Digital Analytics and Robotics for Sustainable Forestry

CL4-2021-DIGITAL-EMERGING-01
Grant agreement no: 101070405

DELIVERABLE 3.4

Field Results: Unified Mission Planning

Due date: month 33 (May 2025)
Deliverable type: R
Lead beneficiary: NTNU

Dissemination Level: PUBLIC

Contents

1 Introduction

Navigation in forest environments poses significant challenges due to their inherently complex, unstructured, and dynamic nature. Dense vegetation, uneven terrain, and limited visibility impose severe constraints on robot mobility, and path planning, thus demanding advanced multi-agent mission planning. By distributing tasks across a heterogeneous team of robots, the system can exploit parallelization and spatial distribution to improve coverage efficiency, operational resilience, and mission flexibility.

However, effective coordination in multi-robot systems is complicated by the platform-specific challenges faced by different robotic agents. Aerial robots must contend with obstacle-dense canopy layers, narrow vertical corridors, and limited flight time. In contrast, legged robots operating on the ground face highly variable terrain topologies and frequent obstacles. These constraints necessitate not only diverse capabilities but also planning strategies that are aware of and adapted to the unique operational envelopes of each platform.

As a result, this document presents a unified planning strategy for multi-agent exploration while accounting for each specific system's capabilities, limitations, and operational contexts. Such a system is key to enabling effective, collaborative deployment in forest environments.

This document is organized as follows. First, an overview of a cross-platform planning architecture, designed to support mission-level coordination across heterogeneous teams is proposed. It includes a detailed breakdown of robot-specific planner operations, tailored to the aerial and legged platforms. Then, we present results from field deployments of a legged–aerial marsupial system, showcasing the planning framework in real-world forest conditions.

2 Unified Planning Architecture

2.1 Unified Exploration Architecture

Exploring unknown environments autonomously is a critical capability for robots operating in GPS-denied, complex environments such as underground tunnels, collapsed buildings, or forest. GBPlanner 2.0 is an advanced graph-based exploration path planner that addresses this challenge with a robust and efficient planning framework designed to maximize exploration while ensuring safety and reliability. At the core of GBPlanner 2.0 is its bifurcated planning architecture, a two-layer approach that integrates the agility of local decision-making with the strategic oversight of global planning. This architecture enables robots to navigate their immediate surroundings effectively while also managing long-term objectives, such as revisiting high-interest areas or safely returning to their starting point. Through this dual-layered strategy, GBPlanner 2.0 achieves a balance between rapid exploration and mission-level coordination, making it well-suited for both aerial and ground robotic platforms operating in demanding environments.

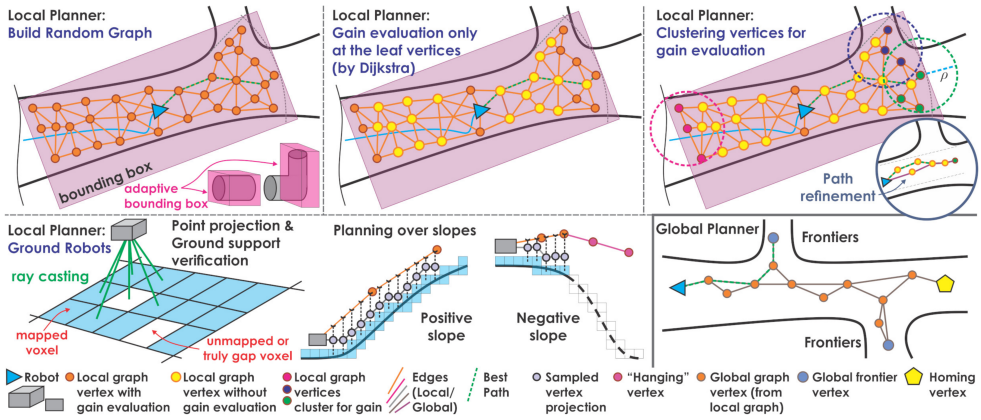

Figure 1: Outline of the key functional steps of the unified exploration architecture.

2.2 Methodology Overview

GBPlanner 2.0 employs a bifurcated planning architecture that divides the planning process into two distinct but complementary modules: a local planner and a global planner. This design enables the system to dynamically adapt to the robot's immediate environment while also maintaining high-level exploration goals, as shown in Fig. 1.

The local planner is responsible for immediate, short-horizon decision-making. It operates within a local bounding box centered around the robot, where it constructs a 3D graph by randomly sampling vertices in known free space. These vertices are connected with admissible edges based on the robot's physical and environmental constraints. To determine the most efficient paths through the graph, the planner computes the shortest paths from the robot's current location to all vertices using Dijkstra's algorithm. Each candidate path is evaluated using an exploration gain metric, which estimates the amount of new, unmapped space the path would reveal.

Simultaneously, a global graph is incrementally maintained throughout the exploration process, logging visited vertices and regions of high gain. This global graph serves as a long-term memory, allowing the planner to retain knowledge about previ-

ously identified frontiers and potential goals. If the local planner fails to find a viable or informative path due to depleted frontiers or blocked access, the global planner takes over. It queries the global graph to locate high-gain regions and guides the robot toward them for renewed exploration. If the mission is nearing completion or time constraints demand a return, the global planner also ensures safe navigation back to the homing point. This dual-layer planning structure ensures efficient coverage, adaptive behavior, and safe operation in complex environments.

