Wild Visual Navigation: Fast Traversability Learning via Pre-Trained Models and Online Self-Supervision

Matias Mattamala[1*†], Jonas Frey[2,3*†], Piotr Libera[2], Nived Chebrolu[1], Georg Martius[3,4], Cesar Cadena[2], Marco Hutter[2], Maurice Fallon[1]

[1]Dynamic Robot Systems Group, University of Oxford, 23 Banbury Road, Oxford, OX2 6NN, Oxfordshire, United Kingdom.
[2]Robotic Systems Lab, ETH Zurich, Leonhardstrasse 21, Zurich, 8092, Zurich, Switzerland.
[3]Autonomous Learning Group, Max Planck Institute for Intelligent Systems, Max-Planck-Ring 4, Tübingen, 72076, Baden-Württemberg, Germany.
[4]Computer Science Department, University of Tüüingen, Maria-von-Linden-Strasse 6, Tübingen, 72076, Baden-Württemberg, Germany.

*Corresponding author(s). E-mail(s): matias@robots.ox.ac.uk; jonfrey@ethz.ch;
†These authors contributed equally to this work.

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 present Wild Visual Navigation (WVN), an online self-supervised learning system for visual traversability estimation. The system is able to continuously adapt from a short human demonstration in the field, only using onboard sensing and computing. One of the key ideas to achieve this is the use of high-dimensional features from pre-trained self-supervised models, which implicitly encode semantic information that massively simplifies the learning task. Further, the development of an online scheme for supervision generator enables concurrent training and inference of the learned model in the wild. We demonstrate our approach through diverse real-world deployments 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, previously unseen outdoor terrains.
Code: https://bit.ly/498b0CV - Project page: https://bit.ly/3M6nMHH

1 Introduction

Traversability estimation is a core capability needed to allow robots to autonomously navigate in field environments. It is understood as the *affordance* [1] 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 under the idea of occupancy mapping [2], 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 [3].

Existing approaches, which build upon deep neural models for semantic segmentation [4] or anomaly detection [5], have demonstrated navigation in off-road environments; however there are recurring problems with the collection and labeling of large amounts of relevant training data. Self-supervised systems

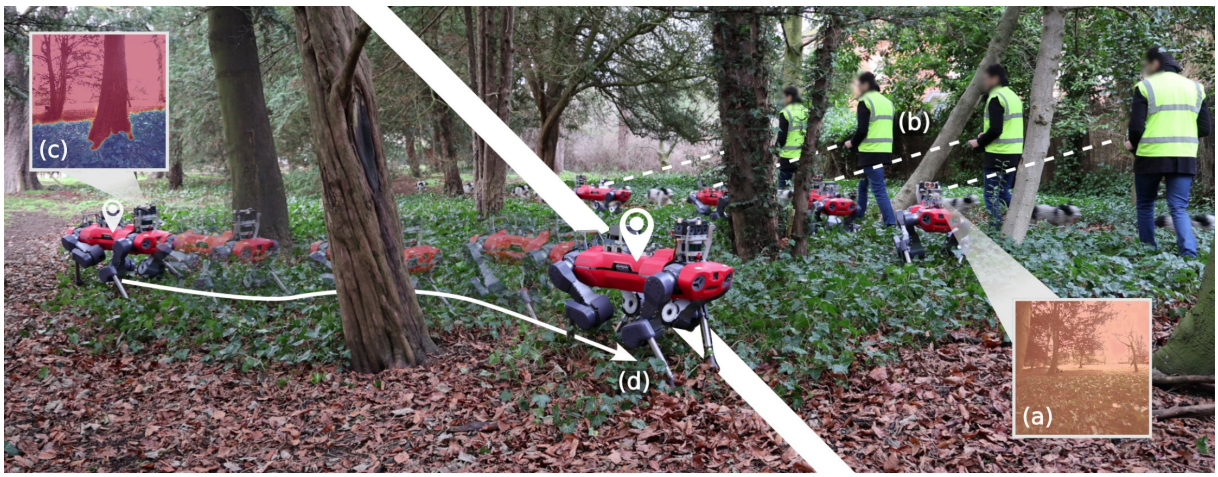

Fig. 1 Wild Visual Navigation (WVN) learns to predict traversability from images via online self-supervised learning. Starting from a randomly initialized traversability estimation network without prior assumptions about the environment (a), a human operator drives the robot around areas that are traversable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

have addressed this challenge by generating labeled datasets from past robot deployments, using classification carried out in hindsight [6] or using predictions of the robot motion [7]. Nevertheless, these previous methods are still trained on robot-specific datasets and subsequently deployed without further adaptation. The Learning Applied to Ground Vehicles (LAGR) program [8, 9] was a first effort towards systems able to adapt in the field, where robots generated its own supervision signals during deployment, facilitating the training of machine learning models within off-road environments.

In this work, we present WVN, a system inspired by the aforementioned approaches to achieve self-supervised, online traversability estimation, solely requiring a few minutes of demonstrations in the field. It combines visual input and proprioceptive information to generate supervision signals while the robot operates, enabling it to simultaneously train the traversability model and use it for online inference (Fig. 1). A key idea in WVN is exploiting high-dimensional features from pre-trained models. This simplifies the learning task while also exploiting the semantic correspondences implicitly learned by these models via offline self-supervised training on large datasets.

This article extends the system presented by Frey and Mattamala et al. [10], addressing some of the limitations raised in the original formulation and introducing additional features for system integration and field deployment. The contributions are:

1. **An online, multi-camera self-supervision pipeline** that extends the original approach by enabling the use of multiple vision sources for supervision and inference.
2. **The use of pre-trained models** as feature extraction backbones, namely DINO-ViT[11] and, in addition to our previous work, STEGO [12]. We demonstrate that this eases the overall traversability prediction training process.
3. **A feature sub-sampling strategies** to efficiently process pixel-wise high-dimensional features. We present additional strategies to SLIC [13] used in our previous work, which further exploit the semantic priors already encoded in the features.
4. **Real-world experiments**, demonstrating hardware integration and onboard execution of WVN with one and multiple cameras, achieving real-world navigation tasks after minutes of training.
5. **Open-source implementation** of the WVN system with ROS [14] integration, with a set of baseline model weights trained on diverse environments.

2 Related Work

2.1 Traversability from Geometry

Classical approaches for traversability estimation analyze the geometry of the environment using 3D sensing [2]. Solutions from the DARPA SubT Challenge [15], used representations such as point clouds and meshes to evaluate navigational metrics like risk or stepping difficulty [16, 17].

However, a purely geometric analysis has proven insufficient and data-driven methods have bridged this gap by learning platform-specific traversability from real data or simulations. Chavez-Garcia et al. [18] used simulations of a ground robot moving on an elevation map. Yang et al. [19] extended this approach for legged platforms, capturing the risk of failure, energy cost and time required for navigation. Recently, Frey et al. [20] expanded this approach to volumetric data and massive parallelization in data collection from simulation. Nevertheless, using geometry-only could be insufficient to represent natural growth such as high grass, branches or bushes.

2.2 Traversability from Semantics

Semantic segmentation methods aim to address the aforementioned challenges by assigning semantic classes to the representations, with different navigation costs. Bradley et al. [21] presented a scene understanding system for a legged platform, trained and evaluated using geographically diverse data. Maturana et al. [4] demonstrated autonomous off-road navigation using semantics projected onto 3D map around a wheeled platform. Schilling et al. [22] used semantically segmented features that were classified into fixed classes using a random forest classifier. Belter et al. [23] developed a semantic terrain analysis module to guide a whole-body planner in a multi-legged platform. Recently, Shaban et al. [24] presented an approach for off-road navigation that learns a dense traversability map from sparse point-clouds, while Cai et al. [25] mapped terrain semantics to vehicle speed profiles as a proxy for traversability.

Most of these methods rely on pre-trained or fine-tuned semantic segmentation models with pre-defined class labels. In this work we exploit the advances in self-supervised models, such as DINO-ViT [11], to determine semantically similar regions without manual supervision.

