Autonomous Robots (2024) 48:4
https://doi.org/10.1007/s10514-024-10158-4

Terrain traversability prediction through self-supervised learning and unsupervised domain adaptation on synthetic data

Giuseppe Vecchio[1] · **Simone Palazzo**[1] · **Dario C. Guastella**[1] · **Daniela Giordano**[1] · **Giovanni Muscato**[1] · **Concetto Spampinato**[1]

Received: 9 June 2023 / Accepted: 17 March 2024

Abstract
Terrain traversability estimation is a fundamental task for supporting robot navigation on uneven surfaces. Recent learning-based approaches for predicting traversability from RGB images have shown promising results, but require manual annotation of a large number of images for training. To address this limitation, we present a method for traversability estimation on unlabeled videos that combines dataset synthesis, self-supervision and unsupervised domain adaptation. We pose the traversability estimation as a vector regression task over vertical bands of the observed frame. The model is pre-trained through self-supervision to reduce the distribution shift between synthetic and real data and encourage shared feature learning. Then, supervised training on synthetic videos is carried out, while employing an unsupervised domain adaptation loss to improve its generalization capabilities on real scenes. Experimental results show that our approach is on par with standard supervised training, and effectively supports robot navigation without the need of manual annotations. Training code and synthetic dataset will be publicly released at: https://github.com/perceivelab/traversability-synth.

Keywords Terrain traversability · Unmanned ground vehicle · Self-supervised learning · Domain adaptation · Synthetic data · Sim-to-real

1 Introduction

Identifying traversable paths and, accordingly, taking proper control actions is a fundamental requirement for a mobile robot to safely and autonomously navigate in non-urban environments. This capability is needed in several robotic applications where autonomous vehicle navigation plays a crucial role, such as search and rescue (Delmerico et al. 2019), precision farming (Yandun Narváez et al. 2018) and planetary exploration (Hewitt et al. 2017). Several methods have been proposed for autonomous navigation of ground vehicles in unstructured environments. Among them, *terrain traversability analysis* is a widely adopted approach, which has recently gained momentum thanks to the development of learning-based methods (Guastella and Muscato 2021; Borges et al. 2022).

Some works address terrain traversability analysis as a traversal cost regression problem, using inverse reinforcement learning (Pflueger et al. 2019; Zhu et al. 2019 to derive a map of costs for a subsequent path planning phase. Other approaches identify the terrain type (Rothrock et al. 2016; Gonzalez and Iagnemma 2018) or pose the problem as a binary classification task (i.e. “traversable” or “non-traversable”) (Chavez-Garcia et al. 2018; Holder and Breckon 2018) or as a per-region regression task (Palazzo et al. 2020).

One of the main limitations of existing learning-based approaches tackling traversability estimation is the need for *annotated* training data. Regardless of the specific formula-

✉ Dario C. Guastella
dario.guastella@unict.it

Giuseppe Vecchio
giuseppe.vecchio@phd.unict.it

Simone Palazzo
simone.palazzo@unict.it

Daniela Giordano
daniela.giordano@unict.it

Giovanni Muscato
giovanni.muscato@unict.it

Concetto Spampinato
concetto.spampinato@unict.it

1 Department of Electrical, Electronic and Computer Engineering, University of Catania, Via Santa Sofia 64, 95123 Catania, Italy

tion of traversability, it is necessary to collect a large set of images, generally requiring a robot to be operated on the target environment (or a similar one). Then, a human operator needs to analyze each individual frame and mark traversable areas. A second limitation of existing approaches is the lack of strategies to use traversability predictions for navigation. Although existing methods show promising results in terms of prediction accuracy, it is not obvious how to use their outputs to drive a ground vehicle.

In this work, we propose a method to tackle both limitations, by (1) training a model on a combination of annotated synthetic data, used for supervised training, and unannotated real data, used for unsupervised domain adaptation; (2) developing a simple, yet effective navigation strategy using the estimated traversability.

In particular, a synthetic dataset is first generated using the MIDGARD (Vecchio et al. 2022) simulation environment. Training is then carried out by combining self-supervision (Caron et al. 2020) and domain adaptation (Ganin and Lempitsky 2015) techniques to ensure that the model, trained in a supervised way on synthetic images and annotations, behaves correctly when processing real-world data.

Furthermore, we propose a navigation algorithm that employs the egocentric outcome of the traversability prediction by defining a control law to directly steer the vehicle towards the safest area. On-field and simulated tests show that the proposed approach is capable of autonomously exploring unknown environments, while avoiding obstacles and harsh terrains.

To summarize, the contributions of our paper are the following:

- We create a photorealistic synthetic dataset for traversability estimation in non-urban environments, with computer-generated annotations for supervised training of learning-based models.
- We propose a deep learning method for traversability estimation from RGB images, employing supervised training on synthetic data, self-supervised pre-training of the traversability predictor, and unsupervised adaptation on real data, thus relieving the user from the burden of manual annotation. Performance analysis shows that the proposed model outperforms existing approaches when real annotations are used.
- We perform both simulated and on-field validation of the proposed method and define an autonomous navigation approach based on the traversability outcome.

2 Related work

2.1 Terrain traversability analysis

Traversability anticipation has been proposed as a long-range prediction problem based on visual data since early works (Howard et al. 2006; Hadsell et al. 2008), as an alternative to the limited perception range of LIDARs or stereo cameras. More recent deep-learning models perform image segmentation for terrain classification, either as a binary (i.e., traversable or not) Holder and Breckon 2018 or as a multi-class classification problem (Rothrock et al. 2016; Valada et al. 2016; Maturana et al. 2018). However, the literature does not offer an established approach for translating the traversability outcome into driving commands for the vehicle. The most common solution is to project scene classification output from the image plane to a polar top-view map of the vehicle's surroundings (Howard et al. 2006; Hadsell et al. 2008; Maturana et al. 2018), thus falling back into a path planning problem. Other works leveraging inverse reinforcement learning aim at directly inferring traversability costmaps and planning trajectories from the demonstrations by an expert and the environment point cloud (Zhang et al. 2018; Zhu et al. 2020), but rely heavily on 3D LIDARs rather than passive RGB cameras.

Besides terrain traversability analysis, *end-to-end* methods have recently been proposed for off-road navigation, where a direct mapping from exteroceptive data (typically a combination of geometric and visual data) and driving commands is performed (Pan et al. 2020; Nguyen et al. 2020). Successful examples have also been reported on the navigation of multi-rotors for outdoor unstructured environments, such as forests (Smolyanskiy et al. 2017; Giusti et al. 2016). However, end-to-end methods tend to make it more difficult to grasp the relationship between the perceived environment and the chosen driving action, due to the lack of interpretable intermediate representations.

Inspired by Palazzo et al. (2020), our approach provides an immediate interpretation of a scene, without the need to construct a top-view costmap, that can be directly used to drive the vehicle towards traversable regions within the camera field of view. The proposed approach mainly differs from Palazzo et al. (2020) in the usage of computer-generated annotations on a synthetic dataset, bridging the gap between synthetic and real images by means of a self-supervision approach (specifically adapted to the problem at hand; see Sect. 3.5) and unsupervised domain adaptation

(Sect. 3.7). Additionally, we improve the model architecture by pre-training and freezing the feature extraction backbone network, significantly speeding up training, while increasing the complexity of the traversability estimation network to compensate for the extraction of more generic backbone features (Sect. 3.4). Finally, we replace the "greedy" navigation algorithm in Palazzo et al. (2020) with a more robust approach (Sect. 4), and provide an extensive analysis of the performance of our approach in a real-world scenario (Sect. 6).

2.2 Synthetic data collection in simulated environments

Training on synthetic data has proven to be a suitable alternative to training on real-world data for autonomous navigation. Recent work has focused on the creation of large-scale, high-resolution synthetic datasets (Haltakov et al. 2013; Richter et al. 2016; Gaidon et al. 2016; Richter et al. 2017; Müller et al. 2021), as well as the development of embodied simulators for training (Vecchio et al. 2022; Skinner et al. 2016; Kolve et al. 2017; Dosovitskiy et al. 2017; Shah et al. 2018; Xia et al. 2018; Savva et al. 2019; Song et al. 2020; Kadian et al. 2020). Several of these simulation platforms have been used to support dataset generation for navigation tasks, both in structured (Dosovitskiy et al. 2017; Savva et al. 2019) and unstructured (Shah et al. 2018; Song et al. 2020) environments:

- AirSim (Shah et al. 2018) is an open source simulator which supports software- and hardware-in-the-loop simulation, providing both an interactive mode for live agents' training and a data collection mode.
- OAISYS (Müller et al. 2021) is a photorealistic terrain simulation pipeline for unstructured outdoor environments, built on top of Blender.[1] It is designed for the collection of synthetic datasets and is capable of generating large varieties of scenes with automatic annotations in terms of instance segmentation, semantic segmentation, and depth.
- MIDGARD (Vecchio et al. 2022) features an interactive mode as well as a data collection mode for creating automatically-annotated datasets. It provides a wide variety of sensors including depth, semantic and instance segmentation, and a traversability-annotator tool.

In this work, we use MIDGARD to automatically collect the synthetic training dataset, as described in Sect. 5.2.

[1] https://www.blender.org

2.3 Domain adaptation

It is recognized that the availability of annotated data is often limited by the required efforts for their collection. Moreover, the performance of deep learning models typically degrades when applied to a data distribution that does not match the training one (either real or synthetic), which represents a further difficulty in the usage of pre-trained models on a custom task. Several *transfer learning* and *domain adaptation* methods have therefore been proposed in the literature (Wang and Deng 2018), which aim at dealing with the distribution shift between different data domains. In this work, we focus on *unsupervised* domain adaptation: we assume that we have an annotated *source* dataset, on which a model can be trained supervisedly, and an unannotated *target* dataset, on which we intend to ultimately employ the trained model. A straightforward approach to unsupervised domain adaptation treats it as a classification task on the target domain, using *pseudo-labels* estimated for target samples by a model trained on the source domain (Yan et al. 2017; Saito et al. 2017; Zhang et al. 2015; Long et al. 2016). The success of these approaches usually depends on the similarity between the source and target distributions, and thus on how accurate pseudo-labels are. Other approaches aim, instead, at minimizing the difference between feature statistics on the two domains: many of these methods are based on introducing a *maximum mean discrepancy* loss term to the training objective (Borgwardt et al. 2006; Ghifary et al. 2014; Long et al. 2015; Zellinger et al. 2017). A similar objective can be pursued by leveraging generative adversarial networks (GANs) (Goodfellow et al. 2014; Liu and Tuzel 2016; Yoo et al. 2016; Shrivastava et al. 2017; Bousmalis et al. 2017). Inspired by the adversarial competition of GANs, the *gradient reversal layer* (GRL) method (Ganin and Lempitsky 2015) learns an intermediate representation that is designed to *maximize* domain classification error, so that similar features are extracted from the two domains. In this work, we opt for this approach due to its simplicity and its recent success in complex domain adaptation scenarios (Palazzo et al. 2020; Bellitto et al. 2020).