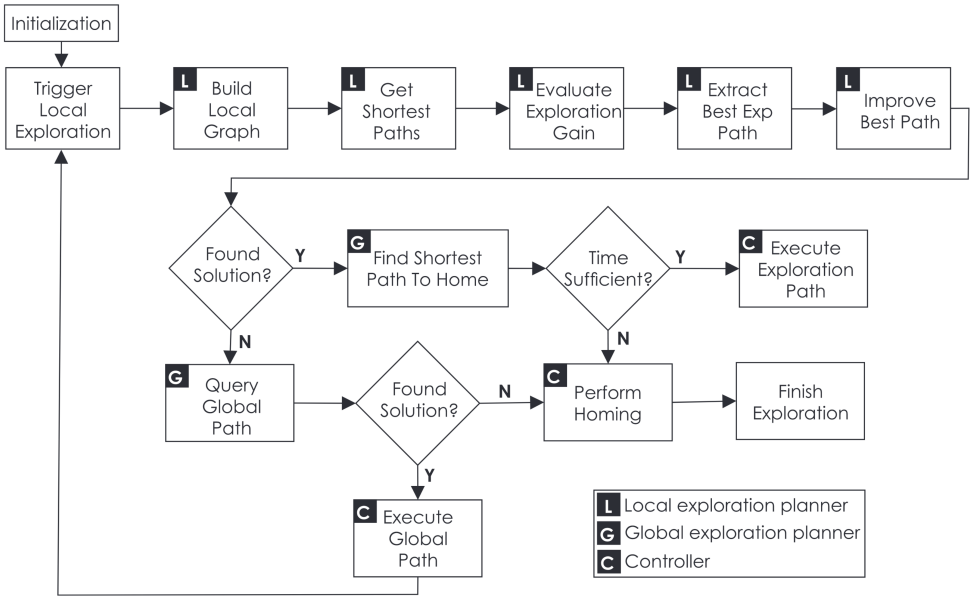

Figure 2: Flowchart of the unified exploration framework.

2.2.1 Aerial Robot-specific Planner Operations

For aerial robots, the planning process is optimized for full 3D mobility and environmental adaptability. The local planner samples points within an adaptive bounding box, which dynamically resizes based on the robot's surrounding environmental structure. Edges connecting these sampled vertices are considered admissible only if they lie entirely within known free space, ensuring collision-free paths through the air. To reduce computational load without sacrificing performance, exploration gain is calculated only for the leaf nodes of the generated graph, those at the ends of the shortest paths, as these are most likely to open up new, unmapped volumes. This targeted gain evaluation accelerates decision-making while maintaining exploration effectiveness. The global planner leverages the maintained global graph to identify previously high-gain regions and guides the aerial robot to them, often capitalizing on its ability to traverse large distances efficiently. This combination of adaptive local bounding box, selective gain calculation, and long-range planning makes the aerial-specific operations robust and scalable in large, complex environments.

2.2.2 Ground Robot-specific Planner Operations

For ground robots, path planning incorporates additional constraints such as terrain inclination, surface continuity, and obstacle avoidance to ensure safe and feasible

navigation. The local planner projects randomly sampled points onto the ground and uses an elevation map to verify the traversability of edges between these points. This elevation map provides detailed information about surface height variations, which is critical for determining whether a path respects the robot's inclination limits and physical capabilities. Edges are considered admissible only if they lie on continuous ground, remain obstacle-free, and maintain slopes within allowable thresholds. This ensures that the planned paths do not lead to unstable or impassable terrain. The global planner then leverages this terrain-aware information to evaluate long-range navigation options. It considers elevation-based accessibility and terrain structure when selecting paths to high-gain regions or planning safe returns, thereby maintaining both exploration efficiency and operational safety in uneven or cluttered environments.

2.3 Alternative Planning Methods

2.3.1 Efficient Mapping and Planning for Autonomous UAV Missions

For autonomous exploration in forest environments on drones, it is crucial to perform all algorithms efficiently to preserve battery life and enable long-reaching missions. With this motivation, we propose a submap-based exploration framework that uses volumetric mapping and is tailored to limited compute ressources, which can run on the drone while it's flying without requiring external processing. The volumetric map efficiently assigns the space to free, occupied, or unknown regions. The proposed approach splits the environment into discrete, locally consistent volumetric *submaps*. This decision is twofold: the data structure used to save the maps is more shallow compared to a monolithic map, allowing for faster data allocation and less memory usage; as we tightly couple our mapping strategy with the leveraged state-estimator, we can rigidly deform these submaps to account for the trajectory updates of the estimator, therefore obtaining a geometrically consistent submaps.

Each submap is an accurate local representation, constructed using the multi-resolution volumetric mapping framework Supereight2 [5]. Sensor measurements from camera or LiDAR sensors are integrated into the currently active submap. New submaps are initiated when the geometric overlap between the current observation and the previous submap is below a predefined threshold, ensuring there is overlap to generate constraints between the geometric information of the submaps.

Global consistency across the entire forest map is maintained through a tightly coupled optimization scheme. This involves incorporating LiDAR- or Camera- derived residuals directly into a SLAM posegraph as both frame-to-map factors, which constrain live sensor data to the active submap, and map-to-map factors, which align overlapping submaps with respect to each other. This process propagates loop-closure corrections across relative submap poses, dynamically updating the global map, which is constituted as an assembly of the rigid submaps.

