Autonomous Mapless Navigation on Uneven Terrains

Hassan Jardali, Mahmoud Ali, and Lantao Liu

arXiv:2402.13443v1 [cs.RO] 21 Feb 2024

Abstract—We propose a new method for autonomous navigation in uneven terrains by utilizing a sparse Gaussian Process (SGP) based local perception model. The SGP local perception model is trained on local ranging observation (pointcloud) to learn the terrain elevation profile and extract the feasible navigation subgoals around the robot. Subsequently, a cost function, which prioritizes the safety of the robot in terms of keeping the robot's roll and pitch angles bounded within a specified range, is used to select a safety-aware subgoal that leads the robot to its final destination. The algorithm is designed to run in real-time and is intensively evaluated in simulation and real-world experiments. The results compellingly demonstrate that our proposed algorithm consistently navigates uneven terrains with high efficiency and surpasses the performance of other planners. The code can be found here: https://rb.gy/3ov2r8.

I. Introduction

Autonomous navigation in outdoor unstructured environments poses a significant challenge due to unpredictable, rugged terrains. Advanced navigation algorithms, utilizing sensors like LiDARs and cameras, have been developed to tackle these issues, enabling robots to navigate complex and challenging terrains more precisely and in real-time. In order to accomplish autonomous navigation, various modules such as sensing, perception, localization, planning, and control are essential. Our main contribution is a GP-based local perception model capable of extracting feasible, traversable navigation points from the robot's immediate environment. To date, the approaches used to navigate challenging terrains can be categorized as map-based methods and mapless methods, where for the map-based methods, a global map is built based on which the path planing techniques are utilized to drive the robot; and for mapless methods the robot uses only its current sensor reading to reach a pre-specified goal without referring to any global map. Mapless navigation offers several advantages such as less computational power as it obviates the necessity for continuous map updates. It also facilitates rapid response to dynamic obstacles and alterations in the environment.

We have recently developed a novel method to process LiDAR pointclouds by converting them into a "truncated" *Occupancy Surface* where the pointclouds received within a pre-specified radius are projected onto this surface [1]. The occupancy surface represents a more compressed form of the environment which enables an efficient communication of the

robot's local observation. Additionally, this occupancy surface was used in navigation and exploration as demonstrated in [2]–[4]. This occupancy surface serves as training data for a GP occupancy model, which predicts a continuous form of occupancy around the robot. In this paper, we leverage the uncertainty inherent in the GP occupancy model to estimate the traversability of the uneven terrain and identify local navigation points. Subsequently, a cost function is applied to evaluate these navigation points, selecting the most promising point for guiding the robot towards its final destination as the subsequent navigation subgoal. This selection process takes into account constraints on the robot's orientation, ensuring that they remain within its safe bounds. Our approach has been verified through rigorous simulations, comparative benchmarks and real-world demonstrations. The results show that our proposed method enables the robot to perform efficient mapless navigation in uneven terrains.

II. Related Work

Autonomous navigation frameworks can be generally categorized into two paradigms: sense-plan-act and end-to-end. The first involves mapping, traversability analysis, planning, and control, whereas the second employs learning-based approaches like deep learning and reinforcement learning [5]. In this study, we focus on methodologies within the sense-plan-act paradigm. Terrain traversability analysis assesses the difficulty of a vehicle might face when navigating a particular terrain section. It aids robots in choosing the most optimal route between multiple possible paths [6]. Grid-based, 2.5D elevation [7] and 3D OctoMap [8] mapping methods are common approaches to solve the map representation problem. In [9], a multi-layer 2D map is obtained from a 3D OctoMap then employed to navigate staircases and slopes, however, its effectiveness is constrained in navigating intricate environments [10]. [7] introduced a real-time elevation mapping technique that mitigates localization drift in legged robots. This approach creates a probabilistic terrain estimate using a grid-based elevation map, where each cell provides an estimated height along with confidence boundaries. The studies detailed in [6], [11] leverage elevation maps to craft strategies for guiding legged robots safely and efficiently across irregular, unstructured terrains. Additionally, in [6], terrain material is classified using learning methods based on RGB data, which is subsequently integrated in the traversability analysis as a navigability criterion. These frameworks extract geometric properties from the terrain's elevation information within each

Authors are with the Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN 47408 USA (e-mail: {hjardali, alimaa, lantao}@iu.edu).

grid cell, producing a traversability score that is logged in a traversability cost map. This score, derived from the terrain's geometric features, is computed by considering factors such as roughness, slope, sparsity, step height, and flatness [6], [10]–[13]. Path planning algorithms such as A* search [13], [14], Dijkstra's algorithm [15], and the rapidly-exploring random tree (RRT) algorithm [16], are then used to generate the path after building the map. A motion command is finally sent to the robot's controller after the path is generated. Optimization methods are sometimes included in the controller such as in [10], [13]. In [13], the authors introduce a distinctive field construction algorithm in 3D space, named the Valid Ground Filter (VGF), that processes the original map taking into account the robots navigation capabilities. In [10], 3D grid map is used to build the map, along PF-RRT*, an RRT variant based on the plane fitting method to include the traversability information while generating the path. The framework in [10] depends also on a SLAM module, which makes it prone to cumulative errors and difficult to scale on large maps.

Navigation systems that operate without reliance on maps, frequently utilize reactive behavior methodologies such as the dynamic window approach (DWA) [17], the vector field histogram (VFH) [18], and Artificial Potential Fields (APF) [19] approaches. Although these methods were originally devised for 2D navigation, advancements in robotics and research have seen their extension to 3D scenarios [20].

