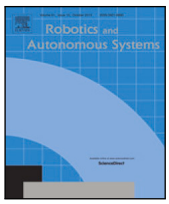
Robotics and Autonomous Systems 170 (2023) 104524

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Pyramidal 3D feature fusion on polar grids for fast and robust traversability analysis on CPU

Daniel Fusaro, Emilio Olivastri, Ivano Donadi, Daniele Evangelista *, Emanuele Menegatti, Alberto Pretto

Department of Information Engineering, University of Padova, Italy

ARTICLE INFO

Keywords:
Traversability analysis
3D LiDAR semantic segmentation
Autonomous driving
Machine learning

ABSTRACT

Self-driving vehicles and autonomous ground robots require a reliable and accurate method to analyze the traversability of the surrounding environment for safe navigation. This paper proposes and evaluates a real-time machine learning-based traversability analysis method that combines geometric features with a pyramid-polar space representation based on SVM classifiers. In particular, we show that by fusing geometric features with information stemming from coarser pyramid levels that account for a broader space portion, as well as integrating important implementation details, allows for a noticeable boost in performance and reliability. The main goal of this work is to demonstrate that traversability analysis is possible with effective results and in real-time even on cheaper hardware than expensive GPUs, e.g. CPU-only PCs. The proposed approach has been compared with state-of-the-art deep learning approaches on publicly available datasets of outdoor driving scenarios, running such algorithms both on GPU and CPU to compare runtimes. Our method can be fully executed on CPU and achieves results close to the best-in-class methods, runs faster, and requires fewer and less expensive hardware resources, consuming less than 30% electrical power with respect to deep learning models on embedded processing units. We release with this paper the open-source implementation of our method.

1. Introduction

Traversability analysis is a fundamental task in the fields of robotics and autonomous driving. It acts as a guideline, or reference, for autonomous vehicles to effectively navigate through all kinds of scenarios.

The ability to correctly detect non-traversable regions of the terrain is closely related to the type of vehicle on which the analysis is performed, and, according to [2], its computational complexity increases with the diversity of the surrounding terrain. As in [3], a grid cell is said to be *Non-traversable* if its characteristics exceed certain thresholds for the vehicle specification: sizes, workloads, risk awareness and so on.[1]

A binary terrain classification that discriminates between *Traversable* and *Non-traversable* areas needs to be reliable and efficient. The reliability requirement is essential, it affects how much the vehicle is able to move in the environment, and it is strictly connected to the robot's safety. At the same time, the real-time requirement is critical for this task, since a delayed detection may negatively affect the vehicle's reaction speed.

In this paper, we propose a real-time machine learning-based polar grid segmentation method, in which point clouds acquired from a 3D LiDAR (Light Detection and Ranging) are used for traversability analysis. The focus of our work is to bring the traversability analysis task also on CPU-only devices, such as simple, cheap and light PCs. The goal is to obtain a method whose accuracy and other important metrics are comparable to modern state-of-the-art deep learning approaches. Deep learning solutions have cutting-edge performances but require expensive hardware as powerful GPUs that are not always available in many practical applications. On the other hand, not all modern deep learning approaches are able to run in real-time, not even on powerful GPUs. Our method instead, enables real-time traversability analysis on CPU.

In our approach, a point cloud is arranged in buckets of 3D points, creating a 2D polar grid centered on the origin of the LiDAR coordinate system. Compared to a uniform squared grid, the polar grid naturally takes into account the non-uniform density of point clouds captured with 360-degree 3D LiDARs commonly used in robotics and

* Corresponding author.
E-mail addresses: fusarodani@dei.unipd.it (D. Fusaro), olivastrie@dei.unipd.it (E. Olivastri), donadiivan@dei.unipd.it (I. Donadi), evangelista@dei.unipd.it (D. Evangelista), emg@dei.unipd.it (E. Menegatti), alberto.pretto@dei.unipd.it (A. Pretto).

[1] Note that this definition differs from the *driveability* stated in [1].

https://doi.org/10.1016/j.robot.2023.104524

Available online 9 September 2023

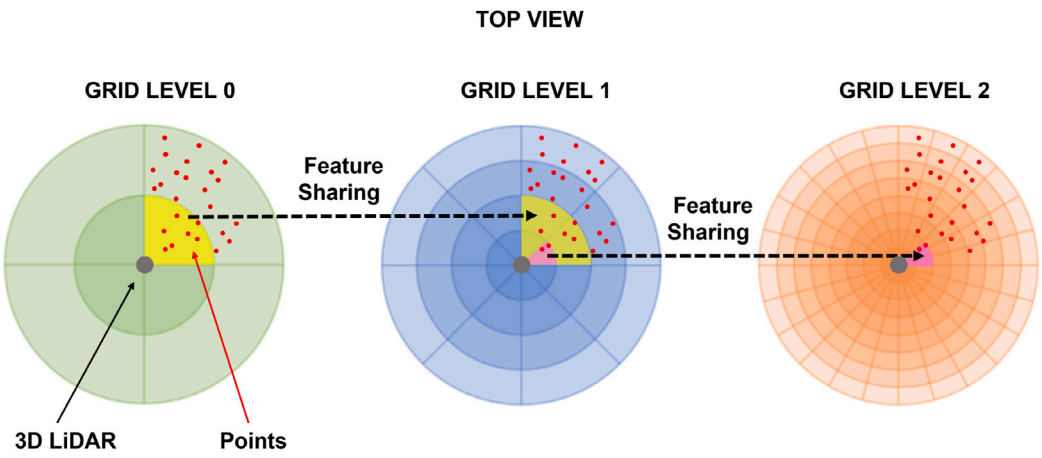


Fig. 1. Overview of the pyramid-polar space representation used in our method. The input point cloud acquired by a 360-degree 3D LiDAR is arranged on a multi-level grid, in the figure three example levels are reported from left to right in a coarse-to-fine progression. The left circle represents the coarsest level (level 0). The point cloud is further subdivided into fine-grided cells at higher levels. The figure also illustrates the employed feature sharing policy: all four magenta cells located on the right-most circle, which represents the level 2 grid, inherit features from the corresponding magenta cell at the upper level, level 1, depicted in the center of the image. Similarly, the four light-yellow cells highlighted at level 1 inherit features from the light-yellow cell at level 0, illustrated on the leftmost part of the image. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

automotive. A grid-based approach is suitable for getting immediate traversability feedback, at the price of a low cloud density. In our method, we aim to classify each cell of the collected polar grid: we compute a set of features representing the group of points in each cell by using a coarse-to-fine approach that integrates multi-scale information (Fig. 1) , then by using such features we evaluate the level of traversability of the cells using a binary classifier. Another challenge of this work is to use only geometric elements in a mixed urban environment: roads, sidewalks, mixed vegetation (e.g., trees, near-road vegetation, urban parks, forests), dynamic agents, and many other urban elements that make the classification of the 3D cloud more difficult and ambiguous.

1.1. Contributions

The contributions of this work can be summarized as follows:

1. The development of a geometric-based approach for a Support Vector Machine (SVM)-based traversability analysis and a feature fusion module on a pyramid-polar space representation, that are able to run in real-time on CPU-only devices;
2. The comparison of the proposed approach with state-of-the-art semantic segmentation based methods on large publicly available datasets;
3. An open-source implementation of the proposed method is released for public usage.[2]

2. Related work

The concept of traversability is one of the pillars of autonomous driving. Even so, there is no explicit formulation of traversability, but usage of this concept can be found in the motion planning context where "traversability maps" are used for the classification of a cell-divided map into *Traversable* or *Non-traversable* categories [4–7]. Following this definition, traversability analysis can be achieved by processing the outputs of sensors such as LiDARs, radars and/or cameras to infer which areas of the surrounding environment can or cannot be traversed. The traversability analysis problem can also be interpreted as a subclass of the generic multiclass semantic segmentation problem, in which the classes of interest for an element (e.g. pixel, voxel, etc.) are essentially two: traversable and non-traversable. On the other hand, many multiclass semantic segmentation approaches, especially those based on deep learning, can be easily converted to traversability analysis by reducing the number of classes or simply by re-mapping the original classes. For this reason, in this literature review, we report also many semantic segmentation approaches designed for outdoor applications.

