2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Gaussian Process-based Traversability Analysis for Terrain Mapless Navigation

Abe Leininger, Mahmoud Ali, Hassan Jardali, and Lantao Liu

***Abstract*— Efficient navigation through uneven terrain remains a challenging endeavor for autonomous robots. We propose a new geometric-based uneven terrain mapless navigation framework combining a Sparse Gaussian Process (SGP) local map with a Rapidly-Exploring Random Tree* (RRT*) planner. Our approach begins with the generation of a high-resolution SGP local map, providing an interpolated representation of the robot's immediate environment. This map captures crucial environmental variations, including height, uncertainties, and slope characteristics. Subsequently, we construct a traversability map based on the SGP representation to guide our planning process. The RRT* planner efficiently generates real-time navigation paths, avoiding untraversable terrain in pursuit of the goal. This combination of SGP-based terrain interpretation and RRT* planning enables ground robots to safely navigate environments with varying elevations and steep obstacles. We evaluate the performance of our proposed approach through robust simulation testing, highlighting its effectiveness in achieving safe and efficient navigation compared to existing methods. See the project GitHub[1] for source code and supplementary materials, including a video demonstrating experimental results.**

***Index Terms*— Off-road navigation, Traversability-analysis, Gaussian process (GP).**

I. Introduction

In recent years, the utilization of ground autonomous robotics in uneven terrain has significantly expanded, paving the way for a variety of applications in exploration, surveillance, and environmental monitoring. Within this context, the navigation task can be defined as the mission to steer a robot from its current pose to a predetermined goal, optimizing for safety, efficiency, and speed. Navigation approaches are typically categorized into either map-based or mapless methods. Map-based strategies require generating an extensive global map of the environment for the robot to plan and navigate through, offering high precision at the expense of substantial computational demands and potential scalability limitations. Conversely, mapless approaches eliminate the need for a global map, instead relying on the robot's immediate sensory inputs for on-the-fly navigation, which sacrifices global optimality in favor of real-time computational efficiency.

In this research, our primary contribution is the construction of a mapless navigation framework composed of two main components. The first one involves utilizing Sparse Gaussian Process (SGP) to create a local map that accurately extracts the geometric characteristics from the robot's immediate surroundings. This process is central to forming a traversability map, which provides the robot with a detailed insight into the navigable parts of the local terrain. The second component provides a method for footprint assessment and sub-goal selection built on the RRT* local planner for safe and efficient navigation. The RRT* algorithm excels at identifying the most efficient traversable path to local sub-goals or "frontier" nodes, thereby facilitating a gradual progression of the robot toward its global goal. To validate the efficacy of our approach, we carried out extensive simulation tests on various challenging terrain conditions. These tests confirmed the efficiency of our framework. We also compared our method against a state-of-the-art map-based navigation approach, where we demonstrate a competitive advantage in navigating complex environments.

Abe Leininger is a senior undergraduate student in the Department of Computer Science at Indiana University, Bloomington, IN 47408 USA. aleinin@iu.edu. Other authors are with the Department of Intelligent Systems Engineering at Indiana University, Bloomington. {alimaa, hjardali,lantao}@iu.edu.

[1]https://github.com/abeleinin/gp-navigation

II. Related Work

Frameworks guiding autonomous systems in navigating uneven terrain generally fall into two categories: geometric-based and learning-based approaches. The former leverages real-time sensor data, primarily from LiDAR and RGBD cameras, while the latter relies on deep or reinforcement learning models [1], [2]. In this work, we are mostly interested in geometric approaches. These approaches usually construct a map, e.g., an occupancy grid map – a matrix where each cell represents a spatial area and contains the probability of that area being occupied [3]. For 3D navigation frameworks, 3D Octomaps [4], where the 3D space is subdivided into smaller octants, or the grid-based 2.5D elevation maps, where each grid holds the elevation value [5], are used. Both methods rely on probabilistic approaches.

Once the map is constructed, a traversability analysis is performed. This analysis can yield a traversability map featuring binary values, indicating whether a grid is traversable or not, or it can employ a cost function to assign a score to each grid [6]. The traversability score is often calculated by extracting geometric features like slope, roughness, and step height. The slope is determined as the inclination angle originating from a plane fitted to the surrounding terrain, while roughness is measured as the standard deviation of the terrain height values from this plane [7]. Additionally, metrics including flatness, sparsity, and unevenness are sometimes considered in the analysis [7]–[12].

Following the creation of the traversability map, a global planner finds a path from the robot's current position to the global goal using either graph-based methods such as Dijkstra [13] and A* algorithms [14], or probabilistic strategies like RRT [15] and RRT* [16] for trajectory planning

