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

Visual-Geometry GP-based Navigable Space for Autonomous Navigation

Mahmoud Ali, Durgkant Pushp, Zheng Chen, and Lantao Liu

Abstract— **Autonomous navigation in unknown environments is challenging and requires the consideration of both geometric and semantic information to assess the navigability of the environment. In this work, we propose a novel space modeling framework, Visual-Geometry Sparse Gaussian Process (VG-SGP), that simultaneously considers the semantics and geometry of the scene. Our proposed approach can overcome the limitation of visual planners that fail to recognize geometry associated with the semantic and the geometric planners that completely overlook the semantic information which is very critical in real-world navigation. The proposed method leverages dual Sparse Gaussian Processes in an integrated manner; the first is trained to forecast geometrically navigable spaces while the second predicts the semantically navigable areas. This integrated model is able to pinpoint the overlapping (geometric and semantic) navigable space. The simulation and real-world experiments demonstrate that the proposed VG-SGP model, coupled with our innovative navigation strategy, outperforms models solely reliant on visual or geometric navigation algorithms, highlighting a superior adaptive behavior. We provided a demonstration video[1] and open-sourced our code [2].**

I. Introduction and Related Work

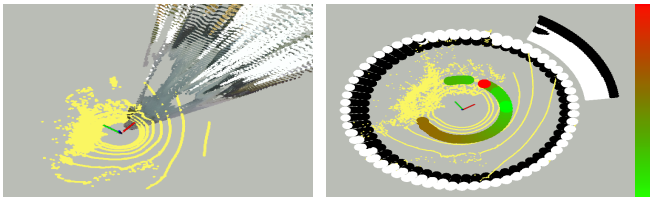
It is a challenging task for autonomous robots to navigate in real-world, unpredictable field environments which typically contain considerable complexity, with obstacles that may include large rocks, tree stumps, as well as terrains of mixed composition such as asphalt, sand, and mud, in addition to dense plant life like high and low grasses, shrubs, and trees. When deploying a robot in such a setting, it must determine the navigable portions of the scene it captures. One approach to identifying the navigable space around the robot is to use the geometry of the objects around the robot regardless of its semantic interpretation, we call this approach the geometry approach. The geometric approach usually exploits the 3D representation of the environment surrounding the robot in the form of either point cloud or depth observation [1], [2]. Geometric methodologies commonly employ either a global or local representation of the navigable space derived from a geometric analysis of the environment. Global representations include the use of an occupancy grid map, which encodes the likelihood of occupancy within a 2D spatial domain [3], a 2.5D elevation map [4] that outlines the terrain elevation of the environment, or 3D maps that model the 3D space using voxels, such as Octomaps [5]. On the other hand, local representations such as cost and traversability maps are exploited by local planners [6], [7]. Another approach to identifying the navigable spaces around the robot is the semantic approach which is accomplished by teaching the robot to perform semantic segmentation of the scene to predict its traversability using visual input [8]–[10]. Visual collision-free navigation has been approached through various methods, including the use of motion information from the desired object or optical flow features [11], [12]. Appearance-based techniques [13] involves storing images or templates of the environment alongside control actions, while an information-theoretic approach [14] employs image signal entropy for controlling non-holonomic robots. Also, Feature tracking methods are explored in [15]. Additionally, image segmentation techniques based on sub-region growing for pathfinding are presented in [16], where control parameters were estimated from path boundary information. While these approaches effectively address obstacle avoidance, they may fall short when it comes to achieving goal-oriented navigation in challenging environments where visually appealing navigable scenes (such as high slope grass land) are non-navigable geometrically.

The Gaussian Process (GP) stands as a reputable framework for modeling continuous spatial phenomena [17]–[20]. Also, GP is employed as a navigability map, grounded in either geometric principles [21] or semantic interpretations [22]. Nonetheless, the conventional GP is burdened by its considerable time complexity denoted by $\mathcal{O}(n^3)$. This aspect narrows down its utility in real-time applications. A prominent approach to relax the computational burden of the standard GP is to identify a crucial subset of the data, m inducing points, to represent the complete training dataset through the application of Bayesian rules [23]–[25], this approach is known as the Sparse GP (SGP). The computation complexity of the SGP, $\mathcal{O}(nm^2)$ is dependent on m, however, it is always less than the full GP's computation complexity. In this work, we use a variational sparse variant of GP introduced in [25]. The geometric-based navigation approaches (based on a continuous navigable space representation, such as GP-based maps) are able to ensure continuity in the representation of a robot's surroundings and demonstrate robustness against weather variations by utilizing LIDAR technology. However, they may incorrectly identify navigable spaces due to a lack of semantic consideration. Conversely, the visual-based navigation approaches, while flexible in defining navigable spaces through individual pixel analysis, are susceptible to weather conditions due to their reliance on visual inputs. The proposed strategy essentially merges visual and geometry-based navigable spaces to improve robot navigation behavior across a variety of tasks by introducing

All authors are with the Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN 47408 USA. {alimaa, dpushp, zc11, lantao}@iu.edu
This work was supported by National Science Foundation with grant numbers 2006886 and 2047169

[1]Video: https://youtu.be/0s6VSj5Z1dg

[2]Code: https://github.com/mahmoud-a-ali/vg-nav



(a) (b)

Fig. 1: Visual-Geometry Navigable Space: (a) shows LiDAR's pointcloud (yellow) and camera's colored-pointcloud; (b) shows the geometry navigable space (grey-coded circular surface, white regions represent free space), the visual navigable space (grey-coded vertical plane, white regions represents the navigable classes in the camera field of view), and the local navigation points (colored-circles, where color represents the go-to-goal cost).

adaptive behavior in different scenarios.

