Fast Traversability Estimation for Wild Visual Navigation

Jonas Frey[⋆,1,3] Matias Mattamala[⋆,2] Nived Chebrolu[2] Cesar Cadena[1] Maurice Fallon[2] Marco Hutter[1]
[1] ETH Zurich [2] University of Oxford [3] Max Planck Institute for Intelligent Systems
[⋆] Equal contribution. jonfrey@ethz.ch, matias@robots.ox.ac.uk

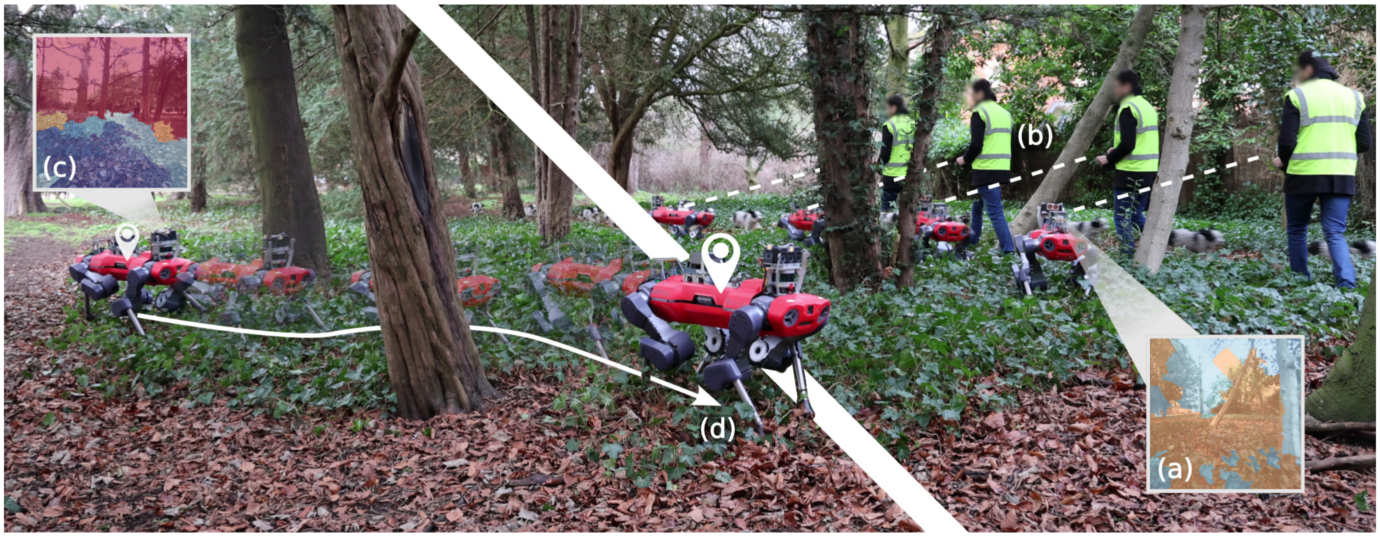

Fig. 1: Wild Visual Navigation (WVN) learns to predict traversability from images via online self-supervised learning. Starting from a randomly initialized traversability estimation network without prior assumptions about the environment (a), a human operator drives the robot around areas that are traversable for the given platform (b). After a few minutes of operation, WVN learns to distinguish between traversable and untraversable areas (c), enabling the robot to navigate autonomously and safely within the environment (d).

***Abstract*—Natural environments such as forests and grasslands are challenging for robotic navigation because of the false perception of rigid obstacles from high grass, twigs, or bushes. In this work, we propose Wild Visual Navigation (WVN), an online self-supervised learning system for traversability estimation which uses only vision. The system is able to continuously adapt from a short human demonstration in the field. It leverages high-dimensional features from self-supervised visual transformer models, with an online scheme for supervision generation that runs in real-time on the robot. We demonstrate the advantages of our approach with experiments and ablation studies in challenging environments in forests, parks, and grasslands. Our system is able to bootstrap the traversable terrain segmentation in less than 5 min of in-field training time, enabling the robot to navigate in complex outdoor terrains — negotiating obstacles in high grass as well as a 1.4 km footpath following. While our experiments were executed with a quadruped robot, ANYmal, the approach presented can generalize to any ground robot. Project page: bit.ly/3M6nMHH**

I. Introduction

Traversability estimation is a core capability needed to allow robots to autonomous navigate in field environments. It is understood as the *affordance* [14] necessary for a robot to navigate within its environment, i.e to understand which areas can be accessed and navigated through and at what cost. While the topic has been widely studied for wheeled or flying robots supported by 3D sensors using the traditional approach of occupancy mapping [27], the development of new platforms with advanced mobility skills, such as legged robots, prompts a reconsideration of current definitions of traversability, as new and more complex types of natural terrain can be traversed [25].

It is difficult to infer traversability within natural terrains, such as high grass or forest undergrowth. Occupancy-based navigation systems based on 3D sensing are often confused by high grass and incorrectly classify such terrain as an obstacle which is untraversable — even if the platform is actually able to pass through. Semantic understanding is important in such environments to determine which terrain is actually passable for a particular robot.

Existing approaches, which build upon deep neural models for semantic segmentation [24] or anomaly detection [37], have demonstrated navigation in off-road environments; however there are recurring problems with the collection and labelling of large amounts of relevant training data. In addition to the effort required to curate these datasets, specific class labels (tree, branch, bush, grass) do not directly correspond to the capabilities of the robotic platform.

Self-supervised systems have addressed this challenge by generating labeled datasets from past robot deployments, using classification carried out in hindsight [36] or using predictions of the robot motion [13]. Nevertheless, these previous methods are still trained on robot-specific datasets and subsequently deployed without further adaptation. If they were to be tested in a new environment or on a new robotic platform, new data would need to be collected and the models would need to be retrained, limiting applicability.

Achieving online self-supervision and learning are key to overcoming these aforementioned limitations, as traversability could be more easily learned in the field. The Learning Applied to Ground Vehicles (LAGR) program [19, 15] pioneered this direction, generating supervision during the mission and training machine learning models for traversability estimation on the fly. They showed first demonstrations of autonomous robots navigating off-road environments under this approach.

In this work, we build upon such advances and present a system for vision-based traversability estimation that achieves online, self-supervised adaptation, by improving on various components of previous works. We name the approach Wild Visual Navigation (WVN), as it is capable of learning which terrain is traversable by a robot after a few minutes of manual demonstrations *in the wild*. The system builds upon four core ideas that we consider the key contributions of our work:

- **A self-supervision system** designed for real-time operation, which concurrently generates supervision signals from vision and traversability measurements from proprioception and control performance.
- **A learning approach** that leverages high-dimensional, self-supervised visual features extracted using pre-trained vision transformer models, which are fed into a small neural network and efficiently trained online.
- **A new formulation for traversability estimation** that combines supervised learning with anomaly detection, accounting for uncertainties due to the sparse supervision.
- **Closed-loop integration** with local mapping and planning methods which demonstrate that these traversability estimates are suitable for autonomous navigation tasks.

We have extensively validated our approach with ablation studies which compare against similar approaches that are trained in an offline fashion. Further, we deployed our system on real hardware, the ANYbotics ANYmal C robot, showing that it can be easily trained for navigation tasks in environments where it would not be possible to traverse using geometric mapping alone. We demonstrate fast adaptation within minutes to determine the traversable areas in a natural environment, achieving closed-loop operation in a forest with different understory foliage and terrain, and a kilometer-scale path-following task in a park where the semantic class of the path was not explicitly labeled. While our experimental results have been demonstrated on a legged robot, the principles we present are general and applicable to other ground platforms.

II. Related Work

A. Traversability from geometry

Classical approaches for traversability estimation analyze the geometry of the environment using 3D sensing [27]. Recent examples from the DARPA SubT Challenge [7], used different representations such as point clouds [4] and meshes [16] to evaluate navigational metrics such as risk or stepping difficulty [9].

However, a purely geometric analysis is not sufficient to completely capture traversability for a given platform. Data-driven methods can bridge this gap by learning platform-specific traversability from data or simulations. For example, Chavez-Garcia et al. [6] learned traversability from simulations of a ground robot moving on an elevation map. Yang et al. [40] extended this approach for legged platforms, capturing the risk of failure, energy cost and time required for navigation. Recently, Frey et al. [11] expanded this approach to volumetric data and massive parallelization in data collection from simulation. While we recognize the contribution those previous works make in using robot-specific geometric data to determine traversability, using geometry-only does not succeed in natural environments containing natural growth such as high grass, branches or bushes. We instead focus on vision-based methods herein, as they describe semantic features that are challenging to capture otherwise.

B. Traversability from semantics

Semantic segmentation methods have been proposed to address the aforementioned challenges by assigning navigation costs to the different semantic classes. Bradley et al. [3] presented a scene understanding system trained and evaluated using geographically diverse data. Maturana et al. [24] demonstrated autonomous off-road navigation using semantics projected onto 3D map around a wheeled platform. Schilling et al. [33] used semantically segmented features that were classified into fixed classes using a random forest. Recently Shaban et al. [35] presented an approach for off-road navigation that learns a dense traversability map from sparse point-clouds.

We note that these methods typically rely on pre-trained or fine-tuned semantic segmentation models. This requires specific class labels to be defined; and these semantic classifiers can be difficult to reuse in different environments. Nevertheless, new advances in self-supervised models, such as DINO-ViT [5], are able to segment semantically meaningful classes without manual supervision. We exploit these promising tools in this work.

C. Traversability from self-supervision

Purely semantic methods are challenging to use *in the wild* because (1) it is difficult to represent the relevant classes of these operating environments without labeled data, and (2) assigning a traversability cost to each class, which can also be arbitrary. Kim et al. [19] exploited information about the areas visited by a wheeled robot to determine positive traversable samples, while a bumper was used for negative labeling.

Bajracharya et al. [2] used a similar approach combined with a height heuristic to determine long-range negative samples.