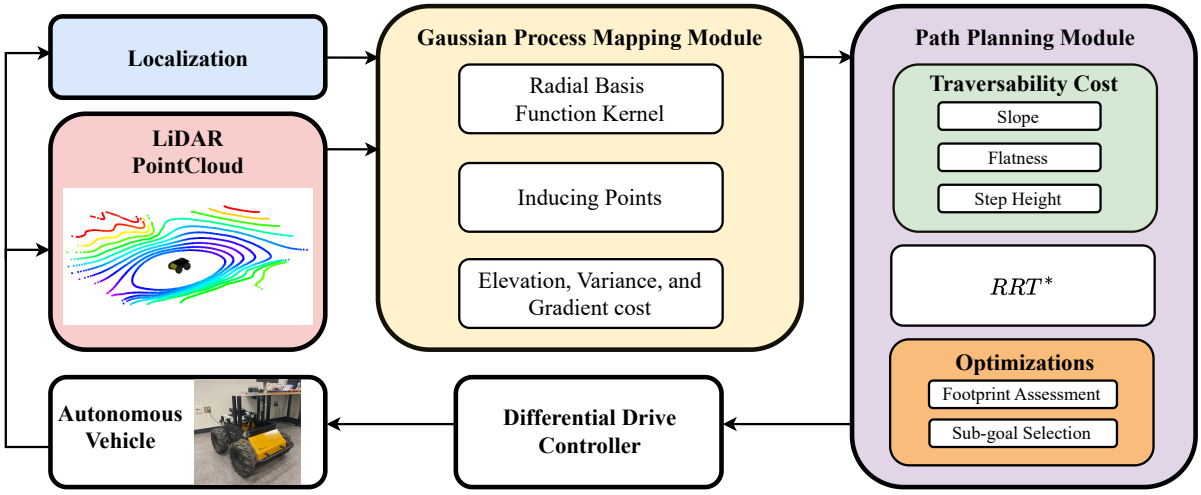


Fig. 1: Overview of the proposed uneven terrain navigation framework. From left to right: Localization and LiDAR PointCloud data informs the Gaussian Process Mapping Module. An RBF Kernel trains a model on the induced PointCloud points within this module, yielding Elevation, Variance, and Gradient local cost maps. These maps feed into the Path Planning Module, which constructs a traversability local cost map based on terrain attributes. RRT* planning occurs within the local traversability map and plans a path to the goal. During planning optimizations, including a footprint-based assessment approach and local sub-goal selection. Finally, we send waypoints to the Differential Drive Controller, which sends control commands to the autonomous vehicle.

[6]. The standard RRT [15] builds a space-filling tree that swiftly explores the state space by randomly sampling states and connecting them to the nearest vertices in the tree. The RRT* variant enhances RRT by ensuring optimality, constantly rewiring the tree to find more optimal paths. After the path is constructed, the waypoints are sent to a controller. Alternatively, there are mapless strategies such as DWA [17], VFH [18], and APF [19], originally designed for 2D navigation, have evolved with advancements in robotics, enabling their use in 3D environments [20].

Gaussian Process (GP) is a well-established framework to model continuous spatial phenomena [21]–[24]. However, despite its proven efficiency, the standard GP comes with a significant drawback which is its high time complexity, $\mathcal{O}(n^3)$. This limitation reduces its applicability in real-time scenarios and with modeling massive datasets. Addressing this issue, various approaches have been explored to lessen the computational load of GPs. One strategy is the Sparse GP (SGP), which reduces the computational complexity of the standard GP by selecting the most relevant subset of the data, *m inducing points*, to describe the entire training data using Bayesian rules [25]–[27]. The computation complexity of SGP, $\mathcal{O}(nm^2)$, is typically much smaller than the full GP's computation complexity $\mathcal{O}(n^3)$. In our previous work, we proved that an SGP-based local perception framework is able to navigate the robot in complex cluttered environment [28]–[30]. Following the paradigm of mapless methods, our approach integrates a Sparse Gaussian Process (SGP) model to build a traversability local map, coupled with an RRT* planner to generate local navigational paths that will lead the robot toward the final goal.

III. Methodology

We propose a new framework for terrain navigation that combines a GP-based traversability map and an RRT* planner. A systematic overview of the proposed framework is shown in Fig. 1. The framework converts point cloud data into a GP elevation map, which models the terrain geometry around the robot. The prediction of the GP model is then used to analyze the terrain traversability based on three components: local slope, local step height, and neighborhood flatness. The RRT* planner finds a traversable path within the local map while considering the robot's footprint. Finally, we provide an approach for selecting sub-goals while navigating to the global goal. Subsequent sections provide detailed explanations of each module.

A. GP Local Elevation Map

In order to construct the traversability map, our initial step involves the creation of a GP-based elevation map, which is built upon the LiDAR observations. Each point in the LiDAR point cloud is represented by a tuple (x,y,z), denoting its spatial coordinates in the 3D space relative to the robot. To ensure that these data points align horizontally to the world, we perform a transformation using the current roll and pitch angles of the robot to align the point's orientation to the global world frame orientation. This transformation allows us to maintain a consistent geometry analysis with respect to the world frame. For elevation mapping, each point is decomposed into the point location in the 2D plane, $\mathbf{p}_i = (x_i, y_i)$, and its associated elevation z_i. Consequently, a dataset denoted as $\mathcal{D} = \{(\mathbf{p}_i, z_i)\}_{i=1}^{n_p}$, is formulated comprising the n_p observed points. The proposed elevation local map is a 2D SGP regression model, trained on $\mathcal{D}$, capable of predicting the elevation value for any point in the continuous space. The SGP elevation map can be defined as follows:

$$f(\mathbf{p}) \sim \mathrm{SGP}\left(m(\mathbf{p}), k_{\mathrm{se}}(\mathbf{p},\mathbf{p}')\right),$$
$$k_{\mathrm{se}}(\mathbf{p},\mathbf{p}') = \sigma_{\mathrm{se}}^2 \exp\left(-\frac{\|\mathbf{p}-\mathbf{p}'\|^2}{2\ell_{\mathrm{se}}^2}\right), \quad (1)$$