II. Methodology

We propose a new framework that combines the geometry and the semantics of the robot's surroundings to identify the navigable spaces around the robot. Briefly, the output of the RGBD camera and LiDAR are processed in a parallel way where the RGB image is segmented by labeling each pixel with a unique class (grass, tree, asphalt, etc), and the pointcloud is represented by an occupancy surface around the robot. Consequentially, the segmented image is converted into a binary navigability image (navigable and non-navigable pixels) based on a defined set of navigable classes. The navigability is then projected on the camera's depth pointcloud which is represented as a *Visual* SGP model (V-SGP) to find the (*visual*) navigable spaces in front of the robot. On the other hand, the occupancy surface is represented by a *Geometry* SGP model (G-SGP) to assess the free and occupied spaces and to identify the (*geometric*) navigable space around the robot. The visual-based and the geometry-based navigable spaces are coupled to calculate a more accurate navigable space, based on which the Local Navigation Points (LNPs) are generated to drive the robot to its destination, see Fig. 1.

A. Geometry Navigable Space: The G-SGP Model

The LiDAR's pointcloud, $\mathbf{P}_l = \{(x_i, y_i, z_i)\}_{i=1}^{n_l}$, is converted to the occupancy surface, $\mathcal{S}_g$, representation [20], where each (geometry) point, $\mathbf{p_g}_i$, on $\mathcal{S}_g$ is defined by its azimuth, α_i, and elevation, β_i, angles, and given an occupancy value, Ω_i, equals to the difference between the surface radius, ρ_g, and the point radius, ρ_i, as follows $\Omega_i = \rho_g - \rho_i$. The surface regions with projected points represent the *occupied space* of $\mathcal{S}_g$. In contrast, other regions with no points represent the *free space* of $\mathcal{S}_g$, see Fig. 2b. The n_g projected points on $\mathcal{S}_g$ form the geometric data set $\mathcal{D}_g = \left\{\left(\mathbf{p_g}_i, \Omega_i\right)\right\}_{i=1}^{n_g}$, where $\mathbf{p_g}_i = (\alpha_i, \beta_i)$, and Ω_i is the occupancy of $\mathbf{p_g}_i$. Subsequently, the Geometric data set $\mathcal{D}_g$ is employed to train a 2D variational SGP (G-SGP), to model the probability of occupancy, $f_g(\mathbf{p_g}_i)$, over $\mathcal{S}_g$ as follows:

$$f_g(\mathbf{p_g}) \sim SGP_g\left(m_g(\mathbf{p_g}), k_g\left(\mathbf{p_g}, \mathbf{p_g}'\right)\right),$$
$$k_g\left(\mathbf{p_g}, \mathbf{p_g}'\right) = \sigma_1^2\left(1 + \frac{\left(\mathbf{p_g} - \mathbf{p_g}'\right)^2}{2\alpha_1 \ell_1^2}\right)^{-\alpha_1}, \quad (1)$$

where $m_g(\mathbf{p_g})$ is the zero mean function, and $k_g\left(\mathbf{p_g}, \mathbf{p_g}'\right)$ is a Rational Quadratics (RQ) kernel with a length-scale ℓ_1, a signal variance σ_1^2, and a relative weighting factor α_1. A Gaussian noise ε_g is added to the predicted occupancy to reflect the measurement noise. The probability of occupancy Ω_g^* for any query point $\mathbf{p_g}^*$, $f_g(\mathbf{p_g}^*)$, on $\mathcal{S}_g$ is calculated by the GP prediction as follows,

$$f_g(\mathbf{p_g}^*) = \mathcal{N}_g(\Omega_g^* | m_{\boldsymbol{\Omega_g}}(\boldsymbol{p_g^*}), k_{\boldsymbol{\Omega_g}}(\boldsymbol{p_g^*}, \boldsymbol{p_g^*}) + \sigma_{n_g}^2),$$
$$m_{\boldsymbol{\Omega_g}}(\boldsymbol{p_g}) = K_{\boldsymbol{p_g} n_g}\left(\sigma_{g_n}^2 I + K_{n_g n_g}\right)^{-1} \boldsymbol{\Omega_g},$$
$$k_{\boldsymbol{\Omega_g}}\left(\boldsymbol{p_g}, \mathbf{p_g}'\right) = k\left(\boldsymbol{p_g}, \boldsymbol{p_g'}\right) - K_{\mathbf{p_g} n_g}\left(\sigma_{n_g}^2 I + K_{n_g n_g}\right)^{-1} K_{n_g \boldsymbol{p_g'}}, \quad (2)$$

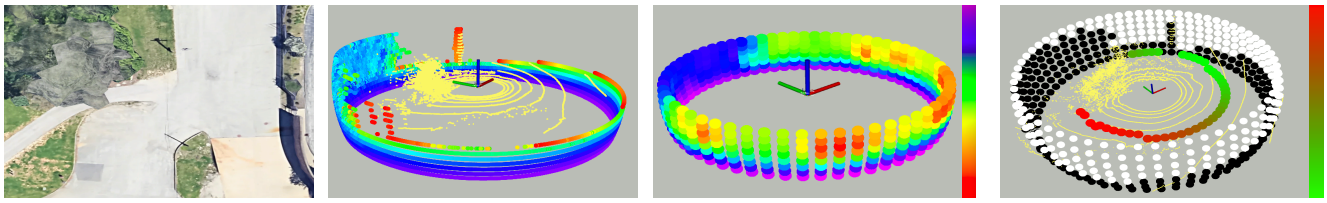
where $m_{\boldsymbol{\Omega_g}}(\boldsymbol{p_g})$ and $k_{\boldsymbol{\Omega_g}}\left(\boldsymbol{p_g}, \boldsymbol{p_g'}\right)$ are the posterior mean and covariance functions [25], $K_{n_g n_g}$ is $n_g \times n_g$ co-variance matrix of the inputs, $K_{p_g n_g}$ is n_g-dimensional row vector of kernel function values between $\boldsymbol{p_g}$ and the inputs, and $K_{n_g p_g} = K_{p_g n_g}^T$. We leverage the variational SGP approach [25] to estimate the kernel hyperparameters Θ and to select the inducing points X_m, more details about the implementation of the G-SGP model can be found in our previous work [20]. Fig. 2c shows the predicted occupancy μ_g on predicted occupancy surface $\mathcal{S}_{\mu_g}$, where the prediction uncertainty σ_g is shown as the variance surface $\mathcal{S}_{\sigma_g}$ in Fig. 2d. Regarding the accuracy of the SGP occupancy model, the reconstructed pointcloud from a G-SGP model with 400 inducing points has an average error of 12 cm [20].