2.3 Traversability from Self-supervision

Self-supervised methods address the challenges of pre-defined classes and costs by using past robot experiences [8, 26]. Modern methods rely on deep neural networks trained from weakly supervised data, and the supervision depends on the robot platform. Wellhausen et al. [6] used the reprojected footholds from a legged robot to provide supervision of walkable areas; Zürn et al. [27] exploited sounds produced by a wheeled robot moving on different terrain as a proxy for supervision; Gasparino et al. [7] instead used the receding-horizon trajectory of a Model Predictive Controller (MPC).

BADGR [28] predicted future robot states and events from images, including its position and crash probability, which can be interpreted as traversability. TerraPN [29] used odometry and IMU signals as supervision to learn a traversability model in $25\,\mathrm{min}$ – including data collection and learning. Guaman Castro et al. [30] predicted traversability based on IMU supervision conditioned on the velocity of the robot. Recently, Jung et al. [31] presented a system that shares with WVN the use of pre-trained models for self-supervision.

While WVN follows similar self-supervision strategies, we aim for concurrent supervision signal generation and learning achieving online adaptation in the field.

2.4 Traversability from Anomalies

Anomaly detection methods are motivated by the imbalance of positive and negative samples in self-supervised methods. Instead of training a discriminative model of traversability, they focus on learning generative models of the traversed terrain. This distribution is used as a proxy to set out-of-distribution (OOD) inputs as untraversable.

Richter and Roy [32] trained an autoencoder to predict OOD scenes from images, switching to safer navigation behaviors when traversing novel environments. Wellhausen et al. [5] used multi-modal sensing from haptics, vision and depth to identify anomalies such as flames and water reflections. Schmid et al. [33] show-case the effectiveness of anomaly detection for identifying safe terrain from vision in an off-road driving scenario. Further, Ji et al. [34] formulated a proactive anomaly detection approach that evaluated candidate trajectories for local planning depending on their probability of failure.

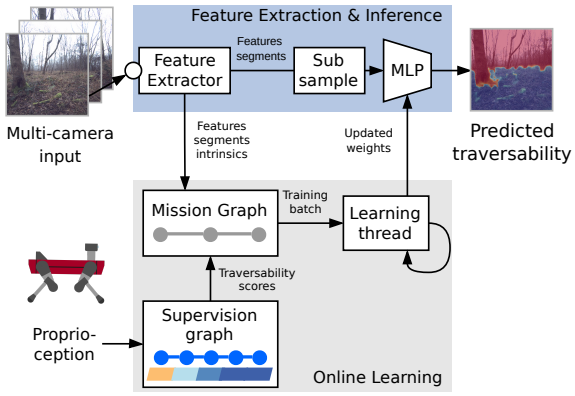

Fig. 2 System overview: WVN only requires monocular RGB images, odometry, and proprioceptive data as input, which are processed to extract features and supervision signals used for online learning and inference of traversability (see Sec. 3).

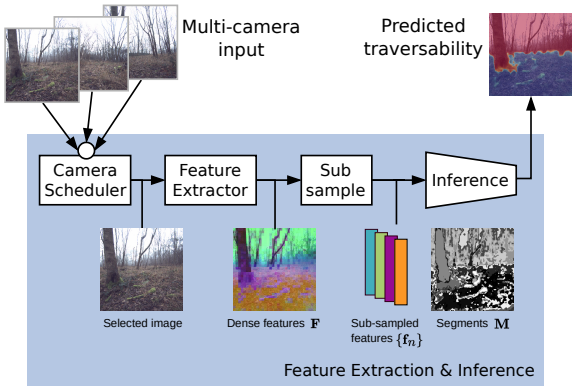

Fig. 3 Feature Extraction & Inference process: The camera scheduler module (Sec. 3.2.1) selects one camera from the available pool, and provides the RGB image to the feature extractor module (Sec. 3.2.2). This extracts dense visual features $\mathbf{F}$ using pre-trained models. Next, the sub-sample module produces a reduced set of embeddings $\{\mathbf{f}_n\}$ using a subsampling strategy based on a weak segmentation system (Sec. 3.2.3). Lastly, the inference module predicts traversability from the image using the embeddings.

Symbol	Definition
$\mathbf{I}$	RGB image with height H and width W
$\mathbf{F}$	Feature map with dim. $E \times H \times W$, $E = 90$ or 384
$\mathbf{M}$	Weak segmentation mask with height H and width W
$\mathbf{S}$	Reprojected supervision with dim. $H \times W \in [0, 1]$
τ	Traversability score $\in [0, 1]$
$\mathbf{f}_n$	Per-segment embedding of dim. $E = 90$ or 384
τ_n	Per-segment traversability score

Table 1 Main definitions used in this work

While we do not explicitly use anomalies to determine traversability, we do use it as a confidence metric to leverage the sparse supervision signals, as it has also been recently explored by Seo et al. [35].

3 Method

3.1 System Overview

The objective of this work is to design a navigation system that estimates dense traversability scores of the terrain from RGB images. We use a neural network model—Multi-Layer Perceptron (MLP)—trained online, in a self-supervised manner, from supervision signals generated by a robot interacting with its environment. The system should require only a brief demonstration from a human operator for data collection and learning.

WVN is implemented as a two-processes system that run at different rates, as shown in Fig. 2. The *feature extraction & inference process* processes images from different cameras, extracts visual features, and performs inference to predict the traversability score pixel-wise. The *online learning process* estimates traversability scores from proprioception, generates the supervision signals from hindsight information, and executes an inner training loop to update the traversability prediction model. While the former supplies visual features for training as images are processed, the latter provides the most updated learned model at a fixed time rate.

The main definitions used in the rest of the paper are summarized in Tab. 1, and the technical details of each process are provided as follows.

3.2 Feature Extraction & Inference

3.2.1 Multi-camera processing

While the original WVN was designed for single-camera processing, it presented limitations during navigation, constraining it to motions within the camera Field of View (FoV).

We enabled multi-camera operation by developing a camera scheduler based on the weighted round-robin algorithm [36]. This ensured that the system only processes a single camera at a time, depending on priorities specified by the cameras being used for training and inference, or inference-only.

3.2.2 Feature extraction

After a camera is selected in each cycle of the scheduler, the following steps are camera-agnostic. Given an RGB image $\mathbf{I}$, we extract dense, pixel-wise visual feature maps (*embeddings*) $\mathbf{F}$. In contrast to previous works based on fine-tuned Convolutional Neural Networks (CNNs), we rely on recent self-supervised network architectures to generate high-dimensional features that encode meaningful semantics without requiring labels.

In our implementation, we integrated the self-supervised DINO-ViT [11], which provides 384-dimensional pixel-wise feature embeddings. We additionally considered STEGO [12], which uses a DINO-ViT backbone with additional layers trained with contrastive learning, providing 90-dimensional features and segmentation mask. Before extracting the features, we resize the input images to a resolution of 224×224. The resulting dense features $\mathbf{F}$ are too large to be stored in GPU memory for online training. Hence, we introduced feature sub-sampling strategies to reduce the dimensionality of $\mathbf{F}$.

3.2.3 Feature sub-sampling

We implemented different sub-sampling strategies to reduce the number of pixel-wise embeddings from 224×224 to a reduced set of $\sim$100 embeddings $\{\mathbf{f}_n\}$. The strategies use a weak segmentation system to partition the image into a set of segments $\mathbf{M}$, and then average the embeddings within each segment:

- **SLIC:** In our original implementation and inspired by Lee et al. [37], we explored the use of SLIC [13] to reduce the dimensionality to $\sim$100 segments per image. While they are fast to compute, they have the disadvantage of being based on texture only, not necessarily grouping pixels by semantics.
- **STEGO:** It provides class-free segments, which implicitly encode semantic affinity, which directly defines the segments $\mathbf{M}$.
- **Random:** We randomly select a set of 100 embeddings from the feature map. In this case, we have no segments but the feature locations only.