Unlike the methods discussed earlier, our approach to autonomous navigation in uneven terrains relies on a novel lightweight perception modeling technique that utilizes the uncertainty of the GP. By using LiDAR readings as training data, the GP model is used to create a representation of the robot's surroundings. The GP uncertainty is then utilized to obtain the terrain elevation profile and identify local navigation subgoals. These subgoals are evaluated based on a cost function that drives the robot towards the final goal while ensuring its safety by imposing penalties on steep paths.

III. PRELIMINARIES

Our local perception model is based on Gaussian Process (GP). In this section, we provide preliminary background of GP. GP is a non-parametric model defined by a mean function $m(\mathbf{z})$ and a covariance function $k(\mathbf{z}, \mathbf{z}')$, which is known as GP kernel, where $\mathbf{z} \in \mathbb{R}^d$ represents the input [21],

$$f(\mathbf{z}) \sim \mathcal{GP}\left(m(\mathbf{z}), k\left(\mathbf{z}, \mathbf{z}'\right)\right). \tag{1}$$

In GP regression, we train a GP model on dataset $D = \{(\mathbf{z}_i, y_i)\}_{i=1}^{n}$, where $y_i = f(\mathbf{z}_i) + \epsilon_i$, with $f(\mathbf{z}_i)$ is the underlying function and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is Gaussian noise. Then, the output y^* of a new input $\mathbf{z}^*$ is calculated using the GP posterior mean and co-variance functions, $m_{\mathbf{y}}(\mathbf{z})$ and $k_{\mathbf{y}}(\mathbf{z}, \mathbf{z}')$, respectively, as follow

$$p(y^*|\mathbf{y}) = \mathcal{N}(y^*|m_{\mathbf{y}}(\mathbf{z}), k_{\mathbf{y}}(\mathbf{z}^*, \mathbf{z}^*) + \sigma^2),$$

$$m_{\mathbf{y}}(\mathbf{z}) = K_{\mathbf{z}n}\left(\sigma^2 I + K_{nn}\right)^{-1}\mathbf{y}, \tag{2}$$

$$k_{\mathbf{y}}\left(\mathbf{z}, \mathbf{z}'\right) = k\left(\mathbf{z}, \mathbf{z}'\right) - K_{\mathbf{z}n}\left(\sigma^2 I + K_{nn}\right)^{-1}K_{n\mathbf{z}'},$$

where $K_{\mathbf{z}n} \in \mathbb{R}^n$ is n-dimensional row vector of kernel function values between $\mathbf{z}$ and the training inputs, with $K_{n\mathbf{z}} = K_{\mathbf{z}n}^T$, while $K_{nn} \in \mathbb{R}^{n \times n}$ refers to the $n \times n$ co-variance

matrix of the training inputs. Maximizing the logarithm of the marginal likelihood, described in eq. 3, is a common approach to optimizing hyper-parameters, including kernel parameters Θ and noise variance σ^2, which in turn leads to more accurate GP predictions.

$$\log p(\mathbf{y}) = \log\left[\mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, \sigma^2 I + K_{nn}\right)\right]. \tag{3}$$

The computational cost of the Gaussian Process (GP), which is $\mathcal{O}(n^3)$, can be prohibitive in some applications. To address this issue, Sparse Gaussian Process (SGP) has emerged as an effective alternative. By using a smaller set of training points, known as the "inducing points" Z_m, SGP achieves lower computational complexity of $\mathcal{O}(nm^2)$ [22]–[24]. In this work, we utilize the Variational SGP (VSGP) approach introduced in [25], which estimates the true posterior of GP $p(f|\mathbf{y})$ with an approximated variational posterior distribution $q(f, f_m)$ by introducing an unconstrained variational distribution $\phi(f_m)$ as follow

$$p(f, f_m|\mathbf{y}) = p(f|f_m)p(f_m|y),$$
$$q(f, f_m) = p(f|f_m)\phi(f_m), \tag{4}$$

where f_m are the values of $f(\mathbf{z})$ at Z_m and $p(f|f_m)$ is the conditional GP prior. Using VSGP, the inducing inputs Z_m and hyperparameters (Θ, σ^2) are estimated by minimizing the Kullback-Leibler (KL) divergence between the true and approximated posteriors, $\mathbb{KL}[q(f, f_m)||p(f|\mathbf{y})]$.

IV. PROBLEM DEFINITION

Consider a differential wheeled mobile robot with a motion model given as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix},$$

with $\mathbf{x} = (x, y, \theta)$ is the robot pose, where θ is the robot's rotation angle around the z-axis (yaw); and $\mathbf{u} = (v, \omega)$ is the control input, where v is the linear velocity and ω is the angular velocity.

Let $\mathbf{x}_s = (x_s, y_s, \theta_s)$ and $\mathbf{x}_f = (x_f, y_f, \theta_f)$ denote the robot's initial and final poses, respectively. The robot has to move from $\mathbf{x}_s$ to $\mathbf{x}_f$ through $\mathbb{R}^3$, where $\mathbb{R}^3$ represents the 3D space of the global environment where the robot moves, with $\mathcal{S}_n \subset \mathbb{R}^3$ and $\mathcal{S}_{nn} \subset \mathbb{R}^3$ representing the global navigable and non-navigable spaces, respectively.