The variance surface $\mathcal{S}_{\sigma_g}$ discriminates efficiently between the free space (white regions with a variance higher than a threshold $V_{g_{th}}$) and the occupied space (dark regions with a variance less than $V_{g_{th}}$) around the robot [26], and reflects the terrain elevation (the boundary between free and occupied space) in the local observation [27]. Therefore, $\mathcal{S}_{\sigma_g}$ is used to define a set of geometrical-feasible LNPs (G-LNPs) around the robot in the free space that are considered navigable based on the robot's maximum roll and pitch angles. G-LNPs are the lowest free points on the variance surface whose elevation angles are bounded by the safe elevations that the robot can climb, (β_{min_s} and β_{max_s}); G-LNPs $= (\alpha_i, \beta_{j^*}) \mid -\pi < \alpha_i < \pi\}$ and $\beta_{min_s} < \beta_j^* < \beta_{max_s}$; and their variance $\sigma_g(\alpha_i, \beta_{j^*})$ is greater than $V_{g_{th}}$.

Formally, a G-LNP is defined as $\mathbf{g}_{lnp_i} = (\alpha_i, \beta_i, \rho_i)$, where α_i defines the direction of $\mathbf{g}_{lnp_i}$ with respect to the robot heading, β_i is the elevation of $\mathbf{g}_{lnp_i}$ with respect to the robot, and ρ_i is the distance between $\mathbf{g}_{lnp_i}$ and the robot predicted by the G-SGP model; $\rho_i = \rho_g - \hat{\Omega}_i$, where $\hat{\Omega}_i = \mu_{g_i}$ and $(\mu_{g_i}, \sigma_{g_i}) = \mathcal{SGP}_g((\alpha_i, \beta_i))$. The Cartesian coordinates of $\mathbf{g}_{lnp_i}$ within the global world frame $\mathcal{W}$ are derived as $(x_i^w, y_i^w, z^w i) = {}^W\mathbf{T}_R \cdot (x_i^R, y_i^R, z^R i)$, where ${}^W\mathbf{T}_R$ represents the robot's localization. The coordinates $(x_i^R, y_i^R, z^R i)$ denote the position of $\mathbf{g}lnp_i$ in the robot frame $\mathcal{R}$, calculated from its spherical coordinates $(\alpha_i, \beta_i, \rho_i)$.

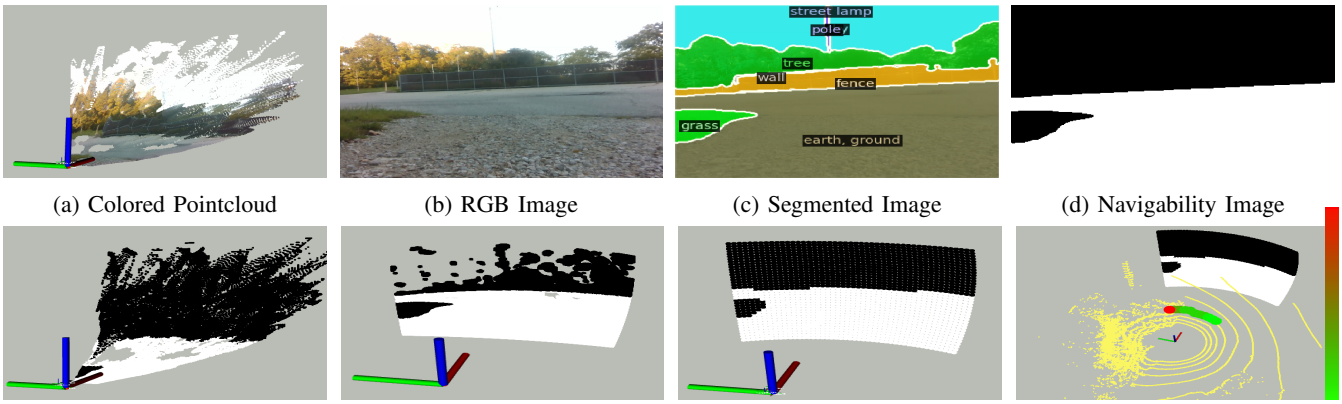
B. Visual Navigable Space: V-SGP Model

The RGB image contains crucial information contributing to obstacle identification, which may not be revealed through LiDAR sensing (geometry-navigable space). The semantic



(a) Environment (b) Occupancy surface $\mathcal{S}_g$ (c) SGP occupancy surface $\mathcal{S}_{\mu_g}$ (d) SGP Variance Surface $\mathcal{S}_{\sigma_g}$

Fig. 2: Geometry-Navigable Space: (b) shows the raw pointcloud in yellow and the original occupancy surface , where warmer colors indicate less occupancy; (c) shows the predicted occupancy surface using the G-SGP model; (d) shows the variance surface, i.e., the uncertainty associated with the predicted occupancy, where white color indicates highly uncertain (free) points. The Geometry-LNPs (G-LNPs) are shown as colored circles, where green indicates lower go-to-goal costs and red indicates higher go-to-goal costs.