where $m(\mathbf{p})$ is the mean function and $k_{\text{se}}(\mathbf{p},\mathbf{p}')$ is the Squared Exponential (SE) kernel, also known as the Radial Basis Function (RBF), with a length-scale ℓ_{se} and a signal variance σ_{se}^2. In GP regression, a noise $\varepsilon_i \sim \mathcal{N}(0,\sigma_{n_p}^2)$ is added to the GP prediction to reflect the measurement noise. The elevation z^* for any query point $\mathbf{p}^*$ on the XY plane is estimated by the SGP elevation model as follows:

$$\begin{aligned} p(z^*|\boldsymbol{z}) &= \mathcal{N}(z^*|m_{\boldsymbol{z}}(\boldsymbol{p}^*), k_{\boldsymbol{z}}(\boldsymbol{p}^*,\boldsymbol{p}^*)+\sigma_{n_p}^2), \\ m_{\boldsymbol{z}}(\boldsymbol{p}) &= K_{\boldsymbol{p}n_p}\left(\sigma_{n_p}^2 I + K_{n_p n_p}\right)^{-1}\boldsymbol{z}, \\ k_{\boldsymbol{z}}(\boldsymbol{p},\boldsymbol{p}') &= k(\boldsymbol{p},\boldsymbol{p}') - K_{\boldsymbol{p}n_p}\left(\sigma^2 I + K_{n_p n_p}\right)^{-1} K_{n_p \boldsymbol{p}'}, \end{aligned} \tag{2}$$

where $\boldsymbol{z} = \{z_i\}_{i=1}^{n_p}$, $m_{\boldsymbol{z}}(\boldsymbol{p})$, and $k_{\boldsymbol{z}}(\boldsymbol{p},\boldsymbol{p}')$ are the posterior mean and covariance functions [31], $K_{n_p n_p}$ is $n_p \times n_p$ covariance matrix of the inputs, K_{pn_p} is an n-dimensional row vector of kernel function values between $\boldsymbol{p}$ and the inputs, with $K_{n_p p} = K_{pn_p}^T$. The GP prediction is in the form of a probability distribution with a mean value μ_{p_i}, which indicates the elevation at the query point $\mathbf{p}_i$, and a variance $\sigma_{p_i}^2$, which indicates the uncertainty of this prediction.

To facilitate the comprehensive assessment of the terrain's elevation and slope around the robot, the area covered by the local observation (LiDAR range) is discretized to form a grid, referred to as the local map $\mathcal{M}$. This map is defined by its width w_m, height h_m, and resolution γ_m. Subsequently, the SGP elevation model operates as the analytical tool to assess the terrain's features, i.e. *elevation* and *slope*. Each attribute reflects varying facets of the terrain's geometry, thereby providing the build blocks of the local traversability cost map around the robot $\mathcal{M}_\tau$. Additionally, the SGP model provides a confidence level for the predicted elevation represented by the GP uncertainty $\mathcal{M}_\sigma$. We utilize this local map to avoid navigation through uncertain regions.

B. Traversability Map

To find a safe trajectory to the goal, we build a local traversability map $\mathcal{M}_\tau$. The local traversability cost map characterizes the navigable terrain in the robot's local environment. The relative danger is analyzed by assessing terrain characteristics like slope, flatness, and elevation change. This allows for safe navigation, which adheres to the vehicle's safety limits for roll ϕ and pitch ψ angles. We propose a geometric approach for terrain analysis by utilizing elevation $\mathcal{M}_h$ and uncertainty $\mathcal{M}_\sigma$ local maps provided by the GP-Module; explained in III-A. Also, using the GP-Module, the slope $\mathcal{M}_\Delta$ map is generated as the gradient of the SGP model. Using these provided local maps; we calculate additional terrain feature maps, including flatness $\mathcal{M}_f$ and step height $\mathcal{M}_\zeta$. Visual depictions of the different local map components are shown in Fig. 2. Together, the slope, flatness, and step height local maps are employed to build a *local traversability map* $\mathcal{M}_\tau$, where traversability is set as a value $[0,1]$, calculated as follows:

$$\mathcal{M}_\tau = \omega_1 \frac{\mathcal{M}_\Delta}{s_{\text{crit}}} + \omega_2 \frac{\mathcal{M}_f}{f_{\text{crit}}} + \omega_3 \frac{\mathcal{M}_\zeta}{\zeta_{\text{crit}}}, \tag{3}$$

where ω_1, ω_2, and ω_3 are weights totaling 1, s_{crit}, f_{crit}, and ζ_{crit} denote robot-specific critical thresholds for maximum slope, flatness, and step height tolerable before reaching unsafe conditions. These thresholds may be sourced from the robot manufacturer's manual or estimated in advance. Additionally, the weighting parameters ω_i can be adjusted based on the robot type; for example, a tracked-wheeled robot may perform better on slopes than one with smooth wheels.