The navigation task can be described as finding the optimal path that guides the robot to move from a starting pose, $\mathbf{x}_s$, to a final goal $\mathbf{x}_f$, safely and efficiently while avoiding obstacles and minimizing costs (e.g. traveled distance, robot's pitch and roll angles, etc). In the context of terrain navigation, safety means preventing the robot from flipping or skidding by keeping the roll ϕ and pitch ψ angles, robot's rotation angles around the X-axis and Y-axis, respectively, in a safe bounded range such as $\phi_{min} \leq \phi < \phi_{max}$ and $\psi_{min} \leq \psi < \psi_{max}$.

Considering only the local perception, we simplify this task by only considering the local space around the robot $\mathcal{S}_l^3 \subset SE(3)$, with $\mathcal{S}_{l_n} \subset \mathcal{S}_l^3$ and $\mathcal{S}_{l_{nn}} \subset \mathcal{S}_l^3$ representing the local navigable and non-navigable spaces, respectively. Subsequently, the local navigable space $\mathcal{S}_{l_n}$ is used to extract a set of feasible subgoals $\mathcal{G}$ around the robot. Then, an optimal subgoal g^* is selected based on a cost function $\mathbf{J}$, which
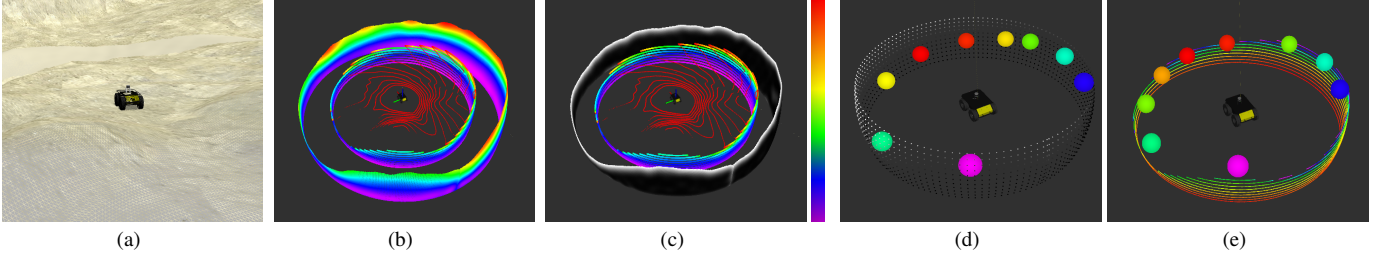
Fig. 1: (a) Husky robot in a simulated hilly terrain environment; (b) Original (inner) vs predicted (outer) occupancy surfaces, where warmer colors indicate less occupancy; (c) Original occupancy (inner) vs. variance (outer) surface; (d) The full variance surface with subgoals shown as colored circles; (e) The original occupancy surface with the extracted subgoals.

minimizes the distance and direction of the robot with respect to the target's pose $\mathbf{x}_f$. Finally, a control input $\mathbf{u}$ is generated to drive the robot towards a subgoal g^*.

V. METHODOLOGY

In this section, we initiate our discussion with the Gaussian Process based perception model, then we cover the process of subgoals generation, and concluding with the formulation of the cost function used to select the most optimal subgoal.

A. Perception Representation using Gaussian Process

This paper extends our previous work [1] which represents the observed pointcloud as *occupancy surface*, then models it using the SGP. The occupancy surface is a 2D circular surface around the sensor center with a predefined radius r_{oc}. Higher r_{oc} values indicate the compression of a greater amount of terrain information along each LiDAR beam. The points of the observed pointcloud are transformed into the spherical coordinates (α_i, β_i, r_i) and projected on the occupancy surface, where (α_i, β_i, r_i) correspond to the azimuth, elevation, and radius values, respectively. Any point $\mathbf{z}_i$ on the surface is described by the azimuth and elevation angles $\mathbf{z}_i = (\alpha_i, \beta_i)$, and given an *occupancy value* $f(\mathbf{z}_i)$, where $f(\mathbf{z}_i) = r_{oc} - r_i$. Thus, the occupancy distribution over the surface, $f(\mathbf{z})$, is modeled as an SGP occupancy model,

$$f(\mathbf{z}) \sim \mathcal{SGP}\left(m(\mathbf{z}), k\left(\mathbf{z}, \mathbf{z}'\right)\right),$$
$$k\left(\mathbf{z}, \mathbf{z}'\right) = \sigma^2 \left(1 + \frac{(\mathbf{z} - \mathbf{z}')^2}{2\gamma \ell^2}\right)^{-\gamma}, \quad (5)$$

where $k\left(\mathbf{z}, \mathbf{z}'\right)$ is a Rational Quadratic (RQ) kernel with a length-scale l, a signal variance σ^2, and a length-scale weighting factor γ. The RQ kernel is used because of its capacity to model functions that vary across diverse length-scales [21]. The zero-mean function $m(\mathbf{z}) = 0$ sets the prior occupancy of the surface to zero.

The prediction of the SGP occupancy model consists of a mean value indicating the predicted occupancy and a variance value revealing the uncertainty of the predicted occupancy at each point on the occupancy surface. Hence, the outcome of the SGP occupancy model consists of two surfaces, the predicted (reconstructed) occupancy surface and the variance (uncertainty) surface, which indicates the uncertainty of the predicted occupancy, see Fig. 1. The LiDAR does not cast any pointclouds if the laser beams do not hit any obstacle; hence, the absence of pointclouds causes high variance in the corresponding areas in the variance surface. Therefore, free

space areas can be conveniently defined as high variance areas where the variance is higher than a certain threshold V_{th}, a tunable parameter that is related to the mean of the variance distribution over the variance surface.