For efficient exploration planning, the framework leverages frontiers, defined as the boundaries between known free space and unknown regions within the forest. Global, environment-wide frontiers are derived from the local frontiers maintained within each submap by checking if a submap's local frontier has been mapped in another submap (Fig. 3). A sampling-based next-best-view exploration planner utilizes these global frontiers to determine the optimal next drone position to maximize information gain and ultimately map the observable forest volume as completely as possible.

The maximization strategy is based on the map entropy along the ray from the point to be observed and the time it would take for the MAV to reach that pose. The map entropy for a voxel v is calculated using Shannon's information theory:

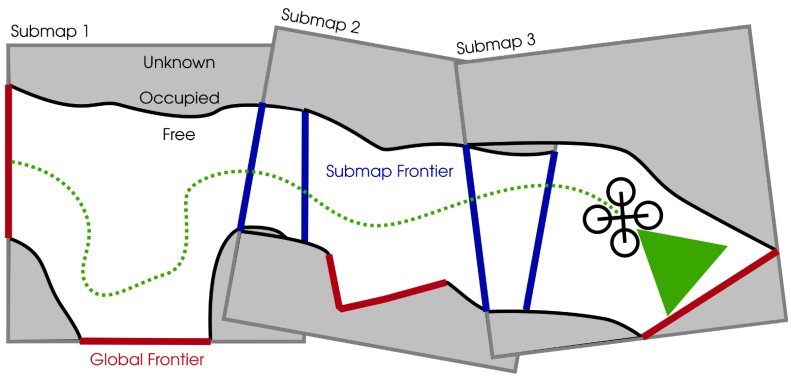

Figure 3: Local-only (blue) and global (red) frontiers in a set of. The local-only submap frontiers correspond to regions mapped in other submaps and are thus not considered to be global frontiers.

$$\mathcal{H}(\mathbf{v}) = -P_0(\mathbf{v})ln(P_0(\mathbf{v})) - (1 - P_0(\mathbf{v}))ln(1 - P_0(\mathbf{v})). \quad (1)$$

Where $\mathcal{H}(\mathbf{v})$ denotes the map entropy and $P_0(\mathbf{v})$ is the occupancy probability of a voxel $\mathbf{v}$. The entropy along the ray is computed as the sum of the entropy of the voxels along the ray until either an occupied voxel is intersected by the ray or the maximum depth range of our sensor is reached. The information gain is then computed as the division of the map entropy with the time to reach the required point by the MAV.

Once the most informative next frontier is determined, an informed RRT* path planning algorithm is used for generating collision-free trajectories that connect the target with the current drone position. Current work expands on this paradigm, targeting the efficient reuse of map queries by employing a cross-submap graph-based planning algorithm.

The proposed algorithmic pipeline is engineered for computational efficiency, enabling real-time execution on resource-constrained edge compute devices carried by the drone. For this, we rely solely on a NVIDIA Jetson Orin NX with 16 GB of shared GPU and CPU memory. This design facilitates autonomous onboard processing of sensor data, concurrent volumetric map construction, and adaptive exploration planning, enabling self-contained operations in remote forest environments. With the given compute resources, we were able to map a plot of 300 m^2 at a voxel resolution of 10 cm in a vision-only mode. In our field trial in Stein am Rhein, we have demonstrated that this map resolution and our planning stack are reliable and enable autonomous exploration and flight through forest environments (Fig. 4).

2.3.2 Traversability-aware Planner

The ANYmal legged robot was used by UOXF and ETH in their terrestrial forest inventory work. Compared to the planning methods used by the aerial platforms this requires some key changes to adapt to this platform. Specifically the robot is not flying and thus its mobility is instead defined by the local traversability which is usually represented as an elevation map (as referred to in Section 2.2.2).

However through learned experience, we identified that planning on the basis of 3D elevation mapping alone was not sufficient as stumps and branches present themselves as being untraversable while actually being traversable; while deep mud or bog can

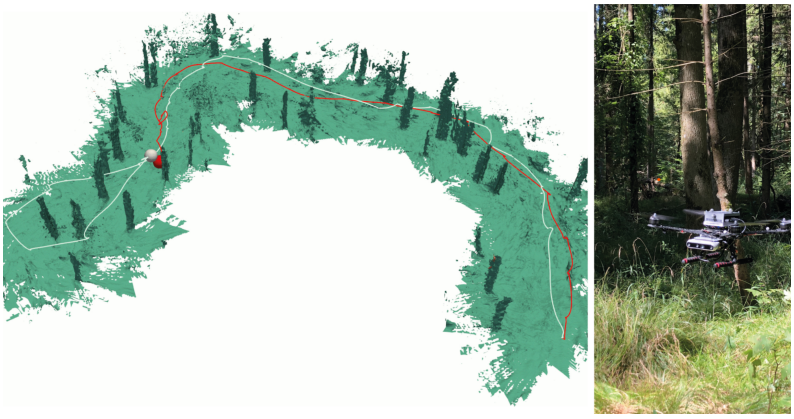

Figure 4: Autonomous deployment of a drone featuring an Intel Realsense D455 camera and a NVIDIA Jetson Orin NX compute module. On the flight, which lasted 15 minutes, we have mapped an area of 300m^2.

appear to be free space but is in fact untraversable. Our work thus went beyond pure geometry by taking a traversability learning approach.