Formally, the slope s is calculated as the magnitude of the *GP-gradient* in the local map $\mathcal{M}$:

$$s \triangleq \nabla k_{\text{se}}(\mathbf{p}^*,\mathbf{p}_i) = k_{\text{se}}(\mathbf{p}^*,\mathbf{p_i})\frac{\mathbf{p}_i - \mathbf{p}^*}{\ell_{\text{se}}^2}. \tag{4}$$

The slope matrix $\mathcal{M}_\Delta$ is then normalized such that $0 < \mathcal{M}_\Delta[i,j] < 1$. Fig. 2c visualizes the slope as the color intensity on the elevation map. Terrain flatness $\mathcal{M}_f$ is analyzed by finding the normal vector of the best-fitting plane to a region of points within our local elevation map $\mathcal{M}_h$ [10]. The region the plane is fit to is equal to the size of the robot footprint m_δ. We solve the plane equation using the least squares method. Step height $\mathcal{M}_\zeta$ is calculated similarly to [7] by finding the max height within a window relative to the current robot height:

$$\mathcal{M}_\zeta[i,j] = max(m_\delta \in \mathcal{M}_h), \tag{5}$$

Finally, the uncertainty map $\mathcal{M}_\sigma$ acts as a *mask* over the traversability map $\mathcal{M}_\tau$, filtering out points with uncertainty above a critical threshold σ_{crit} by marking their traversability as $\tau = 1$, indicating non-traversable or unsafe areas. This process is formalized as follows:

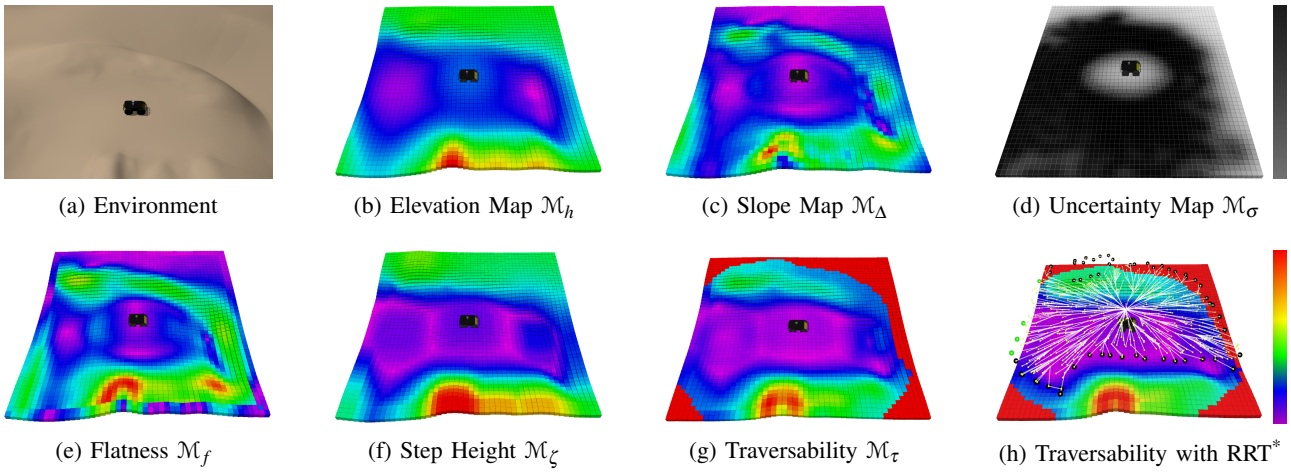
$$\mathcal{M}_\tau[i,j] = \begin{cases} 1 & \text{if } \mathcal{M}_\sigma[i,j] > \sigma_{\text{crit}}, \\ \mathcal{M}_\tau[i,j] & \text{otherwise.} \end{cases} \tag{6}$$

The mask is visually represented as the red boundary region in the traversability map $\mathcal{M}_\tau$, as illustrated in Fig. 2g.

C. Path Planning

Efficient and agile planning modules are essential for generating safe trajectories in uneven environments in real-time. To address this, we utilize the RRT* algorithm [16], an optimal sampling-based method for motion planning. This method extends the foundational RRT algorithm [15] to include a rewiring step, enhancing its capacity to find more efficient paths by reducing the cost in terms of path length, while also guaranteeing asymptotic optimality.

During RRT* planning a sample point is generated within the bounds of the traversability map $\mathcal{M}_\tau$, which is specified as $p_{\text{new}}(x_n, y_n)$. To position the point in 3D space the elevation map $\mathcal{M}_h$ provides an elevation value for the point given by $z_n = \mathcal{M}_h[x_n, y_n]$. We therefore construct a new node $n_{\text{new}} = (x_n, y_n, z_n) \in \mathbb{R}^3$, effectively projecting p_{new} into the 3D space of the local environment. At this stage, the algorithm must assess whether n_{new} qualifies as a traversable leaf node. Our evaluation leverages a *footprint-based* assessment approach [8], by considering the robot footprint $m_\delta \in \mathcal{M}_\tau$. When RRT* creates a segment connecting n_{new} and n_ℓ, where n_ℓ is the initial branching leaf node. This segment is then resampled



(a) Environment (b) Elevation Map $\mathcal{M}_h$ (c) Slope Map $\mathcal{M}_\Delta$ (d) Uncertainty Map $\mathcal{M}_\sigma$

(e) Flatness $\mathcal{M}_f$ (f) Step Height $\mathcal{M}_\zeta$ (g) Traversability $\mathcal{M}_\tau$ (h) Traversability with RRT*

