Path Generation for Wheeled Robots Autonomous Navigation on Vegetated Terrain

Zhuozhu Jian[1], Zejia Liu[2], Haoyu Shao[2], Xueqian Wang[1], Xinlei Chen[3], and Bin Liang[1]

Abstract—Wheeled robot navigation has been widely used in urban environments, but navigation in wild vegetation is still challenging. External sensors (LiDAR, camera etc.) are often used to construct point cloud map of the surrounding environment, however, the supporting rigid ground used for travelling cannot be detected due to the occlusion of vegetation. This often leads to unsafe or non-smooth paths during the planning process. To address the drawback, we propose the PE-RRT* algorithm, which effectively combines a novel support plane estimation method and sampling algorithm to generate real-time feasible and safe path in vegetation environments. In order to accurately estimate the support plane, we combine external perception and proprioception, and use Multivariate Gaussian Processe Regression (MV-GPR) to estimate the terrain at the sampling nodes. We build a physical experimental platform and conduct experiments in different outdoor environments. Experimental results show that our method has high safety, robustness and generalization. The source code is released for the reference of the community[1].

Index Terms—Field Robots Motion and Path Planning Collision Avoidance

I. INTRODUCTION

AUTONOMOUS navigation technology for unmanned ground vehicles (UGVs) has developed rapidly recently for both indoor [1]–[3] and outdoor [4]–[6] scenarios. With other technologies [7], [8], various novel applications become promising, such as city scale sensing [9], [10], post disaster quick response [11] etc. But navigation on uneven vegetated terrain remains a challenging task. Due to the presence of vegetation, the robot's perception of the environment becomes inaccurate and more time-consuming.

During the navigation process, the perception of the environment is very important [12]–[14]. Existing autonomous

[1]Zhuozhu Jian, Xueqian Wang, Bin Liang are with the Center for Artificial Intelligence and Robotics, Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China (e-mail: jzz21@mails.tsinghua.edu.cn; wang.xq@sz.tsinghua.edu.cn; liangbin@mail.tsinghua.edu.cn).

[2]Zejia Liu, Haoyu Shao are with the School of Mechanical Engineering and Automation at Harbin Institute of Technology, Shenzhen 518055, China (e-mail: 200320106@stu.hit.edu.cn; 200320520@stu.hit.edu.cn)).

[3]Xinlei Chen is with the Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China, Pengcheng Lab, Shenzhen 518055, China, RISC-V International Open Source Laboratory, Shenzhen 518055, China (e-mail: chen.xinlei@sz.tsinghua.edu.cn.

[1]Code: https://github.com/jianzhuozhuTHU/PE-RRTstar.

Fig. 1. When wheeled robots navigate autonomously in penetrable environments, support surface estimation is needed to ensure safety and optimization of the generated path.

navigation methods usually take the vegetation as the obstacle, but this method is too conservative, because for the shorter penetrable vegetation, the wheeled robots have the ability to pass through vegetation. Also, traditional methods usually need to build a prior traversability map [15]–[18] for navigation, which takes a lot of time, especially when accurate support ground estimation is needed in vegetated environment. In [4], the authors propose the PF-RRT*(Plane Fitting based RRT*) algorithm to generate traversable path on point cloud surfaces, this work achieves great result on uneven terrain. However, to run the PF-RRT* algorithm in a vegetation environment, there are still two problems: 1) the point cloud of vegetation does not correspond to the rigid geometry of the support ground; 2) Since the safe area is only limited to the radius envelope of the fitting plane, it is easy to collide with obstacles.

This work presents a real-time and safe path generation method to support autonomous navigation on vegetated terrain for wheeled robots. To solve challenge 1), we design a hybrid vegetated terrain estimation method, which fuses proprioception and external perception to generate support plane. The support plane is used to describe the local geometrically rigid terrain. To solve 2), the support plane estimation is integrated to sampling algorithm for path planning, which reduce the process time since the traversability map construction computation is skipped. In addition, the inflation radius is added to the sampling algorithm to enhance safety.

This work offers the following contributions:

1) A novel approach to accurately estimate the support plane is proposed, in which Multivariate Gaussian Process Regression (MV-GPR) based proprioception and external perception are fused considering uncertainty weighting.

2) We extend PF-RRT* to PE-RRT* (Plane Estimation RRT*), in which the safe inflation radius is innovatively introduced in the sampling algorithm to enhance the safety of the generated path.

3) We build the experimental platform and conduct real-world experiments. The effectiveness of our method is confirmed by comparison with existing methods.

II. RELATED WORK

In vegetated terrain, support surface is often invisible to external sensors. Therefore, the accurate perception of the support terrain is the premise of path planning. Some devices are designed to sense directly the ground. In [19], authors use an array of miniature capacitive tactile sensors to measure ground reaction forces (GRF) to distinguish among hard, slippery, grassy and granular terrain types. [20] produces a self-supervised mechanism to train the trafficability prediction model to estimate the trafficability. However, as the length of the trajectory increases and the terrain becomes more varied, the algorithm quality degrades.

Some methods attempt to traverse vegetation based on external sensors. LiDAR is a commonly used external sensor for navigation. In [21], the authors use a fully-trained Deep Reinforcement Learning (DRL) network to compute an attention mask of the environment based on elevation map constructed by LiDAR. [22] uses Markov random fields to infer the supporting ground surface based on LiDAR points. Learning-based methods combined with visual sensors are often used for outdoor navigation. [23]–[25] apply self-supervised learning method based on RGB image information to implement outdoor terrain navigation. [26] defines a regression problem which estimates predicted error between the realized odometry readings and the predicted trajectory. However, learning-based methods often lack robustness and are difficult to ensure safety [27] [28].

Combining proprioception and external perception to improve robustness is considered to be a common and effective approach. [29] provides robustness of hexapod locomotion in high grass by switching between two locomotion modes based on proprioceptive and exteroceptive variance estimates. In [30], the authors propose an attention-based recurrent encoder integrating proprioceptive and exteroceptive input. This approach is applied to quadrupeds and validated experimentally. And in [31], the authors apply Gaussian process regression (GPR) to estimate support surface including the height of the penetrable layer. However, the above work has to build a prior map first, and then analyze the travesability of each foothold, which can cause large computational expense and cost of time. And for the more commonly used wheeled robots, navigation pays more attention to the overall properties of the ground.

In our work, we propose the PE-RRT* algorithm, which avoids the explicit maps by sampling to significantly reduce computational expense. We describe the ground as the set of circular planes, and fuse the height and slope of the planes generated by MV-GPR [32] by taking the variance as the weight. The proposed technology inspires new applications such as city scale fine-grained sensing [33], [34], temporary communication infrastructure construction [35] and city scale 3D sensing [36] etc.
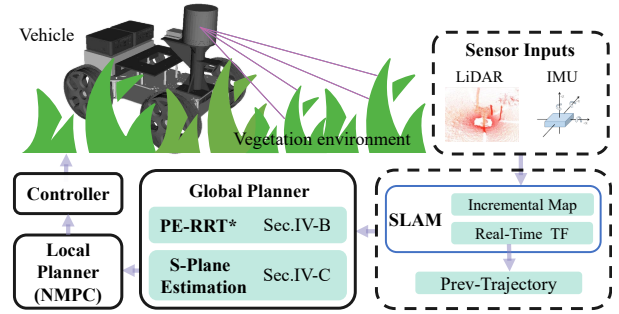

Fig. 2. System Workflow. From left to right: During the movement, the robot receives the data from LiDAR and IMU, and generates a LiDAR-inertial odometry. Based on the odometry, point cloud is registered to build an incremental map. The global planner generates a real-time safe feasible global path. The local planner based on NMPC (Nonlinear Model Predictive Control) pubs control inputs to the robot controller to follow the global path.