Modern methods rely on deep-learned models trained from weakly supervised data, and the supervision strongly depends on the hardware platform. Wellhausen et al. [36] used the reprojected footholds from a legged robot to provide supervision; Zürn et al. [41] exploited sounds produced by the platform moving on different terrain as a proxy for supervision; Gasparino et al. [13] instead used the receding-horizon trajectory of a Model Predictive Controller (MPC). Recently, TerraPN [32] used odometry and IMU signals as supervision and could learn a traversability model in 25 min – including data collection and learning. Our work is inspired by these approaches and follows similar self-supervision strategies but we aim for concurrent supervision signal generation and learning achieving orders of magnitude faster adaptation.

D. Traversability from anomalies

Anomaly detection methods are motivated by one of the key challenges of using self-supervised approaches, namely an imbalance in the number of positive and negative samples. Instead of training a discriminative model of traversability, they focus on learning generative models of the traversed terrain. This distribution can then used as a proxy to set out-of-distribution (OOD) inputs as being untraversable. Richter and Roy [31] developed a visual navigation system that relied on an autoencoder to predict OOD scenes from images, switching to safer navigation behaviors when traversing novel environments. Wellhausen et al. [37] used multi-modal sensing from haptics, vision and depth to classify as anomalies visual elements such as flames and water reflections in navigation tasks. Ji et al. [17] formulated a proactive anomaly detection approach that evaluated candidate trajectories for local planning depending on their probability of failure. While we do not explicitly use anomalies to determine traversability, we do use it as a confidence metric to leverage the sparse supervision signals. Similar ideas have been recently presented by Seo et al. [34] to determine traversability with point cloud data.

E. Traversability from demonstrations

Inverse Reinforcement Learning (IRL) is a framework which can learn a reward function from demonstrations. In our context this can be interpreted as a traversability that encodes the preference to navigate certain areas. Ratliff et al. [30] showed how IRL methods could be used to generate large-scale mission plans from aerial images. Later, Wulfmeier et al. [39] learned cost maps to encode driving preferences using deep neural networks, which was extended by Gan et al. [12] to guide the navigation of a legged robot in natural environments using a local reward map. While our approach uses the examples from a human operator in our self-supervised framework instead of IRL; we show that it can also be used to encode preferences depending on the demonstration and representation of the terrain.

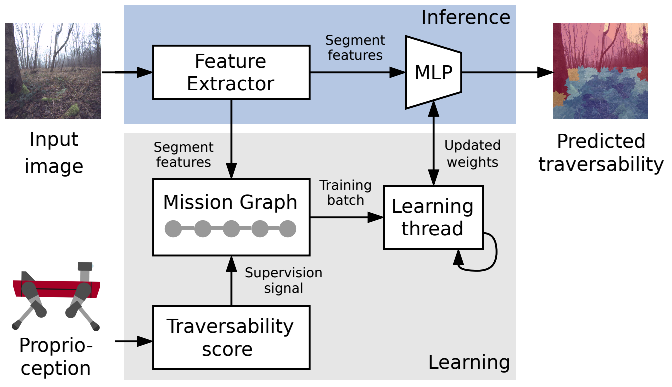

Fig. 2: System overview: WVN only requires monocular RGB images and proprioceptive data as input, which are processed to extract features and supervision signals used for online learning and inference of traversability (see Sec. III).

Symbol	Definition
$\mathbf{I}$	RGB image with height H and width W
$\mathbf{F}$	Feature map with dimensions $E \times H \times W$, $E = 90$
$\mathbf{M}$	Weak segmentation mask with height H and width W
$\mathbf{S}$	Reprojected supervision with dimensions $H \times W \in [0, 1]$
τ	Traversability score $\in [0, 1]$
$\mathbf{f}_n$	Per-segment embedding of dimension $E = 90$
τ_n	Per-segment traversability score

TABLE I: Main definitions used in this work

F. Adaptive traversability estimation

A few works have demonstrated adaptive traversability estimation and navigation behavior. As part of LAGR, Kim et al. [19] ran a clustering algorithm during deployment, and assigned positive and negative labels to the clusters through interactions to determine the traversable areas. Hadsell et al. [15] followed a similar approach to learn different binary classifiers during exploration, using features learned from a Convolutional Neural Network (CNN), making it one of the first works to use neural models for this purpose. Later Lee et al. [21] used Bayesian clustering on SLIC superpixels using color and texture features supervised by motion priors; this approach was able to propagate the traversability labels to similar segments during operation. More recently, BADGR [18] presented an end-to-end policy designed to adapt from experiences, though it was not framed for online learning.

Our approach builds upon the LAGR ideas for self-supervision and online learning but we extend them with more powerful semantic features from a pre-trained visual transformer that allowed us to deploy our system in a variety of natural environments.

III. Method

A. System Overview

The objective of this work is to design a navigation system that estimates dense traversability from RGB images using a neural network model learned online, in a self-supervised manner, using labels generated by a robot interacting with its

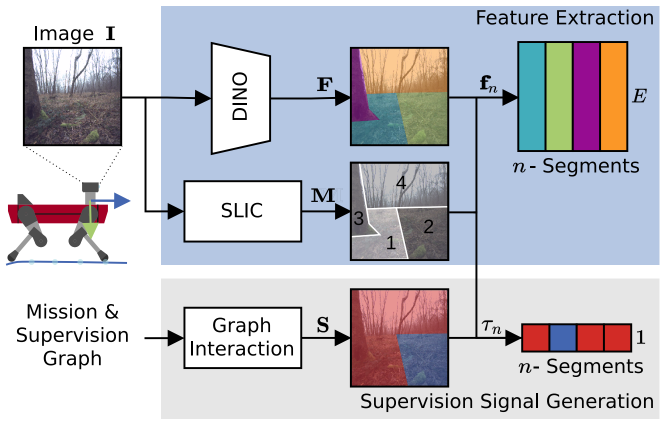

Fig. 3: Feature Extraction: Our approach extracts dense DINO-ViT features $\mathbf{F}$ from an RGB image $\mathbf{I}$, and $n = 100$ segments $\mathbf{M}$ using SLIC. For each of the n segments we average the corresponding features to obtain per-segment embeddings $\mathbf{f}_n$. A traversability score τ_n is computed for each segment based on the score resulting from the graph interaction process (for more detail refer to Sec. III-D).

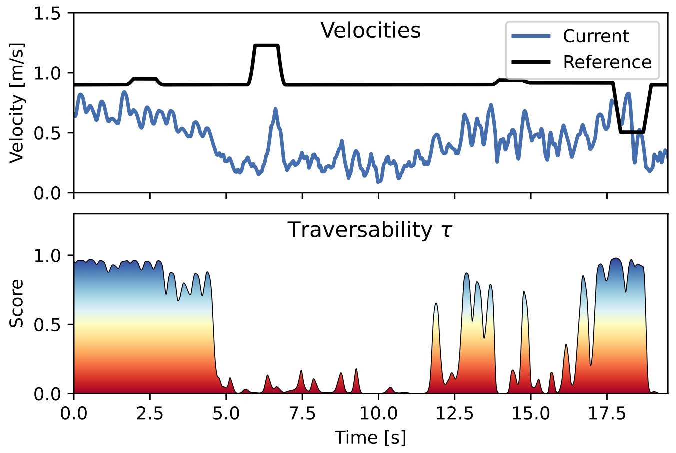

Fig. 4: Traversability score generation: We compute a score based on the difference of an (external) reference velocity command and the current velocity estimated by the robot. Closely tracking the reference is interpreted as moving over traversable terrain (in blue 🟦), high discrepancies indicate otherwise (red 🟥).

environment. We target a system which requires only a brief demonstration from a human operator for data collection and learning.

Our proposed system WVN, builds upon different modules shown in Fig. 2. *Feature extraction* (Sec. III-B) and *traversability score generation* (Sec. III-C) process the incoming sensor data to extract features and traversability scores. The *mission and supervision graphs* (Sec. III-D) manage the data and generate the self-supervision signals, while the *learning thread* (Sec. III-E) performs online traversability and anomaly learning. The interaction between the different modules is illustrated in Fig. 2, and their implementation is covered in the following sections. The main definitions used in the rest of the paper are summarized in Tab. I.

B. Feature Extraction

Given an RGB image $\mathbf{I}$, we first extract dense, pixel-wise visual feature maps (*embeddings*) $\mathbf{F}$. In contrast to previous works based on fine-tuned CNNs, we rely on recent self-supervised network architectures to leverage a Vision Transformer (ViT) trained using the DINO method [5] – DINO-ViT. These learned representations have been demonstrated to encode meaningful semantic and instance information without requiring any labels.

Since we aim for real-time operation, the processing and storage of the full dense features on a mobile robot is prohibited by the limited compute and storage availability. Instead, we follow previous works [21] and compute a weak segmentation mask $\mathbf{M}$ of the input image $\mathbf{I}$ using superpixels. We use SLIC [1] to extract 100 segments per image. We then average the feature maps segment-wise resulting in a single embedding $\mathbf{f}_n$ per segment. Fig. 3 illustrates the complete feature extraction process.

C. Traversability Score Generation

Defining which terrain is traversable or not depends on the capabilities of the specific platform. We define a continuous *traversability score* $\tau \in [0, 1]$, where 0 is untraversable and 1 fully traversable. Previous works have used ground reaction forces, audio supervision, or predictions from a MPC as a proxy for this score. Instead, we adopt a simpler approach that uses the discrepancy between the robot's current linear (x, y) velocity as estimated by the robot $\mathbf{v}$, and the reference velocity command $\bar{\mathbf{v}}$ given by an external human operator or planning systems. The intuition is that when the robot moves on terrain that is easily traversable it should closely track the reference command. In contrast, if the robot struggles to track the reference, the discrepancy will grow, and we can interpret it as a less traversable terrain. We define the mean squared velocity error as:

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

As v_{error} will be a noisy scalar signal, we smooth it with a 1-D Kalman Filter before passing it through a sigmoid function to obtain a valid traversability score:

$$\tau = \text{sigmoid}\left(-k\left(v_{\text{error}} - v_{\text{thr}}\right)\right) \tag{2}$$

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 can be calibrated depending on the motion specifications of each platform and determine how the velocity error is stretched to the $[0, 1]$ interval.