2.4 Self-supervised learning

Model self-supervision aims at learning features from unannotated data, in the attempt to alleviate the need for human-crafted ground truth and reduce the performance gap with supervised networks pre-training (Chen et al. 2020; He et al. 2020; Misra and Maaten 2020. Some approaches formulate self-supervision as an *instance discrimination* task (Dosovitskiy et al. 2015), where each image in the dataset is considered as a single class. Other methods pose the problem by defining a contrastive loss (Hadsell et al. 2006) that attempts to extract similar features from an image and its transformations, while pushing away features from differ-

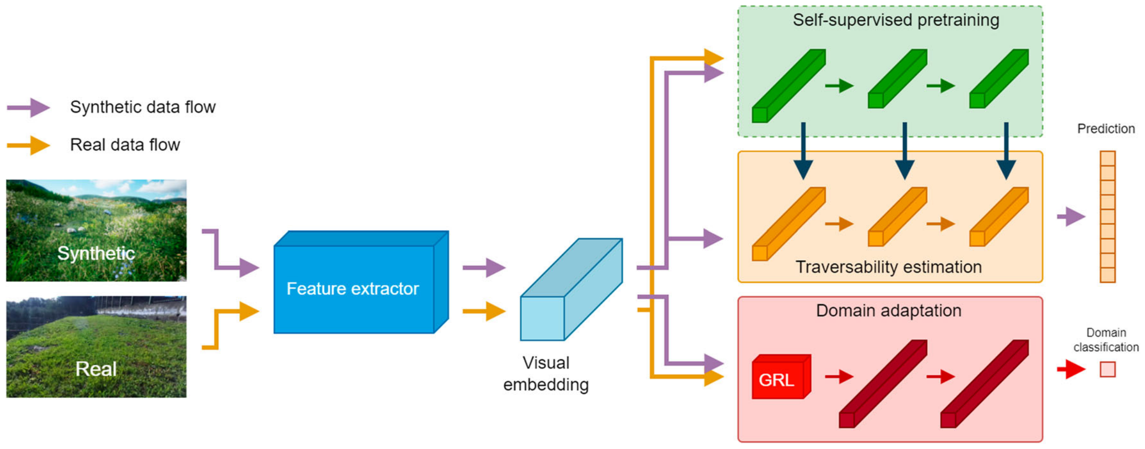


Fig. 1 The proposed framework. Real and synthetic images are fed to a pre-trained feature extractor. Visual embeddings are then fed to a traversability estimation module. The traversability estimator is first pre-trained through self-supervision on both real and synthetic images, and then trained in a supervised way using only annotations for the synthetic data. Features from both domains are fed to a domain classification layer, trained to distinguish between real and synthetic features: gradients estimated through backpropagation are altered by a *gradient reversal layer* (GRL) (Ganin and Lempitsky 2015) to maximize classification loss and match feature distributions between real and synthetic images

ent images. This removes the notion of instance classes by directly comparing image features, enforcing invariance across features of corresponding transformations. Since comparing all possible image pairs in a large dataset is computationally challenging, many works approximate the loss by reducing the number of comparisons to random subsets of images during training (Chen et al. 2020; He et al. 2020; Wu et al. 2018). Among these, a recent approach Caron et al. (2020) proposes to learn features by "swapping assignments between multiple views" (SwAV) of the same image. In this work, we adapt SwAV to make it suitable to our traversability task, in order to initialize the model with features that apply to both real and synthetic data.

3 Traversability prediction

3.1 Overview

An overview of the proposed architecture for traversability prediction is shown in Fig. 1. A feature extraction backbone is used to compute compact representations of the input images. Visual features are then processed by two model branches: one estimating the traversability, initially pre-trained with self-supervision on both real and synthetic images, and then fine-tuned in a supervised way on synthetic images only; the other is, instead, trained to distinguish between real and synthetic images, thus supporting unsupervised domain adaptation to unannotated real images.

3.2 Problem formulation

Following the definition in Palazzo et al. (2020), we pose the traversability estimation as a vector regression problem, where each component of the target vector indicates the traversability score of a corresponding region in the input image. More in detail, we divide the input RGB images into a set of vertical bands and regress an array of traversability scores related to the traversable horizon within each band. Although this formulation may seem to oversimplify the traversability estimation problem, it properly drives the vehicle to avoid potentially dangerous terrain areas, since there is no need to know any further information on the environment beyond the traversable horizon within each band. The navigation capabilities of the robotic platform are implicitly taken into account during the annotation process (e.g., the maximum traversable surface steepness). Automatic annotation for synthetic data is described in Sect. 5.2.

We tackle the problem as an unsupervised domain adaptation task, where we enforce the model, trained in a supervised way on a *source* synthetic dataset, to generalize to an unannotated *target* dataset of real images.

More formally, given a tensor $\mathbf{I} \in \mathcal{I}$ of size $C \times H \times W$ representing an RGB image, this is divided into a set of k vertical bands: $\{\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_k\}$, with:

$$\mathbf{I}_i = \mathbf{I}_{\left[0:C-1, 0:H-1, \frac{iW}{k} : \frac{(i+1)W}{k} - 1\right]}, \tag{1}$$

where $\mathbf{I}_{[\cdot]}$ denotes subtensor indexing and the : operator selects a range of coordinates along the corresponding dimension. Each resulting portion has size $C \times H \times \frac{W}{k}$: if W is

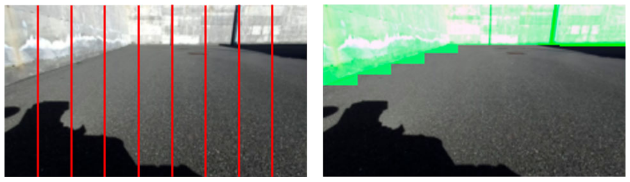


Fig. 2 Example of image division with $k = 9$ vertical bands; for each band, a *traversability score* indicates the position (in terms of image height) of the closest non-traversable elements within the band. Image from Palazzo et al. (2020)

not divisible by k, the width of each vertical area is suitably rounded. The number of vertical bands k is chosen to be an odd number, as the band position within the image frame determines a potential *direction* to drive the vehicle (see Sect. 6). Indeed, when k is odd, the presence of a central band avoids an undesired oscillatory behavior during navigation when the platform is supposed to go straight. Figure 2 shows an example of the resulting division for a given input image. *Traversability scores* for each region are encoded by a vector $\mathbf{t} \in \mathbb{R}^k$ for each of the image subregions, ranging from 0 (not traversable) to 1 (fully traversable).

Given an input image, our objective is to estimate a set of traversability scores $\tilde{\mathbf{t}}$ that approximate target values $\mathbf{t}$, by means of a deep model implementing a function $f : \mathcal{I} \rightarrow [0, 1]^k$. Given an input image $\mathbf{I}$, f is trained to estimate $\tilde{\mathbf{t}} = f(\mathbf{I})$ that is as close as possible to the actual $\mathbf{t}$.

Unlike the approach introduced in Palazzo et al. (2020), we do not aim to learn f in a supervised way on a real dataset, i.e., by training the model with the correct manually-annotated traversability scores. Instead, we assume that data used for training our model can be split into a *source domain* $\mathcal{D}_s$ and a *target domain* $\mathcal{D}_t$. The source domain $\mathcal{D}_s$ includes pairs $(\mathbf{I}, \mathbf{t})$ of synthetic images with computer-generated traversability scores. The target domain $\mathcal{D}_t$, instead, includes real images for which no manual annotations are available during training.

3.3 Feature extraction backbone

A DeepLabV2 Chen et al. (2016) backbone, based on ResNet-101 He et al. (2016) and pre-trained on COCO Stuff 10k Caesar et al. (2016), is employed to extract visual features from an input RGB image. During training, we *freeze* (i.e., do not fine-tune) the parameters of the backbone. This choice aims at reducing overfitting on relatively small datasets, due to the complexity of the backbone. As an additional benefit, pre-computing image features speeds up the training process. Our experiments (Sect. 5) show that freezing the feature extraction backbone indeed improves performance compared to fine-tuning the entire architecture.

Table 1 Architectural details of the traversability estimation network in the proposed model

Layer	Input size	Kernel size	Output size
DeepLabV2	$3 \times 128 \times 227$	–	$2048 \times 17 \times 29$
Conv. 2D	$2048 \times 17 \times 29$	1×1	$256 \times 17 \times 29$
Adapt. pool	$256 \times 17 \times 29$	–	$256 \times 64 \times 64$
ResLayer	$256 \times 64 \times 64$	3×3	$256 \times 64 \times 64$
Max pool	$256 \times 64 \times 64$	2×2	$256 \times 32 \times 32$
ResLayer	$256 \times 32 \times 32$	3×3	$512 \times 32 \times 32$
Max pool	$512 \times 32 \times 32$	2×2	$256 \times 16 \times 16$
ResLayer	$512 \times 16 \times 16$	3×3	$1024 \times 16 \times 16$
Max pool	$1024 \times 16 \times 16$	2×2	$1024 \times 8 \times 8$
ResLayer	$1024 \times 8 \times 8$	3×3	$1024 \times 8 \times 8$
Max pool	$1024 \times 8 \times 8$	2×2	$1024 \times 4 \times 4$
Conv. 2D	$1024 \times 4 \times 4$	1×1	$128 \times 4 \times 4$
Fully-conn	2048	–	9

3.4 Traversability estimation

The architecture of our traversability estimation network is designed to process the features extracted from the DeepLabV2 backbone and regress a traversability score for each image portion. In detail, it receives input features from the DeepLabV2 backbone and processes them through a cascade of convolutional layers, aimed at gradually increasing the number of features while reducing spatial dimensions; a final fully-connected layer with k-dimensional output predicts traversability scores. Architectural details of the layers in the traversability estimation network are presented in Table 1. Each convolutional layer is followed by batch normalization and ReLU activation.

3.5 Self-supervised initialization

Inspired by Palazzo et al. (2020), we employ domain adaptation (described in the next section) to simultaneously train our model to perform traversability prediction on synthetic images and to adapt itself to perform the same task on real images: the objective is to encourage the model to learn features that work equally well on both the synthetic source domain and the real-world target domain. However, unlike (Palazzo et al. 2020), we do not fine-tune the DeepLabV2 backbone during training; hence, the representation it extracts does not adapt to the specific characteristics of the target domain. To mitigate this issue, we perform a self-supervised initialization step, where we pre-train our model on both real and synthetic datasets, without using traversability annotations, in order to learn features applicable to both domains before the supervised training phase.

The self-supervision approach we employ in this work is SwAV (Caron et al. 2020). SwAV is a clustering-based method with the objective of learning a representation of input data such that it is possible to predict the cluster assignment of one view of an image from the representation of another view, thus enforcing consistency between features extracted from variants of the same image.

Formally, given two image representations $\mathbf{z}_s$ and $\mathbf{z}_t$, computed from different views of the same image, a model is trained to compute their "codes" (i.e., cluster assignments) $\mathbf{q}_s$ and $\mathbf{q}_t$ by matching them to a set of learnable cluster prototypes $\{\mathbf{c}_1, ..., \mathbf{c}_K\}$.