Fig. 2: Traversability analysis for one observation (marked by a black square in Fig. 3d) in environment B: (a) shows the simulated environment. (b), (c), and (d) show the local elevation $\mathcal{M}_h$, slope $\mathcal{M}_\Delta$, and uncertainty maps $\mathcal{M}_\sigma$ respectively, generated by the SGP elevation model. (e) and (f) visualize the flatness f and step height ζ maps. (g) shows the final local traversability map $\mathcal{M}_\tau$ with the uncertainty map $\mathcal{M}_\sigma$ applied as a mask. (h) shows the generated RRT* planning on top of the final traversability map. The traversability values are presented on a rainbow spectrum, with purple representing traversable and red representing non-traversable. Similarly, the Uncertainty map is represented on a grey color gradient where white represents uncertain and black indicating certain regions.

to produce equidistant points, each serving as the center of an analysis area defined by m_δ. If m_δ, when centered around n_{new}, lies outside the confines of the local map $\mathcal{M}$, n_{new} will be added to $V_\mathcal{F}$, a vector holding frontier nodes. A frontier is defined as an available sub-goal within the current local map. We analyze the local traversability values inside each m_δ by considering the number of non-traversable cells n_{coll}. We count the number of non-traversable cells within each m_δ that is greater than τ_{crit}.

$$n_{\text{coll}} = \sum_{i=1}^{|m_\delta|}\sum_{j=1}^{|m_\delta|} \mathbf{1}(m_\delta[i,j] > \tau_{\text{crit}}). \tag{7}$$

Notably, τ_{crit} denotes how aggressive the planning will be within $\mathcal{M}_\tau$. If the collision count, n_{coll}, is zero in the evaluated m_δ, it is deemed traversable. Consequently, the corresponding node is added to the vertex set of all leaf nodes, V_ℓ. However, if n_{coll} exceeds a predefined threshold coll_{max}, the new node n_{new} is identified as an edge node and appended to $V_\mathcal{E}$, a vector holding the edge nodes within the local map. These nodes will be removed from the V_ℓ vector, halting the tree's growth when branching into a non-traversable area. coll_{max}, is established based on the number of non-traversable cells in m_δ. See Fig. 4 to view the visual representation of edge and frontier nodes. One benefit of introducing *edge vertices* is that they facilitate avoidance of local minima, preventing the robot from getting stuck.

D. Sub-goal Selection

If the global goal is not attained within the current iteration of the planning algorithm, the branching process persists until a viable sub-goal is identified, to which the robot can navigate and subsequently initiate a re-planning process. It's worth noting the difference between an edge node and a frontier node: an edge node refers to a leaf node leading to a non-traversable area, whereas a frontier node is located at the boundary of the local map, regarded as a potential sub-goal candidate. We employ a weighted-sum approach to determine the optimal sub-goal, denoted as $\mathbf{g}^*$, by evaluating two distance metrics at each frontier node $n_i \in V_\mathcal{F}$. The process is defined mathematically as:

$$\begin{aligned} D(n_i) &= \text{norm}\left(\sqrt{(G_x - n_{i_x})^2 + (G_y - n_{i_y})^2}\right), \\ E(n_i) &= \text{norm}\left(r - \sqrt{(n_{i_x})^2 + (n_{i_y})^2}\right), \\ \mathbf{g}^* &= \arg\min_{n_i \in V_\mathcal{F}} \left(\alpha_1 D(n_i) + \alpha_2 E(n_i)\right). \end{aligned} \tag{8}$$

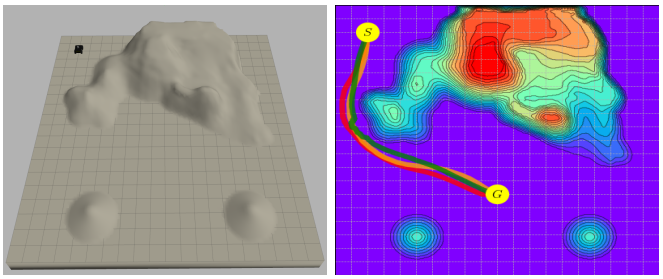
Here, we introduce two normalized distance functions, $D(n_i)$ and $E(n_i)$, that are utilized in a weighted sum to find the minimum and thus select the nearest sub-goal. The function $D(n_i)$ represents the Euclidean distance from the current node n_i to the final goal with coordinates (G_x, G_y) in the robot's frame. Conversely, $E(n_i)$ measures the distance from the frontier node to the boundary of the local map. For simplicity, we consider the local map as a circle around the robot with a radius r equal to half the map width. The weights α_1 and α_2 serve to balance the influences of the two distance metrics.

During path navigation, each node is sequentially sent to the differential-drive controller [32] as a waypoint. As the robot approaches each waypoint, it proceeds to the next one upon reaching a specified proximity to the current waypoint. This approach ensures that the robot maintains a consistent speed, as it does not need to slow down or stop during navigation between waypoints. Similarly, as the robot nears proximity to the local sub-goal, a re-planning phase starts, enabling continuous navigation towards the global goal.