D. Supervision and Mission Graphs

In contrast to other methods that generate the supervision signal in post-processing [36, 13, 32], we execute this process online by accumulating information about the recent history of operation. Our approach is inspired by graphical SLAM

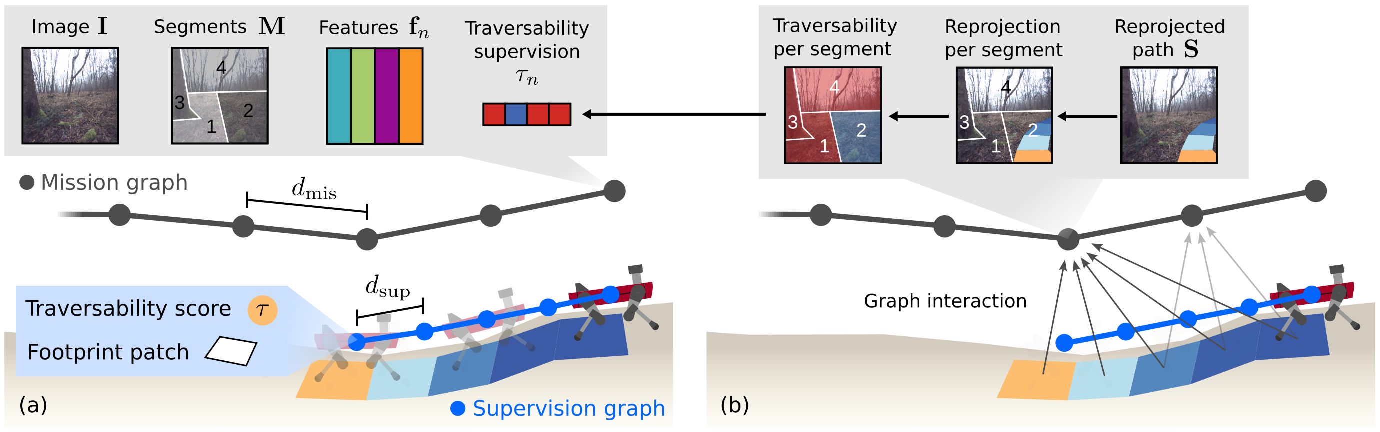

Fig. 5: Supervision and mission graphs: (a) Information stored in each graph over the mission. While the Supervision Graph only stores temporary information about the robot's footprint in a sliding window, the Mission Graph saves the data required for online learning over the full mission. The color 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.

pipelines that often 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.

1) Supervision Graph: The supervision graph stores within its nodes information about the current time, robot pose, and estimated traversability score (Sec. III-C). 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 distance d_{sup}. The product $N_{\text{sup}}\, d_{\text{sup}}$ determines the maximum length (in physical distance) of the supervision graph.

The stored information is a footprint track with traversability scores τ, used to generate supervision signals in hindsight that are reprojected onto previous camera viewpoints as it is explained in the following sections.

2) Mission Graph: On the other hand, the mission graph stores all the information required for learning over the mission. The mission nodes are added to the graph after feature extraction if the distance w.r.t 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.

3) Graph interaction for supervision generation: When a new mission node is added, it triggers an interaction between the graphs to update the supervision labels τ_n stored in each mission node (Fig. 5b). We reproject the footprint track and corresponding traversability scores τ onto all the images of the mission nodes that are within the range of the supervision graph. This ensures the information we reproject stays locally consistent in spite of potential state estimator drift.

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). Then we obtain pairs of per-segment features $\mathbf{f}_n$ and traversability score τ_n for each mission node ready for training.

Fig. 5 illustrates the supervision and mission graphs, and their interaction to generate supervision signals online.

E. Traversability and Anomaly Learning

We aim to 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 allows us to predict the traversability of the scene in front of the robot from a full image by inferring only about those segments. Additionally, we explicitly model the uncertainty about the unvisited (and hence, unlabeled) areas by using anomaly detection techniques to bootstrap a confidence estimate. Our formulation also deals with non-stationary data distributions induced by continuously updating the training data and model weights.

First, we will elaborate on how a confidence score for a segment is obtained; and then we will describe the traversability estimation task which takes as input the confidence and is jointly trained.

1) Confidence Estimation: To obtain a segment-wise confidence estimate we aim to learn the distribution over all traversed segment features $\mathbf{f}_n$. A encoder-decoder network $f^{\theta_r}_{\text{reco}}$ is trained to compress the segment feature $\mathbf{f}_n$ into a low dimensional latent space and consecutively 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:

$$\mathcal{L}_{\text{reco}}(\mathbf{f}_n) = \delta_{\tau_n \neq 0} \frac{1}{E} \sum_{e} \| f^{\theta_r}_{\text{reco}}(\mathbf{f}_{n,e}) - \mathbf{f}_{n,e} \|^2, \qquad (3)$$

where $\delta_{\tau \neq 0}$ is 1 if the segments feature traversability score τ_n is not zero, and 0 otherwise; this ensures that the network only learns to reconstruct the embeddings that are labeled in an anomaly detection fashion. As a consequence, 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. Since the network is trained online, this results in a multi-modal reconstruction loss distribution that evolves over time, as shown in Fig. 6.

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):

$$n_{\text{trav}} = \sum_{\mathbf{f}} \delta_{\tau_n \neq 0}, \tag{4}$$

$$\mu_{\text{pos}} = \frac{1}{n_{\text{trav}}} \sum_{\mathbf{f}\,:\,\tau_n \neq 0} \mathcal{L}_{\text{reco}}(\mathbf{f}_n), \tag{5}$$

$$\sigma_{\text{pos}} = \sqrt{\frac{1}{n_{\text{trav}}} \sum_{\mathbf{f}\,:\,\tau_n \neq 0} (\mathcal{L}_{\text{reco}}(\mathbf{f}_n) - \mu_{\text{pos}})^2} \tag{6}$$

We set the segment confidence to 1 if the loss of the segment is smaller than μ_{pos} and otherwise to 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.

2) Traversability Estimation: A small network $f^{\theta_t}_{\text{trav}}$ with a single channel output is trained to regress on the provided segment traversability score τ. For the untraversed segments with unknown traversability score we follow a conservative approach and we assume it to be zero $\tau = 0$, though we use the confidence score to scale their overall contribution. The loss for traversability estimation is computed using the confidence-weighted MSE:

$$\mathcal{L}_{\text{trav}}(\mathbf{f}) = \delta_{\tau_n = 0} \sum_n (1 - c(\mathbf{f}_n)) \, \|f^{\theta_t}_{\text{trav}}(\mathbf{f}_n) - 0\|^2 + \delta_{\tau_n \neq 0} \sum_n \|f^{\theta_t}_{\text{trav}}(\mathbf{f}_n) - \tau_n\|^2.$$

Effectively, for segments where a traversability score is available by interaction the MSE is computed. For unlabeled segments, the traversability is assumed to be zero but weighted based on the confidence score. Areas similar to the one traversed should be assigned a $c(\mathbf{f})$ close to 1, therefore contributing insignificantly to the total loss. On the other hand anomaly areas (never traversed before, low $c(\mathbf{f})$ score) induce a high loss if predicted with a high traversability score by f_{trav}.

As we aim to provide the estimated traversability as input for a local planning system, it is desired to automatically define a threshold to determine the traversable and untraversable areas. We propose a strategy to select the 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

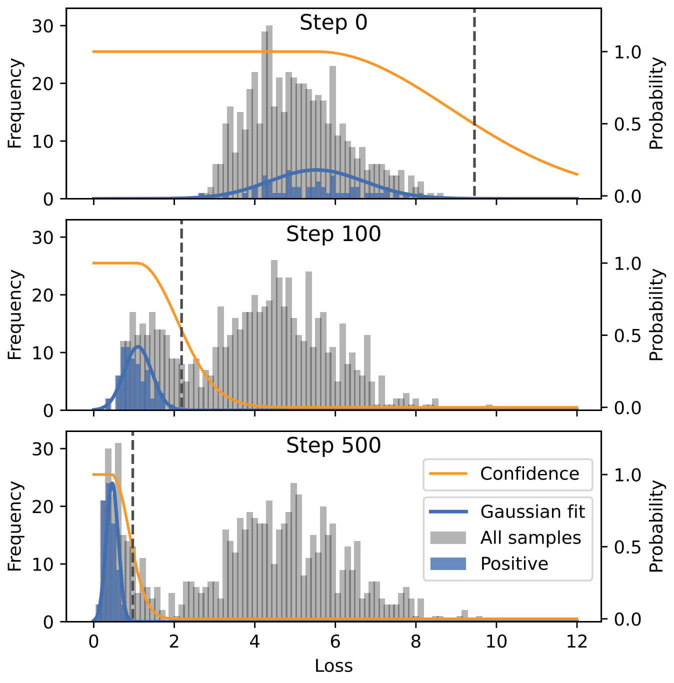

Fig. 6: Histogram of the reconstruction loss $\mathcal{L}_{\text{reco}}$ distribution for all segments within a batch at optimization step 0, 100, and 500. Traversed segments (positive samples) are shown in blue and non-traversed segments in grey. We observed that the total loss distribution becomes bi-modal as the training develops, which we use to determine a threshold to scale the confidence estimate (see Sec. III-E1). At 500 steps all segments with a reconstruction loss $\mathcal{L}_{\text{reco}}$ over 2 are identified as fully anomalous (unconfident) and therefore induce a high loss in the traversability score when identified as traversable. The vertical grey dashed line indicates the decision boundary if a segment is detected as traversable or untraversable by the anomaly detection.

segments as positive labels. Then, we decide on the traversability threshold only by setting the desired False Positive Ratio (FPR), though other metrics such as the Youden's index can also be used.