III. PROBLEM FORMULATION

Our objective is to generate a global path on the rigid geometric surface based on point cloud representing the vegetated environment. In our work, we simplify the local geometric support terrain of a single point into a support plane (S-Plane) $\Phi_S := \{x, y, z, r, p\}$, which contains the roll angle $r \in \mathbb{R}$, pitch angle $p \in \mathbb{R}$ and the 3D coordinates $[x, y, z]^T \in \mathbb{R}^3$ of plane center. We address the problem defined as follows: In the unknown vegetated terrain, given the initial and target state projection $x_{start}, x_{goal} \in \mathbb{R}^2$, search a feasible and optimal global path consisting of W nodes $\Gamma = \{(\Phi_{S,i})_{i=1:W}\}$. Alone path Γ, the wheeled robot can move from x_{start} to x_{goal}. The path should satisfy: 1) the robot can pass safely along the path; 2) avoiding collision with obstacles along the path; 3) reduce time spent on the move; 4) minimizing the risk of the robot being unable to maintain a stable posture.

The workflow of our entire system is shown in Fig.2. Our navigation algorithm is a two-layer structure including global and local planner. The global planner generates a safe and feasible global path in real time, which is the main content of our research. The global planer contains two parts: PE-RRT* which will be detailly described in Sec.IV-B, and S-Plane Estimation which will be detailly described in Sec.IV-C.

IV. IMPLEMENTATION

Commonly used path planning frameworks require building a priori or real-time explicit map. Traversability analysis of the map is performed before path planning, which costs too much time. To solve this problem, we propose PE-RRT*, a sampling-based path planning algorithm. In PE-RRT*, we sample and analyze directly on the point cloud, avoiding building an explicit traversability map. PE-RRT* algorithm will be described in detail in IV-B. For each node, proprioception and external perception are performed in subsection IV-C1 and IV-C2, and parameter gets estimated in real time in subsection IV-C3. The fusion process of proprioception and external perception to generate S-Plane is in subsection IV-C4. For ease of understanding, we first introduce the relevant mathematical basis in IV-A.

A. MV-GPR

Gaussian process regression (GPR) has been proven to be effective in robot navigation [4] [37]. However, the classical
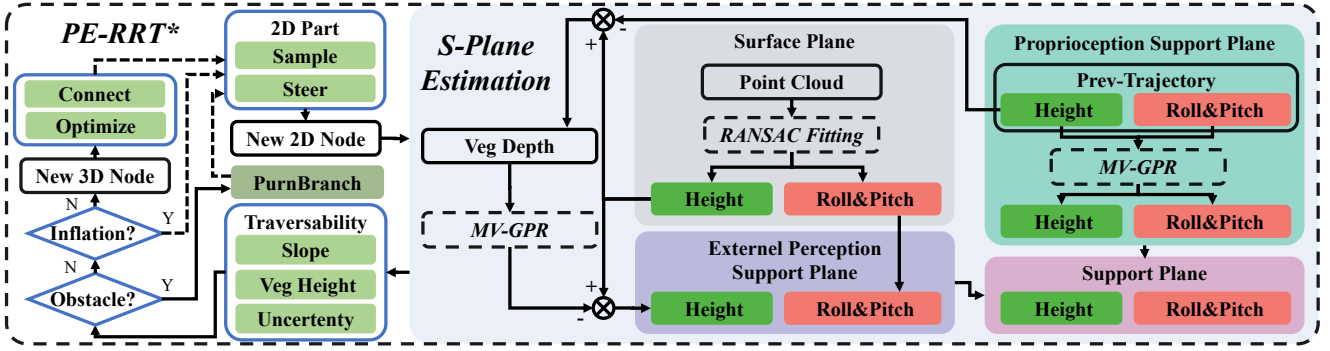
Fig. 3. Algorithm overview. From left to right: To generate the RRT tree, 'Sample' and 'Steer' operations are performed to generate a new 2D node. After combining proprioception and external perception, a S-Plane is obtained. After the 'Obstacle' and 'Infation' check, we can get a new 3D node containing the position (x, y, z), orientation (roll, pitch), and corresponding traversability. After 'Optimize', 'Connect' and 'Purnbranch', the RRT tree is expanded.

GP can't deal with the multi-response problem because of its definition on $\mathbb{R}$. As a result, the correlation between multiple tasks cannot be taken into consideration. To overcome this drawback, [32] proposes multivariate Gaussian process regression (MV-GPR) to perform multi-output prediction. Its precise definition based on Gaussian measures and the existence proof is introduced in [38].

$\boldsymbol{f}$ represents a multivariate Gaussian process with its mean function $\boldsymbol{u} : \mathcal{X} \mapsto \mathbb{R}^d$, kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ and positive semi-definite parameter matrix $\Omega \in \mathbb{R}^{d \times d}$. And Multivariate Gaussian Process (GP) can be denoted as $\boldsymbol{f} \sim \mathcal{MGP}(u, k, \Omega)$. For n pairs of observations $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n, \boldsymbol{x}_i \in \mathbb{R}^p, \boldsymbol{y}_i \in \mathbb{R}^{1 \times d}$, we assume the following model:

$$\boldsymbol{f} \sim \mathcal{MGP}(u, k', \Omega). \tag{1}$$

Different from conventional GPR method, MV-GPR adpots the noise-free regression model, thus $\boldsymbol{y}_i = f(\boldsymbol{x}_i)$ for $i = 1, \cdots, n$. And the noise variance term σ_n^2 is added into the kernel $k' = k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \delta_{ij}\sigma_n^2$, in which $\delta_{ij} = 1$ if $i = j$, otherwise $\delta_{ij} = 0$.

With matrix form $[\boldsymbol{f}(\boldsymbol{x}_1), \cdots, \boldsymbol{f}(\boldsymbol{x}_n)]^T \in \mathbb{R}^{n \times d}$, the joint matrix-variate Gaussian distribution [39] can be represented as:

$$\left[\boldsymbol{f}(\boldsymbol{x}_1)^T, \cdots, \boldsymbol{f}(\boldsymbol{x}_n)^T\right]^T \sim \mathcal{MN}(M, \Sigma, \Omega), \tag{2}$$

where mean matrix $M \in \mathbb{R}^{n \times d}$, covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$, $\Omega \in \mathbb{R}^{d \times d}$ and $X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]^T$ represents the location of training set.