The original STEGO implementation considers the task of semantic segmentation based on a fixed set of classes across a full dataset. To assign each pixel to a semantic class, the authors compute prototype feature vectors across a training dataset offline. A pixel can then be assigned to a semantic class based on its cosine-similarity embedding with respect to the identified prototype feature vectors. This is not applicable in our scenario, where we do not consider a fixed set of classes, nor can identify suitable prototypes vectors before the mission, given that these would strongly depend on the deployment environment. Instead, we compute a fixed number of prototype features per image using KNN-clustering, which guarantees a fixed number of segments per image. Fig. 4 illustrates qualitative examples of the different segments produced by the SLIC and STEGO methods, and in Sec. 5.3.2, we experimentally test these different strategies. We open-sourced the full re-implementation of the modified STEGO version[1].

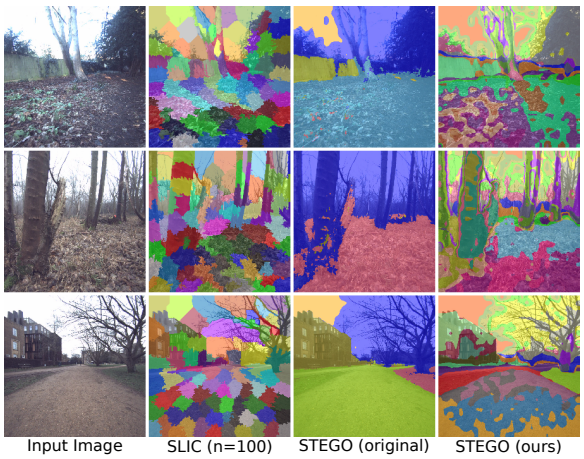

Fig. 4 Comparison feature segmentation methods for 3 example images. SLIC over-segments the image, but fails to construct semantically coherent segments (e.g. top row merging fence and ground into a single segment). The STEGO segmentation aligns with the semantics, but the computation of prototype vectors across a full dataset limits the number of semantic classes, leading to merging of two semantic classes into a single segment (grass and walkway, bottom row). Our modified version of STEGO, over-segments the image but still provides semantically meaningful segments without pre-setting prototype vectors before deployment.

After this sub-sampling step, the subset of embeddings $\{\mathbf{f}_n\}$ and their image locations, segments $\mathbf{M}$ and camera intrinsics are shared with the *online learning* process for training (Sec. 3.3).

3.2.4 Inference

Lastly, this process provides predictions of traversability for all the incoming images from the different cameras. This is achieved by inferencing the MLP model, which is updated at a fixed rate by the online learning

[1] GitHub: https://github.com/leggedrobotics/self_supervised_segmentation

process (Sec. 5.1). The model predicts traversability from the embeddings $\{\mathbf{f}_n\}$ using two different approaches:

- **Segment-wise inference:** This is the approach implemented originally [10], which predicts a traversability score τ_n for each embedding $\mathbf{f}_n$, and assigns the same score for all the pixels corresponding to the given segment.
- **Pixel-wise inference:** Alternatively, we predict fine-grained traversability from the dense features $\mathbf{F}$, given that the MLP forward pass can be executed with low-latency for a batch of features.

Sec. 5.3.2 provides qualitative examples of the improvements that each method provides when the system is deployed in different natural environments.

3.3 Online Learning

3.3.1 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 is fully traversable. We use the terrain *traction* [38], which measures the discrepancy between the robot's current linear velocity as estimated by the robot (v_x, v_y), and the reference velocity command $(\bar{v}_x, \bar{v}_y)$ given by an external human operator or planning system.

We define the mean squared velocity error as:

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

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

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

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

3.4 Supervision and Mission Graphs

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

3.4.1 Supervision Graph

The supervision graph stores within its nodes information about the current time, robot pose, and estimated traversability score (Sec. 3.3.1). This graph is implemented as a ring buffer, which only keeps a fixed number of nodes N_{sup}, separated from each other by a distance d_{sup}.

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

3.4.2 Mission Graph

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

3.4.3 Supervision generation

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

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

3.5 Traversability and Anomaly Learning

We train a small neural network in an online fashion that determines the feature traversability score τ_n from a given segment feature $\mathbf{f}_n$. This reduces the visual traversability estimation problem to simple regression task. Further, we model the uncertainty

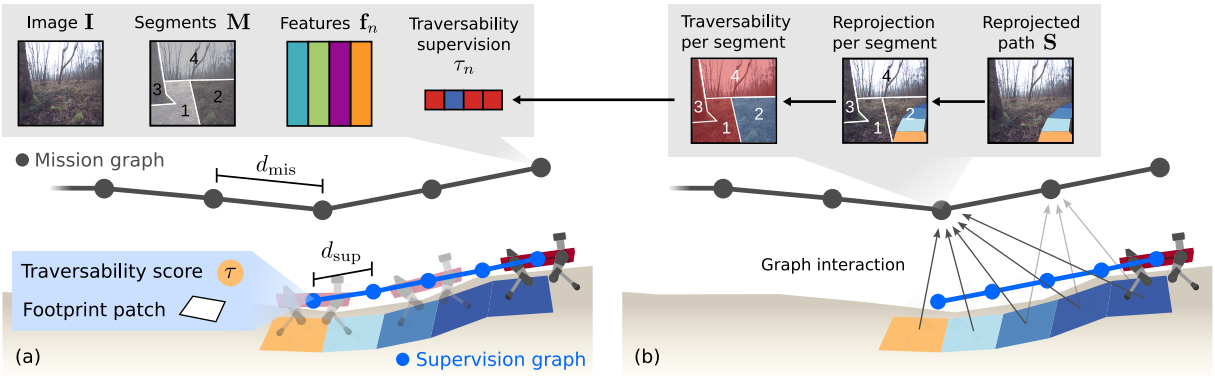

Fig. 5 Supervision and mission graphs: (a) Information stored in each graph over the mission. While the Supervision Graph only stores temporary information about the robot's footprint in a sliding window, the Mission Graph saves the data required for online learning over the full mission. The color of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

about the unvisited (and hence, unlabeled) areas by using anomaly detection techniques to bootstrap a confidence estimate.

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

3.5.1 Confidence Estimation

To obtain a segment-wise confidence estimate, we aim to learn the distribution over all traversed segment features $\mathbf{f}_n$. An encoder-decoder network $f^{\theta_r}_{\text{reco}}$ is trained to compress the segment feature $\mathbf{f}_n$ into a low dimensional latent space and reconstruct the original input features $\mathbf{f}_n$. The reconstruction loss is given by the Mean Squared Error (MSE) between the predicted features and the original feature compute over all channels E:

$$\mathcal{L}_{\text{reco}}(\mathbf{f}_n) = \begin{cases} \frac{1}{E}\sum_e \| f^{\theta_r}_{\text{reco}}(\mathbf{f}_{n,e}) - \mathbf{f}_{n,e} \|^2 & \text{if traversed,} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

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

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

$$\mu_{\text{pos}} = \frac{1}{n_{\text{trav}}} \sum_{n \in \mathcal{T}} \mathcal{L}_{\text{reco}}(\mathbf{f}_n), \tag{4}$$

$$\sigma_{\text{pos}} = \sqrt{\frac{1}{n_{\text{trav}}} \sum_{n \in \mathcal{T}} \left(\mathcal{L}_{\text{reco}}(\mathbf{f}_n) - \mu_{\text{pos}}\right)^2} \tag{5}$$

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

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

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

3.5.2 Traversability Estimation

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

for traversability estimation is computed using the confidence-weighted MSE:

$$\mathcal{L}_{\text{trav}}(\mathbf{f}) = \underbrace{\sum_{n \in \mathcal{T}} \| f^{\theta_t}_{\text{trav}}(\mathbf{f}_n) - \tau_n \|^2}_{\text{Contribution of traversed (\textit{labeled}) segments}} + \underbrace{\sum_{n \in \mathcal{T}^C} (1 - c(\mathbf{f}_n)) \, \| f^{\theta_t}_{\text{trav}}(\mathbf{f}_n) - 0 \|^2}_{\text{Contribution of untraversed segments}}, \tag{7}$$

(8)

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

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

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

3.5.3 Implementation details

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

$$\mathcal{L}_{\text{total}}(\mathbf{f}) = w_{\text{trav}} \mathcal{L}_{\text{trav}}(\mathbf{f}) + w_{\text{reco}} \, \mathcal{L}_{\text{reco}}(\mathbf{f}). \tag{9}$$

with w_{trav} and w_{reco} allowing to weigh the traversability and reconstruction loss respectively. We used Adam [39] to jointly train the networks with a fixed constant learning rate of 0.001. For a single update step, 8 valid mission nodes are randomly chosen to form a data batch, where we defined a node as valid if at least a single segment of the node has non-zero traversability score. For all our experiments we set $k_\sigma = 2$, $w_{\text{trav}} = 0.03$, $w_{\text{reco}} = 0.5$ and use a maximum FPR of 0.15 to determine the traversability threshold. Please refer to our previous publication [10] for ablation studies of the different parameter and design choices.