The self-supervision objective is a "swapped" cluster assignment prediction, where the model is trained to predict $\mathbf{q}_s$ from $\mathbf{z}_t$ and $\mathbf{q}_t$ from $\mathbf{z}_s$, by minimizing the cross-entropy between the target code and the probability distribution obtained by projecting features on the set of prototype vectors:

$$\mathcal{L}(\mathbf{z}_t, \mathbf{z}_s) = -\sum_k \mathbf{q}_s^{(k)} \log \mathbf{p}_t^{(k)} - \sum_k \mathbf{q}_t^{(k)} \log \mathbf{p}_s^{(k)}, \tag{2}$$

where $\mathbf{p}_t^{(k)}$ (and, similarly, $\mathbf{p}_s^{(k)}$) is computed as:

$$\mathbf{p}_t^{(k)} = \frac{\exp \frac{1}{\tau} \mathbf{z}_t^\top \mathbf{c}_k}{\sum_{k'} \exp \frac{1}{\tau} \mathbf{z}_t^\top \mathbf{c}_{k'}}, \tag{3}$$

with τ being a temperature parameter to "flatten" the softmax distribution. Intuitively, $\mathbf{p}_t^{(k)}$ estimates the probability that feature $\mathbf{z}_t$ is associated to cluster k: the objective of training is to ensure that such a probability is maximal for the cluster corresponding to the code of the other view of the image, $\mathbf{q}_s$. As a result, the model learns to project both views to similar representations, hence learning to extract reusable features that take into account visual context and semantics.

In this work, we introduce a variant of SwAV, which differs from the original formulation in two main aspects.

First, SwAV is specifically designed for image classification, and it is based on the assumption that all patches within an image share the same cluster assignment, since they all refer to a single depicted object. In our case, this hypothesis is counter-productive, as different patches within the same image may exhibit significantly different traversability properties. For this reason, we propose a revised consistency assumption, shifting from the original "same image, same cluster" hypothesis to an assumption that places emphasis on local feature similarity rather than global image homogeneity, motivated by the observation that *horizontally contiguous* patches are likely to exhibit similar visual characteristics (see Fig. 2). By focusing on local continuity, our approach encourages the extraction of features that are coherent within smaller spatial regions, thus enhancing the network's sensitivity to subtle variations in terrain. It is important to clarify that our assumption regarding horizontally contiguous patches is not absolute. While we posit that neighboring patches often share similar traversability properties, there will be instances where this local similarity does not hold; however, by leveraging this assumption as a general rule rather than a strict law, our method effectively encourages the learning of discriminative features. In practice, this principle serves as a heuristic guide for the network to learn meaningful representations that are beneficial for distinguishing traversability in complex outdoor environments, without being overly constrained by the occasional exceptions to the rule.

As a second difference from the original formulation, we apply SwAV's clustering-based procedure *to backbone features rather than image patches*. This is motivated by the observation that the pre-trained DeepLabV2 is already able to correctly classify pixels from both real and synthetic input images (see Fig. 3), thanks to the photorealism of our synthetic data. This finding also supports our decision of freezing the DeeplabV2 feature extractor. However, a visualization of DeeplabV2 features with t-SNE (Maaten 2014), in Fig. 4, shows that features extracted from synthetic and real features are markedly clustered in different regions of the projected space, emphasizing a distribution shift between the two sets of data: this may cause issues to the traversability estimation model, which is trained from scratch and might overfit the training distribution, negatively affecting its generalization to the other.

Given these premises, we apply SwAV using features extracted by the backbone as input and enforcing *local* similarity between features computed by the last convolutional layer of the traversability estimation model.

Formally, given a $F \times H \times W$ feature map, where F denotes the number of features and H and W the spatial dimensions, at each iteration of self-supervised training we extract a small $P \times Q$ region, where $P < Q$ to make the shape of the region approximately horizontal. Then, we sample two randomly-sized patches within that region, and apply the SwAV cluster-assignment procedure to features from each patch.

3.6 Supervised training on source domain

The traversability estimation branch of the model is trained in a supervised way on synthetic data. Since we formulated our estimation problem as a regression task, we optimize model parameters by minimizing the mean square error (MSE) loss between the estimated traversability vector $\tilde{\mathbf{t}}$ and the correct $\mathbf{t}$ for a given image:

$$\mathcal{L} = \sum_{(\mathbf{I},\mathbf{t})} \sum_{j=1}^{k} \left(t_j - \tilde{t}_j\right)^2, \tag{4}$$

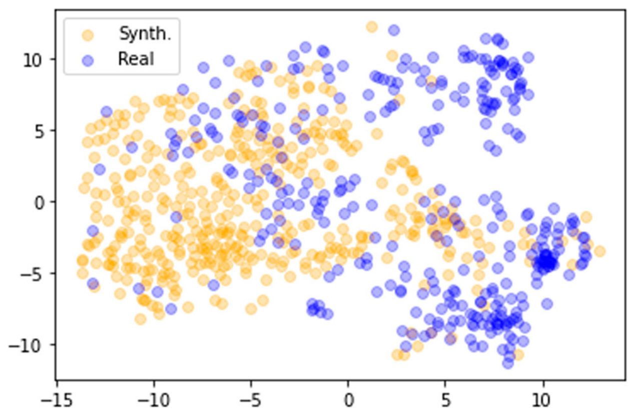


Fig. 3 Distributions of DeeplabV2 features on real and synthetic images, projected through t-SNE. The significant distribution shifts justifies the employment of a self-supervision approach on the features themselves, rather than on input images

where $(\mathbf{I}, \mathbf{t})$ pair consists of an input image and the corresponding target traversability values, and $\tilde{\mathbf{t}} = f(\mathbf{I})$.

We additionally extend the objective function by including the *safety-preserving loss term* introduced in Palazzo et al. (2020). Given t_i and $\tilde{t}_i$ to be, respectively, the ground-truth traversability value for a certain image subregion and the value predicted by the model, it has been observed that the sign of $\left(t_i - \tilde{t}_i\right)$ is significant from a safety perspective: if $t_i < \tilde{t}_i$, the model has estimated a certain path to be more traversable than it actually is, which is something that we want to discourage. Indeed, it is preferable to sacrifice accuracy and provide more conservative predictions, than making overly optimistic decisions that may lead to collisions or overturnings.

The total loss, including L_2 regularization, thus becomes:

$$\mathcal{L}_s = \sum_{(\mathbf{I},\mathbf{t})} \sum_{j=1}^{k} \left[\left(\tilde{t}_j - t_j\right)^2 + \alpha \max\left(0, \tilde{t}_j - t_j\right)^2 \right] + \rho \, \|\boldsymbol{\theta}\|_2^2 \tag{5}$$

where α weighs the importance between prediction accuracy and conservativeness, while ρ controls the strength of regularization on model parameters $\boldsymbol{\theta}$.

3.7 Unsupervised domain adaptation

Self-supervision on synthetic and real data helps to learn initial features for both annotated and unannotated data; however, this does not guarantee that feature activations for inputs from the two domains correspond. Hence, a model trained on synthetic images may not generalize on real images due to the persisting distribution shift.

For this reason, it is necessary to push the feature distributions together, so that the model behaves in a similar way regardless of the domain of the input images. As in Palazzo et al. (2020), we employ *gradient reversal layers* (GRL) (Ganin and Lempitsky 2015) to accomplish this.

Thus, alongside the traversability estimation branch, trained in a supervised way on the synthetic source domain, we introduce a separate *domain classification* branch. As shown in Fig. 1, this classifier receives intermediate features computed from both the source (synthetic) and target (real) domains and aims at discriminating whether an input image comes from one or the other. The key idea of the approach consists in pushing the model to learn intermediate features that *prevent* the domain classification branch from succeeding at its task. Intuitively, if the source and target feature distributions cannot be distinguished, the traversability estimation branch should work equally well on either domain. This mechanism is implemented by introducing a *gradient reversal layer* which changes the signs of gradients (appropriately scaled by a λ hyperparameter) of the domain classification loss. As a result, during training the domain classifier attempts to correctly distinguish the two domains, while features extracted before the GRL are driven to make such classification fail, thus becoming domain-agnostic.

More formally, we can define the training set for the domain classifier as:

$$\mathcal{D}_d = \{(\mathbf{I}, l_s)\}_{(\mathbf{I},\mathbf{t}) \in \mathcal{D}_s} \cup \{(\mathbf{I}, l_t)\}_{\mathbf{I} \in \mathcal{D}_t}, \tag{6}$$

where l_s and l_t are employed as *domain labels* — in practice, they are assigned the values 0 and 1.

Let $h(\mathbf{I})$ be the intermediate features extracted for input image $\mathbf{I}$, and $g(h(\mathbf{I}))$ the output of the domain classifier, which can be interpreted as the likelihood of the input belonging to one of the two domains. The domain classifier is trained with standard binary cross-entropy loss:

$$\mathcal{L}_d = - \sum_{(\mathbf{I},l) \in \mathcal{D}_d} \Big[l \log(g(h(\mathbf{I}))) + (1 - l) \log(1 - g(h(\mathbf{I}))) \Big], \tag{7}$$

with $l \in \{0, 1\}$ being the domain label associated to input $\mathbf{I}$.

In our model, intermediate features are extracted at the output of the last residual layer (see Table 1) and are spatially reduced from $1024 \times 8 \times 8$ to $1024 \times 2 \times 2$ through adaptive max pooling. GRL is inserted at this point, and is followed by the domain classifier, i.e., a multi-layer perceptron with ReLU activations and hidden layers of sizes 1024 and 256. The output of the domain classifier is a scalar value, constrained between 0 and 1 by a sigmoid activation.