To predict variable $\boldsymbol{f}_* = [f_{*,1}, \cdots, f_{*,m}]^T$ with the location $X_* = [\boldsymbol{x}_{n+1}, \cdots, \boldsymbol{x}_{n+m}]^T$ where m represents the test set number, the joint distribution of the training observations $Y = [\boldsymbol{y}_1^T, \cdots, \boldsymbol{y}_n^T]^T$ and $\boldsymbol{f}_*$ is

$$\begin{bmatrix} Y \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{MN}\left(0, \begin{bmatrix} K'(X, X) & K'(X_*, X)^T \\ K'(X_*, X) & K'(X_*, X_*) \end{bmatrix}, \Omega\right), \tag{3}$$

where K' is the covariance matrix of which the (i,j)-th element $\left[K'\right]_{ij} = k'(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Based on marginalization and conditional distribution theorem [40] [41], the predictive distribution is derived as

$$p(\boldsymbol{f}_* | X, Y, X_*) = \mathcal{MN}\left(\hat{M}, \hat{\Sigma}, \hat{\Omega}\right), \tag{4}$$

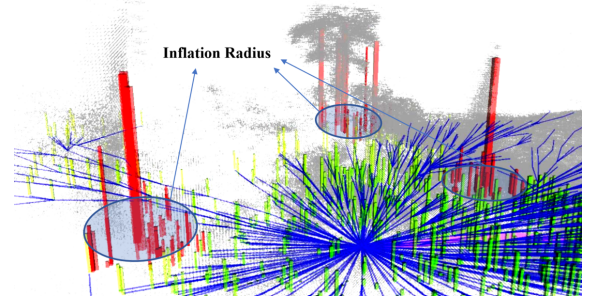$$\hat{M} = K'(X_*, X)^T K'(X, X)^{-1} Y; \tag{5}$$

Fig. 4. Safe inflation radius is integrated into RRT tree generation for enhancing safety. When a node is considered as an obstacle (thick red bar), the area with radius r centered on this node is considered to be the inflation area (thin red bar) of the obstacle.
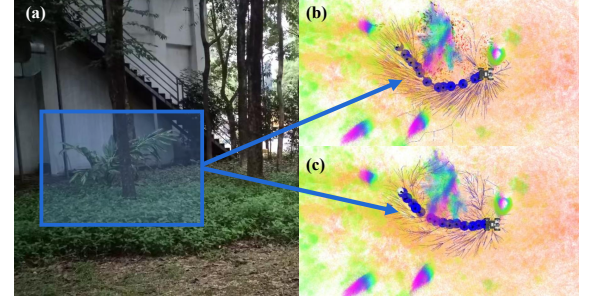

Fig. 5. Generated paths with (b) and without (c) safe inflation radius.

$$\hat{\Sigma} = K'(X_*, X_*) \\ - K'(X_*, X)^T K'(X, X)^{-1} K'(X_*, X); \tag{6}$$

$$\hat{\Omega} = \Omega. \tag{7}$$

According to the above formulas, the expectation and variance are respectively $\mathbb{E}[\boldsymbol{f}_*] = \hat{M}$ and $\text{cov}\left(\text{vec}\left(\boldsymbol{f}_*^T\right)\right) = \hat{\Sigma} \otimes \hat{\Omega}$. When the dimension of the output variable $d = 1$ and covariance matrix $\Omega = I$, it means the process transitions from multivariate to Univariate.

B. PE-RRT*

PE-RRT* algorithm is based on informed-RRT* algorithm [42] which is widely used in the field of path planning, efficiently integrates the process of S-Plane estimation into the RRT tree expansion. The flow of the PE-RRT* is shown in Fig.3 and Alg.1.Especially, we first introduce a safe inflation radius r during the sampling process to enhance safety. When a node is detected as an obstacle, the existing nodes in the RRT tree with this node as the center and r as the radius will

be removed shown as the "red bar" in Fig.4. When the node is not an obstacle but within the range of r as the radius with the existing obstacles as the center, it will be divided into the inflation layer and will not be added to the RRT tree. The advantage of this method is that there is no need to obtain the locations of all obstacles in advance. As shown in Fig.5, when other parameters remain unchanged, the path with $r = 0.5m$ added is farther from the obstacle than the path without the radius which can enhance safety.

Some new subfunctions presented in Alg.1 are described as follows while subfunctions common to the informed-RRT* algorithm can be found in [42] [4]. Surf-Plane Φ_{Surf}, Pro-Plane Φ_{Pro}, EP-Plane Φ_{EP} and S-Plane Φ_S all consist of a 3D plane center point, roll angle and pitch angle.

- **Proprioception**$(\zeta, \bar{\mathbf{x}}_{new})$: Given the robot's Prev-Trajectory ζ and the node's 2D coordinate $\bar{\mathbf{x}}_{new}$, the Pro-Plane is returned. The implementation will be discussed in detail in part IV-C1.
- **ExPerception**$(\Phi_{Surf}, \zeta, \bar{\mathbf{x}}_{new})$: Given the robot's Prev-Trajectory ζ, the Surf-Plane Φ_S and the node's 2D coordinate $\bar{\mathbf{x}}_{new}$, the EP-Plane is returned. The implementation will be discussed in detail in part IV-C2.
- **SupportFuse**(Φ_{EP}, Φ_{Pro}): Given the Pro-Plane Φ_{Pro} and EP-Plane Φ_{EP}, the fused S-Plane is returned. The implementation will be discussed in detail in part IV-C4.
- **ObsCheck**(Φ_{Surf}, Φ_S): Given the S-Plane Φ_S and Surf-Plane Φ_{Surf}, we use the height h of vegetation as the criterion for judging obstacles, and it can be obtained by $h = z_{Surf} - z_S$, where z_S represents the height of Φ_S and z_{Surf} represents the height of Φ_{Surf}. When the vegetation in an area is too high, there are usually rigid trees, which can cause collisions. So we define a threshold value h_{crit}, when $h > h_{crit}$, the node is considered to be obstacle, the function returns "True", otherwise "False".
- **InflationCheck**(Φ_S, Ξ_{Obs}): Given the S-Plane Φ_S and obstacle set Ξ_{Obs}, if for any element in Ξ_{Obs}, its Euclidean distance in 2D $x - y$ space from Φ_S is greater than the inflation radius r, then the function returns "True", otherwise "False".
- **PurnBranch**(T, Φ_S): Given the RRT tree T and S-Plane Φ_S, for each node in T, if its Euclidean distance in 2D $x - y$ space from Φ_S is smaller than r, the node and its branch will be deleted.
- **TraEvaluation**(Φ_S): Given the S-Plane Φ_S, the traversability is obtained from the slope and the uncertainty of Φ_S. The implementation will be discussed in detail in part IV-C4.

C. S-Plane Estimation

In order to generate S-Plane, we perform proprioception and external perception on the node to generate Pro-Plane and EP-Plane respectively.

1) Proprioception:

The Proprioception of the robot usually depends on the sensors of the robot itself (wheel speedometer, IMU, etc.), but usually causes cumulative errors. In our experiments, FAST-LIO2.0 [43] is adopted as an odometer, in which information of IMU and LiDAR is fused to improve the positioning accuracy.

