Fast Traversability Estimation for Wild Visual Navigation

Jonas Frey*[1] Matias Mattamala*[2] Nived Chebrolu[2] Cesar Cadena[1] Maurice Fallon[2] Marco Hutter[1]

[1]ETH Zurich [2]University of Oxford

***Abstract*—This work demonstrates real-world learning and adaptation in the context of traversability estimation. We present a system, Wild Visual Navigation (WVN), which relies on pre-trained high-dimensional features from a self-supervised visual transformer with an online supervision scheme, to achieve on-the-fly traversability learning from a few samples collected in the field. We validate our system with offline experiments and real-world navigation deployments, showing that pre-trained features are fundamental to achieve fast and robust adaptation to new environments.**

I. Introduction

Traversability estimation is a core capability needed by ground robots to autonomously navigate in field environments. Traditionally, occupancy has been used as a proxy for traversability in structured environments, as most of the obstacles can be determined via geometric sensing [13]. However, to achieve navigation in outdoor environments, semantic understanding is further required. Tasks such as following a footpath or navigating through high grass require a high level understanding of the areas that are traversable, which might be challenging to define for each robotic platform.

Previous works in the field have built upon learned geometric models [3, 18, 4] or semantic segmentation [12], which require large labeled datasets. Other approaches designed strategies to label data automatically from previous robot experiences via self-supervision Wellhausen et al. [15], Gasparino et al. [5], anomaly detection [16], or learning from demonstrations Ratliff et al. [14], Wulfmeier et al. [17]. Nevertheless, such methods are still trained on robot-specific datasets and subsequently deployed without further adaptation. The Learning Applied to Ground Vehicles (LAGR) program [8, 6] aimed to overcome this challenge, by showing first examples of systems able to self-supervise machine learning models trained online. While previous work [5, 16] have focused on fine-tuning Convolutional Neural Network (CNN) pre-trained on ImageNet, a recent trend is leveraging expressive features from vision transformer models pre-trained in a self-supervised manner [2]. Recent work by Hamilton et al. [7] showed that the learned features by [2] strongly correlate to the underlying semantics of the scene and linear probing can be leveraged to accurately predict the semantics of the scene.

*Denotes equal contribution. Correspondence to: jonfrey@ethz.ch, matias@robots.ox.ac.uk.

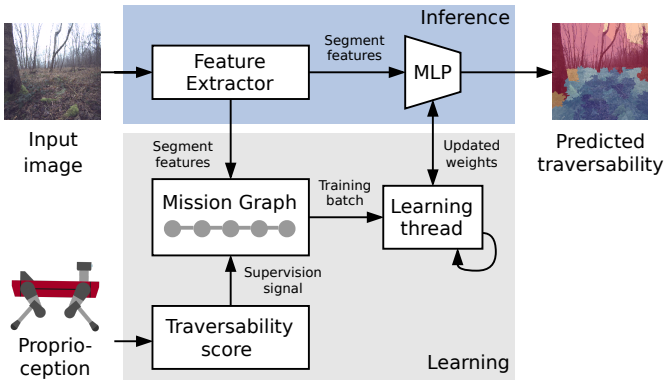

Fig. 1: Wild Visual Navigation (WVN) overview: The system requires monocular RGB images, which are processed by a pre-trained visual transformer to extract high-dimensional features. Proprioceptive data is used to generate supervision signals during operation. Both are used for online learning and inference of traversability (see Sec. II).

This suggests that such features can be exploited for other tasks – in particular traversability estimation.

In this work we present Wild Visual Navigation (WVN), a vision system capable of learning which terrain is traversable by a robot after a few minutes of manual demonstrations *in the wild*. To achieve this, we leverage two main contributions:

- **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.

We validated our approach with ablation studies which compare against similar approaches that are trained in an offline fashion, demonstrating that we achieve comparable or better performance in spite of training online. Further, we deployed our system on the ANYbotics ANYmal C platform to achieve navigation tasks that are otherwise harder to define by geometry alone.