B. Subgoals Placement

The variance surface offers valuable insights into the morphology of the terrain's around the robot. In flat environments [3], the variance surface reveals the distribution of obstacles and identifies free spaces. In uneven terrains, the variance surface displays the varying elevations of the terrain, as depicted in Fig. 2. We opted to utilize the variance surface as it more smoothly and distinctly delineates occupied and free spaces, facilitating the thresholding process and thus the mapping of local terrain elevations. Specifically, Fig. 2 is an unfolded version of the circular variance surface (the out-most surface) shown in Fig. 1(c), where dark regions (lower part) represent occupied space (terrain), while white regions (upper part) indicate the free space. The border between the occupied and free spaces is represented by an uneven curve. This curve is discretized into a set of segments, $\mathbf{S} = \{s_i\}_{i=1}^{Q}$, where each segment s_i reflects an elevation level of the terrain around the robot. Each segment s_i is defined by two azimuth angles, α_{sb_i} and α_{se_i}, which indicate the beginning and the ending points of the segment, respectively, where the segment width w_{s_i} is calculated as $w_{s_i} = r_{oc}(\alpha_{se_i} - \alpha_{sb_i})$. Additionally, each segment has an elevation angle β_{s_i} coupled with a height value, $z_{s_i}^R$, which indicates the terrain elevation in the robot's frame $\mathcal{R}$, see Fig. 2. The segments $\mathbf{S}$ are used to place a set of potential local navigation subgoals, $\mathcal{G} = \{g_i\}_{i=1}^{K}$ around the robot, where the segment height $z_{s_i}^R$ is used to calculate the steepness cost C_{stp} of the corresponding subgoal g_i.

A segment s_i is considered for subgoal positioning *if and only if*: i) the segment width w_{s_i} is greater than the robot width augmented by a safety margin, i.e., $w_{s_i} > w_r + \delta$, where δ is a constant representing the safety threshold; and ii) the segment height $z_{s_i}^R$ falls within a safe slope range, which is determined by the robot's climbing capabilities influenced by its dynamics. For each segment s_i that satisfies the previously mentioned conditions, we initially place a subgoal g_i at the *center* of the segment. This placement is achieved using the spherical coordinates in $\mathcal{R}$, denoted as $g_i = (\alpha_{g_i}, \beta_{g_i}, r_{oc})$, where $\alpha_{g_i} = (\alpha_{se_i} - \alpha_{sb_i})/2$ and $\beta_{g_i} = \beta_{s_i}$. If a segment has a width w_s exceeding $n \cdot (w_r + \delta)$, where $n \geq 3$, we distribute $n - 1$ subgoals at equidistant points along the segment to facilitate a smoother path toward the global goal x_f. The cartesian coordinates of a subgoal g_i, in the global frame $\mathcal{W}$, is
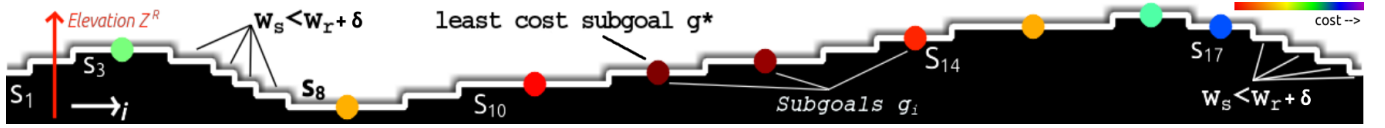
Fig. 2: Unfolded variance surface transformed into a 2D image, where dark areas indicate the location of the observed points, and the white areas above the dark area represent the free space. The horizontal segments $\mathbf{S}\{s_i\}_{i=1}^{Q}$ reflect the discretized terrain's elevation profile of the scene shown in Fig. 1 as seen from the robot's sensor. $Z_{s_i}^{R}$ is the height of a segment s_i in the robot's frame $\mathcal{R}$. A set of subgoals $\mathcal{G} = \{g_i\}_{i=1}^{K}$ is placed at the centers of the corresponding segments whose width satisfies $w_{s_i} < w_r + \delta$. The subgoals' colors indicate the cost assigned to each subgoal g_i by the cost function $\mathbf{J}$, where warmer colored subgoals have less cost.
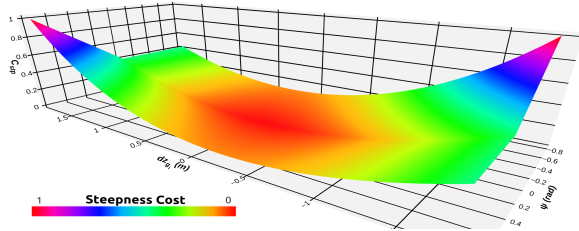

Fig. 3: This graph illustrates the behavior of the steepness cost function $\mathbf{C}_{\text{stp}}$ with respect to the pitch angle ψ and the relative elevation of subgoals (dz_{g_i}). For negative ψ (indicating the robot is inclined upwards), the cost is higher when dz_{g_i} is negative (subgoals are higher than the robot) compared to when dz_{g_i} is positive (subgoals are lower than the robot). Conversely, for positive ψ (indicating the robot is inclined downwards), the function assigns a lower cost to negative dz_{g_i} values and a higher cost to positive dz_{g_i} values. This structure ensures the containment of roll and pitch angles within safe limits.

retrieved as $(x_{g_i}^{w}, y_{g_i}^{w}, z_{g_i}^{w}) = {}^{W}\mathbf{T}_R(x_{g_i}^{R}, y_{g_i}^{R}, z_{g_i}^{R})$, where ${}^{W}\mathbf{T}_R$ is the transformation between $\mathcal{R}$ and $\mathcal{W}$ (robot localization), and $(x_{g_i}^{R}, y_{g_i}^{R}, z_{g_i}^{R})$ are the cartesian coordinates of the subgoal g_i in $\mathcal{R}$.