Algorithm 1: PE-RRT*$(\mathcal{N}_{start}, \mathcal{N}_{goal}, k)$

---

1 $V \leftarrow \{\mathcal{N}_{start}\}, E \leftarrow \emptyset, \sigma^* \leftarrow \emptyset, \Omega_{goal} \leftarrow \emptyset, \Xi_{Obs} \leftarrow \emptyset;$
2 $T = (V, E);$
3 **for** $i = 1$ to k **do**
4   **if** $S^* \neq \emptyset$ **then**
5 $\bar{\mathbf{x}}_{rand} \leftarrow$ **SampleEllipsoid**$();$
6   **else**
7 $\bar{\mathbf{x}}_{rand} \leftarrow$ **RandomSample**$();$
8 $\mathcal{N}_{nearest} \leftarrow$ **FindNearest**$(T, \bar{\mathbf{x}}_{rand});$
9 $\bar{\mathbf{x}}_{nearest} \leftarrow$ **ProjectToPlane**$(\mathbf{Pos}(\mathcal{N}_{nearest}));$
10 $\bar{\mathbf{x}}_{new} \leftarrow$ **Steer**$(\bar{\mathbf{x}}_{nearest}, \bar{\mathbf{x}}_{rand});$
11 $\Phi_{Surf} \leftarrow$ **FitPlane**$(\bar{\mathbf{x}}_{new});$
12 $\Phi_{Pro} \leftarrow$ **Proprioception**$(\zeta, \bar{\mathbf{x}}_{new});$
13 $\Phi_{EP} \leftarrow$ **ExPerception**$(\Phi_{Surf}, \zeta, \bar{\mathbf{x}}_{new});$
14 $\Phi_S \leftarrow$ **SupportFuse**$(\Phi_{EP}, \Phi_{Pro});$
15   **if** !**ObsCheck**(Φ_S, Φ_{Surf}) **then**
16   **if** !**InflationCheck**(Φ_S, Ξ_{Obs}) **then**
17 $\mathcal{N}_{new} \leftarrow$ **TraEvaluation**$(\Phi_S);$
18   **else**
19 $T \leftarrow$ **PurnBranch**$(T, \Phi_S);$
          $\Xi_{Obs} \leftarrow \Xi_{Obs} \cup \{\Phi_S\};$
20   **if** $\mathcal{N}_{new} \neq \emptyset$ **and** $\mathbf{Pos}(\mathcal{N}_{new}) \in X_{trav}$ **then**
21 $\Omega_{near} \leftarrow$ **FindNeighbors**$(V, \mathcal{N}_{new});$
22   **if** $\Omega_{near} \neq \emptyset$ **then**
23 $\mathcal{N}_{parent} \leftarrow$ **FindParent**$(\Omega_{near}, \mathcal{N}_{new});$
24 $V \leftarrow V \cup \{\mathcal{N}_{new}\};$
25 $E \leftarrow E \cup \{(\mathcal{N}_{parent}, \mathcal{N}_{new})\};$
26 $T \leftarrow (V, E);$
27 $T \leftarrow$ **Rewire**$(T, \Omega_{near}, \mathcal{N}_{new});$
28   **if** **InGoalRegion**$(\mathcal{N}_{new})$ **then**
29 $\Omega_{goal} \leftarrow \Omega_{goal} \cup \{\mathcal{N}_{new}\};$
30 $S^* \leftarrow$ **GeneratePath**$(\Omega_{goal});$
31 **return** S^*

In this module, MV-GPR is used for estimate Pro-Plane Φ_{Pro} of new node. To reduce the computational expense, the training size has to be limited [31]. We record the position $\{x_i, y_i, z_{Pro,i}\}_{i=1:N}$ and pose $\{r_{Pro,i}\}_{i=1:N}$, $\{p_{Pro,i}\}_{i=1:N}$ of the previous N steps of the robot. The training input data comprises the horizontal position of the prev-trajectory $X = \left[[x_1, y_1]^T, \cdots, [x_N, y_N]^T\right]^T \in \mathbb{R}^{N \times 2}$, while the output data is defined as $Y_{Pro} = \left[[z_{Pro,1}, r_{Pro,1}, p_{Pro,1}]^T, \cdots, [z_{Pro,N}, r_{Pro,N}, p_{Pro,N}]^T\right]^T \in \mathbb{R}^{N \times 3}$. Note that in order to ensure that the yaw angle make no difference to the slope, we extract roll r and pitch p from rotation matrix R^i, which can be obtained from odometer. And $Y_{Pro,i} = [z_{Pro,i}, r_{Pro,i}, p_{Pro,i}]^T$ for $i = 1 \cdots N$, where

$$p_{Pro,i} = \mathrm{atan2}\left(R_{31,i}, \sqrt{(R_{32,i})^2 + (R_{33,i})^2}\right), \quad (8)$$

$$r_{Pro,i} = \mathrm{atan2}\left(-\frac{R_{32,i}}{\cos(p_{Pro,i})}, \frac{R_{33,i}}{\cos(p_{Pro,i})}\right). \quad (9)$$

Quantifying uncertainty is crucial for assessing the accuracy of plane estimations, which will be discribed in detail in IV-C4. For proprioception, the uncertainty $\sigma_{n,Pro}^2$ of the training set is from TF, which is set to be a constant value in our experiment.

In MV-GPR, the covariance matrix Σ depends on inputs and the kernel function k. Compared to other kernel functions (such as linear, rational quadratic and Matern [44]), squared exponential (SE) kernel is more commonly used due to its simple form and many properties such as smoothness and integrability with other functions. The kernel is defined as:

$$k_{SE}\left(x, x^{'}\right) = s_f^2 \exp^{-\frac{\left\| x-x^{'} \right\|_2^2}{2l^2}}, \qquad (10)$$

where s_f^2 is overall variance and l is kernel length scale. Due to the properties of SE kernel, when the distance between inputs (Euclidean distance) is farther, the variable z, r and p variance becomes larger, which means that the Pro-Plane estimated by proprioception becomes more uncertain.

We take the 2D coordinates $\overline{\mathbf{x}}_{new} = [x_*, y_*]$ of a single node as the input of the test set, and $\{X, Y_{Pro}\}$ as the training set, according to formula 5 6 7, we can get the prediction of height $\hat{z}_{Pro,*}$ and pose $\hat{r}_{Pro,*}$, $\hat{p}_{Pro,*}$ of the node. Thus, we can get the estimation of Pro-Plane is $\hat{\Phi}_{Pro} = \{x_*, y_*, \hat{z}_{Pro,*}, \hat{r}_{Pro,*}, \hat{p}_{Pro,*}\}$. The height variance $\sigma_{Pro,z,*}^2$, roll variance $\sigma_{Pro,r,*}^2$ and pitch variance $\sigma_{Pro,p,*}^2$ can be obtained from the Kronecker product of $\hat{\Sigma}$ and $\hat{\Omega}$.

2) External Perception: Compared with proprioception, external perception relies on point cloud map generated by LiDAR. To get a new EP-Plane Φ_{EP}, we first fit the Surf-Plane Φ_{Surf} corresponding to the 2D node. Compared to the SVD method used in PF-RRT* [4], we adopt RANSAC method to fit a plane, which can avoid the influence of tall rigid obstacles (such as tall trees, large stones) on the slope of the fitted Surf-Plane.