II. Method

A. System Overview

WVN estimates dense traversability from RGB images using a neural network model learned online, in a self-supervised

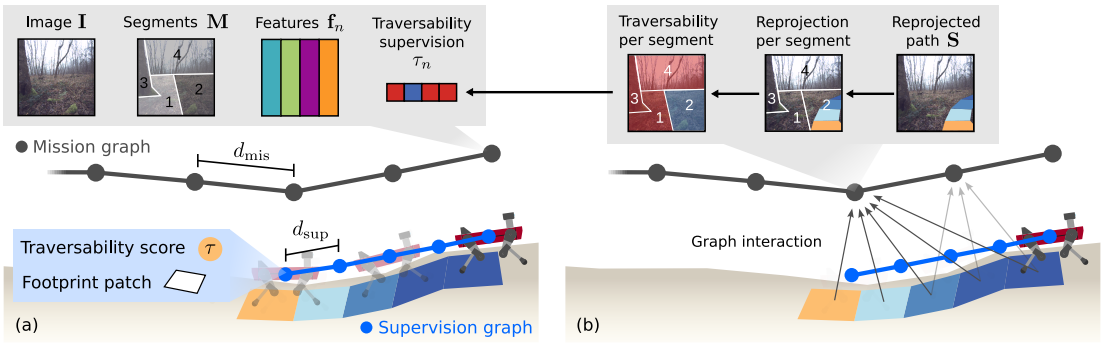

Fig. 2: Supervision and mission graphs: (a) The Supervision Graph only stores information about the robot's footprint in a sliding window; nodes separated by d_{sup} meters. The Mission Graph stores the data required for online learning over the full mission, nodes apart by d_{mis}. (b) Supervision is generated by reprojecting the robot's footprint and traversability scores (indicated by different color) on the images stored in the Mission Graph.

manner, using labels generated by a robot interacting with its environment. Our system only requires a brief demonstration from a human operator for data collection and learning, given that we leverage visual features from a pre-trained feature extraction network. The system overview is shown in Fig. 1.

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 [2] – DINO-ViT.

To enable real-time operation, we follow previous works [10] 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.

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. The traversability score is given by 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. When the robot moves on terrain that is easily traversable it should closely track the reference command; if it struggles to track the reference the discrepancy grows, and we interpret it as a less traversable terrain. Our WVN framework is independent of the exact implementation of the traversability score measure and we chose the velocity tracking error based on its simplicity and interpretability.

D. Supervision and Mission Graphs

To generate online supervision, we accumulate information about the recent history of operation. Our approach relies on a short-horizon graph that works as a buffer for traversability data (*Supervision Graph*), and a mission-long graph that stores the training data generated during a mission (*Mission Graph*); this is shown in Fig. 2.

1) Supervision Graph: The supervision graph is a ring buffer that keeps data within a distance d_{sup}. Each node stores the current time, robot pose, and estimated traversability score τ (Sec. II-C). This graph generates a footprint track with traversability scores τ, as shown in Fig. 2a.

2) Mission Graph: The mission graph works as a memory of the full mission, storing the data required to train the system online. 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) Supervision generation: Upon the creation of a new mission node, 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. The reprojected path is associated to the image segments to generate per-segment features and traversability score used for training (Fig. 2b).

E. Traversability and Anomaly Learning

We train a small neural network that regresses the traversability score τ_n from a given segment feature $\mathbf{f}_n$. It is implemented as two-layer Multi-Layer Perceptron (MLP) with [256, 32] unit dense layers and ReLU non-linear activation functions, with a head that reconstructs the input feature $\mathbf{f}_n$, and another one that predicts the traversability score $\mathbf{f}_n$. The network is trained optimize the following loss:

$$\mathcal{L}_{\text{total}}(\mathbf{f}) = w_{\text{trav}} \mathcal{L}_{\text{trav}}(\mathbf{f}) + w_{\text{reco}} \mathcal{L}_{\text{reco}}(\mathbf{f}). \tag{1}$$

where $\mathcal{L}_{\text{reco}}(\mathbf{f})$ is defined as the Mean Squared Error (MSE) of the input and reconstructed feature, $\mathcal{L}_{\text{trav}}(\mathbf{f})$ is the MSE of the traversability score weighted by an anomaly detection score, and w_{trav} and w_{reco} balance the reconstruction and traversability regression tasks. We train the network using Adam [9] with a fixed constant learning rate of 0.001, by sampling random batches from the Mission Graph, which provides enough diversity to avoid forgetting during the mission. For more details about the exact formulation of the training objective and interaction of anomaly detection and traversability regression, we refer the reader to the Appendix A.