We define a continuous *traversability score* $\tau \in [0, 1]$, where 0 is untraversable and 1 is fully traversable. We use the terrain *traction* [1], which measures the discrepancy between the robot's current linear velocity as estimated by the robot (v_x, v_y), and the reference velocity command $(\bar{v}_x, \bar{v}_y)$ given by an external human operator or planning system.

We define the mean squared velocity error as:

$$v_{\text{error}} = \frac{1}{2}\left((\bar{v}_x - v_x)^2 + (\bar{v}_y - v_y)^2\right) \in \mathbb{R} \tag{2}$$

We smooth v_{error} with a 1-D Kalman Filter before passing it through a sigmoid function to obtain a valid traversability score:

$$\tau = \text{sigmoid}\left(-k\left(v_{\text{error}} - v_{\text{thr}}\right)\right) \tag{3}$$

with k the steepness of the sigmoid, and v_{thr} the midpoint of the sigmoid that assigns a traversability score of 0.5. These values are calibrated depending on the motion specifications of each platform and determine how the velocity error is stretched to the $[0, 1]$ interval.

Supervision and Mission Graphs The system generates supervision signals by accumulating information in hindsight, during operation. Our approach is inspired by graph-based SLAM pipelines that leverage both local and global graphs to integrate measurements: we maintain a *Supervision Graph* to store short-horizon traversability data, and a global *Mission Graph* which stores the generated training data during a mission, shown in Fig. 5.

Supervision Graph The supervision graph stores within its nodes information about the current time, robot pose, and estimated traversability score (Sec. 2.3.2). This graph is implemented as a ring buffer, which only keeps a fixed number of nodes N_{sup}, separated from each other by a distance d_{sup}.

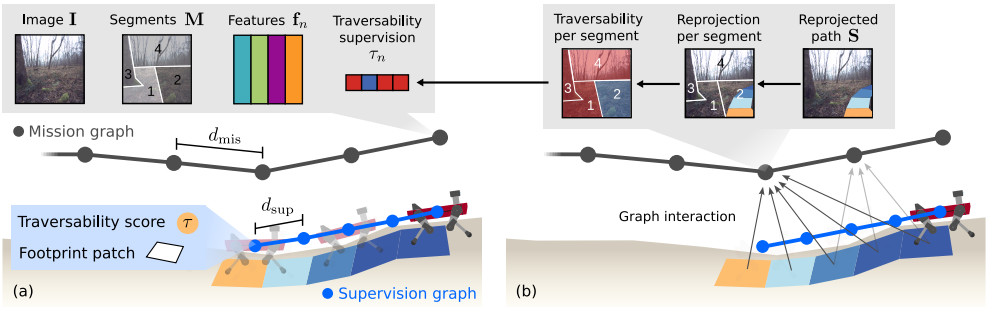

Figure 5: Supervision and mission graphs: (a) Information stored in each graph over the mission. While the Supervision Graph only stores temporary information about the robot's footprint in a sliding window, the Mission Graph saves the data required for online learning over the full mission. The color of the footprint patches indicates the generated traversability score. (b) The interaction between graphs updates the traversability in the mission nodes by reprojecting the robot's footprint and traversability scores.

The stored information is a footprint track with traversability scores τ. It is used to associate traversability scores with features by projecting the footprint track into the previous camera viewpoints.

Mission Graph The mission graph stores all the information required for online training. The mission nodes are added to the graph after feature extraction if the distance with respect to the last added node is larger than d_{mis}. Each mission node contains the RGB image $\mathbf{I}$, the weak segmentation mask $\mathbf{M}$ and per-segment features $\mathbf{f}_n$ with their corresponding traversability supervision τ_n.

Supervision generation When a new mission node is added, we update the supervision labels τ_n by reprojecting the footprint track and corresponding traversability scores τ onto all the images of the mission nodes within a fixed range (Fig. 5b).

Each mission node then has an auxiliary image with the reprojected path, $\mathbf{S}$. We use the weak segmentation mask $\mathbf{M}$ to assign per-segment traversability supervision values τ_n by averaging the score over each segment. Segments that do not overlap with the reprojected footprint track are set to zero (i.e untraversable). The outcome are pairs of per-segment features $\mathbf{f}_n$ and traversability score τ_n for each mission node, used for training.

Traversability and Anomaly Learning We train a small neural network in an online fashion that determines the feature traversability score τ_n from a given segment feature $\mathbf{f}_n$. This reduces the visual traversability estimation problem to simple regression task. Further, we model the uncertainty about the unvisited (and hence, unlabeled) areas by using anomaly detection techniques to bootstrap a confidence estimate.

First, we elaborate on how a confidence score for a feature is obtained, then we describe the traversability estimation learning task.

Confidence Estimation To obtain a segment-wise confidence estimate, we aim to learn the distribution over all traversed segment features $\mathbf{f}_n$. An encoder-decoder network $f_{\text{reco}}^{\theta_r}$ is trained to compress the segment feature $\mathbf{f}_n$ into a low dimensional

latent space and reconstruct the original input features $\mathbf{f}_n$. The reconstruction loss is given by the Mean Squared Error (MSE) between the predicted features and the original feature compute over all channels E:

$$\mathcal{L}_{\text{reco}}(\mathbf{f}_n) = \begin{cases} \frac{1}{E}\sum_e \|f^{\theta_r}_{\text{reco}}(\mathbf{f}_{n,e}) - \mathbf{f}_{n,e}\|^2 & \text{if traversed,} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

This ensures that the network only learns to reconstruct the embeddings that are labeled, in an anomaly detection fashion. Consequently, the trained network reconstructs known (*positive*) feature embeddings, i.e. similar to the traversable segments, with small reconstruction loss; feature embeddings of unknown (*anomalous*) segments the network was never tasked to reconstruct, such as trees or sky, induce a high reconstruction loss.