For slope estimation of EP-Plane, we consider roll and pitch of EP-Plane and Surf-Plane to be the same: $r_{EP,*} = r_{Surf,*}$, $p_{EP,*} = p_{Surf,*}$, due to the assumption of uniformity and continuity of penetrable vegetation. And so is the corresponding variance: $\sigma_{EP,r,*}^2 = \sigma_{Surf,r,*}^2$, $\sigma_{EP,p,*}^2 = \sigma_{Surf,p,*}^2$. The variance of $\sigma_{Surf,r,*}^2$ and $\sigma_{Surf,p,*}^2$ obtained by the empirical formula:

$$\sigma_{Surf,r,*}^2 = \kappa_r \frac{\Sigma_{k=1}^K \left[\boldsymbol{n} \cdot \left(\boldsymbol{x}_{\Phi,*}^k - \boldsymbol{x}_{Surf,*} \right) \right]^2}{K-1}, \qquad (11)$$

$$\sigma_{Surf,p,*}^2 = \kappa_p \frac{\Sigma_{k=1}^K \left[\boldsymbol{n} \cdot \left(\boldsymbol{x}_{\Phi,*}^k - \boldsymbol{x}_{Surf,*} \right) \right]^2}{K-1}, \qquad (12)$$

where the Surf-Plane envelops K points on the point cloud map, the k-th point's 3D coordinate is $\boldsymbol{x}_{\Phi,*}^k \in \mathbb{R}^3$, and the plane center is $\boldsymbol{x}_{Surf,*} \in \mathbb{R}^3$. $\boldsymbol{n}$ represents the normal vector of Surf-Plane. κ_r and κ_p are constant coefficient.

And the estimation of z_{EP} is more complicated. Vegetation depth H is introduced as an intermediate variable for estimating z_{EP}. Take Prev-Trajectory $X = [[x_1, y_1]^T, \cdots, [x_N, y_N]^T$ as the inputs and corresponding vegetation depth $Y = [H_1, \cdots, H_N]^T$ as the outputs. For the i-th vegetation depth H_i, it can be obtained as $H_i = z_{Surf,i} - z_{Pro,i}$, where $z_{Surf,i}$ is the Surf-Plane height. Its uncertainty $\sigma_{H,i}^2 = \sigma_{Pro,z,i}^2 + \sigma_{Surf,z,i}^2$ contains the uncertainty $\sigma_{Pro,z,i}^2$ from TF and the uncertainty $\sigma_{Surf,z,i}^2$ from Surf-Plane due to the independence assumption. And $\sigma_{Surf,z,i}^2$ is defined as:

$$\sigma_{Surf,z,i}^2 = \frac{\sum_{k=1}^K \left(z_{\Phi,i}^k - z_{Surf,i} \right)^2}{K-1}, \qquad (13)$$
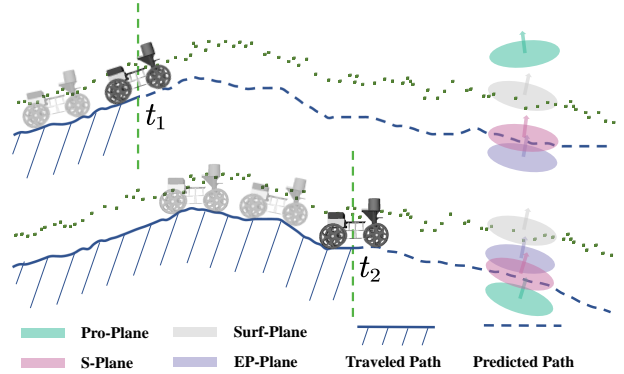

Fig. 6. As the robot moves, the estimated planes for the node changes.

where the height of the k-th point is $z_{\Phi,i,k}$, and the height of the plane center is $z_{Surf,i}$ for i-th Surf-Plane.

Thus the vegetation depth $\hat{H}_*$ of a new node and its variance $\sigma_{H,*}^2$ can be obtained based on equation 5 6 7. And for the new node, the height of EP-Plane is $\hat{z}_{EP,*} = z_{Surf,*} - \hat{H}_*$, and its variance $\sigma_{EP,*}^2 = \sigma_{H,*}^2 + \sigma_{Surf,z,*}^2$ consists of two parts: covariance generated from GPR $\sigma_{H,*}^2$; covariance of Surf-Plane $\sigma_{Surf,z,*}^2$. Note that the $\sigma_{Surf,z,*}^2$ calculation of Surf-Plane is consistent with formula 13.

3) Parameter Estimation: In the process of the robot moving forward, in order to ensure the accuracy of MV-GPR, it is necessary to estimate its parameters in real time. For proprioception which contains a 3-variate Gaussian process, the estimated parameters include kernel matrix parameters s_f^2, l^2, covariance matrix $\Omega = \Phi \Phi^T$, where for $\psi_{11}, \psi_{22}, \psi_{33}, \phi_{31}, \phi_{21}, \phi_{32} \in \mathbb{R}$:

$$\Phi = \begin{bmatrix} e^{\psi_{11}} & 0 & 0 \\ \phi_{21} & e^{\psi_{22}} & 0 \\ \phi_{31} & \phi_{32} & e^{\psi_{33}} \end{bmatrix}, \qquad (14)$$

to ensure the positive definiteness of the matrix. We use the maximum likelihood method to estimate the parameters. For negative log marginal likelihood

$$\begin{aligned} \mathcal{L} = & \frac{nd}{2} \ln(2\pi) + \frac{nd}{2} \ln \det \left(K + \sigma_n^2 \right) + \frac{d}{2} \ln \det (\Omega) \\ & + \frac{1}{2} tr \left(\left(K + \sigma_n^2 \right)^{-1} Y \Omega^{-1} Y^{\mathrm{T}} \right). \end{aligned} \qquad (15)$$

The derivatives of the negative log marginal likelihood with respect to parameter s_f^2, l^2, ψ_{ii} and ϕ_{ij} can be obtained. Formula derivation reference [32].

For external perception witch is Univariate Gaussian process, we only need to estimate the kernel s_f^2, l^2.

4) Plane Fusion: The vegetation height varies in different environments. On the grassland, the vegetation is usually short, and the point cloud returned by the LiDAR is relatively smooth; while in the bushes, the vegetation is usually high and uneven, and the point cloud is rougher; And for tall trees, it is considered to be impassable. As shown in Fig.6, for Pro-Plane, the source of variance is mainly the Euclidean distance, it can more accurately estimate the terrain conditions of the nearby area, but has a poor estimation for the far terrain; for EP-Plane, the source of variance includes both the distance and the the surface condition of the point cloud. In order to accurately estimate the support ground in different environments, the variance is used as a weight to fuse Pro-Plane and EP-Plane. We define the weight as follows:

$$w_{[\cdot]} = \sigma_{EP,[\cdot],*}^2 / \left(\sigma_{EP,[\cdot],*}^2 + \sigma_{Pro,[\cdot],*}^2 \right), \qquad (16)$$

where the symbol $[\cdot]$ here is to refer to r, p and z for simplifying the formula. Thus the estimation of S-Plane $\hat{\Phi}_{S,*} = \{x_*, y_*, \hat{z}_{S,*}, \hat{r}_{S,*}, \hat{p}_{S,*}\}$ can be obtained as:

$$[\hat{\cdot}]_{S,*} = w_{[\cdot]}[\hat{\cdot}]_{Pro,*} + \left(1 - w_{[\cdot]}\right)[\hat{\cdot}]_{EP,*}. \qquad (17)$$

When the point cloud in the area where the robot is driving is relatively cluttered, w_z, w_r and w_p will become larger, and the robot will trust proprioception more; otherwise, the robot will trust external perception more, as shown in Fig.6. In this way, the robustness and safety of the algorithm can be enhanced. Note that when the vegetation height exceeds the threshold h_{crit}, it is considered as an obstacle, and the RRT tree will delete the node and nearby nodes.

