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

Kasidit Muenprasitivej*, Jesse Jiang*, Abdulaziz Shamsah*, Samuel Coogan, and Ye Zhao

***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. 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.

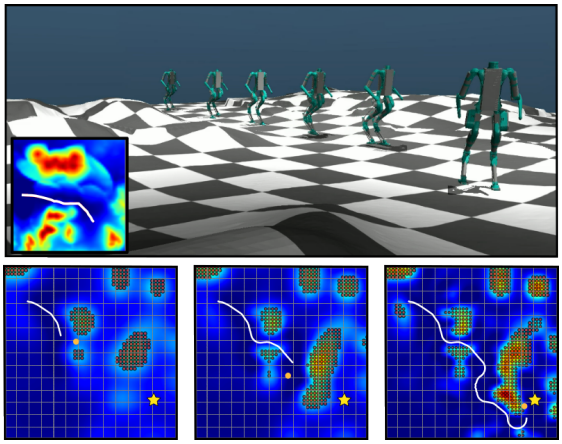


Fig. 1. (Top) The bipedal robot Digit navigates through an environment with rough terrain in our MuJoCo simulation. (Bottom) Snapshots of the trajectory of the bipedal robot at various time instants as it navigates towards the goal (yellow star). The white line depicts the traversed trajectory, and the 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 Figure 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 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 despite environmental occlusion.

The problem of bipedal robot navigation in rough terrain

*Equally contributed authors.

This work was supported in part by a National Science Foundation Graduate Research Fellowship under grant #DGE-2039655.

The authors are with the Institute of Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332, USA (email: {kmuenpra3, jjiang, ashamsah3, sam.coogan}@gatech.edu, ye.zhao@me.gatech.edu).

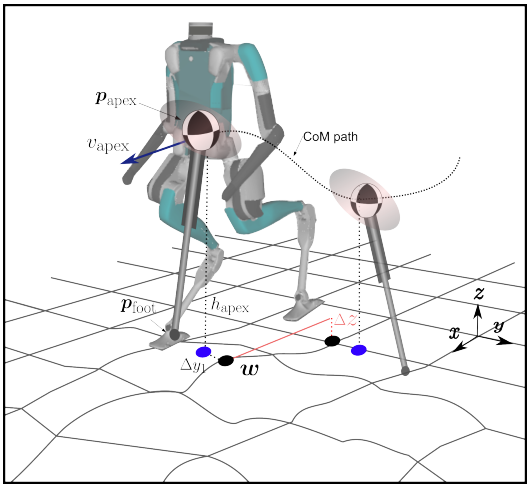


Fig. 2. Prismatic Inverted Pendulum Model (PIPM) model for our robot Digit for traversing over uncertain and uneven terrain.

has not yet been widely explored. The studies in [11], [12] propose methods for identifying stable footstep sequences using sensor data which allow bipedal robots 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. The relative elevation, slope, and friction are used as a traversability cost in the 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.

From a terrain mapping perspective, GPs have been widely used in the literature to quantify terrain uncertainties and learn complex terrain maps. The work in [13] proposes a novel nonstationary kernel which is used to efficiently learn unknown environmental features by prioritizing areas with higher variation for exploration. This kernel is also designed to accurately model rapidly varying terrain by using a mixture model of base kernels and datasets. 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. Another aspect of this work is the use of KD-trees to reduce the dataset size for GP calls, increasing computational tractability.

Leveraging GP approaches to map the terrain has gained increasing attention in the locomotion community. The work in [15] implements a locally adaptive GP for terrain mapping in a legged navigation problem for the Boston Dynamics LittleDog quadruped. The proposed local GP framework seeks to balance the fidelity of the learned model with the computational tractability of training the GP. The authors of [16] use GPs to evaluate candidate trajectories for a hopping robot locomotion planning problem. The work [17] learns a GP-based terrain map from sensor data 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 simulataneously 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 kernels 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 Figure 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 $\boldsymbol{p}_{\text{apex}}$. Δy_1 is the lateral distance between CoM and the high-level waypoint at

apex. Formulating 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 $\sigma^2_{\nu_i}$. 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 $k(\boldsymbol{\xi}) = [\kappa(\boldsymbol{\xi}, \boldsymbol{\xi}^1) \;\; \kappa(\boldsymbol{\xi}, \boldsymbol{\xi}^2) \dots \kappa(\boldsymbol{\xi}, \boldsymbol{\xi}^m)]^T \in \mathbb{R}^m$. 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 $\sigma^2_{g_i,D}$ given by

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

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 kernels for GP are benchmarked for predicting the terrain elevations, namely a radial basis function (RBF) kernel, a Neural Network (NN) kernel [14], and an Attentive Kernel [13].