The unbounded reconstruction loss $\mathcal{L}_{\text{reco}}$ for a segment is mapped to a confidence measure $c(\mathcal{L}_{\text{reco}}) \in [0, 1]$ by first identifying the mode of the traversed segment losses. For this we fit a Gaussian distribution $\mathcal{N}(\mu_{\text{pos}}, \sigma_{\text{pos}})$ over the reconstruction losses per batch of the traversed segments (i.e, positive samples):

$$\mu_{\text{pos}} = \frac{1}{n_{\text{trav}}} \sum_{n \in \mathcal{T}} \mathcal{L}_{\text{reco}}(\mathbf{f}_n), \tag{5}$$

$$\sigma_{\text{pos}} = \sqrt{\frac{1}{n_{\text{trav}}} \sum_{n \in \mathcal{T}} \left(\mathcal{L}_{\text{reco}}(\mathbf{f}_n) - \mu_{\text{pos}}\right)^2} \tag{6}$$

with $\mathcal{T}$ being the set of segments that were traversed, i.e. have a valid traversability score τ_n computed from robot sensing data, and n_{trav} is the total number of traversed segments. We set the segment confidence to 1 if the loss of the segment is smaller than μ_{pos} and otherwise we set it by evaluating the unnormalized Gaussian likelihood:

$$c(\mathcal{L}_{\text{reco}}(\mathbf{f}_n)) = \exp\left(-\frac{(\mathcal{L}_{\text{reco}}(\mathbf{f}_n) - \mu_{\text{pos}})^2}{2(\sigma_{\text{pos}}\, k_\sigma)^2}\right), \tag{7}$$

where we introduce the tuning parameter k_σ, which allows to scale the confidence.

Traversability Estimation We train a small network $f^{\theta_t}_{\text{trav}}$ with a single channel output to regress on the provided segment traversability score τ. For the untraversed segments with unknown traversability score, we follow a conservative approach setting $\tau = 0$ but using the confidence score to scale their overall contribution. The loss for traversability estimation is computed using the confidence-weighted MSE:

$$\mathcal{L}_{\text{trav}}(\mathbf{f}) = \underbrace{\sum_{n \in \mathcal{T}} \|f^{\theta_t}_{\text{trav}}(\mathbf{f}_n) - \tau_n\|^2}_{\substack{\text{Contribution of traversed } (\textit{labeled}) \\ \text{segments}}} \tag{8}$$

$$+ \underbrace{\sum_{n \in \mathcal{T}^C} (1 - c(\mathbf{f}_n)) \, \|f^{\theta_t}_{\text{trav}}(\mathbf{f}_n) - 0\|^2}_{\text{Contribution of untraversed segments}}, \tag{9}$$

with $\mathcal{T}$ the set of traversed segments; $\mathcal{T}^C$ is the complement set of untraversed segments. This formulation enables the learning process to "overwrite" previously unknown samples as new data is used for training:

- If the segment n was traversed: it will contribute to the loss using the assigned traversability score: $\mathcal{L}_{\text{trav}}(\mathbf{f}_n) = \|f^{\theta_t}_{\text{trav}}(\mathbf{f}_n) - \tau_n\|^2$
- If the segment n was untraversed and it does not resemble a positive sample: its confidence will be low $c(\mathbf{f}_n) \to 0$ and $\mathcal{L}_{\text{trav}}(\mathbf{f}_n) \to \|f^{\theta_t}_{\text{trav}}(\mathbf{f}_n) - 0\|^2$
- If the segment n was untraversed but it does resemble a positive sample: its confidence $c(\mathbf{f}_n) \to 1$ and $\mathcal{L}_{\text{trav}}(\mathbf{f}_n) \to 0$, effectively not contributing to the loss anymore. This motivates the network to learn the traversability score measured by physically interacting with the segment as a opposed to being too pessimistic.

As we aim to provide the estimated traversability as input for a local planning system, we automatically define a threshold to determine the traversable and untraversable areas. We select a traversability threshold τ_{thr} by measuring the current performance of the system in a self-supervised manner. We compute the Receiver Operating Characteristic (ROC) throughout training by classifying all segments with confidence under 0.5 as negative and traversed segments as positive labels. Then, we decide on the traversability threshold only by setting the desired False Positive Ratio (FPR).