3) Implementation details: In our implementation, $f^{\theta_r}_{\text{reco}}$ and $f^{\theta_t}_{\text{trav}}$ are implemented by a two-layer Multi-Layer Perceptron (MLP) 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:

$$\mathcal{L}_{\text{total}}(\mathbf{f}) = w_{\text{trav}} \mathcal{L}_{\text{trav}}(\mathbf{f}) + w_{\text{reco}}\, \mathcal{L}_{\text{reco}}(\mathbf{f}). \tag{8}$$

with w_{trav} and w_{reco} allowing to weigh the traversability and reconstruction loss respectively. We used Adam [20] 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; 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 $k_\sigma = 2$, $w_{\text{trav}} = 0.03$,

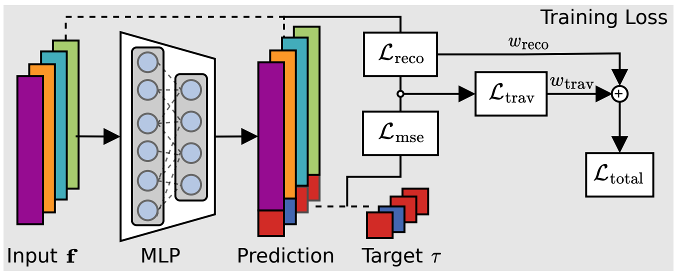
Training Loss
$\mathcal{L}_{\text{reco}}$
w_{reco}
$\mathcal{L}_{\text{trav}}$
w_{trav}
$\mathcal{L}_{\text{mse}}$
$\mathcal{L}_{\text{total}}$
Input $\mathbf{f}$
MLP
Prediction
Target τ

Fig. 7: Network architecture and per-segment losses used for online training.

$w_{\text{reco}} = 0.5$ and use a maximum FPR of 0.15 to determine the traversability threshold.

In Sec. V-C, we present and evaluate different design choices for learning methods, features and architectures. We also evaluate the advantages of our loss formulation compared to other common approaches for traversability, such as anomaly detection (learning a distribution over positive samples) and regression on the traversability score without modeling the uncertainty.

IV. Closed-loop Integration

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.

A. Local terrain mapping

To map the environment surrounding the robot we used an open-source terrain mapping framework [26] to efficiently obtain a robot-centric 2.5D elevation map from the onboard depth cameras and LiDAR sensing. We extended this framework in Erni et al. [8] 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.

B. 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 [10] was used to obtain a Signed Distance Field (SDF), which represents the distance to the closest untraversable object. We implemented a local planner method based on Mattamala et al. [23], which exploits the SDF and the local goal to generate a $\mathrm{SE}(2)$ 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 Miki et al. [25], which is able to traverse rough terrain typically inaccessible to wheeled robots.

C. 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.

V. Experiments

A. Platform Description

For our experiments we used an ANYbotics ANYmal C legged robot. The robot is equipped with an additional NVidia Jetson Orin AGX to run WVN onboard, which was implemented in pure Python code using PyTorch [28] and ROS 1 [29].

The main sensing input for our system are monocular, wide Field of View (FoV) color images from a single global shutter Sevensense Alphasense Core camera. Additionally, the default state estimator provides $\mathrm{SE}(3)$ pose and body velocity measurements. The LiDAR and depth cameras available on the robot were only used for the local terrain mapping module as described in Sec. IV-A.

B. Real-world deployments

We executed different deployments to validate WVN in different environments. The first experiments highlight adaptation to new environments and the advantages of the visual traversability estimation for local planning tasks. The last two experiments demonstrate autonomous navigation among obstacles and fully autonomous large-scale path following.

1) Fast adaptation on hardware: Our first experiment involved teleoperating the robot around 3 loops of a park environment walking on grass and dirt, on open areas and around trees. The goal was to evaluate the fast adaptation capabilities of WVN while running on the robot.

Fig. 8 illustrates the main outcomes of the experiment, showing that the system learned to predict robot-specific traversability over the 3 loops. In particular, section **(a)** shows how the robot starts with a very poor segmentation after 9 steps of training (21 s), this greatly improves after 800 steps (2 min) where it can correctly segment the dirt as traversable terrain while keeping the tree untraversable. Similar behavior occurs in section **(b)** in which the segmentation is conservative at the beginning but it extends across the other grass patches in

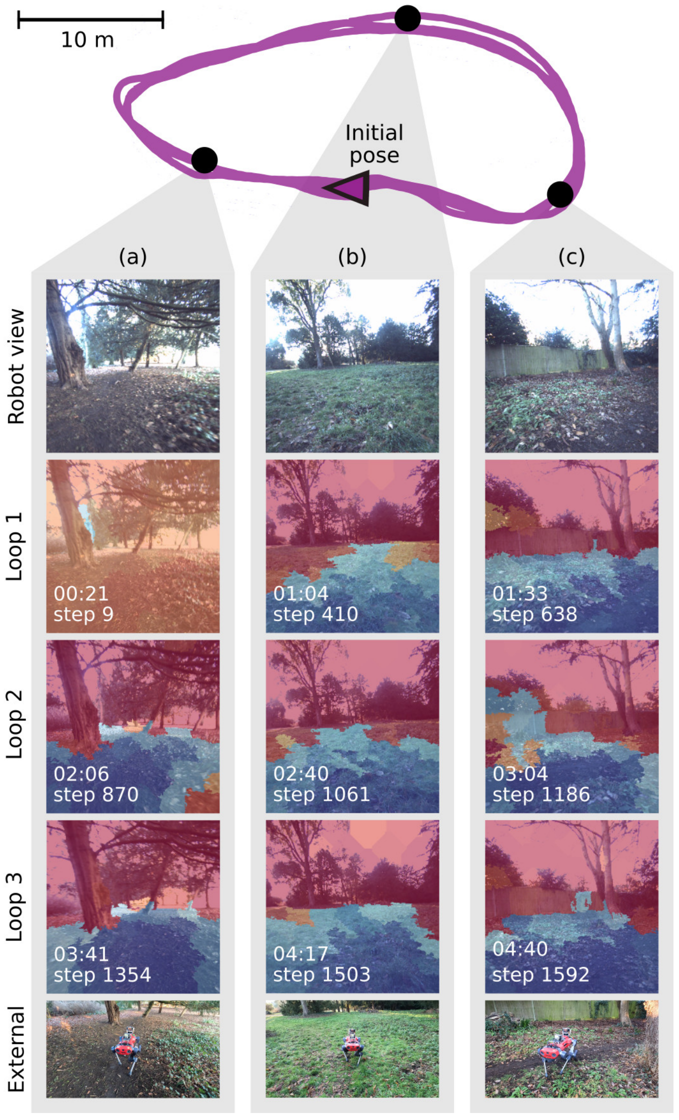

Fig. 8: Adaptation on real hardware: We tested the online adaptation capabilities of our system by teleoperating the robot to complete 3 loops in a park (top, route shown in 🟪). The columns show different parts of the loop (a,b,c); each row displays the improvement of the traversability estimate over time and training steps.

later iterations. Section **(c)** also illustrates some issues related to the SLIC segmentation, as some segments of the wooden wall (step 1186) are incorrectly clustered with patches of the grass, which is not observed in the other captures.

2) Benefits of visual traversability vs geometric methods: Our second experiment aimed to illustrate the advantages of visual traversability estimation in challenging natural environments. Similarly to the previous experiment, we teleoperated the robot — but in a forest with high grass, loose branches, and bushes. Fig. 9, bottom right, shows a representative shot of the experiment, a robot's view input image and WVN's prediction, which illustrates the challenges of the environment for both vision and geometry based approaches.

To compare the different traversability methods, we used the terrain mapping module described in Sec. IV-A, as it allowed us to compare geometry-only and visual traversability. In particular, we compared against two geometric methods that are real-time capable and have been used in previous literature:

- Geometric method based on heuristics such as height and slope of the terrain [38].
- Geometric method based on a learned model of traversability, which is part of the terrain mapping system [26].
- Visual traversability provided by WVN, raycasted onto the terrain map.

The geometric methods only require an elevation representation of the surface, and can directly determine traversability from the 2.5D geometry. For WVN we executed a training procedure driving the robot around the environment for a few minutes first.

Fig. 9 illustrates the output *traversability map* obtained by all the methods (bottom), as well as the corresponding SDFs generated from them. (top) The geometric methods correctly determine the trees as untraversable areas, as they are based on the 2.5D representation. Our system is also able to successfully discriminate the trees, confirming the findings observed in Sec. V-B1. However, the important advantages of our method are observed in high-grass areas, which are represented as elevation spikes in the map that are classified as untraversable by the geometric approaches. WVN correctly characterized the capabilities of the robot to successfully traverse the terrain, as demonstrated by the human operator.

When comparing the SDFs such differences become more evident, as all the areas with low traversability scores become obstacles. Our system correctly determines that all the grass patches are traversable, hence the SDFs displays the right classification and disregards the geometry. The only limitation of our current method is that we use just a single camera and can only predict traversability for the areas in sight. To ensure safety, unknown areas are assumed to be untraversable but future work will take advance of the robot's other cameras.

3) Point-to-point autonomous navigation between trees: After validating our approach in teleoperated settings, we executed closed-loop navigation tasks to demonstrate WVN can easily adapt to a new environment, and the learned traversability estimate can be used to deploy the robot autonomously.

We taught the robot to navigate in a woodland area containing dirt, high grass, and trees. A human operator drove the robot for 2 min through loose dirt and grass — an area that can be easily traversed by the legged platform. Then we commanded the local planner to execute autonomous point-to-point navigation avoiding obstacles, only using the visual traversability for closed-loop planning Sec. IV.