2.1. Traditional approaches

Early algorithms for traversability analysis were based on ML (Machine Learning) techniques and relied mainly on handcrafted, geometric-based features extracted from point clouds. In [8] the point cloud is transformed into a new 2D histogram-based representation. In this new domain, a road line estimation algorithm is applied to obtain the actual road, distinguishing between traversable areas and obstacles by means of a linear classification task in 2D space. In [9] the point cloud is voxelized and these voxels were clustered based on their type (geometric or appearance) and classified using a Random Forest, then the predicted labels were fed to a Conditional Random Field (CRF) to refine the segmentation. [10] proposes a semi-supervised learning approach for traversability analysis which is able to learn compact feature vectors to infer cells' traversability in a 2D occupancy grid with a Naive Bayes classifier, starting from a small set of positive samples (traversable) and unlabeled data. A traversability mapping method that uses Bayesian Generalized Kernel inference can be found within the LeGO-LOAM framework [11]. This work proposes an approach for solving the traversability analysis problem by producing 2.5D elevation grid maps from LiDAR scans. It exploits Bayesian Generalized Kernel (BGK) inference to solve the sparse data problem encountered during terrain mapping. Then, the traversability analysis is performed on those locations intersected by the current LiDAR scan points, and elsewhere it is estimated by BGK traversability inference. In [12], authors train an SVM classifier that uses both geometric-based and appearance-based local descriptors. Experimental results show that purely geometric-based features already provide enough accuracy, but the integration of different types of features, i.e., descriptors taken from 2D camera image processing, increases the robustness of the classifier and improves its classification accuracy.

[2] Code available at: https://github.com/Bender97/TraversabilityAnalysis.

2.2. Deep learning-based approaches

With the rise of deep learning, also traversability analysis moved towards deep architectures, with a strong focus on image-based methods. These methods usually rely on effective and ready-to-use CNNs (Convolutional Neural Networks) designed for multiclass semantic segmentation.

In OFFSEG [13] semantic segmentation is performed on RGB images, then regions predicted as traversable are further classified in terms of color clusters to determine sub-classes like mud, puddle, grass, water, etc., that are very useful information when dealing with off-road scenarios. GONet [14] leverages a Generative Adversarial Network (GAN) to predict from the input image whether the surrounding area is traversable or not. The GAN, trained on traversable examples only, aims to generate an image similar to the input image only if this last depicts a traversable area; a classification module, trained with both positive and negative examples, is used to decide whether the area is traversable by comparing the input and generated images. In [15] traversability estimation is addressed for dynamic environments by proposing a deep network that requires the last two acquired images and a virtual navigation command, i.e., a linear and angular velocity. The network is composed of two CNNs: the first one predicts how the static elements will propagate in the future frame, while the second one predicts how the dynamic elements will behave and change the scene. Finally, the predicted image will be fed to GONet [14] to predict the traversability map. In [16] the traversability prediction is further divided into categories based on the movement modalities of the robots (e.g. legged, wheeled). In [17] the authors present a semi-supervised learning framework for Mars terrain segmentation, which will be used for studying the traversability of the inspected area. The proposed method employs a deep network trained in an unsupervised manner to learn an effective representation. This network will serve as the backbone for the subsequent network that will be trained on a limited number of labeled images. Instead of explicitly learning a segmentation mask for traversability analysis, the authors in [18] propose a network that learns a function that maps images to scores representing terrain traversability. This function is then incorporated as a term into the path planner's cost function, enabling the consideration of terrain traversability during path planning.

Deep learning architectures are also used to segment point clouds, e.g., acquired with 3D LiDARs. A first category of methods relies on a 2D representation of the point cloud by projecting it into a surface. For instance, in SqueezeSeg [19] the point cloud is projected onto a sphere and then a point-wise encoder–decoder CNN performs an initial prediction on the points that will be then refined by a CRF. In order to obtain higher accuracy, SqueezeSeg has been improved by integrating an unsupervised domain adaptation pipeline that trains the network on synthetic data and improves its performance on real data [20] and by exploiting Spatially-Adaptive Convolution (SAC), a technique for tailoring convolutional filters to process different parts of the image according to the input data [21]. In [22] a top-view image extracted from the input point cloud is fed into a fully convolutional neural network (FCN) to distinguish between the two "road" and "non-road" classes. PolarNet [23] addresses the more general multiclass semantic segmentation problem in urban scenarios by using Bird-Eye-View to project a point cloud into a grid, that is expressed in polar coordinates, which alleviates the problem of uneven distribution of the points due to the LiDAR sensor. They employ a Ring CNN to process the polar grid in order to share information between neighboring ring segments. On the opposite side, SalsaNet [24] and its evolution SalsaNext [25] claim that the type of projection, cartesian or polar, does not bring any type of advantage to the segmentation in their work. RangeNet++ [26] uses range images as the intermediate representation and then projects the 2D classification results into the original point cloud by k-nearest neighbors interpolation.

A second category of methods tries to work directly on the raw point clouds. PointNet [27], a seminal work in 3D perception on unordered point clouds, obtains permutation invariance of the input points by exploiting a learned, approximated symmetric function defined in the point set. In RandLA-Net [28] the point cloud is first subsampled to efficiently process large amounts of data. Random sampling is paired with a feature aggregation step to avoid losing relevant information using an attention mechanism. KPConv [29] introduced a new convolution operation that works directly on point clouds, eliminating the need for an intermediate representation. Another approach is to regularize the data in a point cloud using a voxel grid, in order to capture spatial relations with 3D convolutions. In [30] an encoder–decoder model is used to voxelize the point cloud, which will be then processed by a 3D convolutional network to output per-point labels. The key features of this approach are an attentive feature fusion module which is able to aggregate local and global context from the voxel grid and an adaptive feature selection module to better extract global context at different scales. In [31], the input is still a voxelized version of the point cloud, but the label predictions on the voxels are then refined to a per-point level using trilinear interpolation and a CRF module. Cylinder3D [32], uses a cylindrical grid to partition the space, instead of the classical uniform grid, to avoid the LiDAR point cloud density variance. A tailored 3D convolutional network is used to predict labels for the voxels and a further point-wise refinement module is introduced to improve the coarse voxel predictions. In [33], we proposed a hybrid approach that combines geometric and appearance-based features for training deep encoder–decoder architectures to detect the traversability score of a uniform planar grid. The method uses both point clouds and RGB camera images. Point clouds are firstly integrated using odometry estimation, then sorted into the cells of the grid. Geometric features are computed for each cell and then appearance-based features are extracted from RGB by projecting each cell to the image plane. Then, the geometric and appearance feature vector is fed to an encoder–decoder model that outputs the traversability score of each cell.