In the process of RRT tree generation, proper introduction of traversability can improve the safety and stability of the path. When the vehicle is driving, we often pay less attention to the road conditions in small areas (pebbles, clods, etc.). Instead, we are more concerned about the information of the slope s, the uncertainty ε and the vegetation height h. The robot travels on terrain with shallow vegetation and small slope, so it is less likely to slip. The slope s can be obtained from roll and pitch of the S-Plane:

$$s = \arccos\left(\cos\left(\hat{r}_{S,*}\right)\cos\left(\hat{p}_{S,*}\right)\right), \qquad (18)$$

and ε can be obtained from $\sigma_{S,z,*}^2$, $\sigma_{S,r,*}^2 \sigma_{S,p,*}^2$:

$$\varepsilon = \sigma_{S,z,*}^2 + \mu * \left(\sigma_{S,r,*}^2 + \sigma_{S,p,*}^2\right), \qquad (19)$$

where μ is a constant coefficient. Thus, the traversability τ can be described as:

$$\tau = \alpha_1 \frac{s}{s_{crit}} + \alpha_2 \frac{\varepsilon}{\varepsilon_{crit}} + \alpha_3 \frac{h}{h_{crit}}, \qquad (20)$$

where α_1, α_2, and α_3 are weights which sum to 1. s_{crit}, ε_{crit}, and h_{crit}, which represent the maximum allowable slope, uncertainty, and vegetation height respectively, are critical values that may cause collision or rollover. In PE-RRT*, cost includes Euclidean distance d from parent node and traversability: $Cost = d/(1 - \tau)$. When the RRT tree is expanded, the nodes with lower cost will be selected first. With the increase of sampling points, the generated path will gradually tends to be optimal.

V. EXPERIMENTS

We conduct real-world experiments to verify the effectiveness of our work utilizing the physical platform illustrated in Fig.7. Our algorithm works under ROS Melodic operating system, generating the global path at $2Hz$ and local path at $10hz$ by NMPC method using CasADI. The resolution of global map is set to $2cm$, the radius for plane estimation is $15cm$, and the inflation radius is set to $25cm$. Note that the starting point is set to be the origin of the map, i.e $x_{start} = [0,0]^{\mathrm{T}}$.

We conduct experiments in three different scenarios. In the first scenario, the robot needs to traverse across a hillside with grass and trees. And the snapshots of our algorithm in the main scenario are shown in Fig 8. The robot chooses
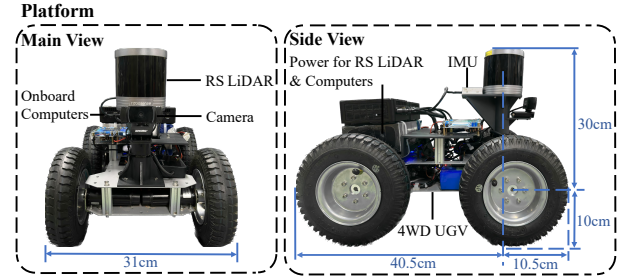

Fig. 7. Robot platform for the experiment. A four-wheel differential-drive mobile robot equipped with RS-Helios 5515 and IMU. RS-Helios 5515 is a 32-beam LiDAR, which boasts a 70° ultra-wide vertical field of view.The camera is only for front view in the video. Two Intel@NUC with an i5 2.4GHz CPU and 16GB memory are used to run the planning algorithm and the SLAM algorithm, respectively. And a battery is installed to power both two computers and the LiDAR.

TABLE I: Comparison between each method

Algorithms	Path len(m)	Safety deg(m)	Comp time(s)	Cons time(s)	Speed dev(m/s)
PE-RRT*	**11.869**	**0.853**	**0.5**	**38**	**0.0069**
PF-RRT*	13.006	0.798	0.5	43.8	0.0343
Pro-RRT*	Collision	0	0.5	-	0.0144
RRT*+PrevMap	18.761	0.691	1.3-2.5	74.3	0.0254

to generate path where the Vegetation Height is smaller, as shown in Fig.8(a). If the robot detects an obstacle (long red bar), it navigates avoiding the obstacle and continues moving forward, as depicted in Fig.8(b). Once the robot enters a safe area with little grass cover, it engages in longer-range global path planning preferring gentle slopes of the supporting plane, as illustrated in Fig8(c). In the end, the robot reaches the target point and stops, as depicted in Fig8(d). Note that compared to the height of the camera, the height of vegetation ranges from from $0.1m$ to $0.2m$, which can seriously block the image captured by the camera as shown in Fig.8(a)(d), which could lead to the failure of the vision-based navigation and obstacle avoidance methods like [23]–[26]. For evaluation, we compare ours with 3 baseline approaches In this scenario:

1) PF-RRT* [4]: RRT* in which each node fits the plane directly on the point cloud map.
2) Pro-RRT*: RRT* in which each node estimates the S-Plane directly based on the Prev-trajectory.
3) RRT*+PrevMap: Estimate the S-Plane based on MV-GPR to generate the previous traversability map. Based on the map, RRT* is used to obtain the global path.

Each algorithm generates trajectories with different colors is shown in Fig.9. And to intuitively compare the performance of different algorithms, we adopt the following indicators to compare the four algorithms:

- Path len: length of the path from the start to the end.
- Safety deg: minimum distance to the obstacle.
- Cons time: consuming time from start to goal.
- Comp time: computation time to generate a global path.
- Speed dev: speed deviation of the robot, reflecting the stability of the robot during navigation.

And the results of the evaluation are presented in Table I. Since the supporting ground cannot be estimated, PF-RRT* chooses to avoid vegetation, which increases its "Path len" and "Comp time". And as shown in Fig.9, on uneven terrain where the vegetation is often more complex, PF-RRT* cannot recognize