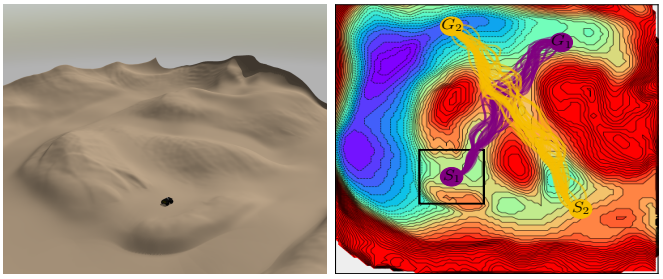
IV. Experimental Design and Results

A. Simulation Setup

The proposed GP local map built is on top of the GPyTorch [33], [34] library. The RRT* algorithm is implemented in Python. Real-time simulation experiments were conducted



(a) Gazebo Environment **A** (b) Topographic Map **A**



(c) Gazebo Environment **B** (d) Topographic Map **B**

Fig. 3: Figure (a) is the Gazebo world for Environment A, provided by [10], and (b) is the associated topographic map with the safest path achieved by each tested algorithm for task T_1. Figure (c) is the Gazebo world for Environment B, and beside it, (d) depicts the topographic map with overlaid paths for all trials run in Tasks 3 and 4. The black box represents the local map $\mathcal{M}$ view, which is visualized in Fig. 2.

in the Gazebo simulator to evaluate the performance of our approach and to compare it to a recent state-of-the-art algorithm for terrain navigation. We selected Plane-fitted Uneven Terrain Navigation (PUTN) Framework [10] as a baseline comparison algorithm. This baseline has already proven its effectiveness and superiority over other methods, and it is open source. During testing, we used two localization modules Advanced-LOAM (ALOAM) [35], and precise ground truth provided by a Gazebo plug-in. ALOAM was selected for consistency with the PUTN localization module.

We assessed the local navigation performance using two environments, A and B. Environment A, sourced from the PUTN GitHub repository [10], features a smooth mountain landscape mainly for testing non-traversable obstacle avoidance seen in Fig. 3a. In contrast, Environment B presents rugged, steep, and slightly undulating terrain to evaluate the framework under more dynamic conditions shown in Fig. 3c.

Our evaluation relied on four metrics: average velocity (v_{avg}), maximum average robot roll (ϕ), maximum average robot pitch (ψ), and path length (ℓ), see Table I. Our goal in the simulation testing is to validate the algorithm's capability to safely navigate uneven terrain abiding by the Clearpath Robotics Husky's [36] angular limits of 0.785 rad for pitch and 0.524 rad for roll. We selected four path objectives, or Tasks across environments A and B, to evaluate various facets of our algorithm's performance. In Task 1, denoted as T_1 and simulated in environment A as depicted in Fig. 3a. We conducted multiple trials using different localization modules for our proposed method and the baseline. In

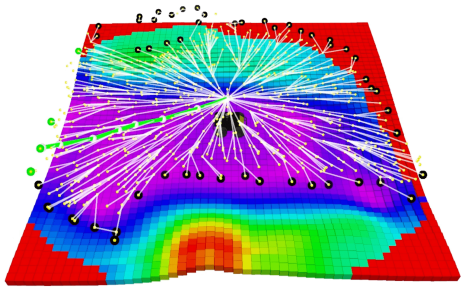


Fig. 4: Expanded Fig. 2h illustrates an RRT* planning iteration, where the green trajectory highlights the route to the chosen subgoal. Black nodes signify edge nodes leading the untraversable areas. While green nodes indicate frontier nodes and potential targets for subsequent planning iterations.

environment B, we conducted three distinct tasks, labeled T_2, T_3, and T_4. The initial and final coordinates for each test are outlined in Table I. It is important to note that the baseline algorithm could not successfully navigate any of the paths in environment B, failing in all tasks even with parameter tuning. For details on parameter configuration for our method, please refer to our GitHub. The tasks tested were designed to assess our algorithm's efficacy and safety across diverse and challenging navigation scenarios.

B. Simulation Results

An overview of our simulation statistics is displayed in Table I. In Environment A, when using ALOAM for localization, both our proposed and the baseline (PUTN) algorithms have approximately equivalent behavior when considering speed and safety. In Task 1, the average velocity PUTN performed was 0.4 ± 0.01m/s, and ours was 0.45 ± 0.02m/s, see Fig. 5. The roll and pitch values are also consistent between both algorithms, with PUTN having a slightly higher average max roll of 0.146 ± 0.33rad compared to GP-RRT of 0.002 ± 0.002rad. The pitch angles are insignificant as the terrain of this map predominantly consists of flat ground with minor variations in elevation. However, an advantage of our algorithm was finding the shortest path within this scenario, with PUTN averaging a path length of 20.2 ± 0.8m and our algorithm having a path length of 19.6 ± 0.6m. During testing, we observed that, despite PUTN being a map-based algorithm, it engaged in planning routes into local minima situated beside the non-traversable hill in half of the trials. Our map-less method successfully avoided this during local planning, because we categorized leaf nodes as edges into non-traversable terrain, separating them from possible planning objectives. When using ground truth localization in this environment, our method improved its performance in speed and finding the optimal path when compared to our method using ALOAM. The average speed increased to 0.71 ± 0.35m/s, and the path length also reduced to 18.0 ± 0.5m.