Another way of representing a point cloud is by using the super-point representation, in which points in the cloud are clustered and the clusters (called super-points) are then connected in a graph representation, which can be processed by Graph Convolutional Neural Networks (GCNNs). In [34] points in the point cloud are clustered together based on geometric similarities, and then a GCNN is used in order to perform segmentation on the super-points. The points belonging to a cluster will share its label. In [35] the GCNN uses a structure-aware loss function that exploits the structure information contained in the super-points, while in [36] the super-point generation is done automatically by using an encoder–decoder architecture that performs both depth-wise and point-wise graph convolutions. Other works try to improve classification accuracy by using information from more than one sensor or by using different representations of the same 3D data at the same time. In [37–39] both images and point clouds have been fed to deep networks in order to extract as much information from the environment to obtain a more robust traversability analysis. In fact, in [37] this approach has been shown to be effective even in unstructured environments. 2DPass [40] still uses both images and point clouds. During training, two independent parallel networks are trained on the task of semantic segmentation, one on the point cloud and one on the related image. Knowledge distillation and feature fusion from the image branch to the point cloud branch are used as a form of regularization. At inference time, the image branch is completely discarded and inference is performed on the point cloud only. Another method leveraging both point clouds and images is [41], where super-points are used as an intermediate representation to connect 3D and 2D features and super-point-based pooling fuses the two features for joint learning. Lastly, RPVNet [42] designs a novel fusion network to merge information coming from point clouds, voxel grids, and range images. The three data representations are processed with parallel networks that share information at multiple levels. In the end, the

network processing the point cloud outputs the final segmentation, after receiving the results from the voxel and range networks.

A recent trend is to use Deep Inverse Reinforcement Learning in order to learn implicitly the traversability map, that will manifest from the navigation behavior of the vehicle. In [43] a behavior-based method to solve traversability analysis in an off-road scenario has been proposed, in which the cost function is extracted from experts' demonstrations, and then that cost function is used in order to learn an optimal policy. Meanwhile, in [44], Deep Inverse Reinforcement Learning is used to predict the trajectories of the other agents in the scene and to produce a traversability map that also takes into account dynamic elements that are in the scene for safer and more robust navigation.

This work builds upon our previous work [45], with substantial improvements, among others: (i) A more difficult traversability analysis task, in which the boundary region between road and sidewalk is considered as *non-traversable*; (ii) The introduction of a polar grid, instead of a uniform grid, to better exploit the point cloud spatial distribution around the 360-degree spinning LiDAR frame; (iii) The design of a pyramid-polar feature sharing mechanism, which enables the model to take into account a wider spatial region for inferencing the traversability of a cell; (iv) The development of a pure-geometric approach, in contrast with the previous work that employed also appearance-based features; (v) Our approach relies on a single point cloud, which is not dependent on odometry estimators. This means that our approach does not need to integrate multiple point clouds together in order to generate a comprehensive representation of the world. This simplicity is beneficial because it reduces the complexity of our model, allowing for faster predictions of the environment.

3. Method

The proposed approach leverages handcrafted geometrical-based features, Principal Component Analysis (PCA), and SVM models. In machine learning and pattern recognition, a feature is an individual measurable property or characteristic of a phenomenon [46]. A geometric feature is calculated based on geometric properties. Features can be manually designed or self-learned as in deep learning models, where they are often hierarchically organized in layers in order to capture both local and global representations of the input. The self-learned features have proven to substantially outperform the handcrafted counterpart. However, this is achieved by using models with a very large number of parameters to train, so using powerful GPUs is often an essential requirement, even at the inference stage. Handcrafted features, on the other hand, have typically reduced footprints and are well-suitable for general-purpose hardware and/or embedded devices. Furthermore, the working mechanisms of the kernels of the hidden layers of deep neural networks are not so easily interpreted [47], and they usually need a large amount of annotated data to properly generalize, whereas handcrafted features can be quickly and easily tailored to specific cases. In this work we run in defense of the latter, proving that in the specific case of traversability analysis, with a hierarchical polar grid organization and with important implementation details, handcrafted features can still represent a valid alternative to deep models in low-cost hardware or when real-time requirements are strict.

In the following, a *feature vector* of a set of points $S = \{p_1, p_2, \dots, p_n\}$ with $p_i \in \mathbb{R}^3$ is expressed in the form of a vector of features

$$\mathcal{F}(S) = [f_1, f_2, \dots, f_k]^T$$

where k is the number of features. This vector of fixed dimension is used to train an SVM model. SVMs are popular decision models used in classification and regression problems. As stated in [46], an important property of SVMs is that the determination of the model parameters corresponds to a convex optimization problem, and so any local solution is also a global optimum. For the sake of brevity, the details on SVM models can be found in [48]. It is sufficient to say that training an SVM model means finding an optimal decision rule to assess the label of a feature vector. The so-called *kernel trick* helps in finding such a decision rule by transforming the original data into a higher dimensional space. By experimental results, we have found that RBF (Radial Basis Function) kernel is the best-performing kernel with respect to Linear and Polynomial kernels in our case. For this reason, we apply RBF in this work.

We employ Principal Component Analysis (PCA) to reduce the dimensionality of the dataset, increasing interpretability but at the same time minimizing information loss [49].

3.1. Pyramidal-polar representation

We assume that the point cloud of the surrounding space is produced by a 3D scanning LiDAR. The traversability of the environment is then evaluated using a polar-based space representation which partitions the point cloud into cells whose size depends on the distance from the sensor frame. A point cloud is better organized using this coordinate system than a Cartesian one since the resulting cells will have a more balanced number of points (it is reasonable to say that points far away from the sensor frame will be sparser than points very close to it). We decided to fix a minimum distance R_{min} and a maximum distance R_{max} from the sensor frame and ignore all points that lie outside this distance interval. The first one is set based on the size of the vehicle that captured the dataset, the second is set according to the size of the area to be analyzed.

In order to let the system get more information than the content of the current cell, we propose a pyramidal-polar representation (see Fig. 1). We divide the point cloud into multiple-resolution polar grids, from coarser to finer. Let us say that the coarser level is level 0. Then, if we use n different polar representations, the level $n-1$ will be the finest. If we compare two cells at the same distance from the sensor frame, a cell at level 0 will be greater in area, hence it will contain possibly more points than a cell at level $n-1$. We can define relations between cells at different levels, in particular between finer cells that are *completely contained* in a coarser cell. We can expect benefits from including coarser features in the features of finer cells as the back-end ML model can integrate higher-level information from the neighborhood of the finer cell. To this end, in our system the features vector of each cell $c_{i,l}$ at level $l > 0$ is expanded by including the features vector of the corresponding cell $c_{j,l-1}$ at level $l-1$, i.e. the one that contains $c_{i,l}$. Included features can be the predicted label or the full geometric feature vector of the containing cell. Recursively, a cell can include all the features of the containing cell, so also the features that the containing cell derived from its containing cell.