Implementation details We implemented $f^{\theta_r}_{\text{reco}}$ and $f^{\theta_t}_{\text{trav}}$ as a two-layer Multi-Layer Perceptrons (MLPs) with [256, 32] unit dense layers and ReLU non-linear activation functions. Both networks share the weights of the hidden layers. $f^{\theta_r}_{\text{reco}}$ has a reconstruction head with E output neurons and $f^{\theta_t}_{\text{trav}}$ a single channel traversability head followed by a sigmoid activation. The 32-channel hidden layer functions as the bottleneck of the encoder-decoder structure. The total loss per segment during training is given by:

$$\mathcal{L}_{\text{total}}(\mathbf{f}) = w_{\text{trav}}\mathcal{L}_{\text{trav}}(\mathbf{f}) + w_{\text{reco}}\,\mathcal{L}_{\text{reco}}(\mathbf{f}). \tag{10}$$

with w_{trav} and w_{reco} allowing to weigh the traversability and reconstruction loss respectively. We used Adam [6] to jointly train the networks with a fixed constant learning rate of 0.001. For a single update step, 8 valid mission nodes are randomly chosen to form a data batch, where we defined a node as valid if at least a single segment of the node has non-zero traversability score. For all our experiments we set $k_\sigma = 2$, $w_{\text{trav}} = 0.03$, $w_{\text{reco}} = 0.5$ and use a maximum FPR of 0.15 to determine the traversability threshold. Please refer to our previous publication [4] for ablation studies of the different parameter and design choices.

Local Planning Scheme The local planner generates velocity commands for the low-level locomotion controller to reach a specified waypoint. To achieve this, it relies on the local scene representation built by the local terrain mapping module.

First, a traversability estimation system processes the local terrain map and extracts geometric features to score the terrain's navigability. The traversability score s_{trav} is specifically tailored to enable navigation in forest environments with branches and twigs.

This traversability information is used by a reactive local planner [7] based on Riemannian Motion Policiess (RMPs) [10]. The approach combines different vector fields, given by analytical expressions or derived from the terrain map. For this application, we compute a continuous cost-to-go $c_{\text{to-go}}$ from the traversability score obtained by the cell-based heuristic:

$$c_{\text{to-go}} = w_{\text{trav}}\,(1.0 - s_{\text{trav}}) + w_{\text{unkn}}\,s_{\text{unkn}}, \tag{11}$$

where s_{unkn} is a fixed score assigned to empty (unknown) cells in the terrain map, and w_{trav}, w_{unkn} are user-defined weights balancing the influence of known and unknown cells. This cost map is used to compute a vector field for collision avoidance (by thresholding the cost map and computing a Signed Distance Field (SDF)), as well as a vector field to guide the robot towards the next waypoint (GDF).

Simultaneously, the local planner continuously monitors progress toward each waypoint to detect potential unfeasible situations, for example if the proposed waypoint is in the middle of bushes [11]. If a waypoint is determined unreachable, the local planner reports this situation to the mission planner, triggering a re-planning of the next waypoint.

Both the local planner and the traversability analysis modules were always executed on the Navigation PC of all the versions of the ANYmal we used.

Locomotion Controller The local planner outputs a 3 DoF velocity command—linear (x, y), angular (yaw)—, which is executed by a low-level RL-based locomotion controller. In our experiments, we used both the perceptive controller presented by Miki et al. [9], as well as the blind, learning-based controller from ANYbotics. We used them with no additional modifications to the architecture or training environment to operate in the forest. In both cases, these ran on the Locomotion PC.

We integrated the learned traversability into a standard navigation pipeline to achieve autonomous navigation with a quadrupedal platform. The details of each module of the system are explained in the following sections.

Local terrain mapping To map the environment surrounding the robot we used an open-source terrain mapping framework [8] to efficiently obtain a robot-centric 2.5D elevation map from the onboard depth cameras and LiDAR sensing. We extended this framework in [2] to fuse our predicted traversability into the local map representation. As our predicted traversability is an image, we used raycasting to take into account the occlusions with the terrain, establishing correspondences between pixel-wise traversability values in the image plane and the local map's grid cells. This procedure allows for temporal fusion of the traversability information in the map while preserving the values of previously observed areas via exponential averaging. Since this indirectly uses geometry, the projection method is susceptible to artifacts due to spikes in the elevation map.

Local planning We used the projected visual traversability from the local map as a costmap for local planning. A median filter removed undesired noise and artifacts before the distance transform method [3] was used to obtain a SDF, which represents the distance to the closest untraversable object. We implemented a local planner method based on [7], which exploits the SDF and the local goal to generate a SE(2) twist command which drives the robot towards the goal while avoiding untraversable terrain. Finally, the twist command becomes the input to a robust learning-based locomotion controller based on the work by [9], which is able to traverse rough terrain typically inaccessible to wheeled robots.

Smart carrot for autonomous exploration Lastly, to generate an autonomous navigation behavior we implemented a simple exploration strategy by analyzing the robot-centric SDF created by the local planner, by choosing a *moving carrot* that drives the robot forward.

The goal pose is given by analyzing a section of the SDF in front of the robot and selecting the position with the largest distance from all obstacles, ensuring the robot stays at the center of the traversable space. The goal is continuously updated when the SDF is recomputed with new traversability information. While this strategy was simple it could safely guide the robot to follow a footpath autonomously using the predicted traversability without requiring a global plan or a large-scale representation of the environment. Nevertheless, further improvement could be achieved by using a sophisticated exploration planning system.