Fig. 10 illustrates the scene used for the experiment and the trajectories used for training and testing autonomous navigation. The robot successfully managed to reach 8 out of 8 goals, where the human operator deliberately chose targets behind trees to challenge the system. This was achieved even

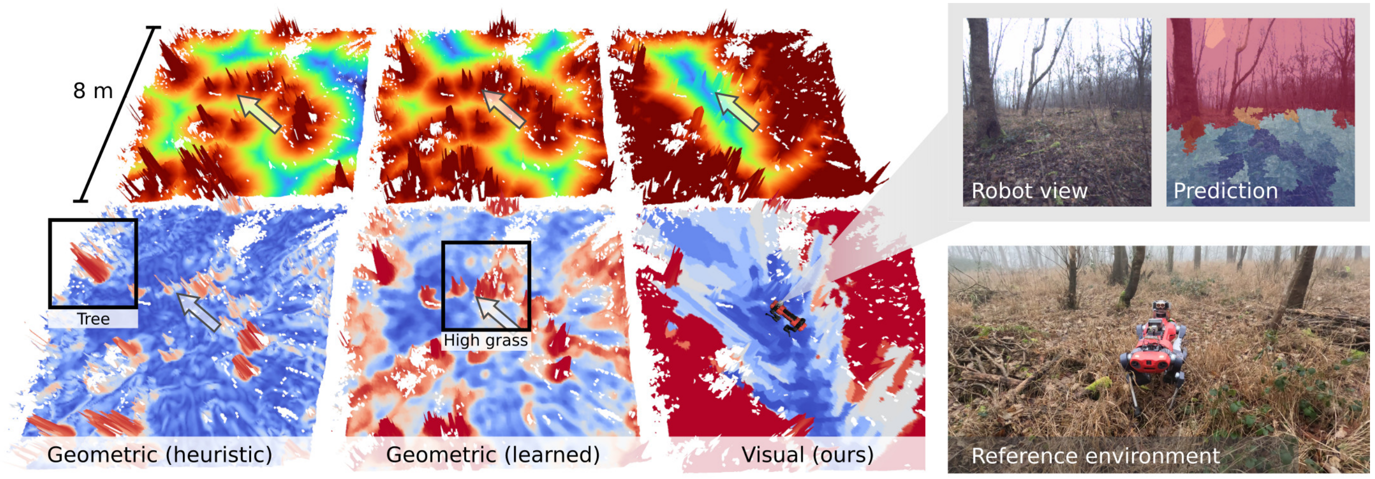

Fig. 9: Visual vs geometric traversability: Illustration of traversability map (bottom row) and corresponding SDF (top row) for three different traversability estimation methods applied to the same terrain patch. Our visual traversability estimate provides clear advantages for local planning compared to geometric methods, where the latter get heavily affected by traversable high grass or branches (bottom row). This is evident when comparing the SDF's, where geometry-based methods are more sensitive to the spikes produced by high grass areas (top row).

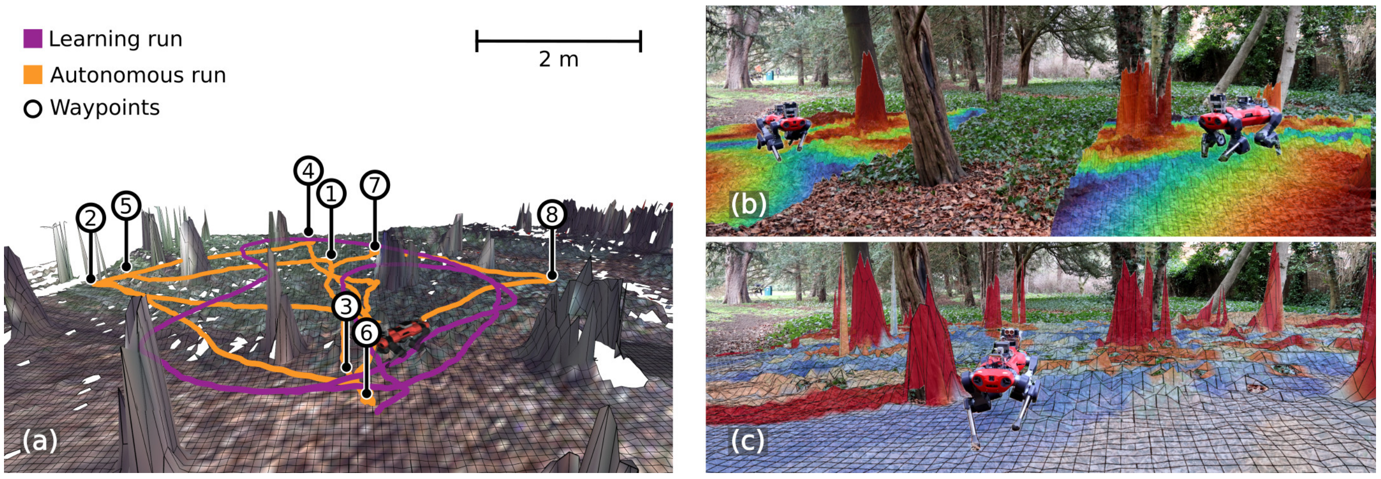

Fig. 10: Point-to-point autonomous navigation: (a) After teleoperating the robot for 2 min (path shown in 🟪), we successfully achieved autonomous navigation in a woodland environment (path shown in 🟧). (b) Some of the SDFs generated from the predicted traversability during autonomous operation. (c) Global 2.5D reconstruction of the testing area and predicted traversability, generated in post-processing to illustrate the capabilities of our approach.

though neither geometry nor any additional assumptions about the environment were used during training.

We also show some examples of the SDFs generated during operation used by the local planner in subfigure (b), which indicate the trees as obstacles. Lastly, in processing we fused the predicted traversability measures into a complete map in subfigure (c), which correctly aligned with the trees. However, we did observe some obstacle artifacts due to limitations of the approach, namely the use of a single camera for the predictions, the coarse segmentation from SLIC, and the raycasting process, which we further discuss in Sec. V-D.

4) Kilometer-scale autonomous navigation in the park: As a last experiment, we demonstrated that WVN can also be used to achieve preference-aware path-following behavior as a result of the human demonstrations and the online learning capabilities of the system.

We executed 3 experiments to demonstrate this in a park. Similarly to our previous experiments, we trained the system for less than 2 min along the footpath. However, we then disabled the learning thread to ensure that the predicted traversability strictly mimics the human preference during the demonstration run. The goal for the robot to follow is given by the *smart carrot* module described in Sec. IV-C, which autonomously guided the robot forward along the path.

In the 3 runs the robot was able to follow the path for hundreds of meters — mostly staying in the center of the path, avoiding grass, bushes, benches, and pedestrians. Fig. 11 shows the trajectories followed in each run, starting from different points in the footpath. For runs 1 and 3 we used the same parameters, $k_\sigma = 2$ and FPR$= 0.15$. In run 2 we

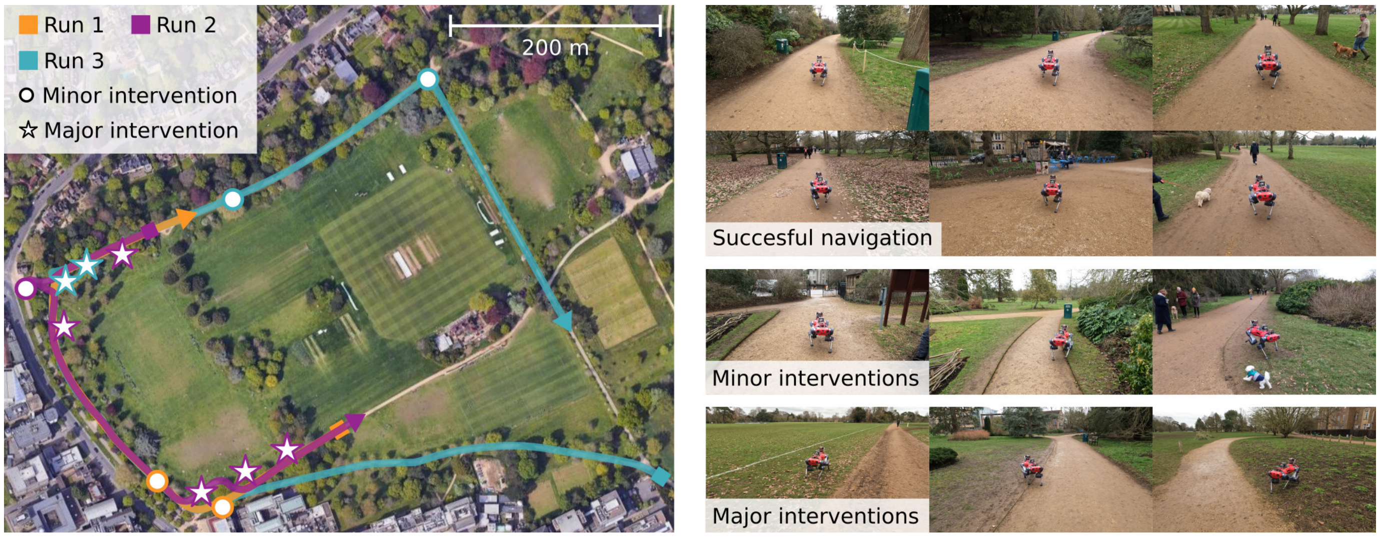

Fig. 11: Kilometer-scale navigation: We deployed our system to learn to segment the footpath of a park after training for a few steps. We executed 3 runs starting from different points in the park: 🟧 *run 1* (0.55 km), 🟪 *run 2* (0.5 km), and 🟦 *run 3* (1.4 km). Minor interventions were applied to guide the robot in intersections; major interventions ($\star$) were required for some areas when the robot miss-classified muddy patches for the path.

relaxed the parameters to $k_\sigma = 3$ and FPR$= 0.3$, producing a less conservative behavior that drove the robot to other visually similar areas in the park (very muddy grass) requiring manual intervention to correct the heading. When the robot approached an intersection we adjusted, if necessary, the heading to follow the desired footpath.

Overall, we achieved autonomous behavior that would have been difficult to achieve using only geometry, as the path boundaries were often geometrically not distinguishable. On the other hand, instead of training and using a semantic segmentation system to learn *all* the possible traversable classes in the park (pavement, gravel path, roadway or grass), we showed that this short teleoperated demonstration of the gravel footpath was enough for WVN to generate semantic cues to achieve the desired path following behavior.

C. Offline validation via ablation studies

To complement the results of our hardware experiments and validate our design decisions, we performed different ablation studies of the individual components of WVN. All the experiments were executed offline on an Nvidia RTX3080 Laptop GPU with Intel i7-11800H CPU.

1) Dataset overview: For offline analysis we used 3 large-scale datasets of new areas not used for the real experiments:

- *Hilly:* a hillside with dense vegetation and fruit trees.
- *Forest:* a fir forest with hiking paths.
- *Grass:* a grassland area with moderate inclines and varying vegetation surrounding a small lake.

The datasets were also recorded with a teleoperated ANYmal C platform and similar sensing setup to the previous experiments. Fig. 12 shows aerial views of the paths that were used for data collection, as well as some samples of the specific areas that were traversed during this operation.