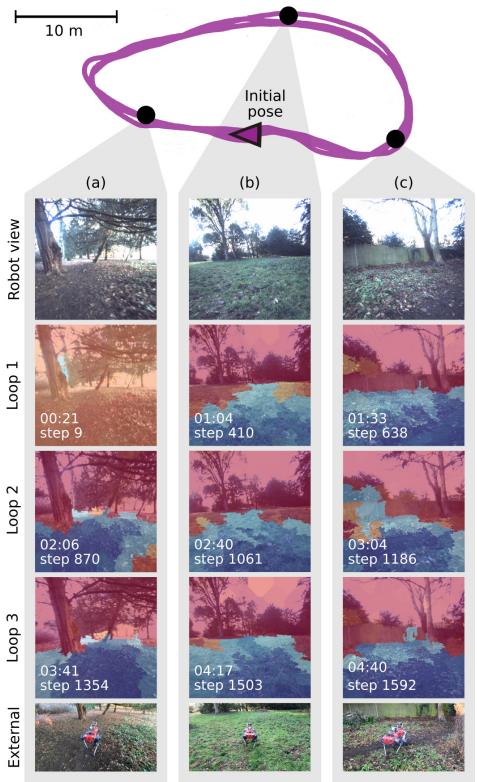

Fig. 3: Adaptation on real hardware: We tested the online adaptation capabilities of our system by driving the robot to complete 3 loops in a park environment (top, route shown in ■). The columns show different parts of the loop (a,b,c); each row displays the improvement of the traversability estimate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

III. Experiments

We evaluated WVN through online and offline experiments. The online experiments were executed on a Jetson Orin board, mounted on an ANYbotics ANYmal C quadruped robot. On the other hand, we ran the offline experiments on a Nvidia RTX3080 Laptop GPU with Intel i7-11800H CPU.

A. Fast adaptation on hardware

We first evaluated the fast adaptation capabilities of WVN while running on the robot. The experiment involved teleoperating the robot around 3 loops of a park environment walking on grass and dirt, on open areas and around trees.

Fig. 3 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 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.

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 I: Scene Adaptation: Traversability Accuracy with respect to the *GT* labels. Each row corresponds to training on a specific environment.

B. Kilometer-scale autonomous navigation in the park

In our second hardware experiment we demonstrated that the system 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. To achieve this, we used the traversability output of WVN to generate a traversability map and a local goal in traversable space. Then a local planner [11] was commanded to follow the proposed goals while staying in traversable space – consequently following the goal staying areas that were designed as traversable during the demonstration.

We executed 3 experiments. In all of them we trained the system for less than 2 min along a footpath, and then we disabled the learning thread. This ensured that the predicted traversability only mimicked the human preference learned during the demonstration run. 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. 4 shows the trajectories followed in each run, starting from different points in the footpath. For runs 1 and 3 we used the same parameters; in run 2 we tested relaxing the learning parameters to achieve a less conservative traversability estimate that required manual interventions.

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. Scene Adaptation

For the subsequent 2 experiments, we used an offline dataset described in the Appendix B. It was collected with a similar robot in different natural environments involving roads, footpaths, grass, bushes, and forests; we named them *Hilly*, *Forest*, and *Grass*. The datasets were split into training, validation, and testing sets. We used an offline version of WVN to generate the training labels from self-supervision. The testing set was hand-labeled into binary classes, based on expert knowledge driving the robot in such environments.