C. Selection of Optimal Feasible Subgoal

After the placement of the subgoals $\mathcal{G} = \{g_i\}_{i=1}^{k}$, a cost function $\mathbf{J}$ is used to evaluate each subgoal g_i. In [3], the distance and the direction between the subgoal g_i and final goal $\mathbf{x}_f$ were used as factors to drive the robot in flat terrains. For the uneven terrain case, we first check the feasibility of the subgoal based on the steepness of the terrain using the steepness cost function $\mathbf{C}_{\text{stp}}$ in Eq. (6), which depends on the elevation of the goal with respect to the robot, $dz_{g_i} = z_{g_i}^{w} - z_r^{w}$, and the robot's pitch angle ψ, where $z_{g_i}^{w}$ and z_r^{w} are the Z-coordinates of the subgoal g_i and the robot in $\mathcal{W}$, respectively. An exponential *pitch-cost*, $\exp(sign(\psi)dz_{g_i})|\psi|$, is used to penalize the subgoals that will increase the absolute value of the robot's pitch angle, and a quadratic *elevation-change* cost, $dz_{g_i}^{2}$, to penalize the subgoals that lead to greater elevation changes $\dot{z}_r$, see Fig. 3. We add to the steepness cost a distance cost $\mathbf{D}_{fg_i}$ [26] and a direction cost α_{g_i} [27]. The $\mathbf{D}_{fg_i}$ represents the distance between each subgoal g_i and the final goal $\mathbf{x}_f$, and α_{g_i} represents the azimuth angle of the subgoal g_i around the Z-axis. This angle is defined as the difference in heading between the robot's current orientation and the direction towards g_i, measured in the robot's frame. The three costs are normalized and combined in one cost function $\mathbf{J}$ as

follows

$$\mathbf{J}(g_i) = k_{dir}\alpha_{g_i} + k_{dst}\mathbf{D}_{fg_i} + k_{stp}\mathbf{C}_{stp},$$
$$\mathbf{D}_{fg_i}(g_i) = ((x_f^{w} - x_{g_i}^{w})^2 + (y_f^{w} - y_{g_i}^{w})^2)^{\frac{1}{2}},$$
$$\mathbf{C}_{stp}(g_i) = (dz_{g_i})^2 + \exp(sign(\psi)dz_{g_i})|(\psi)|, \qquad (6)$$
$$g^{*} = \arg\min_{g_i \in \mathcal{G}}(\mathbf{J}(g_i)),$$

where coefficients k_{dst}, k_{stp}, and k_{dir} can be adjusted through empirical tuning to account for the terrain's flatness and steepness while also considering the dynamics of the robot. The subgoal with the minimum cost g^{*} is selected as the next navigation subgoal. Finally, a control input $\mathbf{u}$ is generated to drive the robot to g^{*} using a simple PID controller with a distance error equals to the distance between the selected subgoal g^{*} and the robot, and a yaw-orientation error of $\alpha_{g^{*}}$.

VI. SIMULATION-BASED EVALUATION

Our method is coded in Python using *gpflow* [28] and integrated with the Robot Operating System (ROS) framework. It runs in *real-time* with a frequency of 10 Hz on a `core-i7` PC with Nvidia RTX-2060 GPU.

A. Simulation Setup

We tested our approach using a simulated *Husky* robot equipped with a 16-beam Velodyne LiDAR in Gazebo simulator. The LiDAR is used to construct the occupancy surface, which spans an azimuth range of $-180^{o} \leq \alpha < 180^{o}$ and an elevation angle range of $-15^{o} \leq \beta < 15^{o}$. The occupancy surface radius r_{oc} is set to 7 meters, while the variance threshold V_{th} is set to 0.7. For the simulation experiments, the weighting factors of the cost function are set as the direction factor $k_{dir} = 0.2$, the distance factor $k_{dst} = 0.3$, and the steepness factor $k_{stp} = 0.5$.

B. Performance Metrics and Simulation Scenarios

The performance of the proposed method is evaluated through extensive simulations and compared to a very recent uneven-terrain navigation method [13], referred here as Valid Ground Filter (VGF). This planner addresses the navigation in uneven terrain using a map-based strategy. The outcomes highlighted in the article demonstrated its superior performance over other methods, and the authors have made it open source. Two sets of performance metrics are considered for uneven terrain navigation: *safety-related* metrics and *motion-related* metrics. The *safety-related* metrics encompass the robot's roll ϕ and pitch ψ angles, the change in elevation $\dot{z}_r$ during the robot's movement from $\mathbf{x}_s$ to $\mathbf{x}_f$, and the vibration $\mathbf{v}_b$, as defined in [29] as the cumulative rate of change for the roll and pitch angles: $\mathbf{v}_b = |\omega_r| + |\omega_p|$, with $\omega_r = \dot{\phi}$ and $\omega_p = \dot{\psi}$. A lower vibration value is indicative of a more stable trajectory.