3.2. Geometric-based feature extraction

The geometric-based features $\mathcal{F}_g(S) = [f_1, f_2, \dots, f_r]^T$ of a polar grid cell are computed based on the geometrical properties of the set of 3D points that lie in it, where r is the number of geometrical features. Most of these features are computed using the eigenvalues and eigenvectors of the correlation matrix of the points in S. This matrix is computed based on the relative space coordinates of the points in S and expresses the dependency between them. It is a 3×3 symmetric positive semi-definite matrix with all real elements, so all of its eigenvalues are real and non-negative. Listing the eigenvalues in descending order $\lambda_1 \geq \lambda_2 \geq \lambda_3$, the corresponding eigenvectors v_1, v_2 and v_3 assume a spatial significance. As explained in [12], the eigenvector v_1 represents the direction of the maximum variance of the points in S and v_2 represents the direction of the second maximum variance of the points. On the contrary, v_3 represents the smallest direction of variance but, at the same time, assuming that the points are arranged in a smooth plane in the space, v_3 is also normal to that plane. In the context of terrain traversability analysis, where roads are generally planar, this assumes a certain relevance. Thus, assuming that $\lambda_1 \neq 0$, among others we use some of the features proposed in [12] and reported in Table 1, where:

Table 1
Geometric features of a point set S computed by the proposed approach to infer the traversability analysis.

linearity = $\frac{\lambda_1 - \lambda_2}{\lambda_1}$	sphericity = $\frac{\lambda_3}{\lambda_1}$
planarity = $\frac{\lambda_2 - \lambda_3}{\lambda_1}$	omnivariance = $\sqrt[3]{\lambda_1 \lambda_2 \lambda_2}$
anisotropy = $\frac{\lambda_1 - \lambda_3}{\lambda_1}$	eigenentropy = $\sum_{i=1}^{3} \lambda_i \log \lambda_i$
sum of eigenvalues = $\sum_{i=1}^{3} \lambda_i$	curvature = $\frac{\lambda_3}{\sum \lambda}$
angle = $\arccos(\vec{n} \cdot \hat{z})$	goodness of fit = $\min(\sigma_i(C))$
roughness = $\frac{1}{\lvert S \rvert} \sum_{i=1}^{\lvert S \rvert} (z_i - \bar{z})^2$	normal vector = $\vec{n}$
inverse cardinality = $\frac{1}{\lvert S \rvert}$	surface density = $\frac{\lvert S \rvert}{d^2}$

- $\vec{n} = v_3$ is the normal vector to the plane fitted to S;
- $\hat{z}$ is the versor in the z direction in the LiDAR coordinate system;
- z_i is the z-coordinate of the i-th point in S, with $\bar{z}$ the average of the z-coordinate of all points in S;
- σ_i is the i-th singular value of the covariance matrix C of S;
- d^2 is the area of the cell that contains S, with d being the side length.

In our approach, another geometric feature is used. The *Zeta Difference* feature represents the maximum difference along the elevation direction, expressed by the normal $\vec{n_p}$ to the plane fitted to the whole point cloud. This is particularly relevant because non-traversable regions (e.g., obstacles) tend to have a wide variation along the normal direction, while points in traversable regions (e.g., roads) tend to have a very small height difference. In order to obtain a consistent metric, each point inside a cell is projected to the scene's normal vector $\vec{n_p}$. The projection p_p of a point p is obtained as follows:

$$p_p = \langle p, \vec{n_p} \rangle \vec{n_p} \tag{1}$$

Let M and m be the points of a cell whose projections have higher and lower values of the z-coordinate. Then:

$$z_{diff} = |M_z - m_z| \tag{2}$$

The feature vector is not fed to the SVM model directly. It is firstly transformed by taking the absolute value of each feature, adding a small quantity to avoid taking the log of zero, and then performing a natural log transform. More formally, each feature f transforms to f_t as follows:

$$f_t = \ln(|f| + 0.0001) \tag{3}$$

Then f_t is normalized to zero mean and unit variance. The normalization process is essential to avoid different features having completely different scales, and it allows the SVM model to train better as it helps to improve the signal-to-noise ratio and remove any bias from data.

3.3. Point clustering and feature extraction

Each point of the point cloud is assigned to the cell it belongs to by looking at the radius and the yaw of the point's 2D coordinates. If it lies outside the polar grid distance interval, it is ignored. For a detailed description of this process, refer to Algorithm 1. A cell is then considered *unpredictable* if it contains fewer points than a threshold value, let's define it τ, otherwise it is considered *predictable*. At this stage, a feature vector is computed for each *predictable* cell and eventually expanded with other features derived from coarser cells. For each predictable cell $c_{i,l}$ at level l, a feature is calculated as described in Algorithm 2.

Algorithm 1: Bin points in polar grid cells at level l

Input:
$P \subset \mathbb{R}^3$: set of points of the point cloud;
$l \geq 0$: cylinder's level;
R_{min}, R_{max}: minimum and maximum radius ranges of the polar grid at level l;
γ_n: number of yaw steps;
r_n: number of radius steps ;
$r_s = \frac{R_{max} - R_{min}}{r_n}$: radius step range (in meters)
Result: The set of point sets $\{S(i,l)\}$, $\forall$ cell c_i at level $l \geq 0$
for *each cell* c_i *at level* l **do**
  $S(i,l) \leftarrow \emptyset$;
end
for *each point* $p \in P$ **do**
  $\rho \leftarrow \sqrt{p_x^2 + p_y^2 + p_z^2}$;
  if $R_{min} \leq \rho \leq R_{max}$ **then**
    $p_\gamma \leftarrow \arctan(p_x, p_y) + \pi$;
    $row \leftarrow \lfloor \frac{\rho - R_{min}}{r_s} \rfloor$;
    $col \leftarrow (\lfloor \frac{p_\gamma \cdot \gamma_n}{2\pi} \rfloor + \frac{\gamma_n}{2}) \bmod \gamma_n$;
    $idx \leftarrow col \cdot r_n + row$;
    $S(idx,l) \leftarrow S(idx,l) \cup p$;
  else
    ignore point p ;
  end
end

Algorithm 2: Compute features for each cell, at each level

Input:
L: number of levels ;
$S(i,l)$: set of points of cell c_i at level l ;
τ: minimum number of points for a cell to be considered predictable;
r: number of geometrical features computed for each cell ;
$\hat{c_i}$: cell at level $l-1$ that contains cell c_i at level l, in other words $c_i \in \hat{c_i}$;
$F_g(c)$: set of geometric features of cell c ;
$F_{in}(c)$: set of features that cell c has inherited from cell $\hat{c}$;
$F_{lab}(c)$: predicted label of cell c (only available at levels $l > 0$) ;
Result: The set of feature sets $\{F(i,l)\}$, $\forall$ cell c_i, level $l \geq 0$
for $l \leftarrow 0$ *to* $L-1$ **do**
  for *each cell* c_i *at level* l **do**
    if $\|S(i,l)\| \geq \tau$ **then**
      compute $F_g(S(i,l)) \leftarrow [f_1, f_2, ..., f_r]$;
      if $l \geq 1$ **then**
        compute $F_{in}(c_i) \leftarrow \{F_{in}(\hat{c_i}), F_g(\hat{c_i}), F_{lab}(\hat{c_i})\}$;
      else
        $F_{in}(c_i) \leftarrow \emptyset$;
      $F(i,l) \leftarrow \{F_g(S(i,l)), F_{in}(c_i)\}$
    else
      label cell c_i at level l as unpredictable

3.4. PCA reduction of geometric features

In order to understand the impact of each geometric feature in the computed features vector we decided to apply Principal Component Analysis. We aimed to explore the possibility that the classifier would be able to take advantage of a feature vector projection into a lower dimensional space. Benefits of this reduction include an analysis of which geometric features really matter for the task, a simplification of the model (a too complex model with many parameters tends to overfit), and a reduced runtime of the system.

3.5. A-priori positive detection

In our approach, we tried another technique, which we refer to as a-priori positive detection. If a cell at coarser levels is classified as *traversable*, then, as per the definition of *traversable* cells (for the definition, refer to Section 4.2), we can be reasonably sure that all the cells at the finer levels that are contained into this cell will be *traversable* too. This can have several advantages in terms of runtime since we can skip many computations. In CPU-only applications with very strict runtime constraints, this technique may boost performance. Of course, it requires SVM models at coarser levels to be as accurate as possible. As shown in Section 5.1, the coarser level accuracy and the F1 score are not good enough to support the prediction of finer levels a-priori for traversability analysis. We report results for comparison in Table 8.