We organized the collected data into training, validation, and testing data, which is summarized in Tab. II. The longest sequence recorded in each site (Fig. 12, shown in purple 🟪) is used for training and validation purposes. The first 80% of the sequence are used to generate training data, with the remaining 20% kept for validation. The remaining sequences of each scene are subsampled and exclusively used for testing.

Regarding the labels, we manually segmented images from the test split into traversable (1) and untraversable (0) classes, which we named *ground truth labels* (*GT*). These binary labels reflect the intuition of an expert robot operator on which places are safely accessible for the robot, and were used for quantitative assessment of the design decisions. While simulation-based evaluations could provide a precise traversability score, we disregarded it because of the limitation of sim-to-real transfer of vision-based methods. Human supervision was straightforward and could incorporate more subtle risks and preferences.

On the other hand, we also used the labels generated by executing WVN over the sequences using the self-supervision approach, which we name *SELF*. Since the *GT* labels were binary due to intrinsic challenges of producing ground truth continuous traversability signals, we also binarized the *SELF* labels for a fair comparison.

Lastly, even though we only had access to binary ground truth labels we did not change the regression formulation presented in Sec. III-E, as we could also obtain a binary output by thresholding the continuous signal proposed by our system. This resulted in a classification task that could be evaluated using metrics such as Accuracy (Acc).

2) Evaluation method: For all of our experiments we trained and tested in the same environment using the splits previously presented, unless stated otherwise. Regarding the

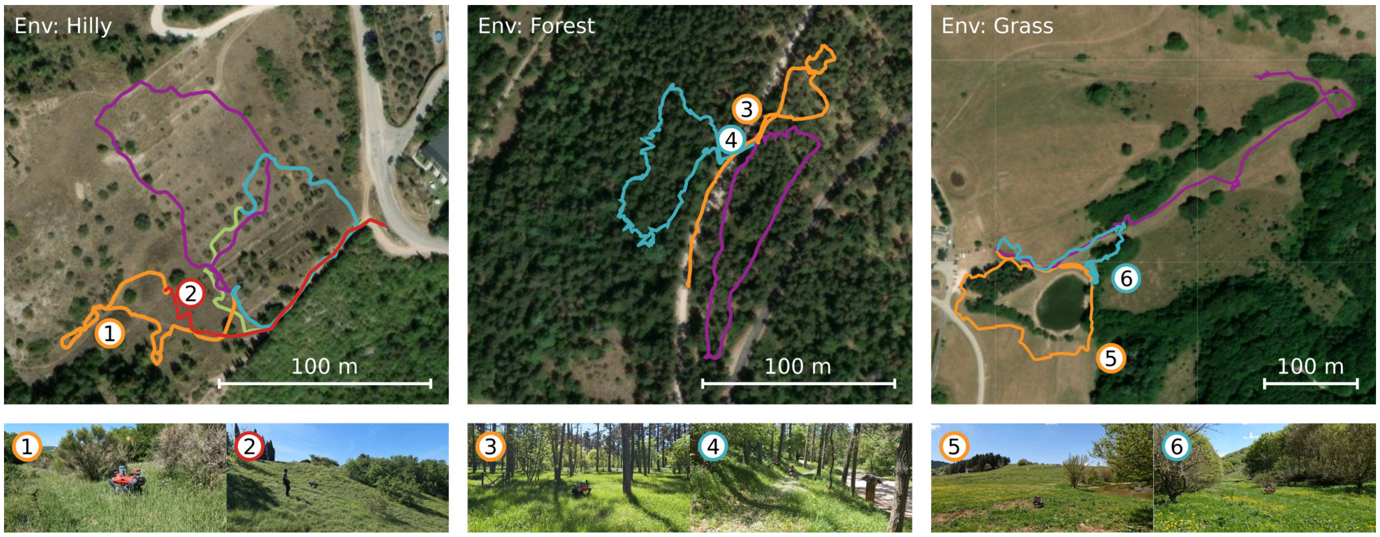

Fig. 12: Aerial views of the 3 environments used for offline testing of our system, illustrating the paths used for data collection and scene examples. The purple 🟪 trajectories are used for training and the remaining for validation.

TABLE II: Dataset Overview

Env	Split	Duration	Distance	# Traj	# Image	Label
Hilly	Train	512.4 s	262 m	1	920	*SELF*
	Val	121.1 s	66 m	1	230	*SELF*
	Test	1202.2 s	840 m	4	55	*GT*
Forest⋆	Train	402.1 s	606 m	1	991	*SELF*
	Val	134.0 s	151 m	1	247	*SELF*
	Test	970.5 s	896 m	2	41	*GT*
Grass	Train	860.3 s	857 m	1	2050	*SELF*
	Val	242.5 s	214 m	1	512	*SELF*
	Test	2196.2 s	1224 m	3	113	*GT*

⋆ Length measured using RTK-GPS and may not reflect the real-distance traversed within the forest.

Env	Metric	RF	SVC-RBF	SVC-Poly	WVN
Hilly	*GT*	71.36 ± 0.0	65.46 ± 0.53	73.96 ± 1.61	**81.05** ± 1.11
	SELF	88.32 ± 0.0	87.51 ± 0.25	84.81 ± 0.31	78.58 ± 0.82
Forest	*GT*	**83.07** ± 0.44	74.66 ± 0.55	76.28 ± 1.34	**82.45** ±1.10
	SELF	81.95 ± 0.44	90.19 ± 0.20	88.09 ± 0.53	85.26 ± 1.24
Grass	*GT*	59.16 ± 0.0	63.64 ± 0.33	69.02 ± 2.00	**78.21** ± 2.39
	SELF	88.14 ± 0.0	87.79 ± 0.14	86.74 ± 0.39	82.60 ± 0.29

TABLE III: Learning Method: Traversability Accuracy of Random Forest (RF), Support Vector Classifier (SVC), and WVN with respect to the binary ground truth labels *GT* and self-supervised labels *SELF* on the ablation environments.

metrics, we evaluated our system in a binary classification setting due to the limitations of the *GT* labels previously discussed. Hence, we reported the Acc for all of our experiments. The accuracy is computed in image space (i.e. pixel-wise) as opposed to segment-wise, which accounts for misclassifications induced by image segments containing both traversable and untraversable terrain.

Lastly, all compared neural network models are trained for 1000 steps with 5 different random seeds, and we report confidence intervals for all the metrics.

3) Study 1: Learning Methods: Our first study compared our approach to *classical* machine learning methods previously used for traversability estimation, namely Random Forest (RF) [33] and Support Vector Classifier (SVC) [2]. For SVC we followed their work and use a polynomial kernel of degree 2 and Radial Basis Function (RBF) kernel. All methods are trained using the DINO-ViT features.

Tab. III summarizes the main results of this study. We observed that *our* method based on an MLP performs consistently good across all three environments with respect to the *GT* labels (see Tab. III). Interestingly, the RF is competitive and still performs strongly across scenes, specially well in the *Forest* scene. The results confirm that our chosen model overall outperforms machine learning methods previously used in the literature, while allowing us to continuously adapt the model using gradient descent during the mission.

4) Study 2: Training Objective: We then studied the impact of various training objectives on the model performance. In particular, we compared four approaches:

- *Trav*: We trained our network to directly regress on the traversability by assuming all untraversed segments are untraversable: no reconstruction loss $w_{\text{reco}} = 0$, with full confidence $c(\mathbf{f}) = 1$, and without self-supervised τ_{thr} thresholding.
- *Fixed threshold*: We fixed the traversability threshold τ_{thr} to a value of 0.5.
- *Anomaly detection*: We used the confidence score to classify features with a confidence over 0.5 as traversable.
- WVN: Our full method.

Our proposed method consistently outperforms the other settings (Tab. IV). A significant performance increase (+5.17%) was achieved over the *Trav* simplest traversability setting by adding the confidence-weighted traversability loss formulation

Env	Metric	Trav	Fixed-Threshold	Anom	WVN
Hilly	*GT*	65.46 ± 0.53	73.96 ± 1.61	72.92 ± 0.82	**81.05** ± 1.11
	SELF	87.51 ± 0.25	84.81 ± 0.31	69.21 ± 0.85	78.58 ± 0.82
Forest	*GT*	74.66 ± 0.55	76.28 ± 1.34	70.31 ± 1.12	**82.45** ±1.10
	SELF	90.19 ± 0.20	88.09 ± 0.53	68.37 ± 1.44	85.26 ± 1.24
Grass	*GT*	63.64 ± 0.33	69.02 ± 2.00	77.64 ± 0.44	**78.21** ± 2.39
	SELF	87.79 ± 0.14	86.74 ± 0.39	75.29 ± 0.21	82.60 ± 0.29

TABLE IV: Training Objective: Traversability Accuracy for different learning objectives with respect to the binary ground truth labels *GT* and self-supervised labels *SELF* on the ablation environments. Refer to *Study 2: Training Objective* for further details.

	Architecture	Param	Time	*GT*	*SELF*
Hilly	DINO-ViT	21M	17.0 ms	**65.46** ± 0.53	87.51 ± 0.25
	DINO-ReN	23.5M	15.9 ms	64.05 ± 0.08	86.71 ± 0.03
	EffNet-B4	17.5M	26.1 ms	62.55 ± 0.07	86.09 ± 0.02
	ResNet-50	23.5M	15.9 ms	61.87 ± 0.08	85.37 ± 0.02
	SIFT	0	-	58.78 ± 0.73	82.80 ± 0.03
Forest	DINO-ViT	21M	17.0 ms	**74.66** ± 0.55	90.19 ± 0.20
	DINO-ReN	23.5M	15.9 ms	74.14 ± 0.24	89.12 ± 0.21
	EffNet-B4	17.5M	26.1 ms	73.00 ± 0.76	89.63 ± 0.29
	ResNet-50	23.5M	15.9 ms	72.71 ± 0.22	88.49 ± 0.10
	SIFT	0	-	60.94 ± 0.00	83.27 ± 0.00
Grass	DINO-ViT	21M	17.0 ms	63.64 ± 0.33	87.79 ± 0.14
	DINO-ReN	23.5M	15.9 ms	**67.08** ± 0.36	88.15 ± 0.13
	EffNet-B4	17.5M	26.1 ms	61.01 ± 0.78	85.91 ± 0.00
	ResNet-50	23.5M	15.9 ms	62.99 ± 0.03	85.23 ± 0.12
	SIFT	0	-	57.61 ± 0.91	83.79 ± 0.02

TABLE V: Comparison of feature extraction backbones.