4 Closed-loop Integration

We integrated the learned traversability estimate into a standard navigation pipeline to achieve autonomous navigation with a quadrupedal platform. The details are explained as follows.

4.1 Local terrain mapping

In order to use the predicted traversability for navigation tasks, we used an open-source terrain mapping framework [40, 41] that produced a robot-centric 2.5D elevation map from the onboard depth cameras and LiDAR sensing. The framework enabled the fusion of the predicted traversability images via raycasting, taking into account the occlusions with the terrain, as well as temporal fusion of the traversability information via exponential averaging.

4.2 Local planning

We used the projected visual traversability as a cost map for a reactive local planner [42] to generate a $\mathrm{SE}(2)$ twist command to drive the robot towards a goal while avoiding untraversable terrain. The twist command was the input to a robust learning-based locomotion controller [3], which is able to traverse rough terrain typically inaccessible to wheeled robots.

4.3 Autonomous Path Following

Lastly, we implemented a navigation strategy to guide the robot in path-like environments. The method continuously spawned new goals in front of the robot by finding the furthest traversable position in the local terrain map, within the FoV of the front-facing camera. This strategy was used to motivate simple autonomous navigation and exploration without requiring a global planner and a large-scale representation of the environment.

5 Experiments

5.1 Platform Description

For our experiments we used the ANYbotics ANYmal C and D legged robots. In both configurations the robots were equipped with an additional NVidia Jetson Orin AGX. We used the manufacturer's state estimator to obtain $\mathrm{SE}(3)$ pose and body velocity measurements. The LiDAR and depth cameras available on the robots were only used for the local terrain mapping module (Sec. 4.1).

For the ANYmal C experiments, we used a single global shutter, wide FoV camera from the Sevensense Alphasense Core unit. For the ANYmal D experiments, we used the RGB images from the integrated front and rear wide-angle cameras.

WVN was implemented in pure Python code using PyTorch [43] and ROS 1 [14]. Both processes ran on the Jetson Orin, and were implemented as separate ROS nodes. Inter-process communication was implemented using ROS publisher-subscriber paradigm, while the trained model weights were shared via write-read operations every 5 s for simplicity.

5.2 Real-world deployments

We executed different deployments to validate WVN in real environments in terms of adaptation to new scenes, the advantages of the visual traversability estimation compared to purely geometric, and autonomous navigation demonstrations.

5.2.1 Fast Adaptation on Hardware

Our first experiment involved teleoperating the ANYmal C robot around 3 loops in University Parks, Oxford, UK, to evaluate the fast adaptation capabilities of WVN when walking on grass and dirt, on open areas, and around trees.

Fig. 6 illustrates the main outcomes of the experiment, showing that the system learned to predict robot-specific traversability over the 3 loops while running onboard. Section **(a)** shows how the robot starts with a very poor segmentation after 9 steps of training (21 s) but 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.

5.2.2 Benefits of Visual Traversability vs Geometric Methods

Our second experiment aimed to illustrate the advantages of visual traversability estimation in challenging natural environments. We teleoperated the ANYmal C robot around high grass, loose branches, and bushes in Wytham Woods, Oxford, UK. Fig. 7, bottom right, shows a representative shot of the experiment, the forward-facing camera image and WVN's prediction.

To compare the different traversability methods, we used the terrain mapping module (Sec. 4.1), as it allowed us to compare geometry-only and visual traversability. We compared against two geometric methods that are real-time capable and have been used in previous works:

- Geometric method based on heuristics such as height and slope of the terrain [44].
- Geometric method based on a learned model of traversability, which is part of the terrain mapping system [40].
- Visual traversability provided by WVN, raycasted onto the terrain map.

The geometric methods only require an elevation representation of the surface to determine traversability from the 2.5D geometry. For WVN we executed a training procedure by driving the robot around the environment for a few minutes, only using images from the forward-facing camera.

Fig. 7 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

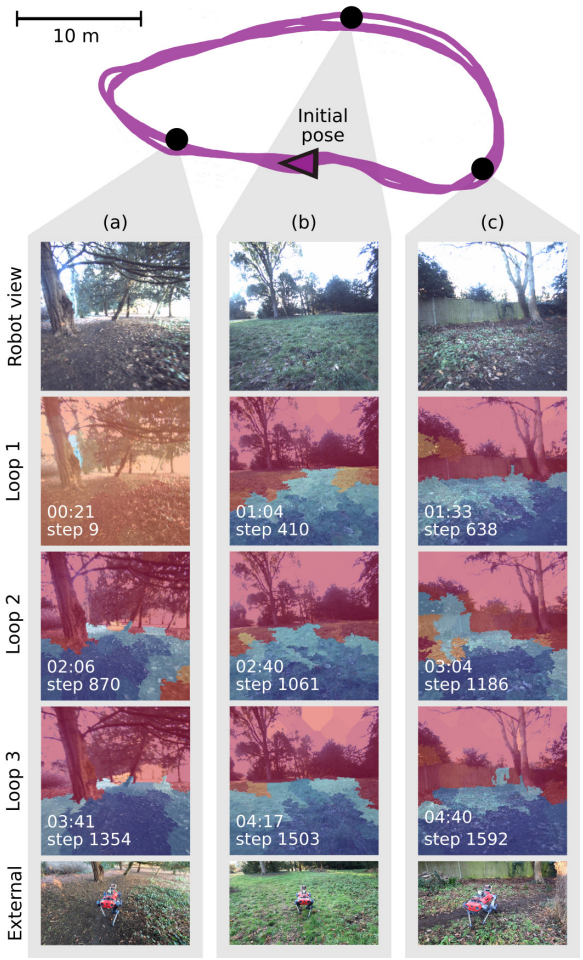

Fig. 6 Adaptation on real hardware: We tested the online adaptation capabilities of our system by teleoperating the robot to complete 3 loops in a park (top, route shown in ■). The columns show different parts of the loop (a,b,c); each row displays the improvement of the traversability estimate over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

untraversable areas. Our system is also able to successfully discriminate the trees, confirming the findings observed in Sec. 5.2.1. Furthermore, the advantages of WVN are observed in high-grass areas, which are represented as elevation spikes in the map that are classified as untraversable by the geometric methods but not by our visual traversability estimate. These differences become more evident in the SDFs where all the areas with low traversability scores become obstacles.

5.2.3 Point-to-point Autonomous Navigation Between Trees

We executed closed-loop navigation tasks to demonstrate that WVN can easily adapt to a new environment, and the learned traversability estimate can be used to deploy the robot autonomously.

We taught the ANYmal C 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. 4.

Fig. 8 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 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 **(b)**, which indicate the trees as obstacles. Lastly, in post-processing we fused the predicted traversability measures into a complete map in **(c)**, which correctly aligned with the trees. However, given that in this experiment we used the SLIC segmentation method from our previous work, we observed some obstacle artifacts. This limitation is addressed in Sec. 5.2.5, where we deploy our multiple-camera setup and the novel segmentation and pixel-wise prediction method.