The *motion-related* metrics include the distance D taken to achieve the final goal and the average trajectory curvature change C_{chg} as in [30], representing the path's oscillations,

$$C_{chg} = \frac{1}{T}\int_0^T |k'(t)|\,dt, \quad k(t) = \left|\frac{\omega(t)}{v(t)}\right|.$$

A superior performance is denoted by reduced values for both safety and motion-related metrics outlined above. To ensure the robot's safety, we adhere to the safety boundaries defined by the robot's manufacturer, *Clearpath Robotics* in our case, which stipulates that the vehicle tends to slide or skid when its roll exceeds 0.524 rad or its pitch exceeds 0.785 rad. Hence, we define a trial as successful if the robot reaches the final goal and its roll and pitch angles remain within the aforementioned constraints; otherwise, the trial is considered a failure. Furthermore, achieving the final destination, $\mathbf{x}_r = \mathbf{x}_f$ is a requisite for a successful simulation. Three distinct simulated environments were used to evaluate the performance of our approach:

1) **The Challenging Hilly Terrain (CHT) Environment**: A versatile environment offering three distinct navigable paths, each with its own set of challenges.
2) **The Martian Landscape (ML) Environment**: Resembling the terrain of the Mars planet.
3) **The Valley Trail Ascent (VTA) Environment**: This environment portrays a valley leading to a trail that ascends to a higher point on a small mountain.
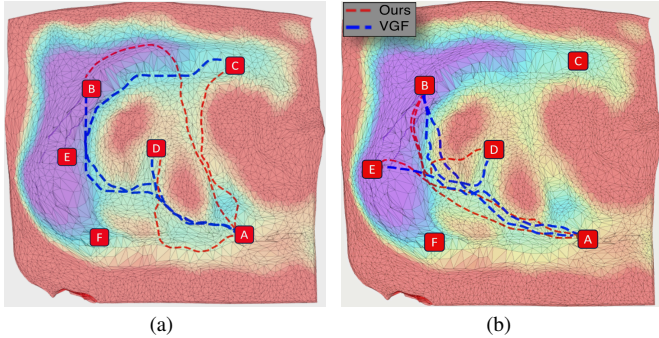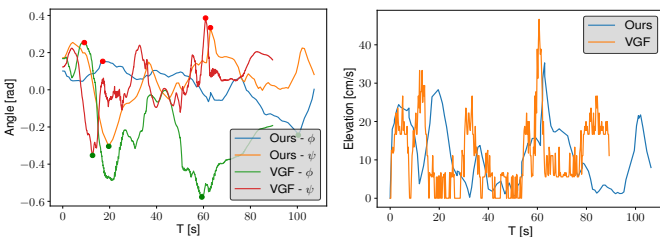
C. Results

(a) (b)

Fig. 4: Path comparison in the CHT environment between our proposed approach and the baseline. The color variations represent the terrain elevation, with red shades denoting higher areas.


(a) (ϕ) and (ψ) angles variation (b) Elevation change $\dot{z}_r$ profile

Fig. 5: Illustrations of roll and pitch angle variations and elevation changes during the AB path trials, as depicted in 4(a).

The performance of our method is evaluated in the CHT environment against the VGF baseline [13] through hundreds of simulation trials. Fig. 4 illustrates the paths pursued by both methods, delineating the starting and ending points.

Fig. 5 exhibits the angle and elevation variations along the AB path. In Table I, where a quantitative comparison of the results derived from a selection of trials is presented, we can notice that our proposed method consistently exhibited lower elevation changes $\dot{z}_r$, reduced vibration $\mathbf{v}_b$, and smaller maximum absolute angles for both roll (ϕ) and pitch (ψ), reflecting an enhanced emphasis on safety. However, in terms of curvature change C_{chg}, the map-based approach yielded superior outcomes, a logical result considering the ability of map-based methods to provide optimized global paths. There were instances where the baseline paths closely skirted steep hills, as illustrated in Fig. 4, leading to significant roll (ϕ) values, like in the AC and AB paths scenario, consequently, the robot encountered multiple failures. For the AB scenario, although our proposed method opted for a lengthier trajectory, it experienced reduced maximum angles compared to the baseline as depicted in 5(a). Moreover, as seen in 5(b), our approach demonstrated a lower rate of elevation change. For the BD path, while the baseline attempted to bypass the hill, it encountered high angles values. In contrast, our method ascended the hill via navigable terrain, resulting in a safer traversal experience. We should emphasize that our analysis was strictly based on the global paths produced by the baseline method, prior to the implementation of the MPC. The data we gathered was obtained from the Husky robot as it navigated the generated waypoints using a PID controller.

It is worth pointing out that a major difference between the two methods is that our method is mapless, while the baseline relies on a map-based strategy. The baseline method initially requires the map to be provided as pointcloud data, followed by analysis and planning using A*. This approach can yield more robust paths. Conversely, our algorithm operates with minimal prior information, limited to the global goal's relative position to the starting point. This characteristic renders our algorithm computationally lightweight and highly adaptable to variations in terrain. The objective of our comparison is to highlight the areas where our proposed method can outperform map-based planning approaches.
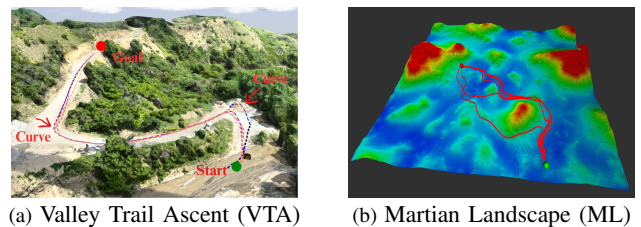
(a) Valley Trail Ascent (VTA) (b) Martian Landscape (ML)

Fig. 6: Simulation environments used to validate our proposed methodology. Some of the paths are shown in these figures.