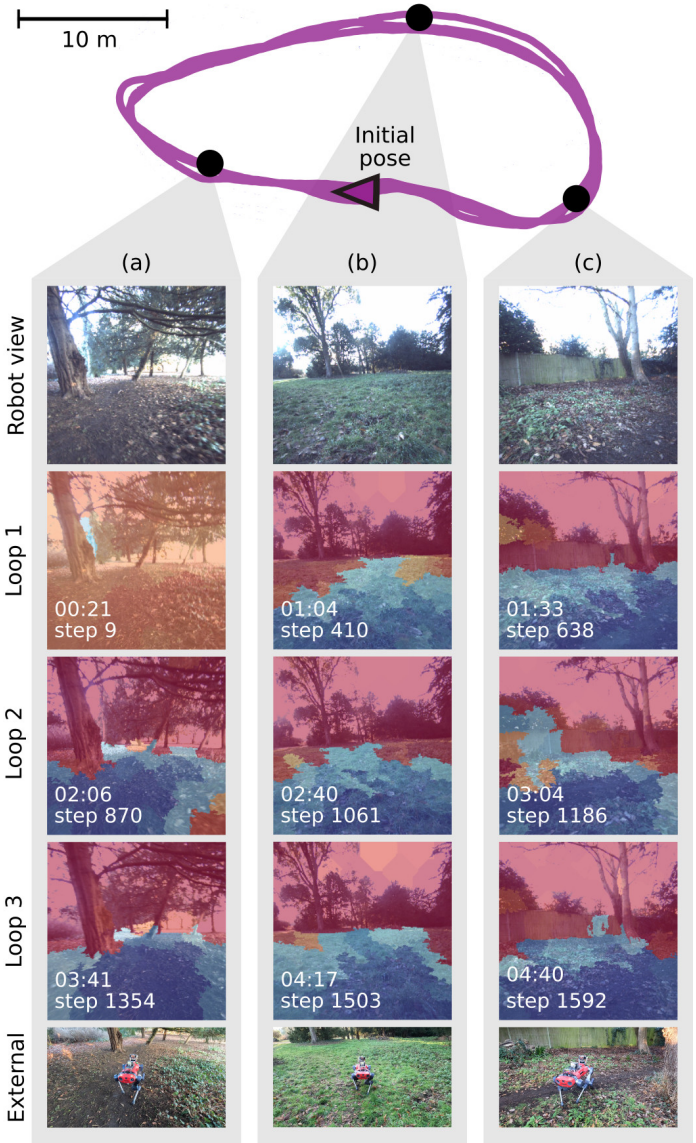

Figure 6: Adaptation on real hardware: We tested the online adaptation capabilities of our system by teleoperating the robot to complete 3 loops in a park (top, route shown in ■). The columns show different parts of the loop (a,b,c); each row displays the improvement of the traversability estimate over time and training steps.

3 Field Results

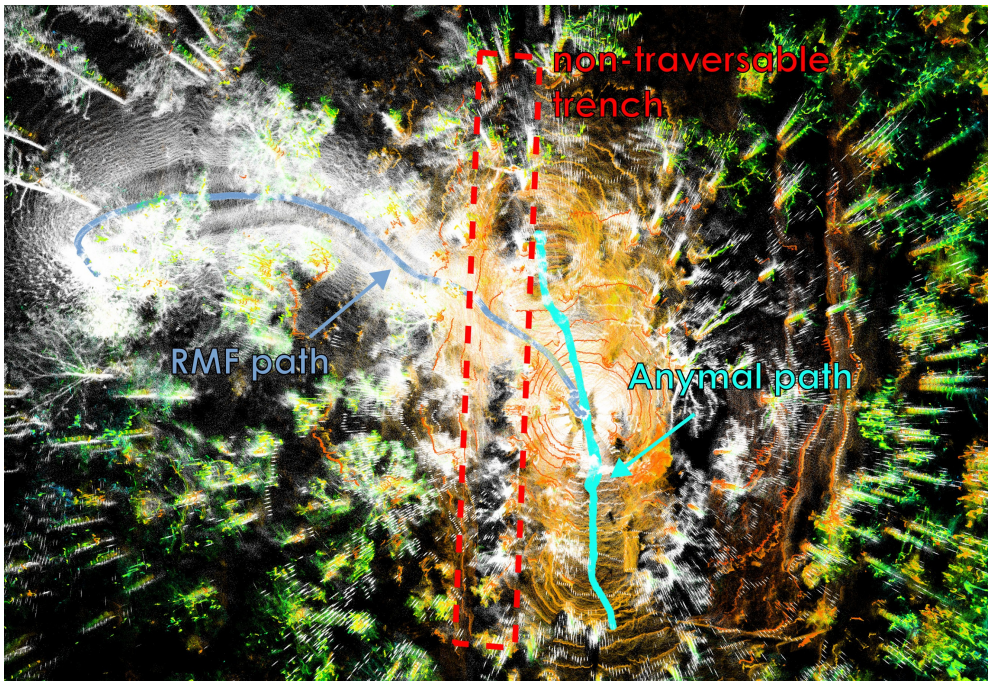

Figure 7: Pointcloud map built in short exploration mission with marsupial system.

To validate the capabilities of the unified planning algorithm in a real-world, unstructured environment, a field test was conducted using a marsupial robot team composed of a ground robot carrying an aerial robot. The test took place in a moderately dense forest environment in Dragvoll, Trondheim, Norway with natural obstacles such as uneven terrain, vegetation, and elevation changes.

The exploration began with the ground robot autonomously navigating the environment using its onboard sensors and elevation-aware local planner. As the robot moved forward, it continuously updated its local and global graphs while checking path feasibility using elevation maps. At one point, the ground robot encountered a non-traversable trench which spanned several meters and featured steep, unstable edges, as shown in Fig. 8. The terrain analysis module flagged this region as untraversable based on slope constraints and discontinuities in elevation.