(a) Colored Pointcloud (b) RGB Image (c) Segmented Image (d) Navigability Image

(e) Visual Pointcloud (f) Visual Surface (g) SGP Navigability Model (h) Visual LNPs

Fig. 3: Visual-Navigable Space: V-LNPs are shown as colored-circles in (h), where the color shows the cost assigned to each V-LNP.

image I_{seg} and the associated class labels I_{cls} can be obtained by $I_{seg}, I_{cls} = g(I_{rgb}, \Theta)$ where $g(.)$ represents any existing image segmentation model with parameter Θ, see Fig. 3c. In this paper we use the *mask2former* segmentation model [10]. Utilizing our domain knowledge regarding the physical properties of objects within the scene, we construct the navigability image, denoted as I_{nav}, by assigning a value 0 to all pixels whose class labels are categorized as non-navigable and a value 255 to all other pixels, see Fig. 3d. The navigability image provides a dynamic categorization of navigable classes which is crucial in defining flexible visually navigable spaces. Depending on the robot's capabilities and the required behavior, various terrains such as grass, snow, or dirt can be designated as navigable or non-navigable, enhancing the robot's adaptive behavior across different scenarios.

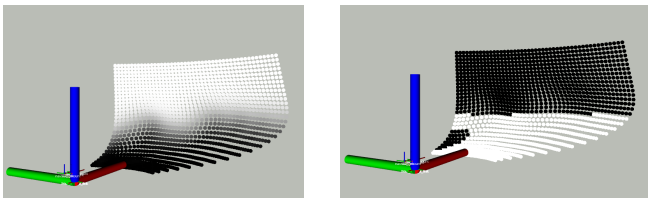
The navigability image I_{nav} is projected on the visual depth pointcloud $\mathbf{P}_c = \{(x_i, y_i, z_i, rgb_i)\}_{i=1}^{n_v}$ by replacing the rgb_i value with the binary navigability value, ι_i, where each point is set to either navigable or non-navigable, resulting in the *navigability pointcloud* $\mathbf{P}_v = \{(x_i, y_i, z_i, \iota_i)\}_{i=1}^{n_v}$, see Fig. 3e. Similar to $\mathcal{S}_g$ in Sec. II-A, the navigability points are projected on a curved visual surface $\mathcal{S}_v$ with a predefined radius ρ_v, where the visual surface span is aligned with the camera field of view (FoV). Each point on $\mathcal{S}_v$ is represented by its azimuth, elevation, radius, and navigability values, $\mathcal{S}_v = \{(\alpha_i, \beta_i, \rho_i, \iota_i)\}_{i=1}^{n_v}$, see Fig. 3f. $\mathcal{S}_v$ is then decomposed into two data sets, the visual navigability dataset $\mathcal{D}_{nav} = \{\mathbf{p}_{\mathbf{v}i}, \iota_i\}_{i=1}^{n_v}$ and the visual depth data set $\mathcal{D}_{dpth} = \{\mathbf{p}_{\mathbf{d}i}, \rho_i\}_{i=1}^{n_v}$, where $\mathbf{p}_{\mathbf{v}i} = \mathbf{p}_{\mathbf{d}i} = (\alpha_i, \beta_i)$. $\mathcal{D}_{dpth}$ is then filtered to include only the navigability points whose ρ less than a constant ρ_d, i.e., $\mathcal{D}_{dpth} = \{\mathbf{p}_{\mathbf{d}i}, \rho_i\}_{i=1}^{n_v} : \mathbf{p}_{\mathbf{d}i} = \mathbf{p}_{\mathbf{v}i} \mid \rho_i < \rho_d$. Thereafter, the visual surface is modeled using two SGP models. The first one is the SGP Depth (D-SGP) model , which is an SGP regression model trained on $\mathcal{D}_{dpth}$ in a similar way as the G-SGP model to predict the occupancy of $\mathbf{p}_{\mathbf{d}i}$ as $\Omega_{\mathbf{d}i} = f_d(\mathbf{p}_{\mathbf{d}i}) + \varepsilon_d$:

$$f_d(\mathbf{p_d}) \sim SGP_d\left(m_d(\mathbf{p_d}), k_d\left(\mathbf{p_d}, \mathbf{p_d}'\right)\right),$$
$$k_{\mathrm{d}}\left(\mathbf{p_d}, \mathbf{p_d}'\right) = \sigma_2^2\left(1 + \frac{(\mathbf{p_d} - \mathbf{p_d}')^2}{2\alpha_2\ell_2^2}\right)^{-\alpha_2}, \tag{3}$$

where $m_d(\mathbf{p_d})$, $k_d(\mathbf{p_d}, \mathbf{p_d}')$, and ε_d are similar to Eq. (1). On the other hand, the second model is the SGP Navigability (N-SGP) model, which is an SGP classification model trained on $\mathcal{D}_{nav}$ to predict the navigability status ι of any point $\mathbf{p}_{\mathbf{v}i}$. The N-SGP classification model uses the same variational SGP framework as the G-SGP model where $\Omega_{\mathbf{v}i} = f_v(\mathbf{p}_{\mathbf{v}i}) + \varepsilon_v$, however, its output probability is thresholded to decide whether $\mathbf{p}_{\mathbf{v}i}$ is navigable or not, see Fig. 3g,

$$f_n(\mathbf{p_v}) \sim SGP_v\left(m_v(\mathbf{p_v}), k_v\left(\mathbf{p_v}, \mathbf{p_v}'\right)\right),$$
$$k_{\mathrm{v}}\left(\mathbf{p_v}, \mathbf{p_v}'\right) = \sigma_3^2\left(1 + \frac{(\mathbf{p_v} - \mathbf{p_v}')^2}{2\alpha_3\ell_3^2}\right)^{-\alpha_3}, \tag{4}$$

$$\iota_{\mathbf{v}_i} = \begin{cases} 255 \text{ "navigable"} & \Omega_{v_i} > \Omega_{v_{th}} \\ 0 \text{ "non-navigable"} & \text{otherwise.} \end{cases} \tag{5}$$