Training	**Hilly**	**Forest**	**Grass**
Hilly	**81.05** ±1.11	82.14 ± 1.78	**82.14** ± 0.63
Forest	75.86 ± 2.18	**82.45** ±1.10	75.80 ± 2.80
Grass	77.49 ± 4.36	73.22 ± 6.38	78.21 ±2.39

TABLE VI: Scene Adaptation: Traversability Accuracy with respect to the *GT* labels. Each row corresponds to a training run on the specific environment and testing on all environments.

(*Fixed-threshold*). Generally, in the *Trav*-setting more regions are classified as untraversable leading to an over-conservative traversability estimation, which performed well on the *SELF*-labels but does not reflect the *GT* traversability. Further improvement of 7.33% can be achieved by adding the online traversability threshold scaling. Here it is important to mention that a fixed threshold is insufficient during deployment given the online adaptation of the network. Lastly, the experiment using only *Anom* detection performs reasonably well, suggesting that a meaningful confidence score was learned across various environments. This indicates that the learned anomaly detection is useful for guiding the traversability learning of our method.

5) Study 3: Features: The quality of extracted segment features can have a significant impact on the performance of traversability prediction, given that we only consider a fixed feature extraction backbone. Hence, we evaluated the performance of the selected self-supervised pre-trained DINO-ViT backbone against popular residual network architectures pre-trained on ImageNet (ResNet-50, EfficientNet-B4), and also trained using self-supervised learning, such as ResNet-50 trained with DINO (DINO-ReN). We also compared against other classical features, such as dense SIFT [22] features over RGB channels. We fed the features to the *trav*-setting model introduced in the previous study, to isolate the impact of the feature extractor on the traversability estimation performance from other procedures. We present the model parameter count for each feature extractor and inference time of all network architectures on a Jetson Orin in Tab. V. The features generated by methods using self-supervised pre-training clearly outperformed pre-trained models on ImageNet, even when using the same architecture, aligning with the findings of Caron et al. [5]. The short inference time measured on the Orin board also validates the choice of DINO-ViT as the backbone suitable for WVN.

6) Study 4: Scene Adaptation: We evaluated the performance of WVN when trained on one environment and tested on all the others, to test the necessity for online adaptation. Tab. VI shows the resulting accuracy for each scene combination. We observed that in general the best performance is achieved when testing on the same training environment as expected, dropping otherwise. The model trained on *Hilly* performs overall the best and showed specially good performance on *Grass*, which we suspect is due to the visual similarity of the scenes (see Fig. 12).

In general, we remark that even though the robot was deployed within scenes featuring similar semantic classes (e.g. trees or high grass), on the same day and within a few kilometers radius, the performance still degraded. This suggests even worse performance drops for changing seasons or urban to natural environment scene changes. We argue that even though this can be hypothetically mitigated by increasing the amount of training data, this is costly, and online adaptation provides a practical solution to enable the deployment of robots in new or changing environments.

7) Study 5: Adaptation Speed & Dataset Size: For our final study we investigated how fast can WVN adapt to the new environments and how many data samples are needed. To examine this we designed an experiment in which we trained the network for 1000 steps for different training dataset sizes, ranging from 10% to 100% of the original size. We measured the accuracy each 2nd step and every 10% increment of the dataset size.

As a result, we obtained heatmaps displaying the performance evolution across these 2 variables, shown in Fig. 13. For all environments starting from a randomly initialized network, we observed that good performance can be achieved within 200 steps. We argue that this is due to use of segments: adding a single image provides 100 new training samples for the network. During continuous training, we also observed some slight fluctuation with respect to the test accuracy. Fig. 14

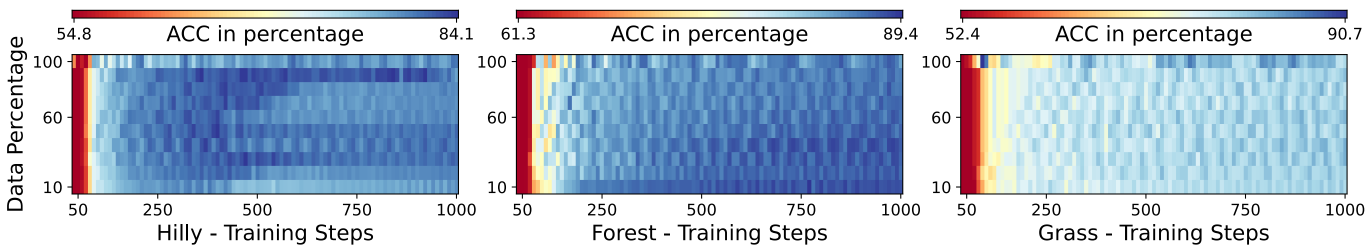

Fig. 13: Adaptation Speed vs Dataset Size: The performance measured by the accuracy increases over training as expected. In the limit case of training for a few steps (< 50), the performance is equally degraded — independent of the dataset size. A small dataset size is sufficient for good performance. Please observe the different color scales for each environment. In the *Grass* environment the color scale is distorted by an outlier when using 100% of the data after 70 steps.

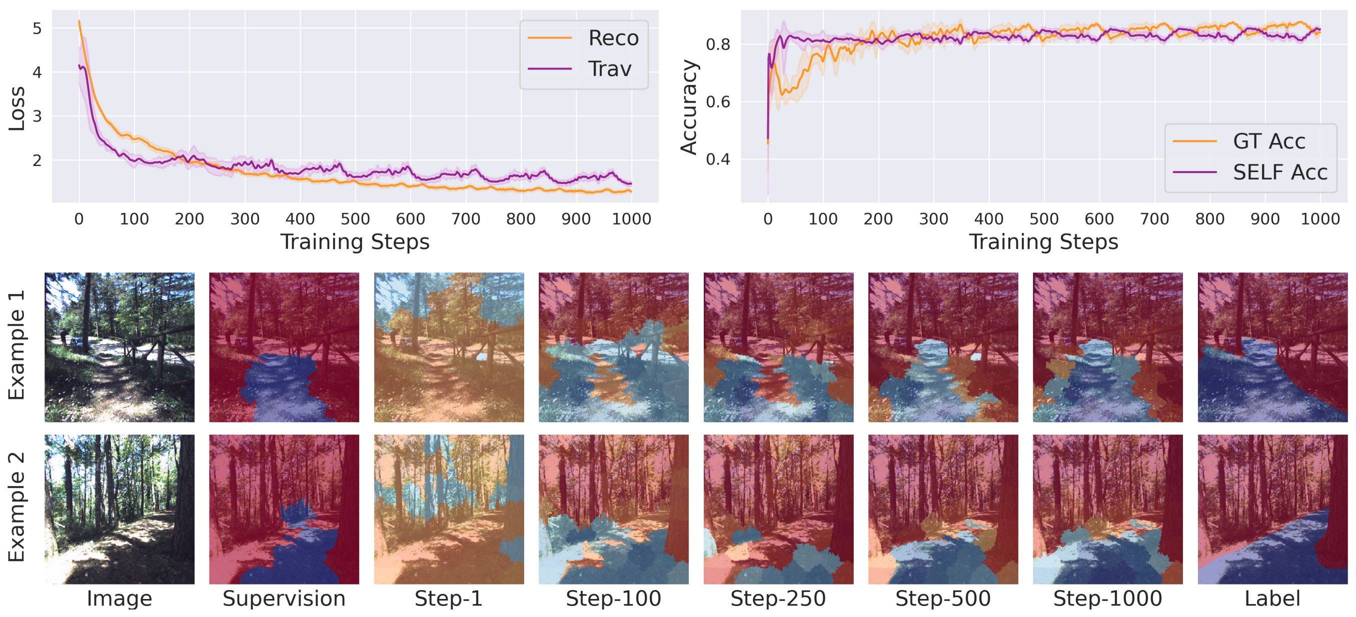

Fig. 14: Training Process: We detail the incremental training process executed by WVN in terms of the loss (top left), accuracy (top right), and visual examples (bottom).

shows the training loss accuracy over time, illustrating some of the fluctuating behavior, as well as example images of the output segmentation over training.

D. Limitations

We have demonstrated that WVN can learned a traversability estimate online, allowing immediate deployment of robots in new environments. However, we observed some limitations during our ablation studies and field deployment.

- WVN runs at 2.5 Hz on the NVidia Orin board with unoptimized code. Improving the efficiency of the pipeline would allow for faster training and better autonomous navigation performance.
- The use of superpixels to reduce the computational complexity limits the accuracy of the output segmentation, as the SLIC segments are only similar from a color-space sense, consequently affecting the computation of features per segment by averaging semantically different features.
- While the use of velocity tracking error was a simple proxy for the traversability score, it does not fully characterize the different interactions the robot can have with the environment. Further investigation is required to determine alternative metrics that can be more suitable for specific platforms.
- For closed-loop integration we projected the traversability prediction onto a local terrain map, which allowed for a straightforward integration with the local planner. However, this presented important drawbacks due to perspective projection, limited FoV due to single camera usage, and raycasting on an inaccurate 2.5D map that required additional filtering stages.
- Lastly, our system could be also framed within the continual learning paradigm. While we do not address it explicitly, we believe that WVN could greatly benefit from the advances of continual learning — not only to adapt within a single mission — but between different test environments.