Our algorithm exhibited successful consistent performance across the three navigation tasks in the more challenging Environment B, however, the baseline failed. Since the baseline operates on a map-based method, it necessitates a

Env	Task	Start	Goal	Method	Loc.	Evaluation Criteria			
						v_{avg}[m/s]	ℓ [m]	ϕ [rad]	ψ [rad]
A	T_1	(-8, 8)	(0, -4)	PUTN	AL	0.4 ± 0.01	20.2 ± 0.8	0.146 ± 0.33	$\mathbf{0.004 \pm 0.002}$
				Ours	AL	0.45 ± 0.02	19.6 ± 0.6	$\mathbf{0.002 \pm 0.002}$	0.008 ± 0.17
				Ours	GT	$\mathbf{0.71 \pm 0.35}$	$\mathbf{18.0 \pm 0.5}$	0.009 ± 0.008	0.006 ± 0.004
B	T_2	(-7, -7)	(3, 11)	PUTN [Fail]	AL	0.3 ± 0.1	26.5 ± 8.8	0.287 ± 0.1	0.41 ± 0.11
				Ours	AL	$\mathbf{0.35 \pm 0.04}$	$\mathbf{38.0 \pm 6.6}$	$\mathbf{0.37 \pm 0.16}$	$\mathbf{0.52 \pm 0.3}$
	T_3	(-7, -7)	(10, 18)	Ours	GT	0.6 ± 0.02	33.6 ± 1.3	0.32 ± 0.085	0.35 ± 0.085
	T_4	(-13, 13)	(21, -7)	Ours	GT	0.51 ± 0.02	42.4 ± 1.4	0.34 ± 0.07	0.34 ± 0.05

TABLE I: The table presents simulation evaluation statistics gathered through a series of tasks. The best statistics are in bold. The column labeled Loc. stands for Localization, and the acronyms used are ALOAM (AL) and Ground Truth (GT). More extensive testing was conducted in the more difficult Environment B with 3 Tasks: T_2 was a localization task where the baseline failed to reach the goal and our method succeeded. T_3 and T_4 are robustness testing of our framework, visualized in Fig. 3d.

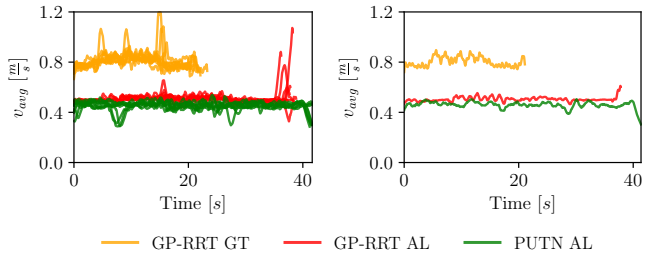


Fig. 5: Environment A velocity graphs. The left graph displays the velocities for 10 trials for each method tested in T_1. AL and GT stand for ALOAM and Ground Truth respectively. The right graph displays a single velocity for the safest path taken displayed in Fig. 3b.

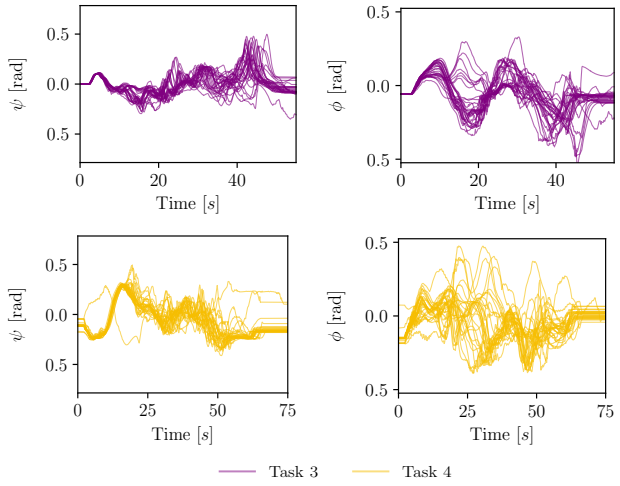


Fig. 6: The graphs in this figure represent the roll and pitch data for **our proposed method** during the 30 trials for each task in Environment B. The top two graphs showcase the pitch and roll of Task 3, while the ones below depict the data from Task 4.

well-constructed map to facilitate fitting a plane along the generated path. However, the complexity of the environment presented significant challenges, hindering the baseline from establishing a suitable map and ultimately leading to its failure. The results for using ALOAM in that environment are shown in T_2 in Table. I. The results demonstrate that our proposed mapless method successfully completed the task, even when utilizing a localization method characterized by a significant amount of noise. The performance metrics gathered from the results are detailed as follows: the average velocity (v_{avg}) was recorded as 0.35 ± 0.04 m/s, the total path length (ℓ) was 38.0 ± 6.6 m, the maximum average robot roll (ϕ) was 0.37 ± 0.16 rad, and the maximum average robot pitch (ψ) was 0.52 ± 0.3 rad. As seen in Fig. 6, the roll and pitch values are within the safety specifications for all test cases, confirming our method's robustness to plan trajectories to the objective in an angularly safe manner. A visualization of all trajectories taken during both tests is displayed in Fig. 3d. In the tasks where the ground truth (GT) was used, the average velocity was 0.6 ± 0.02m/s on Task 3, and 0.51 ± 0.02m/s on Task 4, which shows consistency in terms of speed variation between each of the trials. The average max roll and pitch for Task 3 were 0.32 ± 0.085rad and 0.35 ± 0.085rad, respectively. In Task 4, we observe similar values 0.34 ± 0.07rad and 0.34 ± 0.05rad.

The results from the evaluation in both environments offer valuable insights into the local navigation performance of our framework, indicating its strengths in effectively finding and traversing safe paths in an efficient manner. For additional information, please refer to the project GitHub for video demonstration and source code.