(a) D-SGP (b) Combined V-SGP and N-SGP

Fig. 4: D-SGP and N-SGP models of camera's pointcloud: both (a) and (b) shows the pointcloud generated from the D-SGP model, where in (a) the grey color indicates the uncertainty while in (b) the grey color indicates the predicted navigability class.

where $m_d(\mathbf{p_d})$, $k_d(\mathbf{p_d}, \mathbf{p_d}')$ are similar to Eq. (1), and $\Omega_{v_{th}}$ is a predefined threshold. Fig. 4a shows the pointcloud predicted by the D-SGP model, $\hat{\mathbf{P}}_c = \{(x_i, y_i, z_i, \sigma_{d_i})\}_{i=1}^{N_c}$, where the grey-color intensity indicates the uncertainty σ_d of the predicted radius $\hat{\rho}$, i.e., darker points are more certain. Since the model is trained only on the points whose radius values ρ_i are less than ρ_d, the D-SGP prediction in the range of ρ_d is certain. That is why the dark (certain) points in Fig. 4a are similar, in terms of 3D location, to the flat region (asphalt and grass) of the camera's point cloud $\mathbf{P}_c$ in Fig. 3a, while the white (uncertain) points have an inaccurate prediction. For more details about the accuracy of the generated pointcloud, check [20]. It is worth mentioning that the smoothness property of the GP denoises the pointclouds, where individual points that do not belong to any point cluster are smoothed out. Fig. 4b shows $\hat{\mathbf{P}}_c$ with the color representing the navigability $\hat{\imath}$ predicted by the N-SGP model; $\hat{\mathbf{P}}_v = \{(x_i, y_i, z_i, \imath_i)\}_{i=1}^{N}$. The prediction combination of D-SGP and N-SGP, $\hat{\mathbf{P}}_v$, is a reconstruction of the navigability pointcloud $\mathbf{P}_v$ in Fig. 3e. Only the certain points whose $\hat{\rho} \leq \rho_d$ and $\sigma_d < V_{d_{th}}$, are considered for planning.

The combination of the D-SGP and the N-SGP models is considered as one entity, called the *visual* SGP (V-SGP) model, see Fig. 3. The V-SGP model can predict both the 3D location of the missing points in the camera's FoV (points with a 'nan' return value) and their navigability status, check the difference $\mathbf{P}_v$ in Fig. 3e and $\hat{\mathbf{P}}_v$ predicted by the V-SGP model in Fig. 4b. In the geometry-based approach, G-LNPs are selected solely based on the geometry information encoded in $\mathcal{S}_g$, Contrastingly, $\mathcal{S}_v$ introduces a more nuanced layer to the navigable space assessment by incorporating the navigability status denoted as $\imath$. In the visual-based approach, points falling within the safe elevation boundaries, $\beta_{min_s} < \beta < \beta_{max_s}$, are considered as LNP candidates. Then the navigability cost $\imath$ is estimated using the N-SGP model for all candidates to form the Visual LNPs (V-LNPs), $\boldsymbol{\nu}_{lnp}$, see Fig. 3h.

C. LNPs Selection For Goal-oriented Navigation

In general, to navigate the robot towards a given goal $\mathbf{g} = (x_f, y_f)$ in $\mathcal{W}$, local mapless navigation approaches use different criteria to select one LNP (i.e., an ideal G-LNP ($\mathbf{g}^*_{lnp}$) or V-LNP (ν^*_{lnp})) from the feasible LNPs. For example, the FGM [28] selects the ideal LNP based on the area of the free space around it and its direction to the final goal $\mathbf{g}$. While the admissible gap approach [29] selects the closest LNP to $\mathbf{g}$. We utilize the cost function $C_g(\mathbf{g}_{lnp_i})$ introduced in [27], which accounts for the distance to $\mathbf{g}$, the alignment relative to the robot's heading, and the elevation angle of the LNP,

$$\begin{aligned} C_g(\mathbf{g}_{lnp_i}) &= k_{dst} d_{tg} + k_{dir}\|\alpha_i\| + k_{elv}\|\beta_i\|, \\ d_{tg} &= \rho_i + \sqrt{(x_f - x_i^w)^2 + (y_f - y_i^w)^2}, \\ \mathbf{g}^*_{lnp} &= \arg \min_{\mathbf{g}_{lnp_i} \in \text{LNPs}} (C_g(f_i)), \end{aligned} \tag{6}$$

where k_{dst}, k_{dir}, k_{elv} are weighting factors. The go-to-goal cost $C_g(\mathbf{g}_{lnp_i})$ is applied to the G-LNPs, where in the case of the V-LNPs the navigability $\imath$ is used to mask the cost of the visually non-navigable LNPs to the maximum as follow,

$$C_v(\boldsymbol{\nu}_{lnp_i}) = \begin{cases} C_g(\mathbf{g}_{lnp_i}) & \text{if } \imath_i = 255, \\ 1 & \text{otherwise.} \end{cases} \tag{7}$$

Once the optimal LNP ($\mathbf{g}^*_{lnp}$ / ν^*_{lnp}) is selected, a motion command with linear and angular velocities, v and ω, is generated as follows: $v = k_a \rho_* - k_b\|\alpha_*\|$, and $\omega = k_c \alpha_*$. ρ_* and α_* are the attributes of the optimal LNP ($\mathbf{g}^*_{lnp}$ / ν^*_{lnp}), where k_a, k_b and k_c are tunable coefficients.

