Hereid, and S. Kolathaya, "Linear policies are sufficient to realize robust bipedal walking on challenging terrains," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2047–2054, 2022.

[6] F. Wu, Z. Gu, H. Wu, A. Wu, and Y. Zhao, "Infer and adapt: Bipedal locomotion reward learning from demonstrations via inverse reinforcement learning," in *IEEE International Conference on Robotics and Automation*, 2024.

[7] K. Yang, S. Keat Gan, and S. Sukkarieh, "A gaussian process-based rrt planner for the exploration of an unknown and cluttered environment with a uav," *Advanced Robotics*, vol. 27, no. 6, pp. 431–443, 2013.

[8] F. S. Barbosa, B. Lacerda, P. Duckworth, J. Tumova, and N. Hawes, "Risk-aware motion planning in partially known environments," in *IEEE Conference on Decision and Control*, 2021, pp. 5220–5226.

[9] A. Viseras, D. Shutin, and L. Merino, "Robotic active information gathering for spatial field reconstruction with rapidly-exploring random trees and online learning of gaussian processes," *Sensors*, vol. 19, no. 5, p. 1016, 2019.

[10] Z. Jian, Z. Liu, H. Shao, X. Wang, X. Chen, and B. Liang, "Path generation for wheeled robots autonomous navigation on vegetated terrain," *IEEE Robotics and Automation Letters*, 2023.

[11] D. Kanoulas, A. Stumpf, V. S. Raghavan, C. Zhou, A. Toumpa, O. Von Stryk, D. G. Caldwell, and N. G. Tsagarakis, "Footstep planning in rough terrain for bipedal robots using curved contact patches," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4662–4669.

[12] S. Bertrand, I. Lee, B. Mishra, D. Calvert, J. Pratt, and R. Griffin, "Detecting usable planar regions for legged robot locomotion," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4736–4742.

[13] W. Chen, R. Khardon, and L. Liu, "Ak: Attentive kernel for information gathering," in *Robotics: Science and Systems (RSS)*, 2022.

[14] S. Vasudevan, F. Ramos, E. Nettleton, H. Durrant-Whyte, and A. Blair, "Gaussian process modeling of large scale terrain," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 1047–1053.

[15] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard, "A bayesian regression approach to terrain mapping and an application to legged robot locomotion," *Journal of Field Robotics*, vol. 26, no. 10, pp. 789–811, 2009.

[16] T. Seyde, J. Carius, R. Grandia, F. Farshidian, and M. Hutter, "Locomotion planning through a hybrid bayesian trajectory optimization," in *International Conference on Robotics and Automation*, 2019, pp. 5544–5550.

[17] T. Homberger, L. Wellhausen, P. Fankhauser, and M. Hutter, "Support surface estimation for legged robots," in *International Conference on Robotics and Automation*. IEEE, 2019, pp. 8470–8476.

[18] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[19] Y. Zhao, B. R. Fernandez, and L. Sentis, "Robust optimal planning and control of non-periodic bipedal locomotion with a centroidal momentum model," *The International Journal of Robotics Research*, vol. 36, no. 11, pp. 1211–1242, 2017.

[20] A. Shamsah, Z. Gu, J. Warnke, S. Hutchinson, and Y. Zhao, "Integrated task and motion planning for safe legged navigation in partially observable environments," *IEEE Transactions on Robotics*, pp. 1–22, 2023.

[21] F. Leibfried, V. Dutordoir, S. John, and N. Durrande, "A tutorial on sparse gaussian processes and variational inference," 2021, arXiv: 2012.13962 [cs.LG].

[22] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[23] J. Jiang, S. Coogan, and Y. Zhao, "Abstraction-based planning for uncertainty-aware legged navigation," *IEEE Open Journal of Control Systems*, 2023.

[24] Y. Gong and J. W. Grizzle, "Zero Dynamics, Pendulum Models, and Angular Momentum in Feedback Control of Bipedal Locomotion," *Journal of Dynamic Systems, Measurement, and Control*, vol. 144, no. 12, 10 2022, 121006.

[25] H. Sadeghian, C. Ott, G. Garofalo, and G. Cheng, "Passivity-based control of underactuated biped robots within hybrid zero dynamics approach," in *IEEE International Conference on Robotics and Automation*. IEEE, 2017, pp. 4096–4101.