(a) **(b)** **(c)** **(d)**

Global Path — Prev-Trajectory — Support Plane — Traversability — Vegetation Estimation — Obstacle — Local Path — Target Point
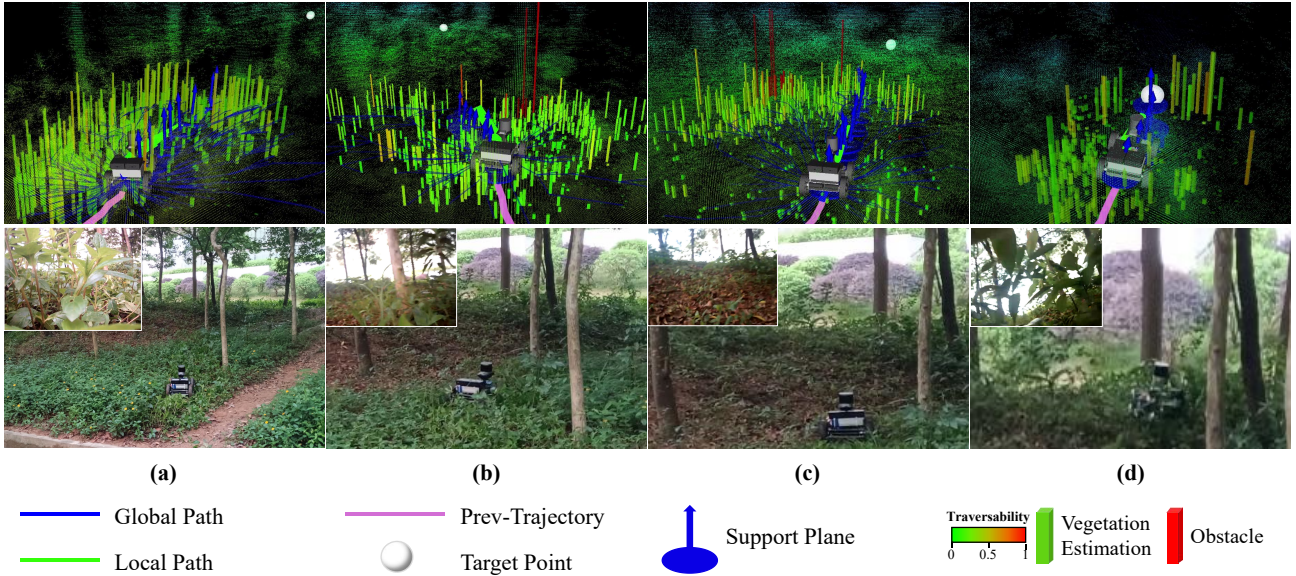
Fig. 8. The blue line indicates the global path generated by PE-RRT*, and the plane with normal vectors indicates the estimation of the support plane at the global path, with passability from blue to red. The green line demonstrates the local path planning generated by NMPC based on the global path, and the pink line shows the Prev-Trajectory used for MV-GPR training. The rectangles correspond to the vegetation estimates at the locations sampled by PE-RRT* with traversability from blue to red, and the red indicate the obstacles and the sampling points within their inflation radius. The figures (a), (b), (c) and (d) show the strategy of our method in different states.
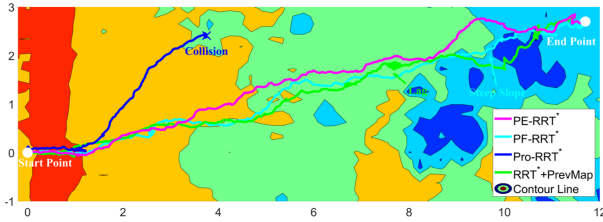

Fig. 9. The trajectory of the four approaches. White dots represent the start and target points, the background is the contour line of global map whose color changes from red to blue with height increasing.
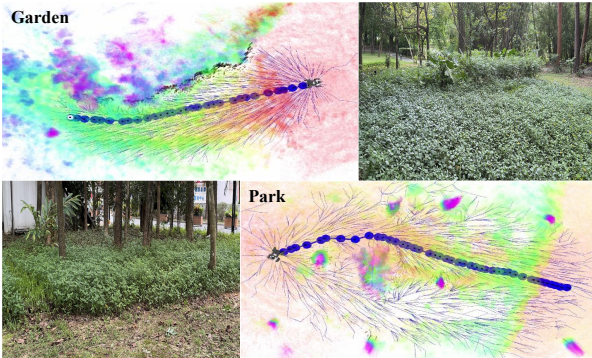

Fig. 10. Experiments in the garden and park. Our algorithm gives the optimal global path of these scenarios with the global point cloud map in the background.

steep slope which can cause danger. By comparison, PE-RRT* is more robust due to the uncertainty-weight based fusion of proprioception and external perception. Pro-RRT* fails and collides with the tree since it does't use the point cloud information to avoid obstacles. RRT*+PrevMap occurs with several lags due to the construction of the explicit traversability map, which is time consuming, in contrast, PE-RRT* only takes 63.27% of the time. Ours efficiently and accurately estimates the height and slope of the node, ensuring the asymptotic optimality of global path generation and smooth obstacle avoidance.

To demonstrate the generalizability of our algorithm, we conduct experiments in other scenarios with same parameters, as shown in Fig 10 (more details at [2]).

VI. CONCLUSIONS AND DISCUSSIONS

This paper proposes a novel path planning method (PE-RRT*) on vegetated terrain based on sampling tree and support plane estimation, in which safe inflation radius is added into RRT tree to avoid collision. Proprioception and external perception are fused to generate support plane based on the uncertainty weight. In addition, we compare our method with three methods (PF-RRT*, Pro-RRT* and RRT*+PrevMap) in real scenarios. The experimental results show that our method is safer and more efficient than other methods in global path planning.

Discussions: The Plane-Estimation algorithm is based on incremental point cloud map and previous trajectory. In addition to LiDAR, depth camera and solid-state LiDAR can also be used to build the map. Also, UWB-based or vision-based positioning algorithms can be used to record historical trajectories. In vegetation environments where sensors can be easily blocked, these two methods have lower accuracy or are easily lost compared to LiDAR-inertial odometry. However, due to the properties of the SE kernel function, the supporting ground and vegetation layer need to satisfy the assumptions of smoothness and continuity. Therefore, the effect of the algorithm will be better in areas where the vegetation and terrain height are relatively uniform. When the terrain is steeper, or the vegetation becomes more complex, the accuracy of support plane estimation decreases.

REFERENCES

[1] D. Chen, A. Xiao, M. Zou, W. Chi, J. Wang, and L. Sun, "Gvd-exploration: An efficient autonomous robot exploration framework based on fast generalized voronoi diagram extraction," *arXiv preprint arXiv:2309.06041*, 2023.

[2] Video: https://youtu.be/EeZ-JXaiXuw.

[2] Z. Jian, Z. Yan, X. Lei, Z. Lu, B. Lan, X. Wang, and B. Liang, "Dynamic control barrier function-based model predictive control to safety-critical obstacle-avoidance of mobile robot," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3679–3685.

[3] Y. Chen, Z. Xu, Z. Jian, G. Tang, L. Yang, A. Xiao, X. Wang, and B. Liang, "Quadruped guidance robot for the visually impaired: A comfort-based approach," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 12 078–12 084.

[4] 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.

[5] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A.-A. Agha-Mohammadi, "Step: Stochastic traversability evaluation and planning for safe off-road navigation," *arXiv preprint arXiv:2103.02828*, 2021.

[6] Z. Xu, Y. Chen, Z. Jian, X. Wang, and B. Liang, "Hybrid trajectory optimization for autonomous terrain traversal of articulated tracked robots," *arXiv preprint arXiv:2306.02659*, 2023.

[7] X. Chen, S. Xu, J. Han, H. Fu, X. Pi, C. Joe-Wong, Y. Li, L. Zhang, H. Y. Noh, and P. Zhang, "Pas: Prediction-based actuation system for city-scale ridesharing vehicular mobile crowdsensing," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 3719–3734, 2020.

[8] S. Xu, X. Chen, X. Pi, C. Joe-Wong, P. Zhang, and H. Y. Noh, "ilocus: Incentivizing vehicle mobility to optimize sensing distribution in crowd sensing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 8, pp. 1831–1847, 2019.