D. Joint Visual-Geometry Model

Our integration methodology excludes the D-SGP model from the V-SGP model, thereby establishing two distinct SGP Models: the unaltered G-SGP model and the N-SGP model. This process essentially substitutes the D-SGP model with a segment of the G-SGP model, specifically the area where the FOVs of the camera and LiDAR overlap, see Fig. 1b. The operational workflow begins with the computation of G-LNPs derived from the G-SGP model. Following this, the N-SGP model is engaged to determine the navigability of the identified G-LNPs to define the Visual G-LNPs (VG-LNPs). For goal-oriented navigation task, one VG-LNP is selected based on the same go-to-goal cost function introduced in Eq. 6, however the navigability mask in Eq. 7 is modified in such away that each G-LNP is allocated a navigability score as follows,

$$C_{vg}(\boldsymbol{\nu}\mathbf{g}_{lnp_i}) = \begin{cases} (1 - k_{nav}) C_g(\mathbf{g}_{lnp_i}) & \text{if } \imath_i = 255, \\ 1 & \text{if } \imath_i = 0, \\ k_{nav} C_g(\mathbf{g}_{lnp_i}) & \text{Out of camera's FoV.} \end{cases} \tag{8}$$

where $0.5 < k_{nav} < 1$ is a parameter allowing the user to control the preference between visually navigable VG-LNPs and G-LNPs outside the camera's FoV. When $k_{nav} = 0.5$, there exists no preference between the two, granting them an equal cost of $0.5 C_g$. Conversely, setting $k_{nav} > 0.5$ augments the propensity to opt for VG-LNPs situated within the camera's FoV.

III. Experimental Design and Results

A. Experimental Setup

The proposed visual-geometry navigation algorithm is built on top of *gpflow* [30] and integrated with the Robot Operating System (ROS) framework. A Clearpath Jackal

robot equipped with a VLP-16 velodyne and Realsense D435 camera is used to validate our algorithm in both simulation and real-world scenarios. For the simulation experiments, we used the grass-mud environment with ground-truth localization and an RGB-based segmentation to generate the navigability image from the RGB image. While for real hardware experiments, we used a microstrain GNSS/INS module for localization and the *mask2former* [10] segmentation to generate the navigability image. The Velodyne pointcloud $\mathbf{P}_l$ is used to construct the occupancy surface $\mathbf{S}_g$ to train the G-SGP model, where the front Realsense D435 RGBD camera is used to generate the navigability poincloud $\mathbf{P}_v$ and construct the visual surface $\mathbf{S}_v$ to train the V-SGP model. Our algorithm runs in real-time with a frequency of 4 Hz in real-world experiment and 10 Hz in simulation due to the time difference required by RGB-Based and the *mask2former* segmentation. Therefore, we limited the maximum velocity in the realworld experiment to 0.3 m/s. We assessed our algorithm's effectiveness by comparing it to two established methods. The first baseline is a purely visual-based navigation named *PovNav* [31], that employs I_{rgb} for generating I_{nav}, similar to our proposed method. However, it uses image-based visual-servoing for motion command generation. The second baseline is GPFrontiers framework [26], [27], local mapless navigation which contrasts PovNav by focusing solely on geometry-based navigation. For clarity, we refer to *PovNav* as V-Nav, GPFrontiers as G-Nav, and our proposed approach as VG-Nav.

B. Simulation Scenarios and Results

In the grass-mud environment, two visual classes are identified: grass and mud, with grass designated as navigable and mud as non-navigable. Furthermore, the environment features two types of terrain: flat terrain and high slope grass (HSG), with HSG being geometrically non-traversable. Our experimental design request the robot to navigate from a start pose of $(15,-4,-\pi/2)$ to a goal pose of $(-4,-16,\pi)$, avoiding mud and HGS regions. Initially facing HSG, the robot must circumvent it, as well as a mud area presented along the straight path to the goal. Successful task completion necessitates the integration of both visual and geometric navigation capabilities to sidestep HSG and mud obstacles. For each approach, we conducted 15 trials to examine its navigation behavior. First, we executed the G-Nav, which, as anticipated, circumvented the HSG area and opted for a direct path to the goal, traversing the mud area due to lack of a visual component. The paths generated by G-Nav are depicted in red in Fig. 5. Subsequently, we deployed the V-Nav which failed to circumvent the HSG, perceiving all grass as navigable due to missing the geometric information. This resulted in all 15 V-Nav trials ending with the robot stuck trying to climb the HSG, depicted as black paths in Fig. 5. To evaluate the V-Nav performance on flat terrain, we reoriented the robot to face flat areas instead of HSG. In this revised configuration (V-Nav:Flat), V-Nav successfully navigated through flat grass, avoiding mud areas in 12 out of 15 trials. However, in 3 trials, V-Nav failed to circumvent mud due to erroneous motion commands. The critical observation was that V-Nav, when detecting only non-navigable classes (mud) within the camera's FoV, tended to reach a local minimum, failing to avoid going into non-navigable areas, check the blue paths in Fig. 5. This behavior is replicated in real-world experiments as well, see Fig. 8a. Finally, we tested our proposed VG-Nav in the original setup with the robot facing HSG. VG-Nav successfully guided the robot to the goal while processing both geometric and visual information, effectively avoiding both HSG and mud areas. The resulting paths are illustrated in cyan in Fig. 5.

V-Nav tries to follow the most visually-navigable space on the navigability image, resulting in a longer path length of 33 m, in contrast to VG-Nav, which prioritizes the goal direction unless avoiding mud areas, achieving a shorter average path length of 29.2 m. Moreover, the average of the achieved maximum velocity over the 15 trials was $1.06m/s$ for VG-Nav and $0.3m/s$ for V-Nav, as detailed in Table I. Simulation results are shown in the supplementary video.

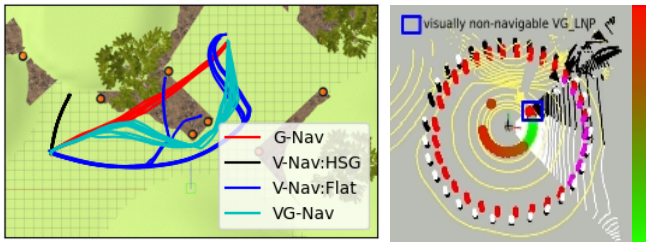


Fig. 5: Simulation Experiments: the right figure shows the cost of the VG-LNPs.