For a more comprehensive demonstration, we tested our algorithm in two distinct scenarios illustrated in Fig. 6 and in Table II, we summarize the performance metrics from these simulation experiments. For the *VTA* environment, the goal $\mathbf{x}_f$ is positioned atop a hill with the starting point $\mathbf{x}_s$ nestled in a valley. The navigable route is illustrated in Fig. 6(a). The paths taken averaged an elevation change rate of $8.5cm/s$, attributed to the hill-valley configuration. The key challenge in this map revolves around bypassing two

Experiment	Method	D (m)	$\lvert\phi\rvert_{\max}$ (rad)	$\lvert\psi\rvert_{\max}$ (rad)	$\mathbf{v}_{b_{avg}}$ (m/s)	$\dot{z}_{r_{avg}}$ (cm/s)	C_{chg} (rad/m)
AD	Ours	63.456	**0.444**	**0.282**	**0.0617**	**5.18**	0.096
	VGF	**46.566**	0.489	0.355	0.1175	9.04	**0.0234**
AE	Ours	74.816	0.492	**0.354**	**0.0702**	**10.39**	0.0693
	VGF	**73.929**	**0.488**	0.475	0.1104	12.49	**0.0236**
BA	Ours	91.860	**0.506**	**0.242**	**0.0391**	**6.49**	0.0732
	VGF	**85.369**	0.508	0.526	0.1067	13.29	**0.0241**
AC	Ours	**71.901**	**0.505**	**0.278**	**0.057**	**8.18**	0.0797
	VGF	140.487	0.649	0.362	0.1065	9.43	**0.0246**
BD	Ours	60.4	**0.52**	**0.53**	**0.094**	**11.07**	0.110
	VGF	**60.3**	0.69	0.721	0.172	17.23	**0.029**

TABLE I: Comparison of metrics between our method and the baseline for selected experiments. Bold values indicate better performance.

TABLE II: Performance statistics in **VTA** and **ML** environments.

Metric	VTA	ML
$D[m]$	64.962 ± 5.207	84.470 ± 2.315
$\max\lvert\phi\rvert[rad]$	0.246 ± 0.075	0.414 ± 0.087
$\max\lvert\psi\rvert[rad]$	0.280 ± 0.065	0.345 ± 0.068
$\mathbf{v}_{b_{avg}}[rad/s]$	0.050 ± 0.007	0.071 ± 0.014
$\dot{z}_{r_{avg}}[cm/s]$	8.5 ± 1.6	4.0 ± 0.6
$C_{chg}[rad/m]$	0.108 ± 0.013	0.112 ± 0.032

pronounced curves. These curves can often lead to failures in mapless algorithms by causing the solution to oscillate between two local navigation goals, which may result in the robot colliding with an obstacle situated between the paths. The steepness and distance components of the cost function are key in addressing these challenges, with further metrics validating the safety of the selected paths.

Turning our attention to the *ML* scenario, the average traveled distance, denoted as D, is 84.47m. Notably, the maximum angles encountered were 0.41 rad and 0.35 rad. The average of elevation change $\dot{z}_{r_{avg}}$ was a modest 4 cm/s, signifying effective hill avoidance by the robot. This aligns with the fact that the safest path between the start and goal points in this terrain does not exhibit significant elevation variations. The curvature metric was registered at $0.112rad/m$, slightly elevated due to the numerous hills the robot circumvented, as depicted in Fig. 6(b). Additionally, each iteration of our algorithm exhibited a latency of approximately $0.11\,s$, encompassing the interval from receiving the point clouds to sending the command to the controller.

(a) Gentle Slope Bypass environment (GSB)
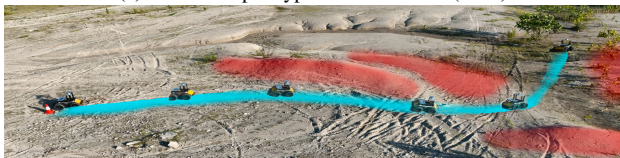
(b) Hilly Terrain Environment (HT)

Fig. 7: Real-world demonstration environments. The paths taken by the robot are highlighted in blue while the hills are highlighted in red. Video clips demonstrating the robot's performance can be found in the supplementary materials.

VII. REAL-WORLD DEMONSTRATIONS

We experimentally demonstrate that our proposed approach can navigate a real robot through uneven terrain safely.

A. Setup

For the experimental validation, we used the simulation setup previously outlined in Section VI-B, except for that the Velodyne is used for robot localization using a *GPS*, and the robot's maximum linear velocity was limited to v_{max} to $0.8m/s$. (We opted not to implement the baseline in the real-world context due to its requirement for an exhaustive scan and meticulous map building of the demonstration site, which is a strong assumption that is not practical in many applications and also a tedious task beyond our present focus.)

Two environments are considered for the real-world demonstration, namely, the *Gentle Slope Bypass* (GSB) environment and the *Hilly Terrain* (HT) environment, see Fig. 7. The *GSB* environment features a small grassy hill; for an optimal path to the goal, the robot should circumvent the hill until it is safe to climb, then the robot should navigate towards the goal. The *HT* environment presents more challenging terrains, where the robot needs to avoid a hilly area to reach its destination.