V. Conclusion

In this paper, we present a new mapless navigation framework for navigating through rugged terrain. Our proposed approach utilizes a Sparse Gaussian Process (SGP) local map with an optimal Rapidly-Exploring Random Tree (RRT*) planner. The GP-based local map serves as the foundation for constructing a traversability map, guiding our navigation planning process. Our findings, based on extensive simulation tests, reveal the great potential of our framework for enhancing both the safety and efficiency of mapless navigation. Our future work will focus on upgrading the controller to optimize its performance and precision. Additionally, we want to explore the integration of vehicle dynamics into the traversability analysis. Furthermore, we aim to leverage this framework by using learning components such as segmentation. These ongoing efforts aim to further advance the capabilities of our navigation framework and contribute to the field of mapless autonomous navigation.

REFERENCES

[1] Kasun Weerakoon, Adarsh Jagan Sathyamoorthy, Utsav Patel, and Dinesh Manocha. Terp: Reliable planning in uneven outdoor environments using deep reinforcement learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9447–9453. IEEE, 2022.

[2] Jing Liang, Kasun Weerakoon, Tianrui Guan, Nare Karapetyan, and Dinesh Manocha. Adaptiveon: Adaptive outdoor local navigation method for stable and reliable actions. *IEEE Robotics and Automation Letters*, 8(2):648–655, 2022.

[3] Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15:111–127, 2003.

[4] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34:189–206, 2013.

[5] Péter Fankhauser, Michael Bloesch, and Marco Hutter. Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robotics and Automation Letters*, 3(4):3019–3026, 2018.

[6] Stepan Dergachev, Kirill Muravyev, and Konstantin Yakovlev. 2.5 d mapping, pathfinding and path following for navigation of a differential drive robot in uneven terrain. *IFAC-PapersOnLine*, 55(38):80–85, 2022.

[7] Annett Chilian and Heiko Hirschmüller. Stereo camera based navigation of mobile robots on rough terrain. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4571–4576, 2009.

[8] Martin Wermelinger, Péter Fankhauser, Remo Diethelm, Philipp Krüsi, Roland Siegwart, and Marco Hutter. Navigation planning for legged robots in challenging terrain. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1184–1189, 2016.

[9] Tiga Ho Yin Leung, Dmitry Ignatyev, and Argyrios Zolotas. Hybrid terrain traversability analysis in off-road environments. In *2022 8th International Conference on Automation, Robotics and Applications (ICARA)*, pages 50–56, 2022.

[10] Zhuozhu Jian, Zihong Lu, Xiao Zhou, Bin Lan, Anxing Xiao, Xueqian Wang, and Bin Liang. Putn: A plane-fitting based uneven terrain navigation framework, 2022.

[11] Jingping Wang, Long Xu, Haoran Fu, Zehui Meng, Chao Xu, Yanjun Cao, Ximin Lyu, and Fei Gao. Towards efficient trajectory generation for ground robots beyond 2d environment. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7858–7864, 2023.

[12] Bo Zhang, Guobin Li, Qixin Zheng, Xiaoshan Bai, Yu Ding, and Awais Khan. Path planning for wheeled mobile robot in partially known uneven terrain. *Sensors*, 22(14), 2022.

[13] Edsger W Dijkstra. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, pages 287–290. 2022.

[14] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[15] Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

[16] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[17] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, 1997.

[18] Johann Borenstein, Yoram Koren, et al. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, 7(3):278–288, 1991.

[19] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[20] Y Peng, WQ Guo, M Liu, JX Cui, and SR Xie. Obstacle avoidance planning based on artificial potential field optimized by point of tangency in three-dimensional space. *Journal of System Simulation*, 26(8):1758–1762, 2014.

[21] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2005.

[22] A. Singh, A. Krause, C. Guestrin, W. Kaiser, and M. Batalin. Efficient planning of informative paths for multiple robots. In *the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2204–2211, 2007.

[23] Ruofei Ouyang, Kian Hsiang Low, Jie Chen, and Patrick Jaillet. Multi-robot active sensing of non-stationary gaussian process-based environmental phenomena. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pages 573–580, 2014.

[24] Mahmoud Ali and Lantao Liu. Light-weight pointcloud representation with sparse gaussian process. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4931–4937, 2023.

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

[26] Rishit Sheth, Yuyang Wang, and Roni Khardon. Sparse variational inference for generalized gp models. In *International Conference on Machine Learning*, pages 1302–1311. PMLR, 2015.

[27] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intell. and statist.*, pages 567–574. PMLR, 2009.

[28] Mahmoud Ali and Lantao Liu. Gp-frontier for local mapless navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10047–10053. IEEE, 2023.

[29] Mahmoud Ali, Hassan Jardali, Nicholas Roy, and Lantao Liu. Autonomous Navigation, Mapping and Exploration with Gaussian Processes. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[30] Ihab S. Mohamed, Mahmoud Ali, and Lantao Liu. Gp-guided mppi for efficient navigation in complex unknown cluttered environments. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7463–7470, 2023.

[31] Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106, 2004.

[32] Mark Rose. Differential-Drive Controller, 2016. GitHub repository.

[33] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[35] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and systems*, volume 2, pages 1–9. Berkeley, CA, 2014.

[36] Clearpath Robotics. Husky unmanned ground vehicle robot.