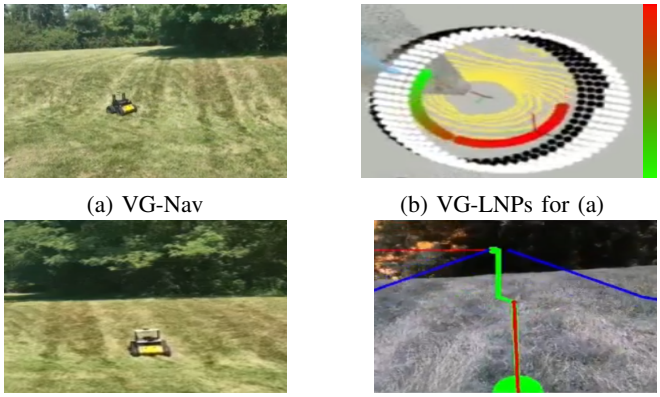
TABLE I: Navigation metrics: G-Nav, V-Nav, and VG-Nav

	Path[m]	$avg(v_{max})$	Skip Mud	Skip Slope
G-Nav	22.5 ± 0.15	1.06 ± 0.10	0%	100%
V-Nav:HSG	-	-	-	0%
V-Nav:Flat	33.0 ± 0.30	0.30 ± 0.01	80%	-
VG-Nav	29.2 ± 0.46	1.03 ± 0.06	100%	100%

C. Real World Scenarios and Results

Three experiments were conducted to validate the outcomes of the simulation studies. In the first setup, the robot is instructed to move towards a goal positioned in a straight line ahead of it. Despite the straightforward path, opting for this route would necessitate crossing over a grassy area, where grass is defined as non-navigable and asphalt/road is defined as navigable. Subsequently, we conducted 3-trials utilizing: G-Nav, V-Nav, and VG-Nav. Fig. 6(left) illustrates the divergent paths chosen by the robot under the influence of each algorithm. Notably, the G-Nav failed to avoid the grass, lacking the component to recognize the grass as a non-navigable zone. In contrast, both V-Nav and VG-Nav successfully navigated around the grassy region, with VG-Nav opting for a shorter path closer to the grass boundary. These results corroborate the findings from the simulation experiments. Fig.6(right) visualizes the cost associated with VG-LNPs. VG-LNPs situated within the non-navigable part of the visual point cloud are assigned the highest cost, in accordance with Eq.8. Conversely, the remaining VG-LNPs are assigned costs based on the go-to-goal cost function,

Fig. 6: Experiment 1: G-Nav can not avoid mud; G-Nav(yellow), V-Nav(green), VG-Nav(cyan)

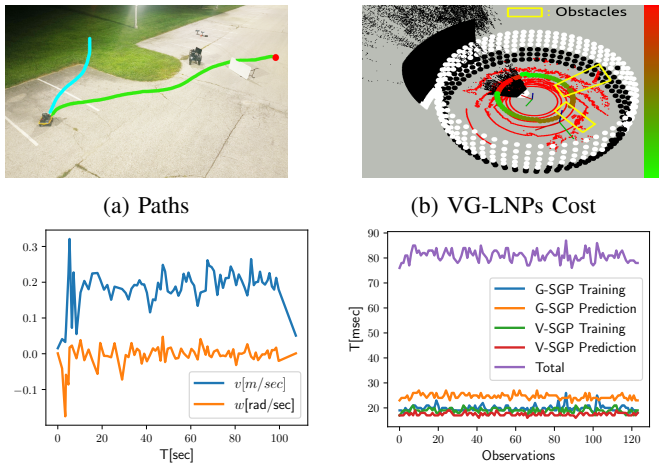


(a) VG-Nav (b) VG-LNPs for (a)

(c) V-Nav (d) Visual Path for (c)

Fig. 7: Experiment 2: V-Nav can not avoid HSG.

$C_g\left(\mathbf{g}_{lnp_i}\right)$. In the second experiment, the robot was positioned in an HSG area with grass classified as navigable. The robot was directed towards a goal straight ahead, yet its path passes through an area of grass with the steepest slope. We conducted 3 trials using V-Nav and VG-Nav. VG-Nav successfully identified HSG and avoided it. As illustrated in Fig.7b, which corresponds to the local observation shown in Fig.7a, no VG-LNPs are present on the robot's right side that leads to HSG, instead, VG-LNPs are positioned forward, backward, and to the left of the robot, in zones within elevation angles with the maximum range β_{max}. Conversely, V-Nav attempted to guide the robot through the HSG along the straight path. Fig.7d depicts the visual path planned based on the local observation in Fig.7c. For V-Nav, we ended the experiment before the robot goes to the steepest slope area. These outcomes further validate the insights from the simulation studies. The third experiment replicated the local minimum behavior observed with V-Nav. In this setup, the robot heading was initiated to face a corner formed by a patch of grass (assigned as non-navgiable), deliberately testing the constraints imposed by a limited visual FoV. Further complicating the path, a table was placed on the direct path to the goal, to introduce a geometric obstacle. Five trials were conducted, where V-Nav failed to get out of the grass corner in all trials due to the local minimum it encounters when the FoV contains only the non-navigable class, check the cyan path in Fig 8b. In contrast, the VG-Nav demonstrated superior adaptability, managing to identify geometrically feasible G-LNPs outside the camera's FoV. All VG-LNPs in camera's FoV were masked to the maximum cost, see Fig 8b, therefore the VG-Nav command the robot to follow the G-LNP with the minimum cost which lies outside the camera's FOV, this is interpreted from the high angular



(a) Paths (b) VG-LNPs Cost

(c) v and ω (d) Computation Time

Fig. 8: Experiment 3: V-Nav local minimum due to limited FoV; V-Nav(cyan), VG-Nav(green).

and low linear velocities values on the first few seconds which led to a sharp rightward rotation, see Fig 8c.