4. Experiments

We implemented our method in C++[3] and performed all the CPU-only experiments on a Laptop PC equipped with an AMD Ryzen 7 5800H CPU (3.2 GHz), 16 GB of RAM, and Linux OS (the internal GPU has not been used). All the deep learning-based approaches were tested on a Desktop PC with an Intel Core i9-10920X CPU (3.50 GHz), 32 GB of RAM, an Nvidia Titan RTX GPU, and a Linux OS As described in Section 4.8, we also run experiments on a Jetson AGX Xavier with an 8-core Nvidia Carmel CPU, a 512-core Nvidia Volta GPU, 16 GB of RAM, and Linux OS.

4.1. SemanticKITTI dataset

To evaluate the performance of the proposed approach, we used the SemanticKITTI dataset [50], a publicly available dataset for semantic scene understanding using point cloud sequences acquired with a 3D LiDAR. This dataset is based on the KITTI Odometry Benchmark [51]. It comes up with 11 different segmented scenarios, named *scenario00-scenario10*, in which a fully sensorized vehicle is driven within an urban context, in low traffic conditions. There are sometimes dynamic agents (other vehicles moving around, people, etc.) and natural elements (grass, parks, trees, etc.).

4.2. Traversability ground truth extraction

The point clouds in the SemanticKITTI dataset are not labeled with the *traversable* and *non-traversable* labels. Indeed, the definition of traversability is in itself quite ambiguous. In order to obtain a ground truth dataset to evaluate the tested methods we pre-processed the point labels and propagated them to the polar grid cells they belong to. We considered as *traversable* the points having the label corresponding to one of the following classes of the SemanticKITTI dataset: *road, sidewalk, parking, lane marking, other ground*. All the other point labels are considered *non-traversable*. The label of a cell is calculated by looking at the LiDAR points that fall inside each polar grid cell. Let τ be the minimum number of points for a cell to be considered predictable: in all our experiments we set $\tau = 4$. If a cell contains less than τ points, it is labeled as *unpredictable*, while if it contains at least τ *non-traversable* points, it is labeled as *non-traversable*. Otherwise, it is labeled as *traversable*. This is done because the SemanticKITTI dataset includes some wrong labels: sometimes a fully *traversable* cell (e.g., a *road* cell) contains an outlier marked as *non-traversable*. By filtering out based on at least τ *non-traversable* points, we get rid of such outliers. An exception is done to make the task more suitable for real-world application: since a vehicle should not cross the roadside up to a sidewalk area, we decided to set some cells as *non-traversable* when they belong to the boundary region including both road and sidewalk areas, i.e. if the cell contains both *road* and *sidewalk* points.

The ground truth collected in this way (an example is reported in Fig. 2(a)) comes with some consequences:

- A number of $\tau = 4$ points is sometimes not sufficient for some geometric properties to completely discriminate cells. This represents a challenge for our method which classifies each cell independently.
- To compare our approach with point-wise segmentation methods, we need to propagate output point labels to cell labels, in the same way we did for SemanticKITTI's ground truth.

[3] Code available at: https://github.com/Bender97/TraversabilityAnalysis

Table 2
Parameters list of the best performing SVM model.

Level	0	1	2
radial steps	8	16	64
yaw steps	16	32	128
ν - nu	0.2028	0.1805	0.1838
γ - gamma	0.098	0.0765	0.1003
C	1.0	1.0	1.0
best method	geom	Geom-L	Geom-L
PCA dimension	17	17	17

4.3. Training and test protocols

The experiments on the SemanticKITTI dataset were performed following the standard procedure which involves testing the approaches on Scenario 08 while all the remaining ten scenarios are used for model training. For our method, we selected the SVM hyperparameters by using the grid search algorithm. We have designed a $n = 3$ level pyramid-polar space representation, based on empirical experiments using alternative strategies (see Section 4.4). Each level has its own dedicated SVM model. The final SVM hyperparameters used for each level are listed in Table 2.

We start by partitioning each SemanticKITTI point cloud into a pyramid-polar space representation. For each level $l = 0, \dots, n-1$, we assign the points to the corresponding polar cells $c_{i,l}$, where i is the index of a cell at level l. For each predictable cell $c_{i,l}$, we assign the ground truth label as described in Section 4.2 and compute the geometric features vector $\mathcal{F}_g(S_{i,l})$ as described in Section 3.2, with $S_{i,l}$ the set of points that belong to $c_{i,l}$. At level $l = 0$, we train the SVM model by using only the geometric features and the corresponding ground truth labels. For $l' > 0$ we also keep track of the predicted label, geometric features, and inherited features $\mathcal{F}_{in}(S_{j,l'-1})$ computed at the previous level for the coarse-grained cell $c_{j,l'-1}$ that contains $c_{i,l'}$, i.e., $c_{i,l'} \in c_{j,l'-1}$ (see also Fig. 1). This additional information can be concatenated to $\mathcal{F}_g(S_{i,l'})$ to contribute to the cell's features vector in some variants of our approach (see Section 4.4). The composed features vector is then used to train a different SVM model for each level $l' = 1, \dots, n-1$. The same point cloud partitioning and features computation strategies are performed in testing. After the inference, the labels estimated at each level of the pyramid are compared with the ground truth labels to compute the metrics reported in Table 3.

4.4. Approaches for feature fusion

We investigate several feature fusion approaches, where for each level features are possibly augmented with coarser levels of information, possibly after PCA re-projection. Please recall that feature sharing is available only at levels $l > 0$.