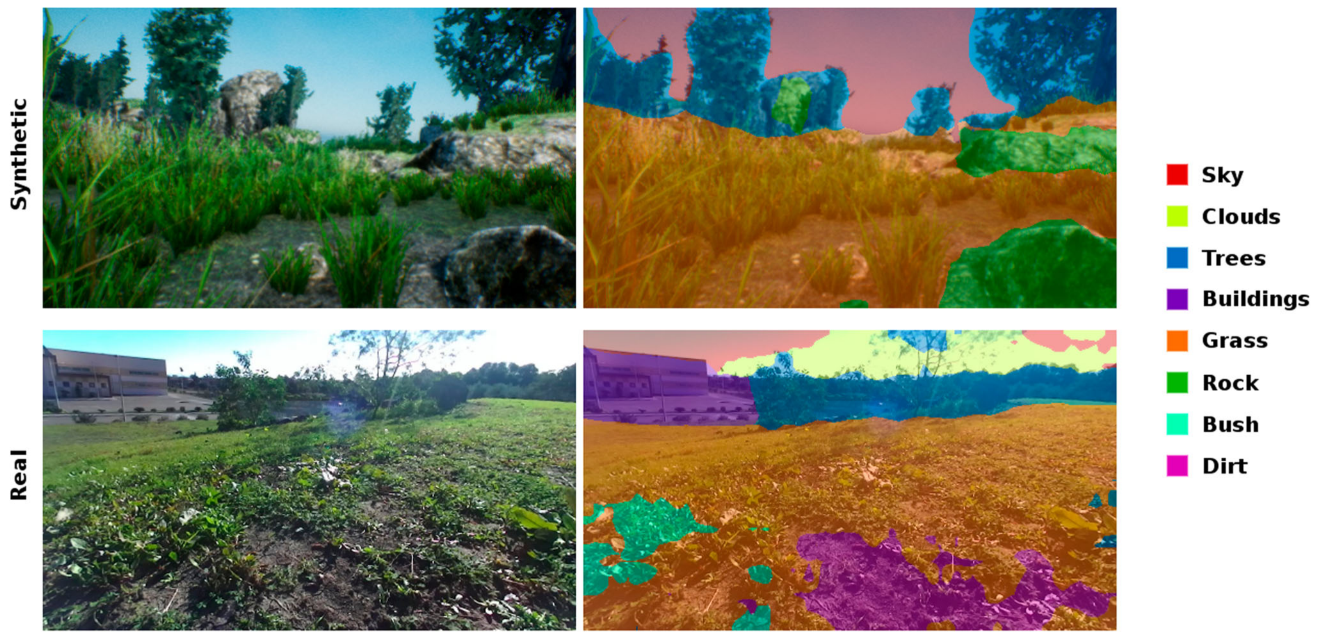


Fig. 4 Segmentation outputs of the employed DeepLabV2 backbone on synthetic and real image samples, demonstrating the high realism of synthetic images and the accuracy of the segmentation model

4 Navigation control

In order to properly assess the effectiveness of the proposed traversability prediction approach in a navigation setting, we hereby propose a strategy to translate the predicted traversability into control commands to the vehicle.

Intuitively, the most straightforward way to identify vehicle direction is to select the band with the highest predicted traversability score. However, if the selected band is next to low-score ones, the vehicle may bump into a close obstacle or move in between two poorly traversable areas. To address this limitation, we design a simple, yet effective, selection strategy, reported in Algorithm 1: given the vector $\tilde{\mathbf{t}} = \{t_1, \ldots, t_k\}$ of predicted traversability scores, the vehicle is directed towards band i_{opt} such that $t_{i_{opt}}$ is the highest score which satisfies the following boolean expression:

$$t_{i_{opt}} - t_{i_{opt}-1} > \delta \wedge t_{i_{opt}} - t_{i_{opt}+1} > \delta, \qquad (8)$$

where δ is a configurable parameter. This rule ensures that the chosen direction is traversable not only in the selected band, but also in the adjacent ones (up to a difference by δ), thus leaving room for trajectory adjustments. In our experiments, setting $\delta = 0.2$ provided a fair trade-off between a too optimistic and a too conservative band selection.

Inspired by Loquercio et al. (2018), the band position i_{opt} within the image frame provides the *direction* of the vehicle, whereas the traversability score t_{opt} modulates the linear forward velocity: the lower the traversability score, the slower the vehicle has to move, and vice versa. Note that

Algorithm 1: Band selection algorithm

```
Input: $\tilde{\mathbf{t}} = \{t_1, \ldots, t_k\}$, predicted traversability
Output: $t_{opt}$, score of the selected band
        $i_{opt}$, index of the selected band

$\tilde{\mathbf{i}} = \{i_1, \ldots, i_k\} \leftarrow$ indices of scores in $\tilde{\mathbf{t}}$ sorted in descending order
for $j = 1 : k$ do
    if $i_j == 1$ or $i_j == k$ then
        continue
    end
    if $t_{i_j} - t_{i_j+1} < \delta \wedge t_{i_j} - t_{i_j-1} < \delta$ then
        $i_{opt} \leftarrow i_j$
        $t_{opt} \leftarrow t_j$
    end
end
```

the boundary bands (i.e., the leftmost and the rightmost ones) are purposefully excluded in the band selection as they would lead to an inherently unsafe choice, since we lack traversability estimates outside of the camera field of view.

If it is not possible to select a proper band or if the identified traversability score t_{opt} is lower than a critical score $t_{crit} = 0.15$, we enable a *recovery mode*: the vehicle starts to slowly rotate in place at 0.1 rad/s in order to find alternative viable paths.

According to the above design principles, the linear velocity v and the angular velocity ω of the vehicle are computed through Eqs. 9 and 10.

$$v = \alpha(t_{opt} - t_{crit}) \qquad (9)$$

$$\omega = \beta(\lceil k/2 \rceil - i_{opt}) \tag{10}$$

The linear velocity is proportional to the difference between the critical score and the chosen band score. The α coefficient is set according to the maximum velocity of the robot. The angular velocity is, instead, proportional to the position of the band with respect to the middle band. The β term is set according to the maximum angular velocity of the platform. In our tests we set α and β, respectively, to 0.6 and 0.1, thus obtaining $v \in [0, 0.51]$ m/s and $\omega \in \{\pm 0.3, \pm 0.2, \pm 0.1, 0\}$ rad/s (since $k = 9$ in our case).

5 Traversability results

In this section, we first introduce the datasets employed in our work: the simulated environment in which traversability annotations are automatically generated, and the real dataset introduced in Palazzo et al. (2020) that we employ for unsupervised domain adaptation.

Then, we evaluate the accuracy of our traversability prediction approach on two different training setups. We first assess model performance with standard supervised training on annotated real images. This analysis allows us to establish an upper bound on the expected accuracy of the proposed approach. Then, we evaluate the quality of traversability predictions when training our model in a supervised way on synthetic images and unsupervisedly on real images. A thorough experimental protocol is followed to evaluate the impact of each component of the proposed architecture.

5.1 Real dataset acquisition

We hereby introduce the acquisition protocol of the real dataset employed in our experiments. Details can be found in Palazzo et al. (2020).

Video sequences are recorded by teleoperating an unmanned rubber-tracked ground robot employed for navigation in rough outdoor environments. The robot is equipped with a ZED stereo camera by Stereolabs acquiring 1280 × 720 RGB images of the terrain in front of the vehicle at 15 fps. Only images from the right stream of the stereo system are employed. The original data acquisition provides video sequences for *on-road* and *off-road* (terrain) scenarios. In this work, we focus on the off-road scenario, containing 419 selected images, since our simulated environment targets non-urban scenes.

Data annotation on this dataset was performed by a human operator. However, the approach presented in this paper does not employ manual annotations for training: we only use them to carry out performance analysis and to run supervised experiments on real images.

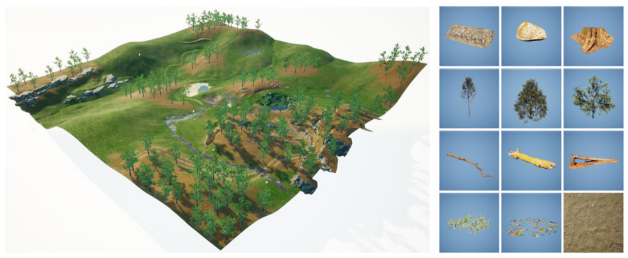


Fig. 5 **Left**: An overview of the map used to collect the synthetic dataset. The map features a grass-covered meadow, some rocky areas and some denser forest regions. It also includes a small lake and river. **Right**: Some sample assets used to populate the synthetic scene. The top three rows contain medium to large assets used as obstacles, while the last row contains ground assets (grass and leaves) and a terrain texture sample

5.2 Synthetic data acquisition

The synthetic dataset is generated using the MIDGARD simulator (Vecchio et al. 2022), which produces photorealistic images and provides the tools for automatic data annotation. The dataset was collected in a custom version of MIDGARD's native meadow scene, consisting of a large map (3, 600 m^2) replicating the features of the real-world dataset. The synthetic dataset was collected in several hand-guided navigation sessions: in each session we introduced some variability in terrain deformation and vegetation distribution. An overview of the entire map and sample elements (obstacles and terrain types) included in the scene are provided in Fig. 5.

We acquire a total of 2271 RGB frames, each with engine-generated traversability annotation, by simulating rays propagated from the robot and detecting collisions to objects in the scene, as exemplified in Figs. 6 and 7.

In detail, the annotation process is carried out in three stages:

1. A set of non-traversable object types is defined (e.g., rocks, trees, branches) and automatically marked with a *NonTraversable* flag, which can be imagined as an invisible overlay over the selected scene elements.
2. A set of rays is projected from the robot perspective: if a ray impacts an object/surface marked as *NonTraversable*, the corresponding portion of the camera view is annotated accordingly.
3. If a ray impacts a ground patch with an average slope angle greater than a threshold it is also annotated as non-traversable, regardless of the *NonTraversable* flag. The threshold is based on the robot's climbing capabilities: in our experiments, we set it to 25°.

To simplify the acquisition and annotation process, we render 512 × 512 frames with a camera field of view of 90°,

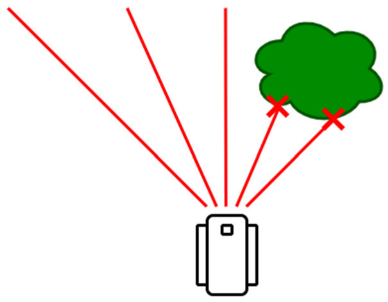


Fig. 6 A simplified aerial-view graphics of the line tracing to detect non-traversable regions in the simulated frame

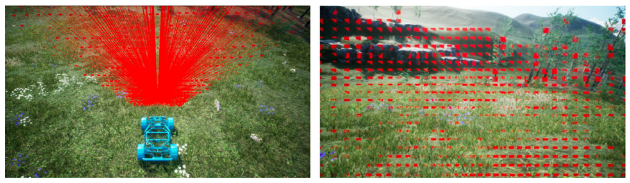


Fig. 7 Visualization of trace-rays cast from the robot perspective for automatic annotation of synthetic images: aerial view of the robot agent (left) and robot camera perspective (right)

then both the frames and the annotations are cropped to 16:9 aspect ratio. To cover the entire 90° × 90° field of view of the captured frames, we split the view into 9 sectors, each covering a range of 10°, both horizontally and vertically, and cast 3 trace-lines for each sector, resulting in an approximation gap between traces of 3.33°, for a total of 729 traces. A sector is considered traversable if all the traces in that sector intersect a traversable surface; otherwise, that sector is considered non-traversable. The 27 vertical traces cast for each horizontal sector are used to detect the maximum traversable horizon height (Fig. 8).

In order to prevent the presence of near-duplicate frames, we ensured that any two consecutive frames in the dataset exhibited a minimum linear or angular displacement, following the same criteria as in Palazzo et al. (2020). Overall, it took about 1.5 h of hand-guided navigation in the simulator to collect the synthetic dataset of 2271 computer-annotated frames.

5.3 Model training and evaluation procedure

Input images are pre-processed by resizing the shortest side to 128 pixels (keeping aspect ratio) and standardizing each color channel. After feature extraction by the DeepLabV2 backbone, the resulting feature maps have the shortest spatial dimension of size 17: input images from the real dataset produce feature maps of size 17 × 29, while feature maps from synthetic images are 17 × 17. When performing self-supervision, we modify SwAV to predict the same cluster assignments to feature patches of spatial size between 2 × 2 and 4 × 4 (with random aspect ratio) extracted from a 4 × 6 area.