We first evaluated the performance of WVN when trained on one environment and tested on all the others, to test the necessity for online adaptation. Tab. I shows the resulting accuracy for each scene combination. We observed that in

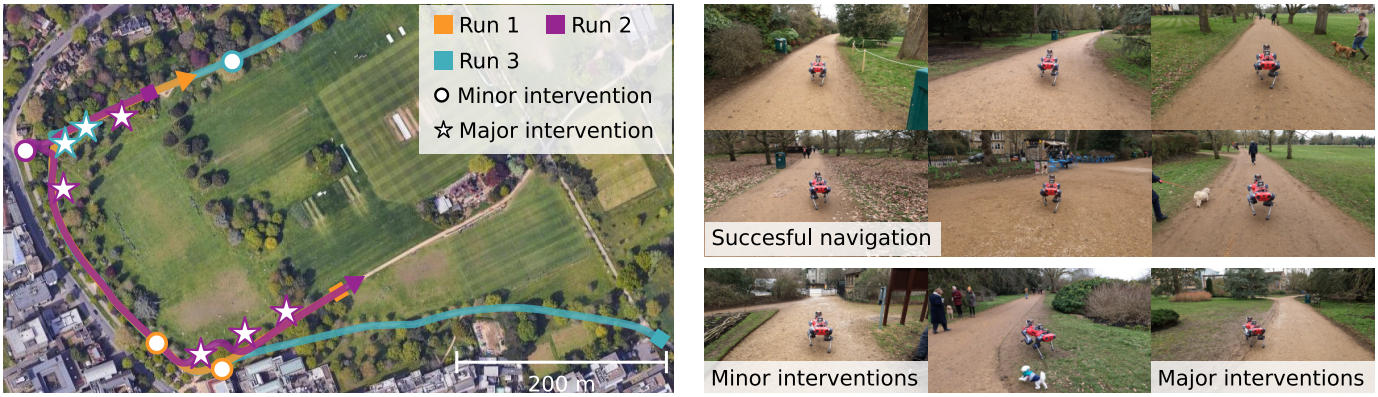

Fig. 4: Kilometer-scale navigation: We deployed our system to learn to segment the footpath of a park after training for a few steps. We executed 3 runs starting from different points in the park: ■ *run 1* (0.55 km), ■ *run 2* (0.5 km), and ■ *run 3* (1.4 km). Minor interventions were applied to guide the robot in intersections; major interventions (⋆) were required for some areas when the robot miss-classified muddy patches for the path.

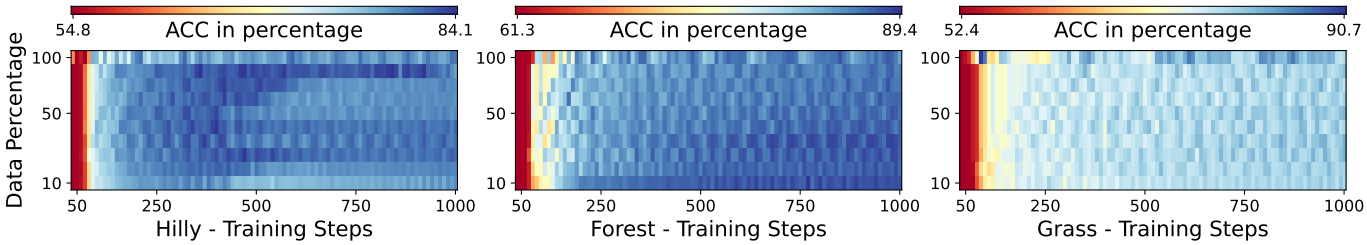

Fig. 5: Adaptation Speed vs Dataset Size: The performance measured by the accuracy increases over training as expected. In the limit case of training for a few steps (50), the performance is equally degraded — independent of the dataset size. A small dataset size is sufficient for good performance. Please 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.

general the best performance is achieved when testing on the same training environment as expected, dropping otherwise.

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 in combination with a pre-trained feature extractor provides a practical solution to enable the deployment of robots in new or changing environments.

D. 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. 5. 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. In the Appendix C the training loss, training accuracy, and example outputs are illustrated.

IV. 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. We validated WVN through different ablation studies and real-world experiments, illustrating its fast adaptation capabilities, and 1.4 km closed-loop navigation experiments in natural scenes. 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. 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.