5.2.4 Kilometer-scale Autonomous Navigation in the Park

We demonstrated that WVN enabled preference-aware path-following behavior as a result of the human demonstrations and the online learning capabilities of the system. This experiment was also executed with the ANYmal C platform.

We executed 3 runs in a footpath at University Parks, Oxford, UK. Similarly to our previous experiments, we trained the system for less than 2 min along the footpath. We then disabled the learning process to ensure that the predicted traversability strictly mimics the human preference during the demonstration run. The autonomous path following system from Sec. 4.3 was used to guide the robot forward along the path.

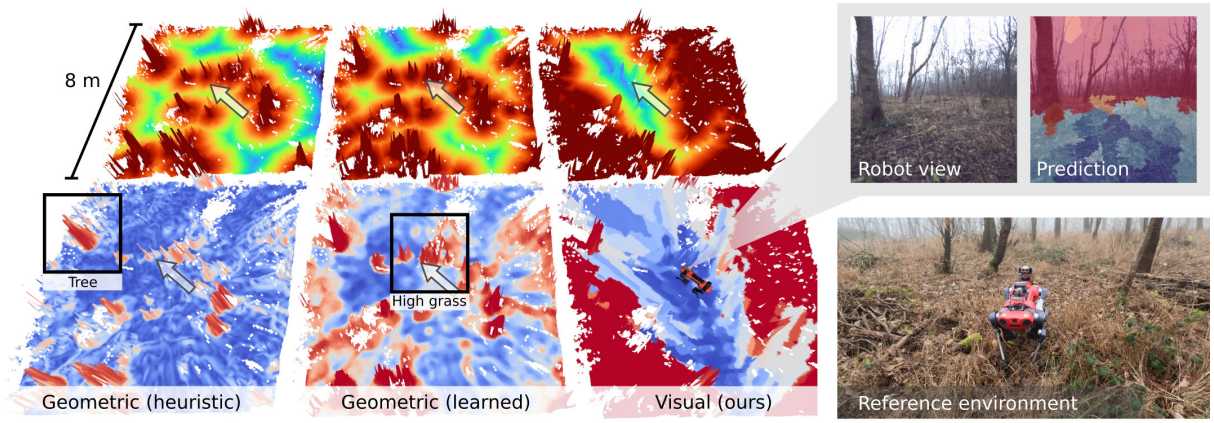

Fig. 7 Visual vs geometric traversability: Illustration of traversability map (bottom row) and corresponding Signed Distance Field (SDF) (top row) for three different traversability estimation methods applied to the same terrain patch. Our visual traversability estimate provides clear advantages for 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).

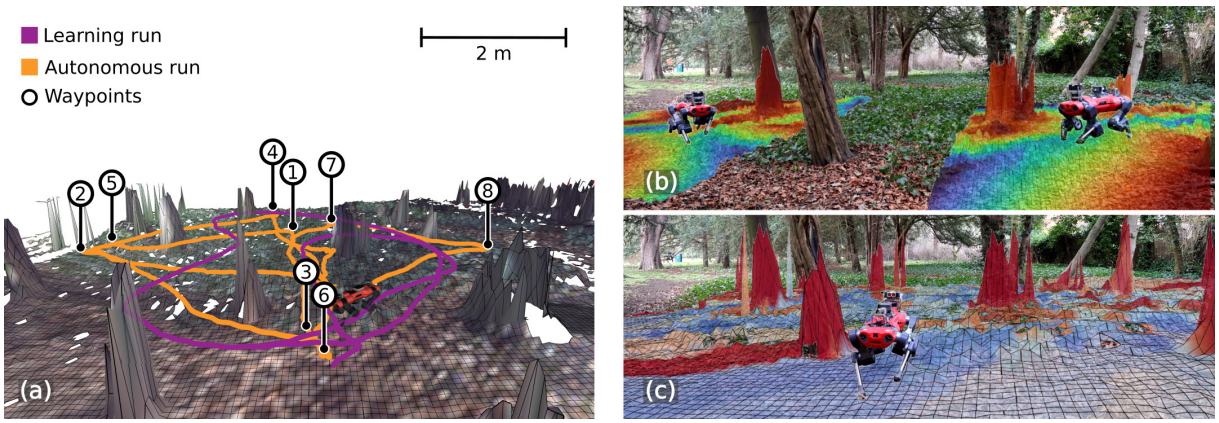

Fig. 8 Point-to-point autonomous navigation: (a) After teleoperating the robot for 2 min (path shown in ■), we successfully achieved autonomous navigation in a woodland environment (path shown in ■). (b) Some of the SDFs generated from the predicted traversability during autonomous operation. (c) Global 2.5D reconstruction of the testing area and predicted traversability, generated in post-processing to illustrate the capabilities of our approach.

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. 9 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 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 (mud patches) 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 indistinguishable . 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

Fig. 9 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.

footpath was sufficient for WVN to generate semantic cues to achieve the desired path following behavior.

5.2.5 Multi-camera Deployment from Indoor to Outdoor Environments

This last experiment demonstrates the adaptation capabilities of WVN and the new multi-camera integration on the ANYmal D quadruped. The deployment was executed at the Max Planck Institute in Tübingen, Germany. We deployed WVN using STEGO segmentation and features during training and perform the inference pixel-wise. Throughout the 7 min teleoperated session, we provide snapshots of the environment, the traversability predictions, as well as the visual and geometric traversability, illustrated in Fig. 10.

The deployment started within a laboratory setting. Upon covering $\sim 8\,\mathrm{m}$ **(a)**, WVN correctly identified the floor as traversable. Transitioning to a corridor after $\sim 40\,\mathrm{m}$ **(b)**, the visual traversability accurately classified windows and closed glass doors as impassable, which the geometric traversability erroneously report as traversable. When exiting the building, WVN correctly predicted the paved walkways as traversable, which can be seen within the courtyard **(c)** at $\sim 86\,\mathrm{m}$, outdoor walkway **(d)** at $\sim 127\,\mathrm{m}$, and the paved road **(e)** at $\sim 148\,\mathrm{m}$. When entering a small grass area with sparse vegetation after walking for a few minutes, ours correctly identified the field as traversable, while the geometric traversability fails to distinguish between trees and penetrable vegetation **(f)** at $\sim 260\,\mathrm{m}$.

The integration of traversability estimates from both cameras enabled us to update the traversability to the front and the back of the robot. This allowed to overcome the restricted field of view limitations shown in Fig. 7. Multiple cameras also allow for more reactive behavior in dynamic environments, where it is crucial for planning to update the belief about the environment constantly.

5.3 Offline analysis

We executed two offline experiments to assess the differences of the feature sub-sampling and inference methods. These experiments were executed in post-processing using logs of previous real-world experiments, on an Nvidia Quadro T2000 Laptop GPU with Intel i7-10875H CPU.

5.3.1 Segments vs Pixel-wise Inference

First, we compared the visual traversability prediction differences when performing segmen-wise and pixel-wise inference. We ran WVN in post-processing, on the recorded logs from the Sec. 5.2.2 and Sec. 5.2.1 experiments.