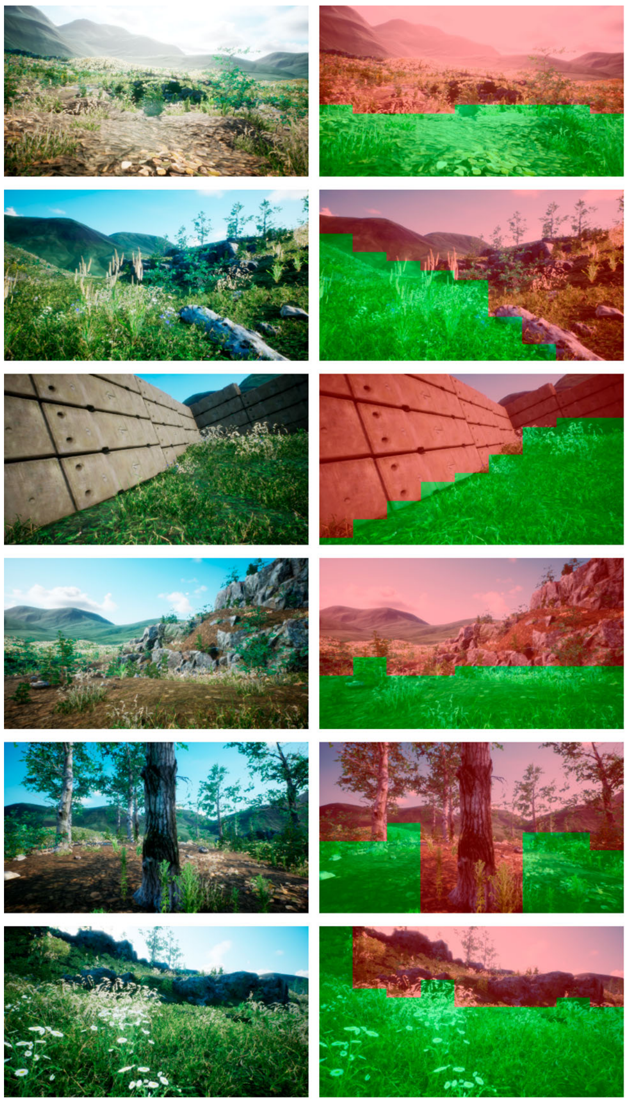


Fig. 8 Examples of the automatic annotation process on the synthetic dataset

The training procedure for self-supervision employs a mini-batch SGD optimizer (batch size: 16), with cosine learning rate annealing from 0.6 to 0.0006, for 400 training epochs.

As for traversability estimation, we train our model with the Adam optimizer for 5000 epochs, using a linear learning rate schedule from 10^{-4} to 10^{-8}. Following Palazzo et al. (2020), we set the α hyperparameter for the safety-preserving loss term to 1.5. Weight decay factor ρ is $5 \cdot 10^{-4}$, and the λ hyperparameter that scales gradient reversal for domain adaptation is set to 0.1. This training setting is applied to both real and synthetic datasets.

Since the real dataset consists of a sequential stream of frames captured while operating the robot, we define the training and test splits by using the first 80% of the sequence for training and the rest for test, thus minimizing the risk of near-duplicates in the two sets. For consistency, we also split the synthetic dataset in the same way.

The model accuracy is computed on the real-dataset test set, in terms of mean absolute error (MAE), estimated by averaging the absolute errors of traversability scores predicted in each image, and then averaging the results over all test images. In order to account for random model initialization and to better assess differences in performance, we report the mean and standard deviation of MAE over 10 runs of the training and evaluation protocol.

Our model is implemented using the PyTorch library. We use an NVIDIA Titan X (Pascal) GPU for training, and an on-board NVIDIA Jetson TX1 (Maxwell) GPU for inference on the robot, which is able to process 15 frames per second.

5.4 Performance analysis in the supervised setting

We hereby present the results achieved by our model (and variants thereof) when it is trained in a supervised way on real images. This setup represents an ideal case, with manual annotations available on dataset created from real acquisitions. In the next section, we will show how our proposed approach, with unsupervised domain adaptation on real images, compares to these results.

Table 2 shows the average MAE for several supervised training configurations of our approach, namely, when synthetic data are used alongside the real ones (*Synth.*) and when self-supervised initialization, as described in Sect. 3.5, is performed (*Self-sup.*). As a reference for comparison, we report the results from the traversability estimation model in Palazzo et al. (2020). For a thorough comparison, since in Palazzo et al. (2020) the feature extraction backbone is also fine-tuned during training, we first carry out a similar experiment by disabling parameter freezing. It is worth to note that, when not freezing the backbone, our approach obtains comparable results with Palazzo et al. (2020); a slight difference in accuracy is due to architectural changes and random model initialization.

The results obtained when introducing the proposed training enhancements show that our model is able to outperform the baseline by a statistically significant margin. In this setup, with the availability of real-image ground-truth annotations during training, the impact of employing synthetic images and of performing self-supervision is limited. This can be expected, since the error signal provided by real images is probably the most important factor that drives learning. We can also notice that backbone freezing has a positive impact on results, reducing model complexity and leading our approach to better generalize on test data. Therefore, we enable backbone freezing in all the following experiments on unsupervised domain adaptation.

Table 2 Performance evaluation, in terms of average MAE on the traversability scores, in the supervised setup with manual annotations on real images

Model	Freez	Synth	Self-sup	MAE ($\times 10^{-3}$)
Palazzo et al. (2020)	–	–	–	112 ± 8 **
Proposed model	–	–	–	114 ± 6 **
	✓	–	–	95 ± 7 **
	✓	✓	–	98 ± 6 *
	✓	–	✓	96 ± 5 *
	✓	✓	✓	**89 ± 7**

The * and ** symbols indicate statistical significance of t-test with *p*-value less than 0.05 and 0.01, respectively

Table 3 Performance evaluation, in terms of average MAE on the traversability scores, in the domain adaptation setup, where no manual annotations on real images are available

Model	Self-sup	Adapt	MAE ($\times 10^{-3}$)
Palazzo et al. (2020)	–	–	170 ± 10 **
Proposed model	–	–	164 ± 11 **
	✓	–	131 ± 6 **
	–	✓	124 ± 8 **
	✓	✓	**104 ± 8**

The ** symbol indicates statistical significance of t-test with *p*-value less 0.01

5.5 Performance analysis in the unsupervised domain-adaptation setting

In this setting, we evaluate the accuracy of our model when no annotations on the real dataset are available. Table 3 shows the average MAE when the model is supervisedly trained on synthetic images only and demonstrates how performance varies when we gradually integrate self-supervision ("Self-sup.") and unsupervised domain adaptation ("Adapt."). Note that when no self-supervision or domain adaptation is employed (as in the case of Palazzo et al. (2020), which we include in the comparison as a baseline), the model is simply trained on synthetic images and tested on real images.

Results show that the model successfully learns traversability features from synthetic images and, unsupervisedly, from real images. It is interesting to note that results without any form of adaptation (i.e., when only synthetic images are used) are already relatively accurate, demonstrating the realism of the proposed simulation framework. Then, the integration of self-supervision and unsupervised domain adaptation independently improve the accuracy of the model. The full variant of our approach further reduces the traversability estimation error, achieving an average MAE of 0.104, which is very close to the value of 0.089 obtained when real images are supervisedly used at training time.

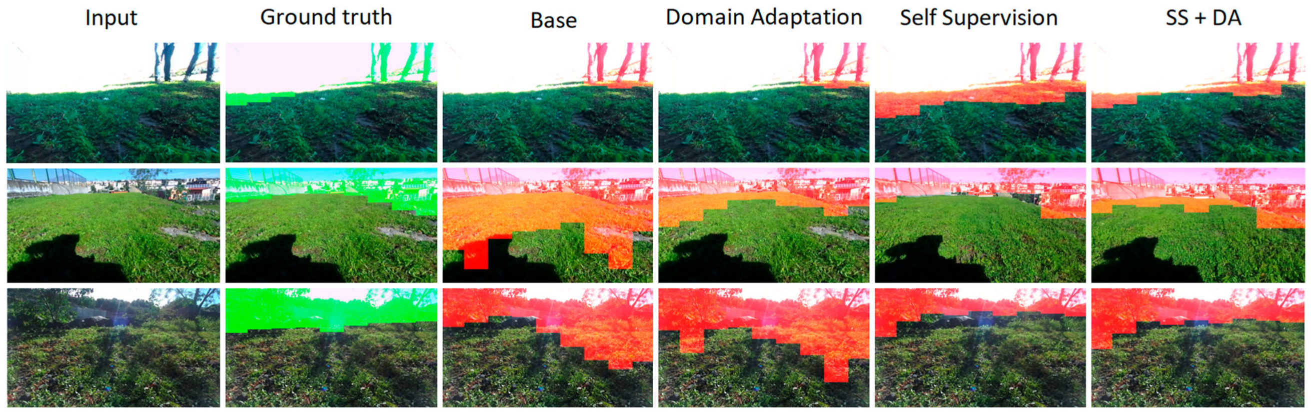


Fig. 9 A comparison between the output of the methods in Table 3 and the ground-truth for real images. From left to right we have: (1) input, (2) ground truth, (3) base (training on synthetic only), (4) domain adaptation on unlabeled real data, (5) self-supervised pre-training, (6) self-supervision + domain adaptation

The outputs of different setups of our model applied to three sample images are compared in Fig. 9, qualitatively showing the improved traversability prediction of our final model. When training on synthetic data and directly testing on real data (*"Base"* column), model predictions exhibit large estimation errors; the integration of domain adaptation alone does improve results on simple terrain (e.g., second row). Self-supervision has a visibly positive impact on the results, though it appears to cause overly optimistic predictions in some regions. Integrating self-supervision and domain adaptation further improves predictions, keeping them close to ground-truth annotations but making them more conservative and safe.

6 Navigation results

In this section we evaluate the performance of the proposed traversability-driven navigation method in both on-simulation and on-field settings.

6.1 On-simulation navigation results

Simulated experiments are designed to evaluate the traversability-driven navigation approach in several challenging scenarios and to compare it to the state of the art. The experiments are carried out in the MIDGARD simulation environment for a total control over scene features and geometry.