VI. Conclusion

We presented Wild Visual Navigation (WVN), a system that leverages the latest advances in pre-trained self-supervised networks with a scheme to generate supervision signals while a robot operates, to achieve online, onboard visual traversability estimation. The fast adaptation capabilities of our system allowed us to easily deploy robots for navigation tasks in new environments after just a few minutes of learning from human demonstrations. We validated WVN through different ablation studies and real-world experiments, illustrating its fast adaptation capabilities, the consistency of its traversability prediction for local planning, and 1.4 km closed-loop navigation experiments in natural scenes. Our experiments show that WVN can enable autonomous robot navigation by learning from small data *in the wild*. We aim to tackle the current limitations of our system by exploring data-driven self-supervised methods for segment extraction, possibly mitigating artifacts induced by segments containing traversable and untraversable terrain. We also plan to extend the current implementation to multiple cameras — allowing the system to learn from different inputs and estimate traversability in different directions for more complex local planning. For future work specific to legged systems capable to negotiate challenging terrain, we aim to further close the loop between WVN's traversability prediction and feedback provided directly by the locomotion policy about the traversability of the terrain.

Acknowledgments

This work was supported by the Swiss National Science Foundation (SNSF) through project 188596, the National Centre of Competence in Research Robotics (NCCR Robotics), the European Union's Horizon 2020 research and innovation program under grant agreement No 101016970, No 101070405, and No 852044, and an ETH Zurich Research Grant. Jonas Frey is supported by the Max Planck ETH Center for Learning Systems. Matias Mattamala is supported by the National Agency for Research and Development (ANID) / DOCTORADO BECAS CHILE/2019 - 72200291 and NCCR Robotics. Maurice Fallon is supported by a Royal Society University Research Fellowship.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012. III-B

[2] Max Bajracharya, Andrew Howard, Larry H. Matthies, Benyang Tang, and Michael Turmon. Autonomous off-road navigation with end-to-end learning for the lagr program. *Journal of Field Robotics*, 26(1):3–25, 2009. II-C, V-C3

[3] David M. Bradley, Jonathan K. Chang, David Silver, Matthew Powers, Herman Herman, Peter Rander, and Anthony Stentz. Scene understanding for a high-mobility walking robot. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1144–1151, 2015. II-B

[4] Chao Cao, Hongbiao Zhu, Fan Yang, Yukun Xia, Howie Choset, Jean Oh, and Ji Zhang. Autonomous Exploration Development Environment and the Planning Algorithms. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, page 8921–8928, 2022. II-A

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. In *International Conference on Computer Vision (ICCV)*, 2021. II-B, III-B, V-C5

[6] R. Omar Chavez-Garcia, Jérôme Guzzi, Luca M. Gambardella, and Alessandro Giusti. Learning Ground Traversability From Simulations. *IEEE Robotics and Automation Letters*, 3(3):1695–1702, 2018. II-A

[7] Timothy H. Chung, Viktor Orekhov, and Angela Maio. Into the Robotic Depths: Analysis and Insights from the DARPA Subterranean Challenge. *Annual Review of Control, Robotics, and Autonomous Systems*, 6(1), 2023. II-A

[8] Gian Erni, Frey Jonas, Miki Takahiro, Matias Mattamala, and Hutter Marco. MEM: Multi-Modal Elevation Mapping for Robotics and Learning. 2023. IV-A

[9] David D. Fan, Kyohei Otsu, Yuki Kubo, Anushri Dixit, Joel Burdick, and Ali-Akbar Agha-Mohammadi. STEP: Stochastic Traversability Evaluation and Planning for Safe Off-road Navigation. In *Robotics: Science and Systems*, 2021. II-A

[10] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance Transforms of Sampled Functions. *Theory of Computing*, 8(19):415–428, 2012. IV-B

[11] Jonas Frey, David Hoeller, Shehryar Khattak, and Marco Hutter. Locomotion Policy Guided Traversability Learning using Volumetric Representations of Complex Environments. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022. II-A

[12] Lu Gan, Jessy W. Grizzle, Ryan M. Eustice, and Maani Ghaffari. Energy-Based Legged Robots Terrain Traversability Modeling via Deep Inverse Reinforcement Learning. *IEEE Robotics and Automation Letters*, 7(4): 8807–8814, 2022. II-E

[13] Mateus V. Gasparino, Arun N. Sivakumar, Yixiao Liu, Andres E. B. Velasquez, Vitor A. H. Higuti, John Rogers, Huy Tran, and Girish Chowdhary. WayFAST: Navigation With Predictive Traversability in the Field. *IEEE Robotics and Automation Letters*, 7(4):10651–10658, 2022. I, II-C, III-D

[14] James J Gibson. *The Ecological Approach to Visual Perception*. Boston: Houghton Mifflin, 1979. I

[15] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning Long-range Vision for Autonomous Off-road Driving. *Journal of Field Robotics*, 26(2):120–144, 2009. I, II-F

[16] Nicolas Hudson, Fletcher Talbot, Mark Cox, Jason Williams, Thomas Hines, Alex Pitt, Brett Wood, Dennis Frousheger, Katrina Lo Surdo, Thomas Molnar, Ryan Steindl, Matt Wildie, Inkyu Sa, Navinda Kottege, Kazys Stepanas, Emili Hernandez, Gavin Catt, William Docherty, Brendan Tidd, Benjamin Tam, Simon Murrell, Mitchell Bessell, Lauren Hanson, Lachlan Tychsen-Smith, Hajime Suzuki, Leslie Overs, Farid Kendoul, Glenn Wagner, Duncan Palmer, Peter Milani, Matthew O'Brien, Shu Jiang, Shengkang Chen, and Ronald Arkin. Heterogeneous Ground and Air Platforms, Homogeneous Sensing: Team CSIRO Data61's Approach to the DARPA Subterranean Challenge. *Field Robotics*, 2(1):595–636, 2022. II-A

[17] Tianchen Ji, Arun Narenthiran Sivakumar, Girish Chowdhary, and Katherine Driggs-Campbell. Proactive Anomaly Detection for Robot Navigation With Multi-Sensor Fusion. *IEEE Robotics and Automation Letters*, 7(2):4975–4982, 2022. II-D

[18] Gregory Kahn, Pieter Abbeel, and Sergey Levine. BADGR: An Autonomous Self-Supervised Learning-Based Navigation System. *IEEE Robotics and Automation Letters*, 6(2):1312–1319, 2021. II-F

[19] Dongshin Kim, Jie Sun, Sang Min Oh, J.M. Rehg, and A.F. Bobick. Traversability Classification using Unsupervised On-line Visual Learning for Outdoor Robot Navigation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 518–525, 2006. I, II-C, II-F

[20] Diederick P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015. III-E3

[21] Honggu Lee, Kiho Kwak, and Sungho Jo. An Incremental Nonparametric Bayesian Clustering-based Traversable Region Detection Method. *Autonomous Robots*, 41(4): 795–810, 2017. II-F, III-B

[22] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. V-C5

[23] Matias Mattamala, Nived Chebrolu, and Maurice Fallon. An Efficient Locally Reactive Controller for Safe Navigation in Visual Teach and Repeat Missions. *IEEE Robotics and Automation Letters*, 7(2):2353–2360, 2022. IV-B

[24] Daniel Maturana, Po-Wei Chou, Masashi Uenoyama, and Sebastian Scherer. Real-time Semantic Mapping for Autonomous Off-Road Navigation. In *International Conference on Field and Service Robotics (FSR)*, pages 335 – 350, 2017. I, II-B

[25] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning Robust Perceptive Locomotion for Quadrupedal Robots in the Wild. *Science Robotics*, 7(62), 2022. I, IV-B

[26] Takahiro Miki, Lorenz Wellhausen, Ruben Grandia, Fabian Jenelten, Timon Homberger, and Marco Hutter. Elevation Mapping for Locomotion and Navigation using GPU. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2273–2280, 2022. IV-A, V-B2

[27] Hans Moravec and Alberto Elfes. High Resolution Maps from Wide Angle Sonar. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 2, pages 116–121, 1985. I, II-A

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 2019. V-A

[29] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2009. V-A

[30] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. Maximum margin planning. In *International Conference on Machine Learning (ICML)*, ICML '06, page 729–736, 2006. II-E

[31] Charles Richter and Nicholas Roy. Safe visual navigation via deep learning and novelty detection. In *Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017. doi: 10.15607/RSS.2017.XIII.064. II-D

[32] Adarsh Jagan Sathyamoorthy, Kasun Weerakoon, Tianrui Guan, Jing Liang, and Dinesh Manocha. Terrapn: Unstructured terrain navigation using online self-supervised learning. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 7197–7204, 2022. II-C, III-D

[33] Fabian Schilling, Xi Chen, John Folkesson, and Patric Jensfelt. Geometric and Visual Terrain Classification for Autonomous Mobile Navigation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2678–2684, 2017. II-B, V-C3

[34] Junwon Seo, Taekyung Kim, Kiho Kwak, Jihong Min, and Inwook Shim. Scate: A scalable framework for self- supervised traversability estimation in unstructured environments. *IEEE Robotics and Automation Letters*, 8 (2):888–895, 2023. II-D

[35] Amirreza Shaban, Xiangyun Meng, JoonHo Lee, Byron Boots, and Dieter Fox. Semantic Terrain Classification for Off-Road Autonomous Driving. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 619–629, 2022. II-B

[36] Lorenz Wellhausen, Alexey Dosovitskiy, René Ranftl, Krzysztof Walas, Cesar Cadena, and Marco Hutter. Where Should I Walk? Predicting Terrain Properties from Images via Self-Supervised Learning. *IEEE Robotics and Automation Letters*, 4(2):1509 – 1516, 2019-04. I, II-C, III-D

[37] Lorenz Wellhausen, René Ranftl, and Marco Hutter. Safe

Robot Navigation Via Multi-Modal Anomaly Detection. *IEEE Robotics and Automation Letters*, 2020. I, II-D

[38] Martin Wermelinger, Péter Fankhauser, Remo Diethelm, Philipp Andreas Krüsi, Roland Siegwart, and Marco Hutter. Navigation Planning for Legged Robots in Challenging Terrain. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016. V-B2

[39] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10):1073–1087, 2017. II-E

[40] Bowen Yang, Lorenz Wellhausen, Takahiro Miki, Ming Liu, and Marco Hutter. Real-time Optimal Navigation Planning Using Learned Motion Costs. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 9283 – 9289, 2021. II-A

[41] Jannik Zürn, Wolfram Burgard, and Abhinav Valada. Self-supervised visual terrain classification from unsupervised acoustic feature learning. *IEEE Transactions on Robotics*, 37(2):466–481, 2021. II-C