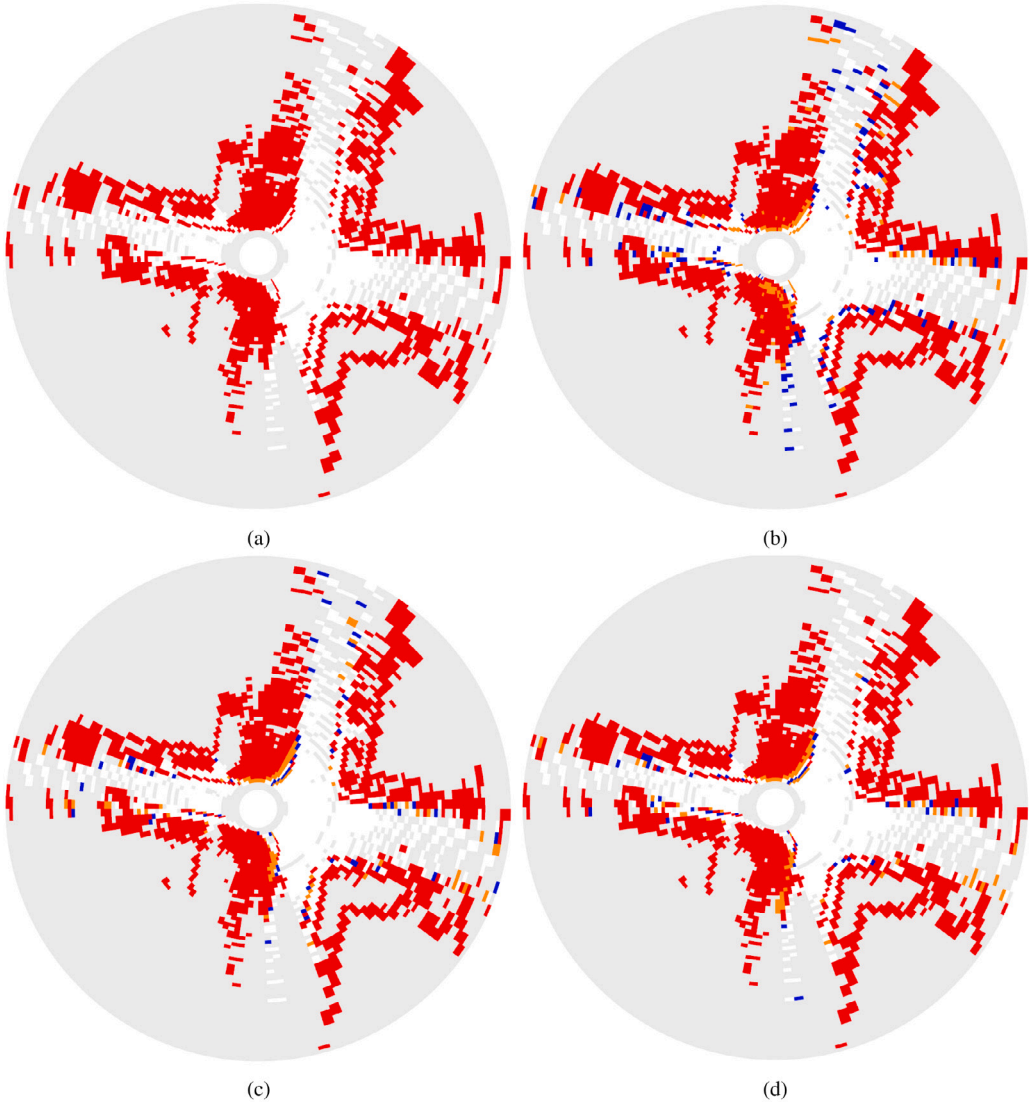


Fig. 2. Example results of polar grid-based traversability analysis applied to a scan of the SemanticKITTI's test Scenario 08: (a) ground truth; (b) our approach; (c) RangeNet++; (d) Point-Voxel KD. Cells correctly classified as traversable and non-traversable are colored in white and red, respectively. Cells incorrectly classified as traversable and non-traversable are colored in orange and blue, respectively. Unknown and unpredictable cells are colored in gray. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- *Geom:* For each cell, we only use geometric features computed at the current level, so the resulting feature vector will be made of the 17 features described in Table 1 plus Eq. (2).
- *Geom-label, or Geom-L:* For each cell, we use the geometric features (*Geom*) computed at the current level l augmented with the predicted labels of the related coarser cells at levels $l-1, l-2, \dots, 0$. Adding the label information is expected to enhance the information available for the inference since it provides a broader look at the neighborhood of each cell. This can be particularly relevant for cells with very few points or belonging to the boundary region between road and sidewalk. The resulting feature vector will be long $17 + l$ features, with l being the level.
- *Geom-all, or Geom-A:* For each cell, we use the geometric features computed at the current level l augmented with the predicted labels *and* the geometric features computed for the related coarser cells at levels $l-1, l-2, \dots, 0$. The resulting feature vector will be long $17 \cdot (l+1) + l$ features, with l being the level.
- Each of the three previous approaches is tested using the PCA reduction technique, to investigate both the usefulness of all the features and also if a re-projection to a different vector space can enhance the performance of the method.

4.5. Metrics

To compute the metrics we used the number of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The performance is measured using the metrics detailed in Table 3. In particular, with the accuracy metric we can get an easily interpretable value representing the percentage of input samples that were correctly classified, but it is strongly biased by imbalances in the number of samples per class; the Intersection over Union - Traversable (IoUT), instead, measures the overlap between the set of ground truth positive samples and the set of predicted positive samples, giving a measure for single-class accuracy; accordingly, the Intersection over Union - Non-traversable (IoUN), measures the overlap between the set of ground truth negative samples and the set of predicted negative samples. When the dataset is dominated by negative samples, a more reliable metric can be found in the F1 score, which balances precision with the rate of detection for the positive class. The F1 score ignores true negatives and can thus be misleading if the dataset is positive-dominant; the Cohen's kappa metric, which in the binary classification case measures fractional improvement over the random classifier, is symmetric and assesses both directions of predictability. True Positive Rate (TPR),

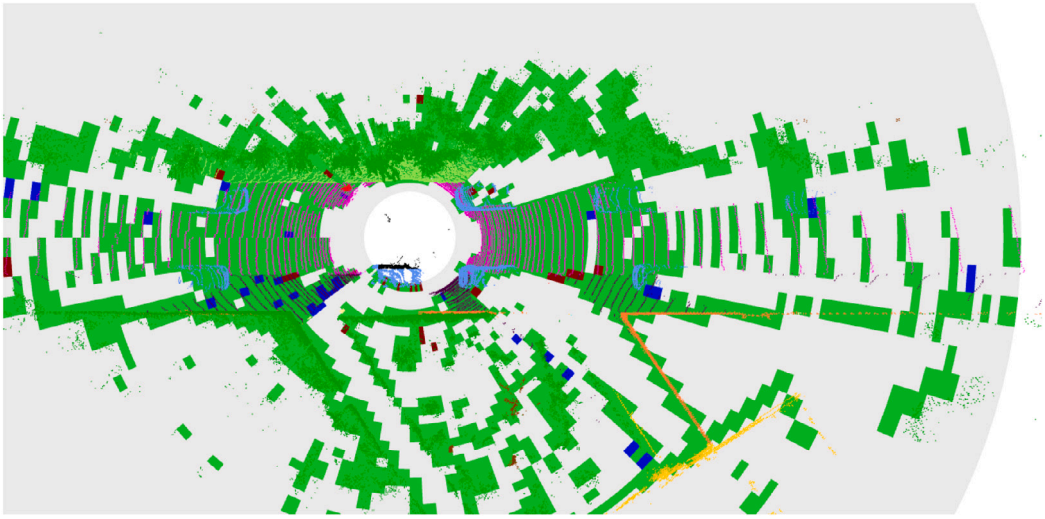


Fig. 3. Qualitative results of the proposed approach applied to a scan from test set Scenario 08. Correctly inferred cells (true positives and true negatives) are colored in green cells, while false positives and false negatives are colored in dark-red and blue, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Metrics used for the quantitative evaluation of the models.

Metric	Definition
Accuracy	$\frac{TP+TN}{TN+FN+TP+FP}$
IoUT	$\frac{TP}{FP+FN+TP}$
IoUN	$\frac{TN}{FP+FN+TN}$
F1 score	$\frac{2TP}{2TP+FP+FN}$
Cohen's kappa	$\frac{2\cdot(TP\cdot TN-FN\cdot FP)}{(TP+FP)\cdot(FP+TN)+(TP+FN)\cdot(FN+TN)}$
TPR	$\frac{TP}{TP+FN}$
TNR	$\frac{TN}{TN+FP}$

Table 4
Average results on SemanticKITTI Scenario 08. The *runtime* value (Rt) is expressed in milliseconds (ms).

Exp	Acc	IoUN	IoUT	F1	Co	TPR	TNR	Rt_{CPU}	Rt_{GPU}
	%	%	%	%	%	%	%	ms	ms
P-SVM	91.7	87.4	80.4	89.2	82.4	89.0	93.4	**49**	–
RN++	95.4	92.8	88.6	94.0	90.3	93.2	96.8	1605	**30**
Cyl3D	96.8	95.0	91.9	95.8	93.2	95.3	97.7	6300	110
PVKD	**96.9**	**95.1**	**92.2**	**95.9**	**93.4**	**95.6**	**97.7**	27000	480

or sensitivity, and True Negative Rate (TNR), or specificity, are also reported. A high TPR means that the model rarely misses positive cases, and conversely, a high TNR means that the model rarely misclassifies negative cases as positive. This is particularly important in traversability analysis, where the cost of False Positive is high as obstacles are not identified as non-traversable cells.