Fig. 12 shows some examples of the traversability predictions when using STEGO and SLIC segments, as well as pixel-wise segmentation. We observed consistencies between the segment-wise and pixel-wise

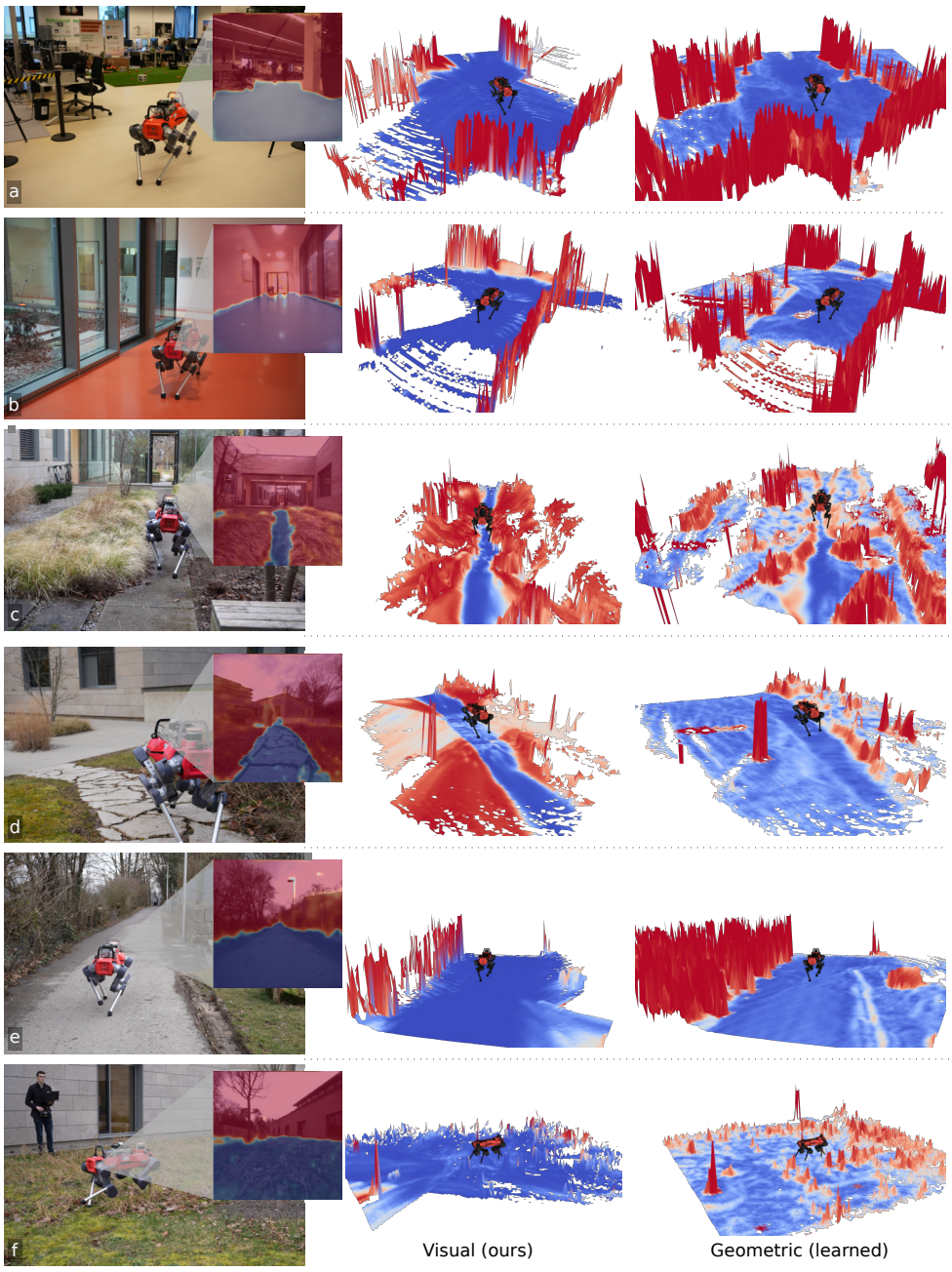

Fig. 10 Deployment with multi-camera setup. Left: Real scene and visual traversability prediction. Center: Visual traversability projected on the local terrain map. Right: Geometric traversability computed from elevation map. The robot was teleoperated throughout the experiment. Each example (a) - (f) is sequentiall and it is discussed in detail in Sec. 5.2.5.

predictions, which we explain due to the intrinsic properties of the features (discussed in Sec. 5.3.2). However, pixel-wise inference shows advantages in providing fine-grained traversability predictions, disregarding the artifacts that weak-segmentation methods such as SLIC induce, and seen in Fig. 11 *(b)* and *(c)* on the tree trunk.

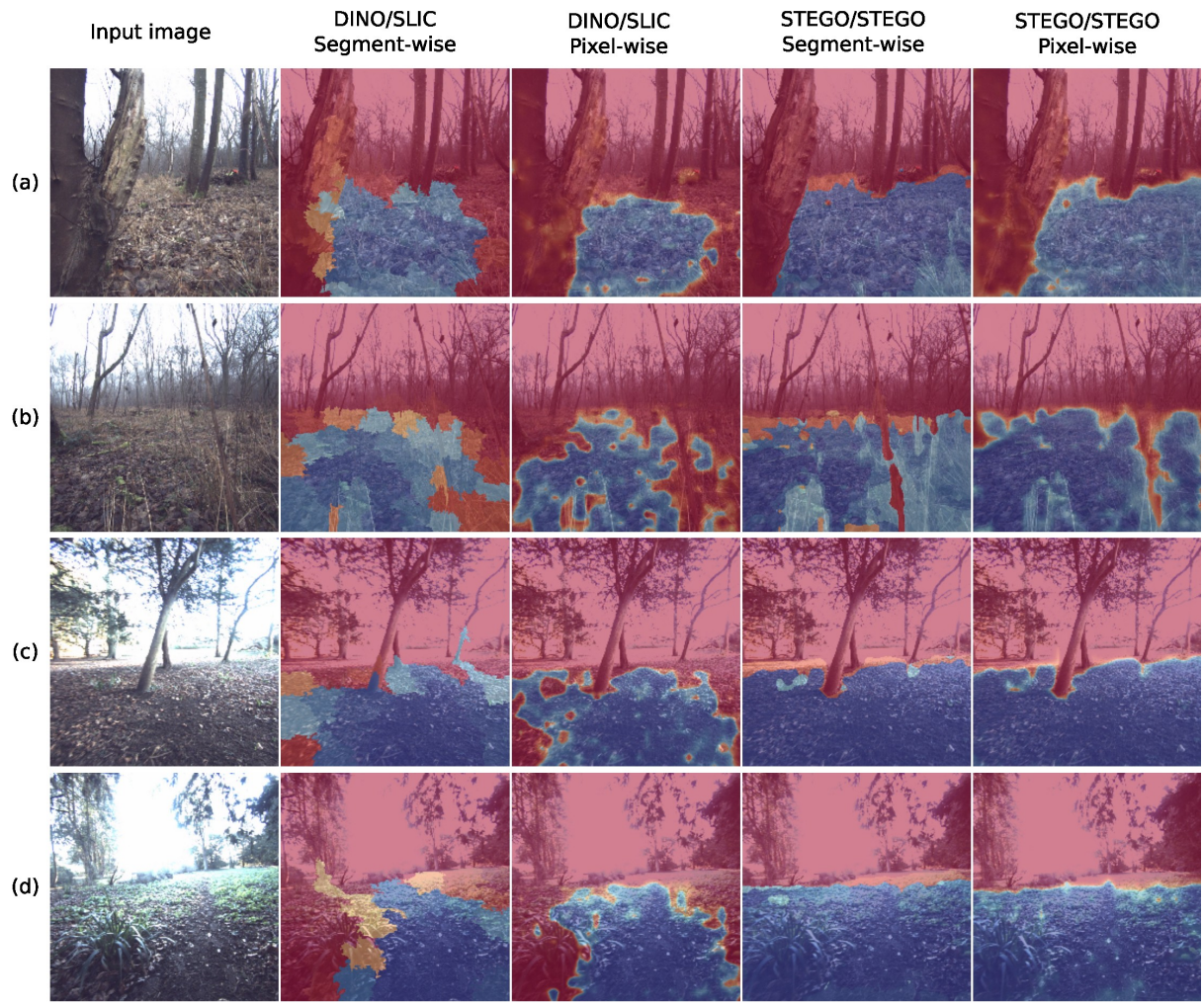

Fig. 11 Inference approaches: We qualitatively compared segment-wise and pixel-wise inference using pre-trained DINO and STEGO features. We observed advantages in executing the inference in a pixel-wise manner, which provided a fine-grained prediction regardless of the pre-trained features.