We first evaluate qualitatively our navigation approach under three challenging scenarios where traversability estimation may lead to erroneous navigation outcomes:

- *Steep slope occluding horizon.* A steep upslope or downslope may limit the estimated traversability value due to the sky covering a large part of the view, causing the agent to stop. Figure 10 presents an extreme example where the horizon is fully occluded. We report the predicted traversability maps in three key points: (1) upslope, before reaching the top of the hill; (2) at the top of the hill; (3) downslope, while going down the hill. In all of these cases, the estimated traversability never falls below the critical threshold of $t_{crit} = 0.15$, at which the control algorithm stops the agent. Note that this behavior also depends on the camera setup, which in our experiments is tilted downward with an angle of 5°, replicating the setup of the real-world robot.
- *Mid-air obstacles*, such as horizontal branches in front of the vehicle camera, may limit visibility and, consequently, the traversability estimation. The example in Fig. 11 shows how the proposed approach suitably lowers the traversability scores as the agent gets closer to the overhead branch up to a point where they reach the critical threshold and the vehicle stops. Suitable camera positioning, at the front top of the robot, is necessary to have occlusions appearing in the middle/lower portion of the frame, in order to be suitably marked as non-traversable.
- *Narrow path.* Narrow paths represent another critical navigation scenario, where small mistakes may cause serious damage to the robot. To validate our navigation algorithm in such a setting, we crafted a scene consisting of a small traversable path between two rocky walls, as shown in Fig. 12. In this test, as exemplified in the sequence of frames captured from the virtual camera, the robot is able to effectively travel the path, without exhibiting any oscillatory behavior that might divert it from the optimal course.

We then perform a quantitative assessment of navigation performance, in terms of average traveled distance and time, and compare it to the state of the art. More specifically, we compare our approach with an end-to-end reinforcement

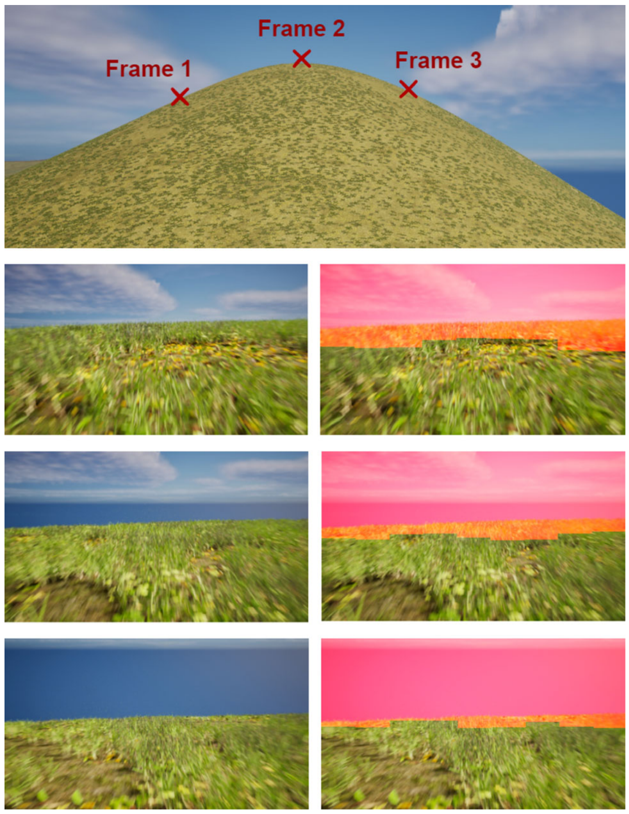


Fig. 10 Traversability prediction in steep slope setting. The top image shows a side-view of the hill with the acquisition locations marked, shown below in order with the corresponding traversability predictions. In all frames, the traversability range is always above the $t_{crit} = 0.15$ threshold

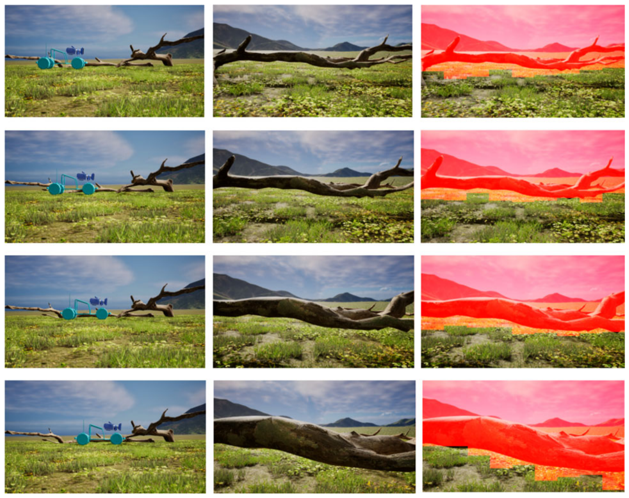


Fig. 11 Examples of traversability estimated when facing a horizontal obstacle in mid-air. Each row of images shows a side view of the scene, the agent's camera view and predicted traversability

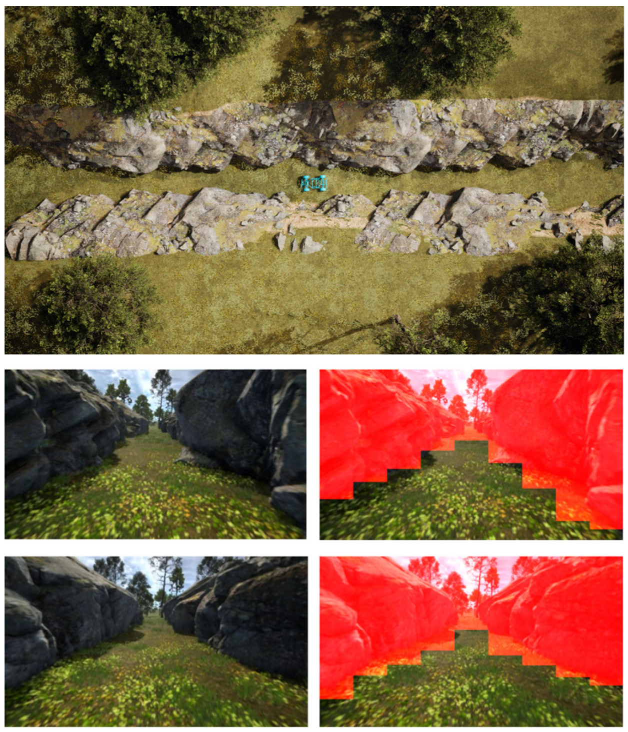


Fig. 12 Top: aerial view of the scene in our narrow path navigation experiment. Bottom: two frames, with corresponding traversability prediction, at two different points of the path

learning navigation method (Xie et al. 2017), which employs a D3QN agent to perform obstacle avoidance from RGB and depth inputs. For a fair comparison to our method (that relies only on RGB), we leave out intermediate depth estimation from Xie et al. (2017) and work on RGB only. The reward function used to train the agent is the same as described in Xie et al. (2017). At each step, the agent receives a reward computed as $r = vT \cos \omega$, where v and ω are the local linear and angular velocity, and T is the time between each step. Additionally, the agent receives a negative reward of -10 when a collision is detected.

In addition, we also test two variants of our navigation approach. Our first variant is designed to investigate the possible presence of a correlation bias between distance and position of elements in the 2D image, which might tend to mark areas in the bottom part of the camera as traversable, possibly erroneously. To assess the impact of this bias, we apply a depth-based post-prediction criterion over the maximum estimated traversability: points in the scene which are further than a certain depth threshold are marked as non-traversable, regardless of the predicted traversability score (we refer to this variant to as *depth-constrained*). Under this constraint, overconfident predictions about far elements in the scene are prevented. In our experiments, we set the depth threshold to 15 ms, which is within the range supported by the real ZED camera.

Our second variant is designed to assess the effect of the proposed navigation approach, compared to a simpler alternative which applies a manually-set convolutional kernel to identify the ideal band. In detail, each traversability score s_i is updated as $s_i \leftarrow s_i - \text{abs}(c)$, where c is obtained by applying a $[-0.5, 1, -0.5]$ convolutional kernel at vector location i. Intuitively, the effect of this operation is to prefer locations that a) have a large traversability score to start with, and b) have similar scores in the neighbor bands. The band with the largest updated score is then selected as the direction towards which the robot should move. Linear and angular velocities are set as per Eq. 9 (with $t_{crit} = 0$) and Eq. 10.

Results, in terms of average traveled distance and navigation time before a collision over 200 navigation episodes, are given in Table 4: our method significantly outperforms the end-to-end learning approach in Xie et al. (2017) in both metrics. Interestingly, including the maximum depth constraint[2] seems to yield worse performance: this may be due to the reduced traversable horizon forced by the distance threshold, which limits navigation options. As for our experiment with the “naïve” navigation rule, results show that this approach, as can be expected, yields low navigation performance. From a manual inspection of the robot’s behavior in this setting, choosing the target direction “greedily” often leads the robot to either be surrounded by non-traversable regions or oscillate between nearby bands, due to the lack of a criterion for ensuring consistency of traversability scores across nearby bands. It is interesting to note that, quantitatively, this causes the robot to navigate for shorter distances before incurring into obstacles, while counter-intuitively increasing the duration of the simulation, since frequently switching directions leads to spending more time performing in-place rotations than linear displacements.

Our findings thus suggest that assessing accurately the height of the traversable region in 2D projected images adequately supports complex navigation tasks, while being more interpretable than end-to-end navigation approaches.

6.2 On-field navigation results

On-field experiments have been carried out in a previously unseen real-world scenario to evaluate the performance of the traversability prediction model, especially in terms of sim-to-real transfer. The test environment features small to large rocks, tree branches and trunks, and high vegetation, thus being far more challenging compared to the real-world dataset used in training. The tracked vehicle and hardware setup introduced in 5.1 and adopted for the dataset acquisition (Palazzo et al. 2020) are used also for running the on-field test campaign.

[2] It should be noted that, in our experiment, depth measurements were accurately simulated; in a real application, noisy depth values might further deteriorate performance.

Table 4 Performance evaluation of navigation methods, in terms of average navigated distance and average navigation duration, before a collision is detected

Method	Avg. distance	Avg. time
Xie et al. (2017)	43.32 m	198.51 s
Palazzo et al. (2020)	44.13 m	192.48 s
Our approach (naïve rule)	31.96 m **	206.88 s **
Our approach (depth-constrained)	47.54 m **	190.17 s **
Our approach	**49.29 m** **	**232.67 s** **

The ** symbol indicates statistical significance of t-test with p-value less 0.01

The proposed approach allows the vehicle to smoothly navigate without experiencing any oscillatory behavior in the angular velocity, despite its discretization, thanks to the stable outcome of the band selection algorithm over time. More in detail, we observe that the selected band position is stable or smoothly transitioning to adjacent bands in consecutive video frames during navigation. A video showing some sessions of autonomous navigation during the on-field tests performed in the considered environment can be found in the supplementary material.

On-field experiments highlight the model’s ability to properly identify traversable areas even in scenes differing from the training dataset, thus demonstrating the generalization capability and robustness of the model. Some examples are reported in Fig. 13, showing convincing predictions of the traversable horizon for the observed scenarios.

Performance decreases when dealing with very close vegetation, resulting in occasional misclassification of non-traversable obstacles such as rocks, thus posing a potential risk to the vehicle. Some failure cases are reported in Fig. 14. We argue that this behavior is caused by the limited number of similar examples in both real and synthetic datasets: as a consequence, obstacles occluded by vegetation can be mistakenly perceived as traversable. Comparing the results in Fig. 13 to those in Fig. 14, it can be observed how the model is still able to recognize the traversable path as long as it has enough contextual awareness and the visual appearance of obstacles is not overly occluded by vegetation.