1) RBF kernel: The RBF kernel is a stationary kernel, commonly adopted in GP regression, in which the predictions are prone to be smooth and have the same degree of 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 nonstationary and models the covariance function of a neural network featuring a single hidden layer with infinitely many nodes and a sigmoid activation function [22]. The NN kernel is defined as

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

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.

3) Attentive Kernel: The Attentive Kernel is nonstationary and adaptive to different variability in the output by applying a weighted sum of many base RBF kernels with different lengthscales. The kernel also utilizes the Instance Selection method, whereby each input location is assigned a membership vector to ensure that training input within the same neighborhood can break their correlation given abrupt changes in the training output. 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 the amplitude constant, $\bar{w}$ and $\bar{z}$ are the weight and membership vector respectively, and $\{\kappa(\boldsymbol{\xi}^i, \boldsymbol{\xi}^j)\}_{m=1}^M$ are base RBF kernels with varying 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, creating obstacles in the environment. Additionally, the terrain elevation is an input to the PSP model, so inaccurate terrain estimations increase deviation and create instability in planned footstep trajectories. Thus, we incorporate an additional objective in the navigation planner to learn the terrain online, improving the dynamical feasibility of planned trajectories.

The primary objective is for the robot to reach a desired location in the environment. However, the terrain elevation is initially unknown, creating untraversable regions and perturbing the motion of the robot. Thus, the robot must sample the environment to learn an accurate representation of the terrain map and feasibly traverse towards the goal. There also exists motion perturbations resulting from the model error between the PIPM used for planning and the full-order dynamics, which complexifies 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.

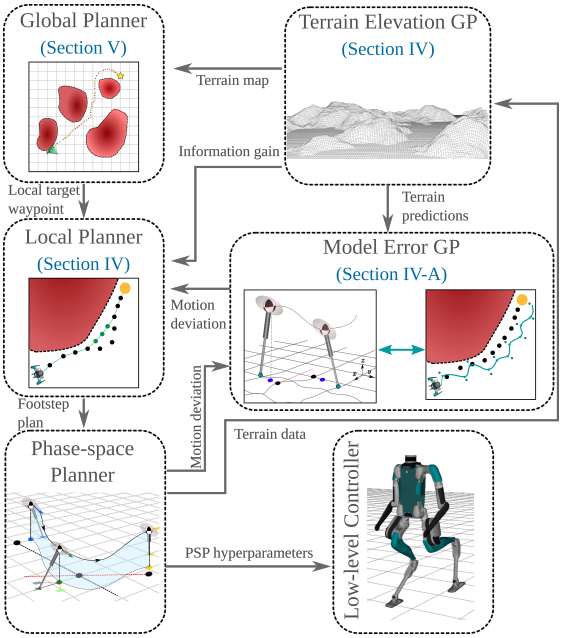


Fig. 3. Overall block diagram of the proposed global-local planning framework for bipedal navigation over rough terrain.

Our approach to this problem is as follows. We first learn the unknown terrain elevation and characterize its effect on the motion of the robot using GPs. Then, we design a local navigation planner which generates dynamically feasible waypoints to avoid untraversable regions and balance objectives of environmental exploration and motion perturbation minimization. Subsequently, we develop a global navigation planner which generates near-horizon targets for the local navigation planner which guide the robot towards the global goal while ensuring dynamically feasible plans. Figure 3 depicts the structure of the proposed approach.

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 $a_{HL,i} = (d_i, \Delta\theta_i, \Delta z_i, \psi_i)$. 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 Figure 4(c). Given a target endpoint generated by a global navigation planner (detailed in Section V), the local navigation planner generates a plan which seeks to reach the target while balancing two objectives. First, the plan encourages exploration so that the robot can more accurately characterize terrain elevation, improving the feasibility of future motion plans. Second, the plan designs a sequence of waypoints which minimize the expected motion perturbation to reach the desired target.