STEGO did not produce significant differences in terms of the output, consistently segmenting the traversed areas for both inference approaches. However, we did observe problems over segmenting certain areas, such as the plants in row (d), which suggest that both the features and the segments 'agreed' on the object being semantically similar to the other traversed areas.

5.3.2 Feature Subsampling

This second experiment compared different subsampling strategies presented in Sec. 3.2.3 in terms of traversability prediction and training. Our methodology involved re-running WVN in post-processing using the recorded signals from the *Autonomous Navigation in the Park* sequence (Sec. 5.2.4). We executed five runs for each case, training the traversability prediction model from scratch without any pre-trained weights. We recorded the produced traversability predictions as well as the learning curves. Fig. 12 shows the training loss, averaged over the five runs with 2σ confidence bounds, and qualitative examples of the traversability predictions when using the pixel-wise inference method.

We observed the most benefit when using the STEGO segments and features, which enabled rapid

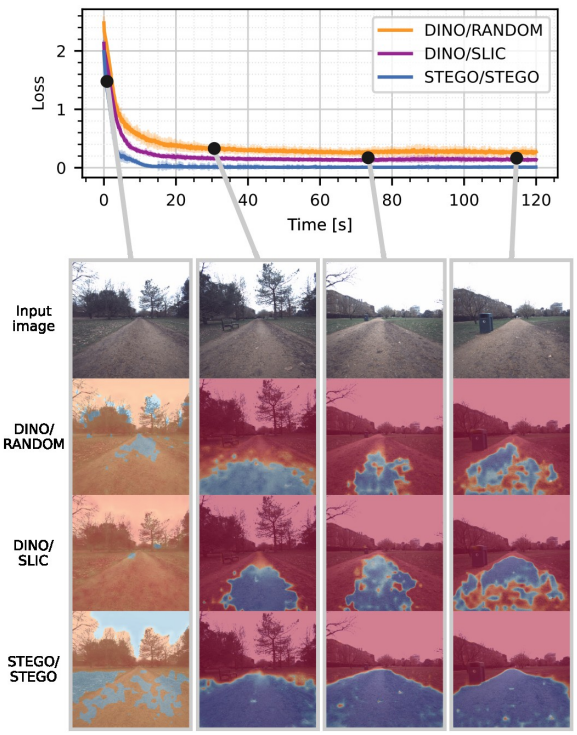

Fig. 12 Feature Sub-sampling: We tested the different sub-sampling methods in the recorded path-following sequence from Sec. 5.2.4. We observed that STEGO provides significant improvements for the path-following task in both traversability prediction fidelity and training stability.

adaptation in terms of segmenting the footpath as traversable (Fig. 12). This was also reflected in the corresponding loss curves, which achieved faster convergence and lower training loss than the other methods.

Regarding the two other sub-sampling methods, random and SLIC, we did not observe significant differences in the predicted traversability. This can be explained by the use of the same DINO-ViT features, which suggests that most of the expressive power is already encoded in the features, and the contribution of the sampling and mean averaging does not considerably affect the predictions. The main difference is the slightly improved training stability reflected on the lower confidence bounds of SLIC compared to random sub-sampling.

6 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 demonstrated WVN through different real-world experiments and offline analyses, illustrating its fast adaptation capabilities, the consistency of its traversability prediction for local planning, and closed-loop navigation experiments, in both indoor and natural scenes. The experiments validated the key idea behind our approach of exploiting the semantic priors from pre-trained models, enabling fast generalization and adaptation in unseen scenarios from small data collected during demonstrations *in the wild*.

Regarding the limitations, the use of traction as the traversability score metric, as well as the closed-loop integration with the local terrain map via raycasting are the main aspects to investigate. These are some of the main open scientific and engineering questions for WVN.

Lastly, to foster further research in the field, we provide the community with the codebase and pre-trained models for different environments as baselines.

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.

The authors also thank Benoit Casseau for technical support, and Pía Cortés-Zuleta for critically proofreading the manuscript.

References

[1] Gibson, J.J.: The Ecological Approach to Visual Perception. Houghton Mifflin, Boston (1979)

[2] Moravec, H., Elfes, A.: High Resolution Maps from Wide Angle Sonar. In: IEEE Int. Conf. Robot. Autom. (ICRA), vol. 2, pp. 116–121 (1985). https://doi.org/10.1109/ROBOT.1985.1087316

[3] Miki, T., Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., Hutter, M.: Learning Robust Perceptive Locomotion for Quadrupedal Robots in the Wild. Sci. Robot. **7**(62) (2022) https://doi.org/10.1126/SCIROBOTICS.ABK2822

[4] Maturana, D., Chou, P.-W., Uenoyama, M., Scherer, S.: Real-time Semantic Mapping for Autonomous Off-Road Navigation. In: Field and Service Robotics, pp. 335–350 (2017). https://doi.org/10.1007/978-3-319-67361-5_22

[5] Wellhausen, L., Ranftl, R., Hutter, M.: Safe Robot Navigation Via Multi-Modal Anomaly Detection. IEEE Robot. Autom. Lett. (RA-L) (2020) https://doi.org/10.1109/LRA.2020.2967706

[6] Wellhausen, L., Dosovitskiy, A., Ranftl, R., Walas, K., Cadena, C., Hutter, M.: Where Should I Walk? Predicting Terrain Properties from Images via Self-Supervised Learning. IEEE Robot. Autom. Lett. (RA-L) **4**(2), 1509–1516 (2019-04) https://doi.org/10.1109/LRA.2019.2895390

[7] Gasparino, M.V., Sivakumar, A.N., Liu, Y., Velasquez, A.E.B., Higuti, V.A.H., Rogers, J., Tran, H., Chowdhary, G.: WayFAST: Navigation With Predictive Traversability in the Field. IEEE Robot. Autom. Lett. (RA-L) **7**(4), 10651–10658 (2022) https://doi.org/10.1109/LRA.2022.3193464

[8] Kim, D., Sun, J., Oh, S.M., Rehg, J.M., Bobick, A.F.: Traversability Classification using Unsupervised On-line Visual Learning for Outdoor Robot Navigation. In: IEEE Int. Conf. Robot. Autom. (ICRA), pp. 518–525 (2006). https://doi.org/10.1109/ROBOT.2006.1641763

[9] Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Scoffier, M., Kavukcuoglu, K., Muller, U., LeCun, Y.: Learning Long-range Vision for Autonomous Off-road Driving. J. Field Robot. **26**(2), 120–144 (2009) https://doi.org/10.1002/ROB.20276

[10] Frey, J., Mattamala, M., Chebrolu, N., Cadena, C., Fallon, M., Hutter, M.: Fast Traversability Estimation for Wild Visual Navigation. In: Robotics: Science and Systems (RSS) (2023). https://doi.org/10.15607/RSS.2023.XIX.054

[11] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging Properties in Self-Supervised Vision Transformers. In: Intl. Conf. on Computer Vision (ICCV) (2021). https://doi.org/10.1109/ICCV48922.2021.00951

[12] Hamilton, M., Zhang, Z., Hariharan, B., Snavely, N., Freeman, W.T.: Unsupervised semantic segmentation by distilling feature correspondences. In: Intl. Conf. on Learning Representations (ICLR) (2022). https://openreview.net/forum?id=SaKO6z6Hl0c

[13] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. IEEE Trans. Pattern Anal. Mach. Intell. **34**(11), 2274–2282 (2012) https://doi.org/10.1109/TPAMI.2012.120

[14] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A.: ROS: an open-source Robot Operating System. In: IEEE Int. Conf. Robot. Autom. (ICRA) (2009)

[15] Chung, T.H., Orekhov, V., Maio, A.: Into the Robotic Depths: Analysis and Insights from the DARPA Subterranean Challenge. Annual Review of Control, Robotics, and Autonomous Systems **6**(1) (2023) https://doi.org/10.1146/ANNUREV-CONTROL-062722-100728