4.6. Comparison

The experiments carried out aim to emphasize the performance brought by the contributions of this work, in particular the use of the pyramid-polar space representation, the geometrical features, and the low runtime due to the SVM models.

We tested our framework (referred to as P-SVM below and in the tables) against different recent semantic segmentation methods: RangeNet++ (RN++) [26], Cylinder3D (Cyl3D) [32] and Point-Voxel KD (PVKD) [52]. At the time of this work being carried out, Point-Voxel KD ranked second on SemanticKITTI Leaderboard for semantic segmentation task but was the first among the open-source approaches that do not exploit the RGB images. Cylinder3D is the second. Interestingly, our approach is the only one of these approaches that does not use the remission values of LiDAR scans and, despite this, achieves similar results. We plan to leverage remission values in our approach in future work. Each deep learning method had to be trained by remapping the original classes into the 3 (+1) classes used for the traversability analysis: *unknown* (not learned and not used in the computation of the loss function), *traversable* (road, parking, lane marking, and other-ground), *sidewalk* (for the detection of the boundary region) and *non-traversable* (all the other classes). The traversability of a cell is then computed based on the predicted label of each point of the cloud. This is done using the exact same method used for determining the polar grid traversability ground truth starting from the segmented point cloud, but this time using the predicted point cloud labels.

None of the deep learning methods were trained from scratch. Instead, we fine-tuned the models from publicly available pre-trained models available on each method's GitHub repository page. We paid close attention to the fact that the models were not trained on the test scenario (*scenario08*).

We report in Table 4 the quantitative results in terms of the metrics from Table 3, while we present some qualitative results in Figs. 2–4. The results of the proposed method are comparable to the other tested methods. Deep learning-based methods generally achieve better results, with PVKD as the clear winner followed by Cyl3D. Despite not utilizing remission values, P-SVM gets close to the other methods in terms of accuracy, F1 score, IoUN, TPR, and TNR. Our approach demonstrates a good balance between sensitivity (89.0%) and specificity (93.4%). This means that it performs reasonably well in both correctly identifying positive cases and accurately identifying negative cases, ensuring a reliable classification outcome overall. This is also confirmed by a consistent F1 score, that describes how the model can balance between false positives and false negatives. P-SVM exhibits decent IoUN (87.4%) and IoUT (80.4%) scores, which measure the overlap between predicted and ground truth traversable/non-traversable regions, respectively. These scores suggest that P-SVM is capable of handling class imbalance, a common challenge in binary classification tasks. In general, all the models exhibit relatively high Cohen's kappa values, indicating good agreement beyond chance. Cyl3D, PVKD, and RN++ particularly demonstrate excellent levels of agreement, while P-SVM also shows a substantial level of agreement.

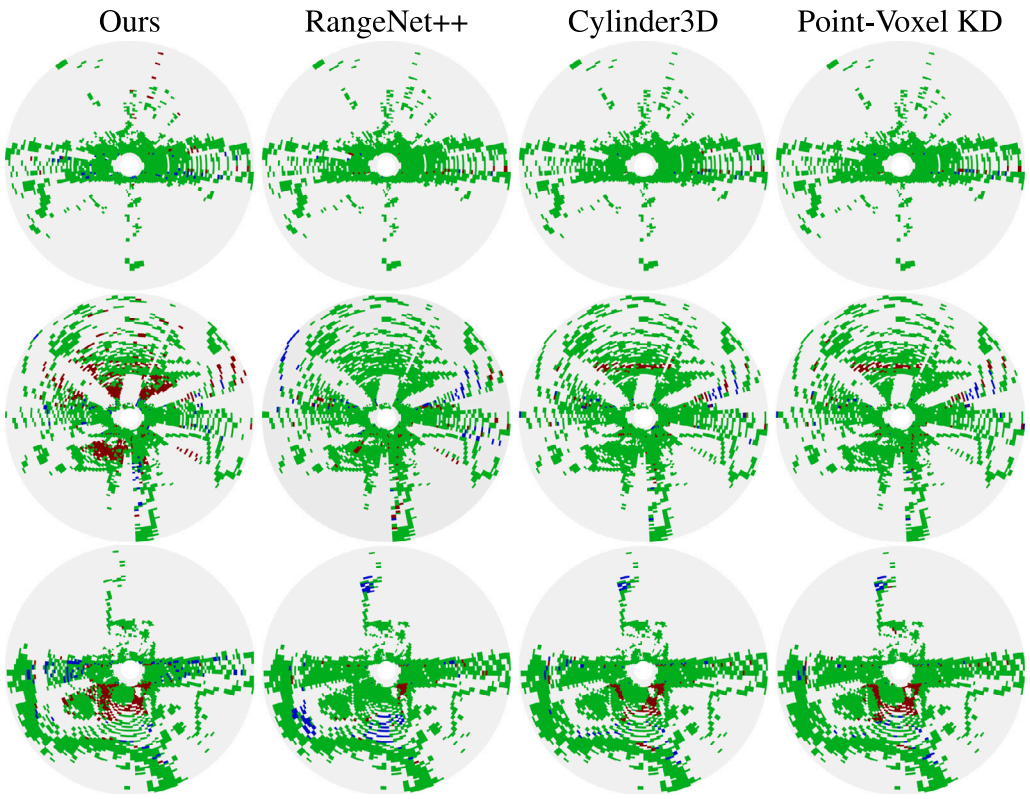


Fig. 4. From left to right, the qualitative results are respectively: our approach, RangeNet++, Cylinder3D, and Point-Voxel KD. Each row in the figure corresponds to different scenes, specifically arranged from top to bottom: a scene where all the methods perform well, the worst-case scenario for our method, and the worst-case scenario for all approaches. Please refer to Fig. Fig. 3 for the color legend.

However, the slightly better performance of deep learning-based methods comes at the cost of a higher runtime despite requiring expensive GPUs in order to achieve acceptable execution times. Only RN++ on GPU slightly outperforms P-SVM on CPU, and both are suitable for real-time scan processing, whereas both PVKD and Cyl3D are not. We also tested all approaches on the same CPU: P-SVM takes 3% of the runtime required by RN++, the fastest deep-learning method, and a mere 0.18% of the runtime required by PVKD. Overall, our method provides effective results that are close to the tested deep learning approaches while demanding nearly two orders of magnitude less computational overhead.

4.7. Generalization

To assess the generalization capability of our approach, we report a set of experiments performed on another publicly available dataset, PandaSet [53], a large-scale collection of data for autonomous driving in urban environments. It consists of multiple types of sensor data, e.g. Cameras, LiDARs, and IMUs. For our purposes, we used the point clouds coming from the main LiDAR, an Hesai Pandar64, as well as the corresponding annotated labels. Its specifications are the following: 360° horizontal FOV, 10 Hz, 64 channels. This sensor is very similar to the Velodyne HDL-64E used in SemanticKITTI.

We directly tested on PandaSet the models previously trained *solely* on SemanticKITTI. The PandaSet test set includes 6080 annotated point clouds. As reported in Table 5 and qualitatively in Fig. 5, our method generalizes well alongside PVKD and Cyl3D, while outperforming RN++, which is affected by a significant performance drop. This probably can be explained by the fact that, differently from the LiDAR used in SemanticKITTI, in PandaSet a LiDAR with non-uniform vertical scan distribution has been exploited, partially invalidating the standard projection function used in RN++.