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. II-B

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

[3] 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. I

[4] 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. I

[5] 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

[6] 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

[7] Mark Hamilton, Zhoutong Zhang, Bharath Hariharan, Noah Snavely, and William T. Freeman. Unsupervised semantic segmentation by distilling feature correspondences. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=SaKO6z6Hl0c. I

[8] 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

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

[10] 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-B

[11] 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. III-B

[12] Daniel Maturana, Po-Wei Chou, Masashi Uenoyama, and Sebastian Scherer. Real-time Semantic Mapping for Autonomous Off-Road Navigation. In *Proceedings of 11th International Conference on Field and Service Robotics (FSR '17)*, pages 335 – 350, 2017. I

[13] 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

[14] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. Maximum margin planning. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 729–736, 2006. I

[15] 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

[16] Lorenz Wellhausen, René Ranftl, and Marco Hutter. Safe Robot Navigation Via Multi-Modal Anomaly Detection. *IEEE Robotics and Automation Letters*, 2020. I

[17] 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. I

[18] 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. I

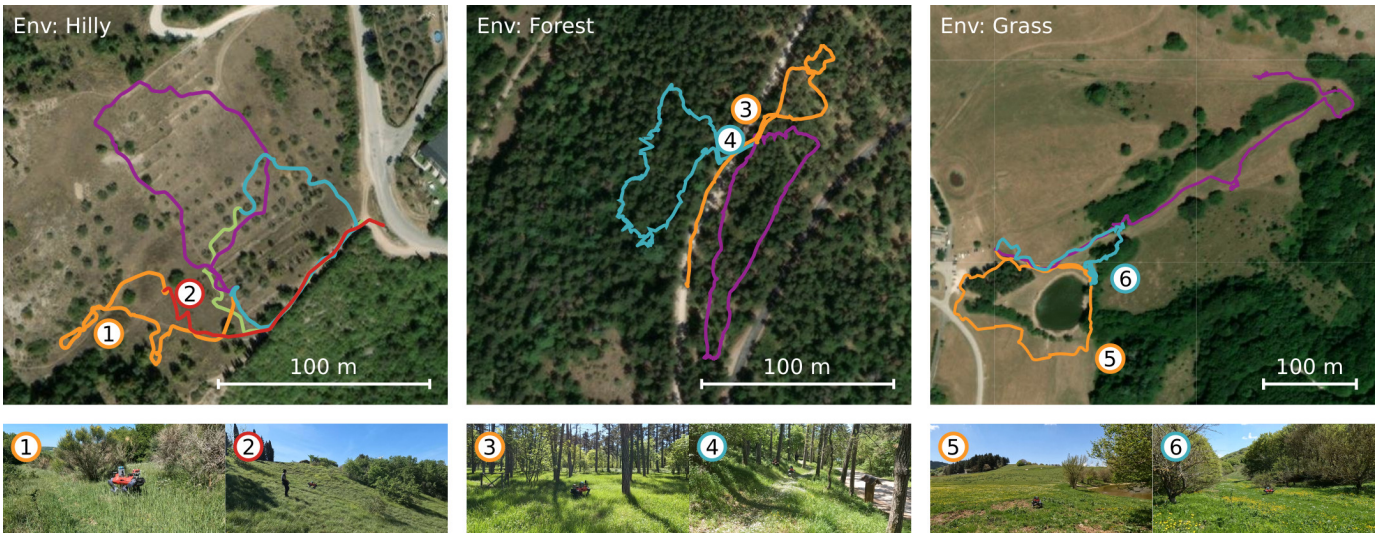

Fig. 6: Aerial views of the 3 environments used for offline testing of our system, illustrating the paths used for data collection and scene examples. The purple ■ trajectories are used for training and validation.

APPENDIX

A. Details: Traversability and Anomaly Learning

During demonstration, we are only capable to collect sparse labels of the environment, where most areas are traversable. We assume that all unlabeled (non-traversed areas) are untraversable. 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 in detail 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 MSE between the predicted features and the original feature compute over all channels E:

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

where $\delta_{\tau \neq 0}$ is 1 if the segments feature traversability score τ_n is not zero, and 0 otherwise. The trained network reconstructs feature embeddings similar to the traversable segments with small reconstruction loss, while feature embeddings of *unknown* segments, i.e, the network was never tasked to reconstruct (e.g. 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):

$$n_{\text{trav}} = \sum_{\mathbf{f}} \delta_{\tau_n \neq 0}, \tag{3}$$

$$\mu_{\text{pos}} = \frac{1}{n_{\text{trav}}} \sum_{\mathbf{f}\,:\,\tau_n \neq 0} \mathcal{L}_{\text{reco}}(\mathbf{f}_n), \tag{4}$$

$$\sigma_{\text{pos}} = \sqrt{\frac{1}{n_{\text{trav}}} \sum_{\mathbf{f}\,:\,\tau_n \neq 0} \left(\mathcal{L}_{\text{reco}}(\mathbf{f}_n) - \mu_{\text{pos}}\right)^2} \tag{5}$$