[9] X. Chen, S. Xu, H. Fu, C. Joe-Wong, L. Zhang, H. Y. Noh, and P. Zhang, "Asc: Actuation system for city-wide crowdsensing with ride-sharing vehicular platform," in *Proceedings of the Fourth Workshop on International Science of Smart City Operations and Platforms Engineering*, 2019, pp. 19–24.

[10] Z. Li, F. Man, X. Chen, B. Zhao, C. Wu, and X. Chen, "Tract: Towards large-scale crowdsensing with high-efficiency swarm path planning," in *Adjunct Proceedings of the 2022 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2022 ACM International Symposium on Wearable Computers*, 2022, pp. 409–414.

[11] X. Chen, A. Purohit, C. R. Dominguez, S. Carpin, and P. Zhang, "Drunkwalk: Collaborative and adaptive planning for navigation of micro-aerial sensor swarms," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, 2015, pp. 295–308.

[12] Z. Jian, F. Zhu, and L. Tang, "Research on human body recognition and position measurement based on adaboost and rgb-d," in *2020 39th Chinese Control Conference (CCC)*. IEEE, 2020, pp. 5184–5189.

[13] J. Zha, L. Han, X. Dong, and Z. Ren, "Privacy-preserving push-sum distributed cubature information filter for nonlinear target tracking with switching directed topologies," *ISA transactions*, vol. 136, pp. 16–30, 2023.

[14] X. Chen, C. Ruiz, S. Zeng, L. Gao, A. Purohit, S. Carpin, and P. Zhang, "H-drunkwalk: Collaborative and adaptive navigation for heterogeneous mav swarm," *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 2, pp. 1–27, 2020.

[15] P. Fankhauser and M. Hutter, "A universal grid map library: Implementation and use case for rough terrain navigation," *Robot Operating System (ROS) The Complete Reference (Volume 1)*, pp. 99–120, 2016.

[16] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 709–715.

[17] Y. Ren, Y. Cai, F. Zhu, S. Liang, and F. Zhang, "Rog-map: An efficient robocentric occupancy grid map for large-scene and high-resolution lidar-based motion planning," *arXiv preprint arXiv:2302.14819*, 2023.

[18] Z. Yan, X. Wu, Z. Jian, B. L. X. Wang, and B. Liang, "Rh-map: Online map construction framework of dynamic objects removal based on region-wise hash map structure," *arXiv preprint arXiv:2307.00599*, 2023.

[19] X. A. Wu, T. M. Huh, R. Mukherjee, and M. Cutkosky, "Integrated ground reaction force sensing and terrain classification for small legged robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1125–1132, 2016.

[20] C. Ordonez, R. Alicea, B. Rothrock, K. Ladyko, J. Nash, R. Thakker, S. Daftry, M. Harper, E. Collins, and L. Matthies, "Characterization and traversal of pliable vegetation for robot navigation," in *Proceedings of the 2018 International Symposium on Experimental Robotics*. Springer, 2020, pp. 293–304.

[21] 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)*. IEEE, 2022, pp. 9447–9453.

[22] C. Wellington, A. Courville, and A. Stentz, "A generative model of terrain for autonomous navigation in vegetation," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1287–1304, 2006.

[23] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter, "Fast traversability estimation for wild visual navigation," *arXiv preprint arXiv:2305.08510*, 2023.

[24] A. J. Sathyamoorthy, K. Weerakoon, T. Guan, J. Liang, and D. Manocha, "Terrapn: Unstructured terrain navigation using online self-supervised learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 7197–7204.

[25] M. G. Castro, S. Triest, W. Wang, J. M. Gregory, F. Sanchez, J. G. Rogers, and S. Scherer, "How does it feel? self-supervised costmap learning for off-road vehicle traversability," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 931–938.

[26] A. Polevoy, C. Knuth, K. M. Popek, and K. D. Katyal, "Complex terrain navigation via model error prediction," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 9411–9417.

[27] M. Daily, S. Medasani, R. Behringer, and M. Trivedi, "Self-driving cars," *Computer*, vol. 50, no. 12, pp. 18–23, 2017.

[28] J. Betz, A. Wischnewski, A. Heilmeier, F. Nobis, T. Stahl, L. Hermansdorfer, B. Lohmann, and M. Lienkamp, "What can we learn from autonomous level-5 motorsport?" in *9th International Munich Chassis Symposium 2018: chassis. tech plus*. Springer, 2019, pp. 123–146.

[29] M. Bjelonic, N. Kottege, T. Homberger, P. Borges, P. Beckerle, and M. Chli, "Weaver: Hexapod robot for autonomous navigation on unstructured terrain," *Journal of Field Robotics*, vol. 35, no. 7, pp. 1063–1079, 2018.

[30] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

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

[32] Z. Chen, B. Wang, and A. N. Gorban, "Multivariate gaussian and student-t process regression for multi-output prediction," *Neural Computing and Applications*, vol. 32, pp. 3005–3028, 2020.

[33] X. Chen, S. Xu, X. Liu, X. Xu, H. Y. Noh, L. Zhang, and P. Zhang, "Adaptive hybrid model-enabled sensing system (hmss) for mobile fine-grained air pollution estimation," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 1927–1944, 2020.

[34] X. Chen, X. Xu, X. Liu, S. Pan, J. He, H. Y. Noh, L. Zhang, and P. Zhang, "Pga: Physics guided and adaptive approach for mobile fine-grained air pollution estimation," in *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, 2018, pp. 1321–1330.

[35] J. Ren, Y. Xu, Z. Li, C. Hong, X.-P. Zhang, and X. Chen, "Scheduling uav swarm with attention-based graph reinforcement learning for ground-to-air heterogeneous data communication," in *Adjunct Proceedings of the 2023 ACM International Joint Conference on Pervasive and Ubiquitous Computing & the 2023 ACM International Symposium on Wearable Computing*, 2023, pp. 670–675.

[36] M. Azam, X. Chen, and S. Xu, "Incentivizing mobility of multi-agent vehicle swarms with deep reinforcement learning for sensing coverage optimization," in *Adjunct Proceedings of the 2022 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2022 ACM International Symposium on Wearable Computers*, 2022, pp. 397–402.

[37] T. Shan, J. Wang, B. Englot, and K. Doherty, "Bayesian generalized kernel inference for terrain traversability mapping," in *In Proceedings of the 2nd Annual Conference on Robot Learning*, 2018.

[38] Z. Chen, J. Fan, and K. Wang, "Multivariate gaussian processes: definitions, examples and applications," *METRON*, pp. 1–11, 2023.

[39] A. P. Dawid, "Some matrix-variate distribution theory: notational considerations and a bayesian application," *Biometrika*, vol. 68, no. 1, pp. 265–274, 1981.

[40] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia, "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario," *Sensors and Actuators B: Chemical*, vol. 129, no. 2, pp. 750–757, 2008.

[41] S. Zhu, K. Yu, and Y. Gong, "Predictive matrix-variate t models," *Advances in neural information processing systems*, vol. 20, 2007.

[42] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2997–3004.

[43] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[44] C. E. Rasmussen and C. K. Williams, "Gaussian processes for machine learning, vol. 1," 2006.