We have also compared our method with the reinforcement learning approach in Xie et al. (2017). While the latter provides appreciable results, thanks to retraining performed in simulation, it also occasionally resulted in collisions with rocks or walls. This kind of navigation mistakes could be ascribed to the inability of the model to deal with the change of lighting conditions or to a potential delay in the decision making of the agent, which results into trajectories colliding with obstacles. A video showing some video samples of scenarios where (Xie et al. 2017) fails, whereas our approach

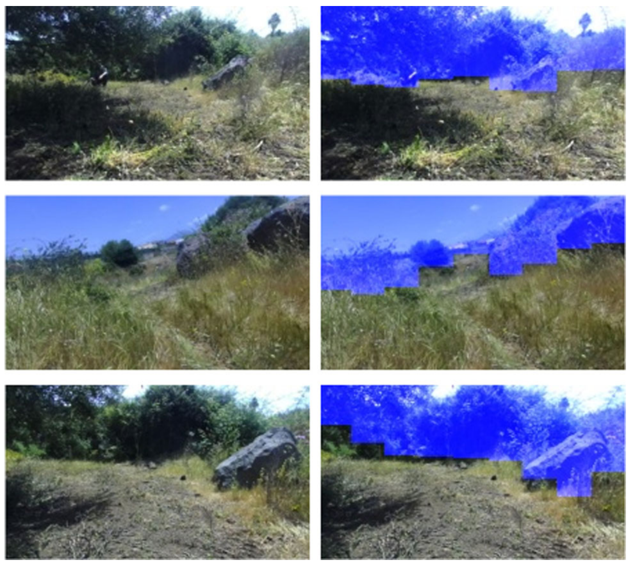


Fig. 13 Successful traversability inference examples from the on-field tests: real images (left), related non-traversable area prediction in blue (right)

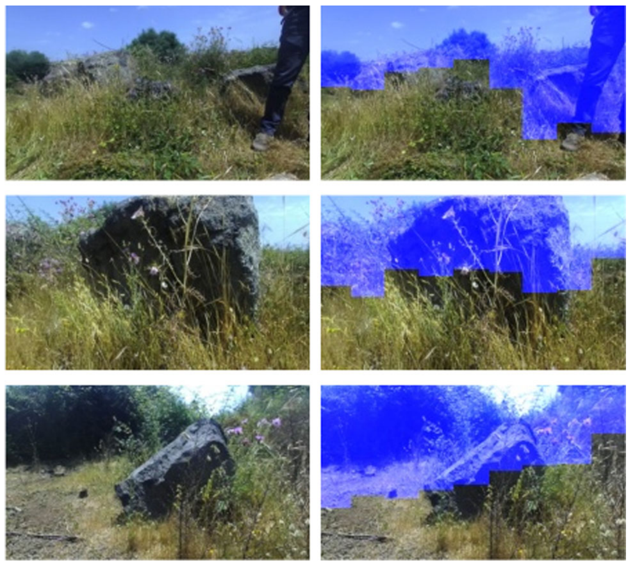


Fig. 14 Failure examples from the on-field tests: real images (left), related non-traversable area prediction in blue (right)

succeeds, can be found in the supplementary material. However, it is worth recalling that it is hard to properly identify potential sources of failures, since in Xie et al. (2017) we do not have a predicted traversability outcome, but the navigation control action. Thus, in the provided video we show the on-board camera view during navigation up to the moment before the robot bumps into an obstacle.

7 Conclusions

In this paper, we introduced a novel method for traversability prediction based on self-supervision and domain adaptation. To enhance the capabilities of the model we created a large, fully annotated, synthetic dataset in a simulated environment, relieving from the burden of manually labeling a real dataset. The proposed approach outperforms previous methods for traversability estimation (Palazzo et al. 2020), while relaxing the need for supervision on real data.

On-field tests, performed in an unseen outdoor and unstructured scenario, confirm the effectiveness of the proposed method to accurately estimate traversability, thus enabling the vehicle to autonomously navigate in the target environment.

Future developments include enhancing model results when dealing with new unseen objects, such as occlusions between vegetation and non-traversable obstacles in rocks. A possible solution may include to integrate traversability predictions with confidence scores by a segmentation model, such as DeepLabV2, in order to detect critical regions (where prediction confidence is expected to be low). Uncertainty regions can be then treated as non-traversable when the prediction confidence falls below a threshold.

A related research direction concerns the limitations of the proposed approach for domain adaptation. Indeed, while numerical experiments and on-field tests demonstrated that our method effectively transfers knowledge from synthetic scenarios to real (and even unseen) environments, our preliminary results show that the model fails in presence of extreme domain changes, e.g., when performing supervised training on a synthetic "volcanic" scene and domain adaptation on a grassy environment, with previously-unseen obstacles such as trees).

In terms of the formulation of the training objective, we also intend to address the lack of distinction between traversability errors performed in the lower part of the image (i.e., closer to the robot) than in the higher part. A possible solution would be to weigh the training samples based on the ground-truth traversability scores, giving more importance to lower ones; to this aim, we should beforehand ensure a uniform distribution over traversability values collected during the hand-guided navigation sessions.

Finally, we intend to investigate sensor fusion approaches to integrate RGB data with depth cameras and/or LIDAR scans, to further improve the accuracy of the model in the presence of ambiguous visual features, as well as to explore model architectures and learning methodologies for point cloud inputs—e.g., 3D convolutions (Huang and You 2016), graph neural networks (Shi and Rajkumar 2020), transformers (Guo et al. 2021). The proposed approach could be further extended by integrating it with learning-based path planners, such as an end-to-end reinforcement learning–based navigation agent.

Supplementary information

A video showing some sessions of autonomous navigation during the on-field tests performed in the considered environment can be found in the supplementary material.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10514-024-10158-4.

Author Contributions G.V., S.P., and D.C.G. developed the method implementation and carried out the experiments, both on the neural network model and on the robotic platform. All authors wrote and reviewed the manuscript.

Funding Open access funding provided by Università degli Studi di Catania within the CRUI-CARE Agreement. We acknowledge financial support from PNRR MUR project PE0000013-FAIR. Dario Guastella acknowledges support by the project PON R&I REACT-EU.

Code availability Training code and synthetic dataset will be publicly released at: https://github.com/perceivelab/traversability-synth.

Declarations

Conflict of interest The authors have no conflict of interest to declare that are relevant to the content of this article.

Ethics approval Not applicable.

References

Bellitto, G., Salanitri, F.P., Palazzo, S., Rundo, F., Giordano, D., & Spampinato, C. (2020). Video saliency detection with domain adaption using hierarchical gradient reversal layers. arXiv preprint arXiv:2010.01220

Borges, P., Peynot, T., Liang, S., Arain, B., Wildie, M., Minareci, M., Lichman, S., Samvedi, G., Sa, I., Hudson, N., et al. (2022). A survey on terrain traversability analysis for autonomous ground vehicles: Methods, sensors, and challenges. *Field Robotics, 2*(1), 1567–1627.

Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., & Smola, A. J. (2006). Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics, 22*(14), 49–57.

Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., & Krishnan, D. (2017). Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3722–3731).

Caesar, H., Uijlings, J. R. R., & Ferrari, V. (2016). COCO-Stuff: Thing and stuff classes in context. CoRR arXiv:1612.03716

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., & Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in neural information processing systems*.

Chavez-Garcia, R. O., Guzzi, J., Gambardella, L. M., & Giusti, A. (2018). Learning ground traversability from simulations. *IEEE Robotics and Automation Letters, 3*(3), 1695–1702.

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597–1607). PMLR.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A.L. (2016). DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. arXiv preprint arXiv:1606.00915

Delmerico, J., Mintchev, S., Giusti, A., Gromov, B., Melo, K., Horvat, T., Cadena, C., Hutter, M., Ijspeert, A., Floreano, D., Gambardella, L. M., Siegwart, R., & Scaramuzza, D. (2019). The current state and future outlook of rescue robotics. *Journal of Field Robotics, 36*(7), 1171–1191.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An open urban driving simulator. In *Conference on robot learning* (pp. 1–16). PMLR.

Dosovitskiy, A., Fischer, P., Springenberg, J. T., Riedmiller, M., & Brox, T. (2015). Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 38*(9), 1734–1747.

Gaidon, A., Wang, Q., Cabon, Y., & Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4340–4349).

Ganin, Y., & Lempitsky, V. S. (2015). Unsupervised domain adaptation by backpropagation. In *International conference on machine learning* (pp. 1180–1189).

Ghifary, M., Kleijn, W. B., & Zhang, M. (2014). Domain adaptive neural networks for object recognition. In *Pacific rim international conference on artificial intelligence* (pp. 898–904). Springer.

Giusti, A., Guzzi, J., Cireşan, D. C., He, F.-L., Rodríguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Caro, G. D., Scaramuzza, D., & Gambardella, L. M. (2016). A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters, 1*(2), 661–667.

Gonzalez, R., & Iagnemma, K. (2018). DeepTerramechanics: Terrain classification and slip estimation for ground robots via deep learning. arXiv:1806.07379

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks. arXiv preprint arXiv:1406.2661

Guastella, D. C., & Muscato, G. (2021). Learning-based methods of perception and navigation for ground vehicles in unstructured environments: A review. *Sensors, 21*(1), 73.

Guo, M.-H., Cai, J.-X., Liu, Z.-N., Mu, T.-J., Martin, R. R., & Hu, S.-M. (2021). PCT: Point cloud transformer. *Computational Visual Media, 7*(2), 187–199.

Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society* *conference on computer vision and pattern recognition (CVPR'06)* (vol. 2, pp. 1735–1742). IEEE.

Hadsell, R., Erkan, A., Sermanet, P., Scoffier, M., Muller, U., & LeCun, Yann (2008). Deep belief net learning in a long-range vision system for autonomous off-road driving. In *2008 IEEE/RSJ international conference on intelligent robots and systems* (pp. 628–633).

Haltakov, V., Unger, C., & Ilic, S. (2013). Framework for generation of synthetic ground truth data for driver assistance applications. In *German conference on pattern recognition* (pp. 323–332). Springer.

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9729–9738).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hewitt, R. A., Ellery, A., & Ruiter, A. (2017). Training a terrain traversability classifier for a planetary rover through simulation. *International Journal of Advanced Robotic Systems, 14*(5), 1729881417735401.

Holder, C. J., & Breckon, T. P. (2018). Learning to drive: Using visual odometry to bootstrap deep learning for off-road path prediction. In *2018 IEEE intelligent vehicles symposium (IV)* (pp 2104–2110).

Howard, A., Turmon, M., Matthies, L., Tang, B., Angelova, A., & Mjolsness, E. (2006). Towards learned traversability for robot navigation: From underfoot to the far field. *Journal of Field Robotics, 23*(11–12), 1005–1017.

Huang, J., & You, S. (2016). Point cloud labeling using 3D Convolutional Neural Network. In: *2016 23rd international conference on pattern recognition (ICPR)* (pp. 2670–2675). IEEE.

Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., Savva, M., Chernova, S., & Batra, D. (2020). Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters, 5*(4), 6670–6677.

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., & Farhadi, A. (2017). AI2-THOR: An Interactive 3D Environment for Visual AI. arXiv preprint arXiv:1712.05474

Liu, M.-Y., & Tuzel, O. (2016). Coupled generative adversarial networks. arXiv preprint arXiv:1606.07536

Long, M., Cao, Y., Wang, J., & Jordan, M. (2015). Learning transferable features with deep adaptation networks. In *International conference on machine learning* (pp. 97–105). PMLR.

Long, M., Zhu, H., Wang, J., & Jordan, M. I. (2016). Unsupervised domain adaptation with residual transfer networks. arXiv preprint arXiv:1602.04433

Loquercio, A., Maqueda, A. I., del-Blanco, C. R., & Scaramuzza, D. (2018). DroNet: Learning to fly by driving. *IEEE Robotics and Automation Letters, 3*(2), 1088–1095.

Maaten, L. (2014). Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research, 15*(1), 3221–3245.

Maturana, D., Chou, P.-W., Uenoyama, M., & Scherer, S. (2018). Real-time semantic mapping for autonomous off-road navigation. In M. Hutter & R. Siegwart (Eds.), *Field and service robotics* (pp. 335–350). Springer.

Misra, I., & Maaten, L.v.d. (2020). Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6707–6717).

Müller, M. G., Durner, M., Gawel, A., Stürzl, W., Triebel, R., & Siegwart, R. (2021). A Photorealistic Terrain Simulation Pipeline for Unstructured Outdoor Environments. In: *2021 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 9765–9772). IEEE.

Nguyen, A., Nguyen, N., Tran, K., Tjiputra, E., & Tran, Q.D. (2020). Autonomous navigation in complex environments with deep multimodal fusion network. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 5824–5830).

Palazzo, S., Guastella, D. C., Cantelli, L., Spadaro, P., Rundo, F., Muscato, G., Giordano, D., & Spampinato, C. (2020). Domain adaptation for outdoor robot traversability estimation from RGB data with safety-preserving loss. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 10014–10021).

Pan, Y., Cheng, C.-A., Saigol, K., Lee, K., Yan, X., Theodorou, E. A., & Boots, B. (2020). Imitation learning for agile autonomous driving. *The International Journal of Robotics Research, 39*(2–3), 286–302.

Pflueger, M., Agha, A., & Sukhatme, G. S. (2019). Rover-IRL: Inverse reinforcement learning with soft value iteration networks for planetary rover path planning. *IEEE Robotics and Automation Letters, 4*(2), 1387–1394.

Richter, S. R., Hayder, Z., & Koltun, V. (2017). Playing for benchmarks. In *Proceedings of the IEEE international conference on computer vision* (pp. 2213–2222).

Richter, S. R., Vineet, V., Roth, S., & Koltun, V. (2016). Playing for data: Ground truth from computer games. In *European conference on computer vision* (pp. 102–118). Springer.

Rothrock, B., Kennedy, R., Cunningham, C., Papon, J., Heverly, M., & Ono, M. (2016). SPOC: Deep learning-based terrain classification for mars rover missions. In *AIAA SPACE 2016*.

Saito, K., Ushiku, Y., & Harada, T. (2017). Asymmetric tri-training for unsupervised domain adaptation. In *International conference on machine learning* (pp. 2988–2997). PMLR.

Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., & Malik, J., & Parikh, D. (2019). Habitat: A platform for embodied AI research. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9339–9347).

Shah, S., Dey, D., Lovett, C., & Kapoor, A. (2018). AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics* (pp. 621–635). Springer.

Shi, W., & Rajkumar, R. (2020). Point-GNN: Graph neural network for 3D object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1711–1719).

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., & Webb, R. (2017). Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2107–2116).

Skinner, J., Garg, S., Sünderhauf, N., Corke, P., Upcroft, B., & Milford, M. (2016). High-fidelity simulation for evaluating robotic vision performance. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2737–2744). IEEE.

Smolyanskiy, N., Kamenev, A., Smith, J., & Birchfield, S. (2017). Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 4241–4247).

Song, Y., Naji, S., Kaufmann, E., Loquercio, A., & Scaramuzza, D. (2020). Flightmare: A flexible quadrotor simulator. arXiv preprint arXiv:2009.00563

Valada, A., Oliveira, G., Brox, T, & Burgard, W. (2016). Towards robust semantic segmentation using deep fusion. In *Workshop on limits and potentials of deep learning in robotics at robotics: Science and systems (RSS)*.

Vecchio, G., Palazzo, S., Guastella, D. C., Carlucho, I., Albrecht, S. V., Muscato, G., & Spampinato, C. (2022). MIDGARD: A simulation platform for autonomous navigation in unstructured environments. arXiv preprint arXiv:2205.08389

Wang, M., & Deng, W. (2018). Deep visual domain adaptation: A survey. *Neurocomputing, 312*, 135–153.

Wu, Z., Xiong, Y., Yu, S., & Lin, D. (2018). Unsupervised feature learning via non-parametric instance-level discrimination. arXiv preprint arXiv:1805.01978

Xia, F., Zamir, A. R., He, Z., Sax, A., Malik, J., & Savarese, S. (2018). Gibson Env: Real-world perception for embodied agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 9068–9079).

Xie, L., Wang, S., Markham, A., & Trigoni, N. (2017). Towards monocular vision based obstacle avoidance through deep reinforcement learning. arXiv preprint arXiv:1706.09829

Yan, H., Ding, Y., Li, P., Wang, Q., Xu, Y., & Zuo, W. (2017). Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2272–2281).

Yandun Narváez, F., Gregorio, E., Escolà, A., Rosell-Polo, J. R., Torres-Torriti, M., & Auat Cheein, F. (2018). Terrain classification using ToF sensors for the enhancement of agricultural machinery traversability. *Journal of Terramechanics, 76*, 1–13.

Yoo, D., Kim, N., Park, S., Paek, A. S., & Kweon, I. S. (2016). Pixel-level domain transfer. In: European conference on computer vision (pp. 517–532). Springer.

Zellinger, W., Grubinger, T., Lughofer, E., Natschläger, T., & Saminger-Platz, S. (2017). Central moment discrepancy (cmd) for domain-invariant representation learning. arXiv preprint arXiv:1702.08811

Zhang, Y., Wang, W., Bonatti, R., Maturana, D., & Scherer, S. (2018). Integrating kinematics and environment context into deep inverse reinforcement learning for predicting off-road vehicle trajectories. In *Conference on robot learning* (pp. 894–905).

Zhang, X., Yu, F. X., Chang, S.-F., & Wang, S. (2015). Deep transfer network: Unsupervised domain adaptation. arXiv preprint arXiv:1503.00591.

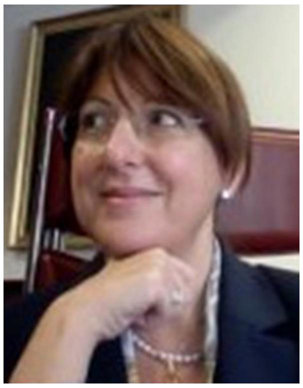
Zhu, Z., Li, N., Sun, R., Xu, D., & Zhao, H. (2020). Off-road autonomous vehicles traversability analysis and trajectory planning based on deep inverse reinforcement learning. In *2020 IEEE intelligent vehicles symposium (IV)* (pp. 971–977).

Zhu, Z., Li, N., Sun, R., Zhao, H., & Xu, D. (2019). Off-road autonomous vehicles traversability analysis and trajectory planning based on deep inverse reinforcement learning.

Giuseppe Vecchio received his Ph.D. in Computer Vision and Deep Learning at the University of Catania, Catania, Italy, where he focused on the intersection between computer vision, graphics, and AI to bridge the gap between simulation and reality. His work, published in high-quality peer-reviewed international conferences and journals, includes generative methods for 3D graphics, autonomous navigation in outdoor unstructured environments, and sim-to-real approaches.

Simone Palazzo is Assistant Professor at University of Catania, Italy. His research interests include machine learning, artificial intelligence and pattern recognition, with particular focus on computer vision, medical imaging, explainable AI and bio-inspired methods for machine learning. In addition to participating in large-scale research project, he is principal investigator for University of Catania in Italian RESILIENT project (PRIN 2022 PNRR call) and in EU project ECS4DRES.

Dario C. Guastella received his PhD at the University of Catania, Italy, in 2019. Since January 2022 he is Assistant Professor at the Dep. of Electrical Electronic and Computer Engineering of the University of Catania. His research activity is carried out within the Robotic Systems Group and focuses on cooperative mobile robots (both ground and aerial vehicles), terrain traversability analysis and Artificial Intelligence for autonomous navigation.



Daniela Giordano received her MS degree in Computer Engineering in 1990 (University of Catania) and her PhD from Concordia University (Montreal, Canada) in 1998. She is Full Professor at University of Catania, Department of Electrical, Electronics and Computer Engineering, where she teaches Cognitive Computing and Artificial Intelligence. From 2012 to 2020 she has been the Director of the M.S. degree in Computer Engineering at the University of Catania, and from 2019 she is Responsible for the Catania node of the CINI National Laboratory in Artificial Intelligence and Intelligent Systems (AIIS LAB). She has led several research projects, at regional, national and European level concerned with applying and advancing state of the art AI and pattern recognition technologies to various fields, including biomedicine, information systems for mobility and logistics, and educational systems. Her current research interests are in the area of cognitive systems (both artificial and natural), language and speech understanding, and affective computing. She is a member of the IEEE Systems Man and Cybernetics, IEEE Engineering in Medicine & Biology, IEEE ACM Human-computer Interaction, ACM Knowledge Discovery and Data Mining.

Giovanni Muscato received the Electrical Engineering degree from the University of Catania, Catania, Italy, in 1988. After completing graduation, he was with the Centro di Studi sui Sistemi, Turin, Italy. In 1990, he joined the DIEEI University of Catania, where he is currently a Full-Time Professor of robotics and automatic control and since 2018, Director of the Department. His current research interests include service robotics and the cooperation between ground and flying robots. He was the coordinator of the EC project Robovolc and is the local coordinator of several national and European projects in robotics. He is the author of more than 300 papers in scientific journals and conference proceedings and three books in the fields of control and robotics. He is also with the Board of Trustees of the Climbing and Walking Robots (CLAWAR) Association and Senior member of the IEEE. Web site: www.muscato.eu.

Concetto Spampinato earned his Laurea degree and completed his PhD in Computer Engineering at the University of Catania in 2004 and 2008, respectively. He currently serves as an Associate Professor at the same institution. He spent a research term at the University of Edinburgh during 2008-2009, concentrating on object detection and recognition. Since October 2016, he has been a Courtesy Faculty member at Center for Research in Computer Vision at the University of Central Florida. Dr. Spampinato's research focuses on learning-based computer vision and pattern recognition, especially using deep learning techniques. In 2014, he founded the Pattern Recognition and Computer Vision Laboratory at the University of Catania. He has authored over 200 publications and led various EU, national, and regional projects on foundational AI and its applications.