A. Gaussian Process Learning of Terrain and Model Errors

We first detail the GP structure 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 $\hat{z}(x, y)$ for which the input is a global location (x, y) and the output is the corresponding terrain height at that location, z. The kernel of the GP is either the RBF, NN, or Attentive Kernel discussed in Section II-C. We evaluate the respective performance of these approaches in Section VI. At runtime, the GP is trained and updated using 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 the variance is utilized 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 in the lateral direction at each step using the GP $\Delta\hat{y}_1(d_c, \Delta\theta_c, \Delta z_c, d_n, \Delta\theta_n, \Delta z_n)$. The input parameters $d_c, \Delta\theta_c, \Delta z_c$ represent the difference in distance, heading angle, and terrain elevation, respectively, between the previous waypoint and the current waypoint at a given step. The parameters with subscript n represent the same values measured between the current waypoint and the next waypoint. The PSP controller is designed to achieve the desired sagittal distance exactly and then minimize lateral error as a secondary objective. Thus, we choose to focus on modeling lateral deviation.

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 stance of the robot also affects the motion perturbation. Steps taken with the left foot result in lateral deviation to the left of the waypoint (in the local robot frame), whereas steps taken with the right foot deviate to the right of the targeted waypoint. For efficient learning, we learn the absolute value of the lateral deviation with a single model error GP taking into account both left and right footsteps. Then, when calling the model error GP, we assign a positive value for the deviation on left footsteps, and we assign a negative value for the deviation on right footsteps.

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. The GP input parameters $d, \Delta\theta$ can be extracted directly from the waypoint sequence, but the parameter Δz depends on the unknown terrain elevation at each waypoint. Thus, we choose to use the mean $\mu_{\hat{z}}$ of the terrain elevation GP prediction at the waypoints to approximate Δz at each step. 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.

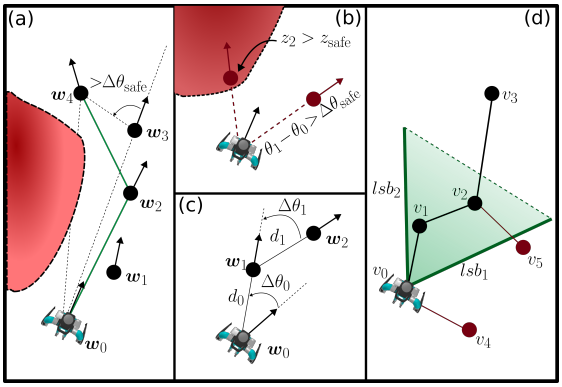


Fig. 4. (a) Illustration of the smoothing algorithm. The goal is to find the smoothest path between the start $\boldsymbol{w}_0$ and the target $\boldsymbol{w}_4$. Connecting $\boldsymbol{w}_0$ and $\boldsymbol{w}_4$ directly is invalid because the line between them passes through an obstacle. Connecting to $\boldsymbol{w}_3$ is also invalid because the resultant heading angle exceeds the limit $\Delta\theta_{\text{safe}}$. Thus, the smoothing algorithm connects $\boldsymbol{w}_0$ to $\boldsymbol{w}_2$ 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. The solid green lines are the locomotion safety barriers, and the shaded green area is their convex hull. Starting from vertex v_0, the connection to v_4 is invalid because v_4 is outside the locomotion safety barriers. The connection from v_2 to v_4 is invalid as it crosses a locomotion safety barrier. The connection from v_2 to v_3 is valid, as the dashed green line is not a locomotion safety barrier.

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

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 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. Figure 4(a) illustrates the smoothing algorithm, Figure 4(b) shows the vertex safety constraints, and Figure 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 $\mathcal{A}^*_{\boldsymbol{w}\to\boldsymbol{w}'}$

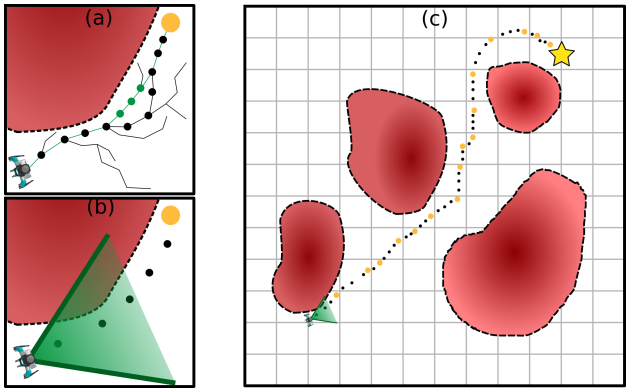


Fig. 5. (a) Trajectory generated by the LDA-L-RRT* algorithm for reaching a local waypoint. The green dots show the trajectory modifications made by the proposed smoothing algorithm. (b) Illustration of the locomotion safety barrier region around the start vertex in the global navigation planner (a zoomed-in version of the one shown in (c)). (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. The green triangle indicates the locomotion safety barrier region constraining vertices near the start point.