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:

$$c(\mathcal{L}_{\text{reco}}(\mathbf{f}_n)) = \exp\left(\frac{(\mathcal{L}_{\text{reco}}(\mathbf{f}_n) - \mu_{\text{pos}})^2}{2(\sigma_{\text{pos}}\, k_\sigma)^2}\right), \tag{6}$$

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. The loss for traversability estimation is simply computed by 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}.

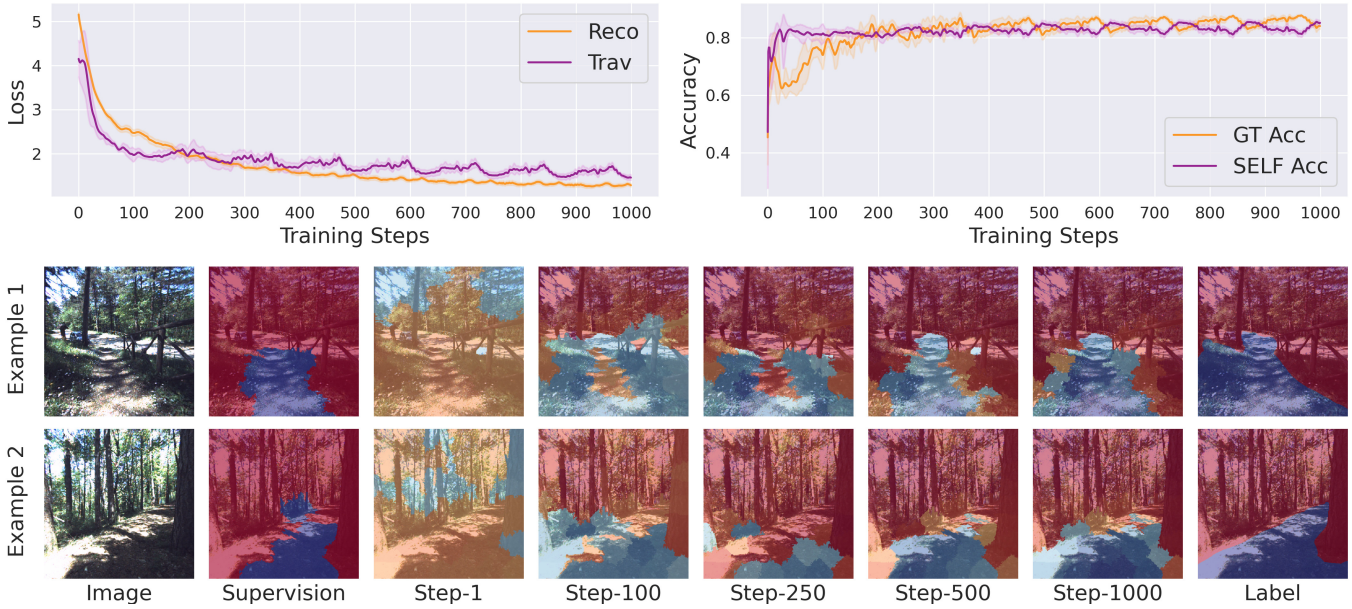

Fig. 7: Training Process: We detail the incremental training process executed by WVN in terms of the loss (top left), accuracy (top right), and visual examples (bottom).

B. Dataset Overview

TABLE II: Dataset Overview: *SELF* indicates that only sparse positive labels are available where the robot walked; *GT* indicates human-annotated ground truth segmentation into traversable and untraversable areas.

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.

For offline analysis we used 3 large-scale datasets, namely:

- *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 are recorded with a teleoperated ANYmal C platform. 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. Fig. 6 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. 6, 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.

C. Training Process

Fig. 7 shows the training loss accuracy over time, illustrating some of the fluctuating behavior, as well as example images of the output segmentation over training.