In response, the system triggered the deployment of the aerial robot. Before takeoff, the ground robot shared its locally explored map with the aerial robot. The aerial robot then performed co-localization using its onboard sensors and the shared map to align its pose estimate within the global frame established by the ground robot. This ensured that both robots operated within a unified map, allowing for seamless data fusion and coordinated exploration.

Following successful localization, the aerial robot took off and activated its own local planner, constructing a 3D graph to explore the environment beyond the trench. With full 3D mobility, it was able to fly over the obstacle and identify high-gain regions that were otherwise inaccessible to the ground robot.

The aerial robot transmitted its mapped data and updated global graph back to the ground platform, enabling the system to maintain a unified and up-to-date view

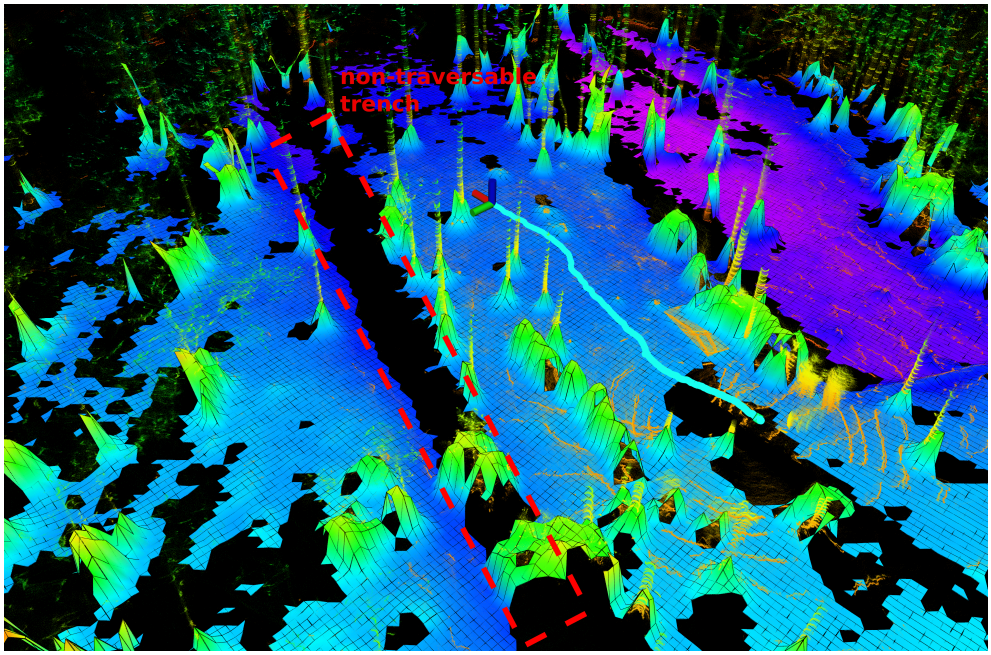

Figure 8: Elevetion map built in short exploration mission by the ground robot.

of the explored environment, as illustrated in Fig. 7. This test demonstrated the effectiveness of coordinated deployment between heterogeneous robots, highlighting the system's ability to adaptively switch modalities in response to environmental constraints and maintain exploration continuity in complex terrain.

References

[1] Xiaoyi Cai et al. "EVORA: Deep Evidential Traversability Learning for Risk-Aware Off-Road Autonomy". In: *CoRR* abs/2311.06234 (2023). DOI: 10.48550/ARXIV.2311.06234.

[2] Gian Erni et al. "MEM: Multi-Modal Elevation Mapping for Robotics and Learning". In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2023. DOI: 10.1109/IROS55552.2023.10342108.

[3] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. "Distance Transforms of Sampled Functions". In: *Theory of Computing* 8.19 (2012), pp. 415–428. DOI: 10.4086/toc.2012.v008a019.

[4] Jonas Frey et al. "Fast Traversability Estimation for Wild Visual Navigation". In: *Robotics: Science and Systems (RSS)*. 2023. DOI: 10.15607/RSS.2023.XIX.054.

[5] Nils Funk et al. "Multi-resolution 3D mapping with explicit free space representation for fast and accurate mobile robot motion planning". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3553–3560.

[6] Diederick P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *Intl. Conf. on Learning Representations (ICLR)*. 2015.

[7] Matias Mattamala, Nived Chebrolu, and Maurice Fallon. "An Efficient Locally Reactive Controller for Safe Navigation in Visual Teach and Repeat Missions". In: *IEEE Robot. Autom. Lett. (RA-L)* 7.2 (2022), pp. 2353–2360. DOI: 10.1109/LRA.2022.3143196.

[8] Takahiro Miki et al. "Elevation Mapping for Locomotion and Navigation using GPU". In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2022, pp. 2273–2280. DOI: 10.1109/IROS47612.2022.9981507.

[9] Takahiro Miki et al. "Learning Robust Perceptive Locomotion for Quadrupedal Robots in the Wild". In: *Sci. Robot.* 7.62 (2022). DOI: 10.1126/SCIROBOTICS.ABK2822.

[10] Nathan D. Ratliff, Jan Issac, and Daniel Kappler. "Riemannian Motion Policies". In: *CoRR* abs/1801.02854 (2018). arXiv: 1801.02854.

[11] Sebastian Scherer et al. "Resilient and Modular Subterranean Exploration with a Team of Roving and Flying Robots". In: *Field Robotics* 2 (2022), pp. 678–734. DOI: 10.55417/fr.2022023.