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 $error$ predicted by the model error GP over the waypoint sequence and maximizes the information gain $info$, which rewards the traversal of areas currently having high uncertainty in the terrain elevation GP. The parameters α, β tune the importance of these two objectives.

V. Global Navigation Planner

In a large, global environment, it is computationally expensive to perform footstep-by-footstep local planning in order to reach a global goal. Thus, in this section we propose a coarse global navigation planner which plans waypoints as inputs for the LDA-L-RRT* algorithm as described in Section IV. This global navigation planner guides the robot towards the global goal. Then, we detail the complete local-global planning framework.

A. Locomotion-dynamics-aware Global RRT*

For the global navigation planner, we propose another modified RRT* algorithm which incorporates bipedal locomotion constraints while reducing computational costs as compared to the local navigation planner of Section IV for planning in large environments.

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

$$lsb_1 = \left\{ \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + 2d_{\text{step}}\gamma \begin{bmatrix} \cos(\theta_0 - \Delta\theta_{\text{safe}}) \\ \sin(\theta_0 - \Delta\theta_{\text{safe}}) \end{bmatrix} \right\}, \gamma \in [0,1],$$

$$lsb_2 = \left\{ \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + 2d_{\text{step}}\gamma \begin{bmatrix} \cos(\theta_0 + \Delta\theta_{\text{safe}}) \\ \sin(\theta_0 + \Delta\theta_{\text{safe}}) \end{bmatrix} \right\}, \gamma \in [0,1],$$

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}:

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

Figure 4(d) illustrates the locomotion safety barriers.

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 $\{v_i, v_1, \cdots, v_j\}$ 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.

The purpose of the locomotion safety barriers is to ensure that the first several steps of the robot are feasible with respect to the heading angle change for each step.

In practice, we select the step size d_{step} for the LDA-G-RRT* algorithm larger than the step size d_{safe} for the local LDA-L-RRT* algorithm. 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, whereas LDA-L-RRT* constrains every vertex. These two

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

features result in the computational efficiency of LDA-G-RRT*. Figure 5(b),(c) illustrates LDA-G-RRT*.

B. Overall Framework

We now detail the overall planning framework. We assume that there exists *a priori* a small dataset of terrain elevation data points, and we initialize the terrain GP by training on this dataset. This assumption can be relaxed by, *e.g.*, initializing the hyperparameters of the terrain GP to have a conservative level of uncertainty throughout the environment. The model error GP is trained offline as in Section IV-A. We then run the LDA-G-RRT* algorithm to obtain a local target waypoint, which we send to the LDA-L-RRT* algorithm to generate a footstep-by-footstep motion plan. The robot executes this motion plan, collecting terrain elevation data along the trajectory. Once the robot reaches the local target waypoint, the terrain GP is retrained on the newly collected data. Then, LDA-G-RRT* is run again and the rest of the process repeats until the global target waypoint is reached. The complete framework is summarized in Algorithm 1.

VI. Results

We evaluate our framework on simulations of a Digit bipedal robot navigating three environments with varying terrain. Each environment is 20×20 meters in size, with terrain elevation varying between 0 and 0.5 meters. The terrain GP $\hat{z}$ model is initialized on prior information consisting of 100 evenly-spaced points across the terrain. Then, the GP model is updated online as new data is collected as the robot traverses the environment. The sensor is programmed to simulate the collection of 10 sample points within a 3-meter radius of the robot during each step. The model error GP $\Delta\hat{y}_1$ is trained offline as described in Section IV-A. Across all simulations, the average motion perturbation per step was 1.75e-2 meters, and the average prediction error of $\Delta\hat{y}_1$ per step was 2.06e-4 meters. Thus, $\Delta\hat{y}_1$ accurately evaluates the motion perturbation. Finally, for the LDA-L-RRT* local navigation planner, we evaluate three candidate trajectories for each local target.

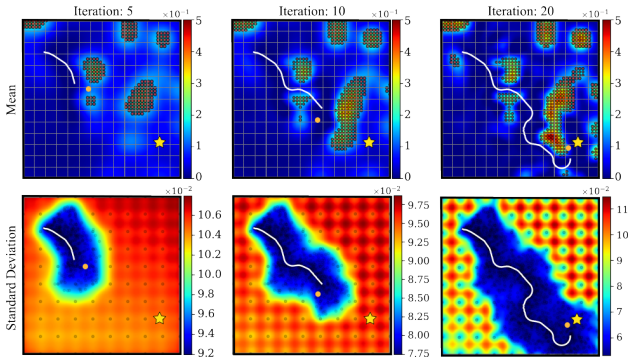


Fig. 6. Snapshots of the state of the system at various points throughout one sample run. In all plots, the white line is the CoM trajectory, the orange dot is the local target waypoint, and the yellow star is the global goal. In the left plot, which is early in the trajectory, the robot has only characterized the terrain in a small region around the start point. In the middle plot, the robot has safely passed through the first set of obstacles based on the earlier terrain data gathered, and it has learned most of the large obstacle blocking the goal. Finally, the last plot shows the completed run, where the robot has learned the terrain sufficiently to reach the goal.