B. Results

The performance of the proposed approach during the real-world demonstration in both environments, **GSB** and **HT**. The performance statistics show that our approach successfully navigates the robot in the tested environments. In **GSB**, the maximum roll angle $\phi_{max} = 0.2rad$ exceeds the maximum pitch angle $\psi_{max} = 0.12rad$. This is attributed to the robot's trajectory; it traversed the lowest part of the slope diagonally until reaching a section where it was feasible to curve towards the goal, as seen in 7(a). The average change in elevation $\dot{z}_{r_{avg}}$ is $2cm/s$ and the total traveled distance is around $50m$. In **HT**, the terrain's incline aligns with the robot's direction, hence the pitch angle $\psi_{max} = 0.2rad$ predominantly exceeds the roll angle $\phi_{max} = 0.15rad$. This scenario also records an average elevation change of $\dot{z}_{r_{avg}}$ of $1.1cm/s$ and a traveled distance of $29m$. The observed low vibration rates signify the stability of our method. The high averages curvature changes, denoted as C_{chg}, can be attributed to our mapless approach where subgoal placements are determined by the real-time readings from the LiDAR, without the incorporation of a global planner.

VIII. CONCLUSION AND FUTURE WORK

This paper proposes a new mapless navigation approach for robots operating in uneven terrain. The approach utilizes an

efficient variant of Sparse Gaussian Process to learn the steepness and flatness of the robot's surrounding environment. The simulation and real-world experiments' results have shown that the proposed algorithm is able to navigate the robot in uneven terrain without the need for a global map. In forthcoming research endeavors, we intend to refine the methodology by incorporating obstacle consideration within navigable paths, enhancing the cost function's responsiveness to negative slopes such ditches, and integrating an optimization-based controller for improved navigation efficiency.

REFERENCES

[1] M. Ali and L. Liu, "Light-weight pointcloud representation with sparse gaussian process," pp. 4931–4937, 2023.

[2] M. Ali, H. Jardali, N. Roy, and L. Liu, "Autonomous Navigation, Mapping and Exploration with Gaussian Processes," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[3] M. Ali and L. Liu, "Gp-frontier for local mapless navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 10 047–10 053.

[4] I. S. Mohamed, M. Ali, and L. Liu, "Gp-guided mppi for efficient navigation in complex unknown cluttered environments," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 7463–7470.

[5] B. Zhou, J. Yi, X. Zhang, L. Chen, D. Yang, F. Han, and H. Zhang, "An autonomous navigation approach for unmanned vehicle in outdoor unstructured terrain with dynamic and negative obstacles," *Robotica*, vol. 40, no. 8, pp. 2831–2854, 2022.

[6] T. H. Y. Leung, D. Ignatyev, and A. Zolotas, "Hybrid terrain traversability analysis in off-road environments," in *2022 8th International Conference on Automation, Robotics and Applications (ICARA)*, 2022, pp. 50–56.

[7] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3019–3026, 2018.

[8] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[9] P. Kim, J. Chen, J. Kim, and Y. K. Cho, "Slam-driven intelligent autonomous mobile robot navigation for construction applications," in *Advanced Computing Strategies for Engineering: 25th EG-ICE International Workshop 2018, Lausanne, Switzerland, June 10-13, 2018, Proceedings, Part I 25*. Springer, 2018, pp. 254–269.

[10] Z. Jian, Z. Lu, X. Zhou, B. Lan, A. Xiao, X. Wang, and B. Liang, "Putn: A plane-fitting based uneven terrain navigation framework," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 7160–7166.

[11] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krüsi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1184–1189.

[12] A. Chilian and H. Hirschmüller, "Stereo camera based navigation of mobile robots on rough terrain," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 4571–4576.

[13] J. Wang, L. Xu, H. Fu, Z. Meng, C. Xu, Y. Cao, X. Lyu, and F. Gao, "Towards efficient trajectory generation for ground robots beyond 2d environment," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 7858–7864.

[14] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[15] A. Ammar, H. Bennaceur, I. Châari, A. Koubâa, and M. Alajlan, "Relaxed dijkstra and a* with linear complexity for robot path planning problems in large-scale grid environments," *Soft Computing*, vol. 20, pp. 4149–4171, 2016.

[16] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[17] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[18] J. Borenstein, Y. Koren, *et al.*, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.

[19] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[20] Y. Peng, W. Guo, M. Liu, J. Cui, and S. Xie, "Obstacle avoidance planning based on artificial potential field optimized by point of tangency in three-dimensional space," *Journal of System Simulation*, vol. 26, no. 8, pp. 1758–1762, 2014.

[21] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press, 2005.

[22] N. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse Gaussian process methods: The informative vector machine," in *Proceedings of the 16th annual conference on neural information processing systems*, 2003, pp. 609–616.

[23] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," *Advances in neural information processing systems*, vol. 18, p. 1257, 2006.

[24] M. K. Titsias, "Variational model selection for sparse gaussian process regression," *Report, University of Manchester, UK*, 2009.

[25] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Artificial intelligence and statistics*. PMLR, 2009, pp. 567–574.

[26] K. Yan and B. Ma, "Mapless navigation based on 2d lidar in complex unknown environments," *Sensors*, vol. 20, no. 20, p. 5802, 2020.

[27] C. Ye and P. Webb, "A sub goal seeking approach for reactive navigation in complex unknown environments," *Robotics and Autonomous Systems*, vol. 57, no. 9, pp. 877–888, 2009.

[28] A. G. d. G. Matthews, M. Van Der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman, "Gpflow: A gaussian process library using tensorflow." *J. Mach. Learn. Res.*, vol. 18, no. 40, pp. 1–6, 2017.

[29] J. Liang, K. Weerakoon, T. Guan, N. Karapetyan, and D. Manocha, "Adaptiveon: Adaptive outdoor navigation method for stable and reliable motions," *arXiv preprint arXiv:2205.03517*, 2022.

[30] M. Mujahed and B. Mertsching, "The admissible gap (ag) method for reactive collision avoidance," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1916–1921.