We also analyzed the computational costs associated with implementing our VG-Nav. To facilitate a granular understanding of the system's performance, we computed the time cost for each individual observation. This encompassed the time required for the training phase of both the G-SGP and V-SGP models. Moreover, we considered the time consumed to predict $\mathcal{S}_g$ alongside its uncertainty $\mathcal{S}_{\sigma_g}$, and the time for predicting the navigability of the G-LNPs by the G-SGP model. Fig. 8d shows the time costs for each component and their sum as the the total time. The average training time of both the G-SGP and the V-SGP models is around $19msec$, while the average prediction time of the G-SGP ($24.5msec$) is higher than the average navigability prediction time of the V-SGP model ($17.5msec$) because the V-SGP only predicts the navigability of the feasible G-LNPs identified by the G-SGP model. The average total time cost for each observation is around ($81msec$). It is worth mentioning that the time cost does not explode over the operation time (number of observations) because we consider each observation individually.

IV. Conclusion

We present the Visual Geometry Combined Spaces (VG-SGP) model along with a corresponding navigation strategy designed to adeptly guide a robot to its destination. This method leverages the environment analysis of two separate SGP models to pinpoint areas that are navigable based on both visual and geometric characteristics in the robot's surroundings. By synthesizing both visual and geometric data, our approach facilitates more reliable and adaptive navigation. Simulation and real-world experiments demonstrate that our VG-SGP model and its integrated navigation strategy outperform systems reliant solely on either visual or geometric navigation algorithms, showcasing superior adaptive behavoir required for accomplishing flexible tasks.

REFERENCES

[1] K. Weerakoon, A. J. Sathyamoorthy, U. Patel, and D. Manocha, "Terp: Reliable planning in uneven outdoor environments using deep reinforcement learning," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 9447–9453, IEEE, 2022.

[2] J. Liang, K. Weerakoon, T. Guan, N. Karapetyan, and D. Manocha, "Adaptiveon: Adaptive outdoor local navigation method for stable and reliable actions," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 648–655, 2022.

[3] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous robots*, vol. 15, pp. 111–127, 2003.

[4] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)* (A. Koubaa, ed.), ch. 5, Springer, 2016.

[5] 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.

[6] J. Shin, D. Kwak, and K. Kwak, "Model predictive path planning for an autonomous ground vehicle in rough terrain," *International Journal of Control, Automation and Systems*, vol. 19, pp. 2224–2237, 2021.

[7] I. S. Mohamed, M. Ali, and L. Liu, "Gp-guided mppi for efficient navigation in complex unknown cluttered environments," *arXiv preprint arXiv:2307.04019*, 2023.

[8] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3357–3364, IEEE, 2017.

[9] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi, "Visual semantic navigation using scene priors," *arXiv preprint arXiv:1810.06543*, 2018.

[10] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," 2022.

[11] A. Bernardino and J. Santos-Victor, "Visual behaviours for binocular tracking," in *The Second EUROMICRO Workshop on Advanced Mobile Robots*, pp. 2–7, 1997.

[12] C. Boretti, P. Bich, Y. Zhang, and J. Baillieul, "Visual navigation using sparse optical flow and time-to-transit," in *2022 Int. Conf. on Robotics and Automation (ICRA)*, pp. 9397–9403, 2022.

[13] S. R. Bista, P. R. Giordano, and F. Chaumette, "Appearance-based indoor navigation by ibvs using line segments," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 423–430, 2016.

[14] A. Dame and E. Marchand, "A new information theoretic approach for appearance-based navigation of non-holonomic vehicle," in *2011 IEEE Int. Conf. on Robotics and Automation*, pp. 2459–2464, 2011.

[15] D. Kim, D. Lee, H. Myung, and H. Choi, "Object detection and tracking for autonomous underwater robots using weighted template matching," in *2012 Oceans - Yeosu*, pp. 1–5, 2012.

[16] M. Zhang, Y. Li, and J. Yang, "Autonomous visual navigation guided by path boundaries for mobile robot," in *2008 International Conference on Computer Science and Software Engineering*, vol. 6, pp. 344–348, 2008.

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

[18] 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, pp. 2204–2211, 2007.

[19] R. Ouyang, K. H. Low, J. Chen, and P. 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*, pp. 573–580, 2014.

[20] M. Ali and L. Liu, "Light-weight pointcloud representation with sparse gaussian process," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4931–4937, 2023.

[21] M. G. Jadidi, J. V. Miro, and G. Dissanayake, "Warped gaussian processes occupancy mapping with uncertain inputs," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 680–687, 2017.

[22] M. G. Jadidi, L. Gan, S. A. Parkison, J. Li, and R. M. Eustice, "Gaussian processes semantic map representation," *arXiv preprint arXiv:1707.01532*, 2017.

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

[24] R. Sheth, Y. Wang, and R. Khardon, "Sparse variational inference for generalized gp models," in *International Conference on Machine Learning*, pp. 1302–1311, PMLR, 2015.

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

[26] M. Ali and L. Liu, "Gp-frontier for local mapless navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10047–10053, 2023.

[27] H. Jardali, M. Ali, and L. Liu, "Autonomous mapless navigation on uneven terrains," *arXiv preprint arXiv:2402.13443*, 2024.

[28] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm:"follow the gap method"," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1123–1134, 2012.

[29] M. Mujahed, D. Fischer, and B. Mertsching, "Admissible gap navigation: A new collision avoidance approach," *Robotics and autonomous systems*, vol. 103, pp. 93–110, 2018.

[30] 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.

[31] D. Pushp, Z. Chen, C. Luo, J. M. Gregory, and L. Liu, "Povnav: A pareto-optimal mapless visual navigator," *arXiv preprint arXiv:2310.14065*, 2023.