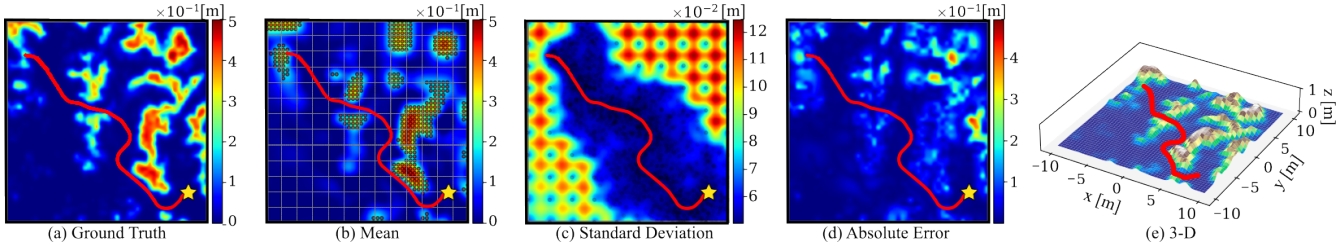
Figure 6 depicts snapshots during a sample run of our framework using the Attentive Kernel for the terrain GP, in which the robot safely traverses the environment, improving its estimation of the terrain until it is able to reach the global goal by navigating around the obstacle regions. Figure 7 shows the final results for the same run, illustrating 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 Figure 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., CoM velocities, and heading change) 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.

TABLE I
Benchmarking for Evaluated GPs

Metric	Attentive Kernel	RBF Kernel	NN Kernel
Avg. Error (Path)	3.05e-3	**7.73e-4**	2.02e-4
Avg. Error (Env.)	**78.9**	81.4	126
Avg. Time (sec)	231	**167**	310
Avg. Steps	196	188	**161**
Success Rate	**90%**	60%	<53%

Finally, in Table I we benchmark the efficacy of the Attentive Kernel, RBF, and NN Kernel GPs in our case studies. For each of the three environments, we run experiments using each GP until we achieve three successes. Overall, the Attentive Kernel framework exhibits the highest success rate of reaching the global goal and minimizes the prediction error on the global terrain map (Env.), *i.e.*, the entire environment. The RBF kernel is the least expensive to implement computationally and achieves the lowest prediction error along the traversed trajectory (Path), *i.e.*, where the



(a) Ground Truth (b) Mean (c) Standard Deviation (d) Absolute Error (e) 3-D

Fig. 7. A sample run of our proposed framework using an Attentive-Kernel-based GP. In each graph, the red line represents the CoM trajectory; the yellow dots are the discrete footsteps; and the yellow star is the global goal location. (a) The ground truth elevation map with the heatmap. Regions with elevation above 0.15 meters are untraversable. (b) The final GP estimation of the terrain for the Attentive Kernel method. (c) The standard deviation of the terrain GP. Higher values indicate a higher uncertainty. Note that, as expected, the robot has a high confidence around its actual trajectory and low confidence far from where it has sampled. (d) The absolute error of the terrain GP (b) compared to the ground truth (a). The estimation shows high accuracy near the actual trajectory, and the robot does not need an accurate estimation of terrain far away from its trajectory in order to reach the goal. (e) A 3D view of the Digit robot navigation trajectory through the environment.

robot actually stepped, but has a lower success rate than the Attentive Kernel. Finally, the NN Kernel requires the fewest number of steps on average to reach the global goal but it fails to find a successful path on one of the environments, thus making it the least reliable. The NN kernel, although non-stationary, correlates test points with both close-by and far-off points in the terrain, thus hindering performance in regions with rapidly varying terrain.

VII. Conclusion

In this work, we propose a novel hierarchical planning framework for bipedal navigation in rough and uncertain terrain environments using GP learning of uncertainty. Future work will implement this planning framework on hardware experiments of Digit navigating through outdoor fields.

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.