[16] Cao, C., Zhu, H., Yang, F., Xia, Y., Choset, H., Oh, J., Zhang, J.: Autonomous Exploration Development Environment and the Planning Algorithms. In: IEEE Int. Conf. Robot. Autom. (ICRA), pp. 8921–8928 (2022). https://doi.org/10.1109/ICRA46639.2022.9812330

[17] Fan, D.D., Otsu, K., Kubo, Y., Dixit, A., Burdick, J., Agha-Mohammadi, A.: STEP: Stochastic Traversability Evaluation and Planning for

Safe Off-road Navigation. In: Robotics: Science and Systems (RSS) (2021). https://doi.org/10.15607/RSS.2021.XVII.021

[18] Chavez-Garcia, R.O., Guzzi, J., Gambardella, L.M., Giusti, A.: Learning Ground Traversability From Simulations. IEEE Robot. Autom. Lett. (RA-L) **3**(3), 1695–1702 (2018) https://doi.org/10.1109/LRA.2018.2801794

[19] Yang, B., Wellhausen, L., Miki, T., Liu, M., Hutter, M.: Real-time Optimal Navigation Planning Using Learned Motion Costs. In: IEEE Int. Conf. Robot. Autom. (ICRA), pp. 9283–9289 (2021). https://doi.org/10.1109/ICRA48506.2021.9561861

[20] Frey, J., Hoeller, D., Khattak, S., Hutter, M.: Locomotion Policy Guided Traversability Learning using Volumetric Representations of Complex Environments. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS) (2022). https://doi.org/10.1109/IROS47612.2022.9982190

[21] Bradley, D.M., Chang, J.K., Silver, D., Powers, M., Herman, H., Rander, P., Stentz, A.: Scene understanding for a high-mobility walking robot. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 1144–1151 (2015). https://doi.org/10.1109/IROS.2015.7353514

[22] Schilling, F., Chen, X., Folkesson, J., Jensfelt, P.: Geometric and Visual Terrain Classification for Autonomous Mobile Navigation. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 2678–2684 (2017). https://doi.org/10.1109/IROS.2017.8206092

[23] Belter, D., Wietrzykowski, J., Skrzypczynski, P.: Employing natural terrain semantics in motion planning for a multi-legged robot. J. Intell. Robotic Syst. **93**(3-4), 723–743 (2019) https://doi.org/10.1007/S10846-018-0865-X

[24] Shaban, A., Meng, X., Lee, J., Boots, B., Fox, D.: Semantic Terrain Classification for Off-Road Autonomous Driving. In: Faust, A., Hsu, D., Neumann, G. (eds.) Conf. on Robot Learning (CoRL). Proceedings of Machine Learning Research, vol. 164, pp. 619–629 (2022)

[25] Cai, X., Everett, M., Fink, J., How, J.P.: Risk-aware off-road navigation via a learned speed distribution map. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 2931–2937 (2022). https://doi.org/10.1109/IROS47612.2022.9982200

[26] Bajracharya, M., Howard, A., Matthies, L.H., Tang, B., Turmon, M.: Autonomous off-road navigation with end-to-end learning for the lagr program. J. Field Robot. **26**(1), 3–25 (2009) https://doi.org/10.1002/ROB.20269

[27] Zürn, J., Burgard, W., Valada, A.: Self-supervised visual terrain classification from unsupervised acoustic feature learning. IEEE Trans. Robot. **37**(2), 466–481 (2021) https://doi.org/10.1109/TRO.2020.3031214

[28] Kahn, G., Abbeel, P., Levine, S.: BADGR: An Autonomous Self-Supervised Learning-Based Navigation System. IEEE Robot. Autom. Lett. (RA-L) **6**(2), 1312–1319 (2021) https://doi.org/10.1109/LRA.2021.3057023

[29] Sathyamoorthy, A.J., Weerakoon, K., Guan, T., Liang, J., Manocha, D.: TerraPN: Unstructured terrain navigation using online self-supervised learning. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 7197–7204 (2022). https://doi.org/10.1109/IROS47612.2022.9981942

[30] Guaman Castro, M., Triest, S., Wang, W., Gregory, J.M., Sanchez, F., Rogers, J.G., Scherer, S.: How does it feel? self-supervised costmap learning for off-road vehicle traversability. In: IEEE Int. Conf. Robot. Autom. (ICRA), pp. 931–938 (2023). https://doi.org/10.1109/ICRA48891.2023.10160856 . IEEE

[31] Jung, S., Lee, J., Meng, X., Boots, B., Lambert, A.: V-STRONG: visual self-supervised traversability learning for off-road navigation. In: IEEE Int. Conf. Robot. Autom. (ICRA) (2024)

[32] Richter, C., Roy, N.: Safe visual navigation via deep learning and novelty detection. In: Robotics: Science and Systems (RSS), Cambridge, Massachusetts (2017). https://doi.org/10.15607/RSS.2017.XIII.064

[33] Schmid, R., Atha, D., Schöller, F., Dey, S., Fakoorian, S., Otsu, K., Ridge, B., Bjelonic, M., Wellhausen, L., Hutter, M., Agha-mohammadi, A.-a.: Self-supervised traversability prediction by learning to reconstruct safe terrain. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 12419–12425 (2022). https://doi.org/10.1109/IROS47612.2022.9981368

[34] Ji, T., Sivakumar, A.N., Chowdhary, G., Driggs-Campbell, K.: Proactive Anomaly Detection for Robot Navigation With Multi-Sensor Fusion. IEEE Robot. Autom. Lett. (RA-L) **7**(2), 4975–4982 (2022) https://doi.org/10.1109/LRA.2022.3153989

[35] Seo, J., Kim, T., Kwak, K., Min, J., Shim, I.: Scate: A scalable framework for self- supervised traversability estimation in unstructured environments. IEEE Robotics and Automation Letters **8**(2), 888–895 (2023) https://doi.org/10.1109/LRA.2023.3234768

[36] Katevenis, M., Sidiropoulos, S., Courcoubetis, C.: Weighted round-robin cell multiplexing in a general-purpose ATM switch chip. IEEE J. Sel. Areas Commun. **9**(8), 1265–1279 (1991) https://doi.org/10.1109/49.105173

[37] Lee, H., Kwak, K., Jo, S.: An Incremental Nonparametric Bayesian Clustering-based Traversable Region Detection Method. Auton. Robot. **41**(4), 795–810 (2017) https://doi.org/10.1007/S10514-016-9588-7

[38] Cai, X., Ancha, S., Sharma, L., Osteen, P.R., Bucher, B., Phillips, S., Wang, J., Everett, M., Roy, N., How, J.P.: EVORA: deep evidential traversability learning for risk-aware off-road autonomy. CoRR **abs/2311.06234** (2023) https://doi.org/10.48550/ARXIV.2311.06234

[39] Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: Intl. Conf. on Learning Representations (ICLR) (2015)

[40] Miki, T., Wellhausen, L., Grandia, R., Jenelten, F., Homberger, T., Hutter, M.: Elevation Mapping for Locomotion and Navigation using GPU. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 2273–2280 (2022). https://doi.org/10.1109/IROS47612.2022.9981507

[41] Erni, G., Frey, J., Miki, T., Mattamala, M., Hutter, M.: MEM: Multi-Modal Elevation Mapping for Robotics and Learning. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS) (2023). https://doi.org/10.1109/IROS55552.2023.10342108

[42] Mattamala, M., Chebrolu, N., Fallon, M.: An Efficient Locally Reactive Controller for Safe Navigation in Visual Teach and Repeat Missions. IEEE Robot. Autom. Lett. (RA-L) **7**(2), 2353–2360 (2022) https://doi.org/10.1109/LRA.2022.3143196

[43] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Intl. Conf. on Neural Information Processing Systems (NeurIPS) (2019)

[44] Wermelinger, M., Fankhauser, P., Diethelm, R., Krüsi, P.A., Siegwart, R., Hutter, M.: Navigation Planning for Legged Robots in Challenging Terrain. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS) (2016). https://doi.org/10.1109/IROS.2016.7759199