Table 5
Average of the results obtained on PandaSet.

Exp	Acc %	IoUN %	IoUT %	F1 %	Co %	TPR %	TNR %	Rt_{CPU} ms	Rt_{GPU} ms
P-SVM	81.0	71.2	64.0	78.1	61.5	71.8	89.2	**50**	–
RN++	66.4	58.6	36.1	53.1	30.9	40.3	**89.8**	1800	**35**
Cyl3D	86.2	76.5	75.1	85.8	72.5	87.9	84.7	6700	115
PVKD	**87.9**	**78.6**	**78.4**	**87.9**	**75.9**	**92.6**	83.9	30000	490

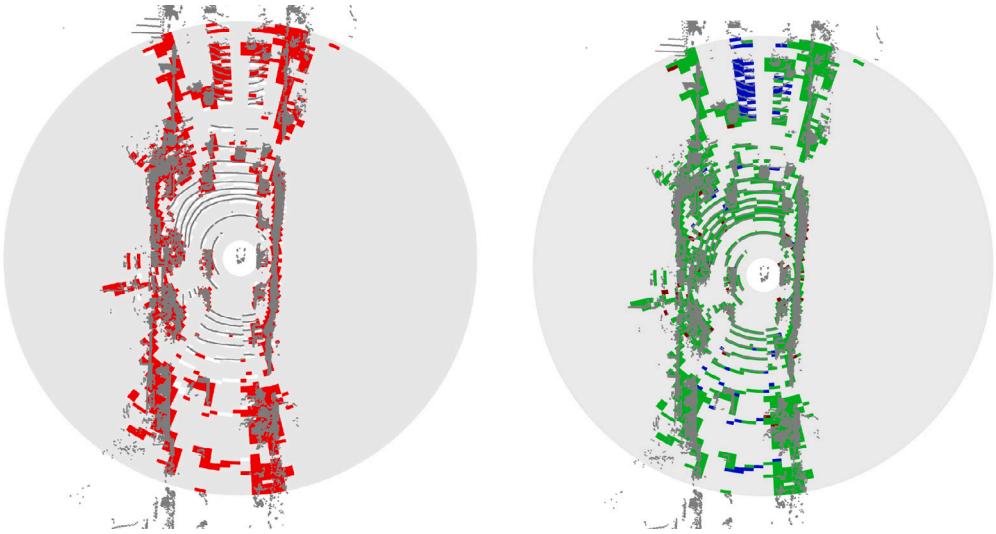
The runtime of our method on CPU is 50 ms, which is still very low when compared to the other methods' runtime, both on CPU and GPU. Only RN++, on GPU, is faster than P-SVM.

4.8. Performance on embedded devices

We also assess the runtime and power consumption performance of all methods on a Jetson Xavier board, an embedded device equipped with a small but powerful GPU specifically designed for navigation and perception tasks for autonomous vehicles [54]. The results, reported in Table 6, include the runtime, in milliseconds (ms), and the power consumption, in Watts (W), of each method. P-SVM runs on CPU while the deep learning methods also use the GPU. P-SVM outperforms, in both runtime and power consumption, the other methods. The runtime is very close to the real-time constraint (100 ms), while none of the deep learning methods are able to get close to this constraint even in GPU. Similarly, P-SVM's power consumption is less than 30% of the energy needed to run deep learning approaches.

5. Ablation study

In this section, we evaluate the different variants of feature fusion of our method (see Section 4.4) to select the best-performing one. In



(a) (b)

Fig. 5. Qualitative results of the proposed approach in a frame from PandaSet dataset. In the left Fig. 5(a), in white are cells classified as *Traversable*, in red as *Non-traversable*, in dark gray *unknown* cells. In the right Fig. 5(b), green cells are correctly classified cells (TP and TN), dark-red cells are FP and blue cells are FN. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 6
Runtime and power consumption measurements of the different methods on a Jetson Xavier board.

Method	Rt_{CPU} ms	Rt_{GPU} ms	Power Consumption W
P-SVM	**110**	–	**8.260**
RN++	3700	**320**	28.250
Cyl3D	12000	1020	27.570
PVKD	33600	3950	31.700

particular, we compared the *Geom*, *Geom-L*, and *Geom-A* variants, with and without PCA feature projection/dimensionality reduction. In all cases, at level 0 we start with *Geom*. Table 7 reports the results in terms of all metrics presented in Table 3. The **PCA** column reports the dimension of the transformed space, which in some cases could have the same dimensionality as the input space. For the sake of brevity, for $l > 0$ we only report the best-performing PCA dimension.

The best results at level $l = 0$ are obtained using a PCA projection on a vector space having the same dimension, 17. This model obtains the best Accuracy, F1, and IoU scores, and thus it is the most suitable for sharing its features to the finer levels. The runtime of each model is very low, usually ≤ 4 ms.

At level $l = 1$ the best performing model is *Geom-L* with a PCA projection to the same number of dimensions, being 17. Similar but worse results are obtained by methods *Geom*, projected to 17 dimensions, and *Geom-L*, without any PCA reduction. It is worth noting how the use of label inheritance and PCA projection both contribute substantially. Also, we can appreciate how all the geometric features are meaningful for the inference of the cells' labels. The runtime of each model is still very low, usually ≤ 7 ms.

At level 2 the best-performing model is *Geom-L* with a PCA projection to the same number of dimensions, 17. Very similar results are obtained by *Geom-L* without any PCA projection and by *Geom* followed by PCA projection to 17 dimensions. Methods exploiting label inheritance with or without PCA reduction show lower false positives, which is an important metric when dealing with Autonomous Driving applications, and better overall accuracy. These augmented methods outperform the pure-geometric methods. This can be explained by the fact that the predicted labels largely depend on the goodness of the

Table 7
Comparison of the different feature fusion variants. For a description of each method, refer to Section 4.4. For each one, we report the best results obtained using grid-search for parameter tuning.

	Model	PCA	Acc %	IoUN %	IoUT %	F1 %	Co %	TPR %	TNR %
Level 0	*Geom*	–	97.4	97.3	55.6	71.5	70.2	66.7	98.9
	Geom	**17**	**97.5**	**97.4**	**56.1**	**71.9**	**70.6**	66.7	**98.9**
	Geom	15	97.4	97.3	55.5	71.4	70.0	66.7	98.9
	Geom	13	97.4	97.3	56.1	71.9	70.5	**68.8**	98.8
Level 1	*Geom*	–	94.0	93.2	67.5	80.6	77.1	**81.6**	96.3
	Geom-L	–	94.1	93.2	67.3	80.4	76.9	79.7	96.6
	Geom-A	–	93.6	92.7	65.6	79.3	75.5	80.4	95.7
	Geom	17	94.1	93.2	67.3	80.5	77.0	80.3	96.6
	Geom-L	**17**	**94.2**	**93.4**	**67.8**	**80.8**	**77.4**	79.6	**96.8**
	Geom-A	34	93.8	93.0	66.6	80.0	76.3	80.4	96.2
Level 2	*Geom*	–	91.1	86.4	79.5	88.6	81.3	90.1	91.7
	Geom-L	–	91.5	87.2	80.0	88.9	82.1	88.3	**93.5**
	Geom-A	–	90.8	85.9	78.9	88.2	80.6	90.3	91.1
	Geom	17	91.3	86.7	79.7	88.7	81.6	89.6	92.4
	Geom-L	**17**	**91.7**	**87.4**	**80.4**	**89.2**	**82.4**	89.0	93.4
	Geom-A	51	90.6	85.6	78.6	88.0	80.3	**90.6**	90.6

coarser levels, so the model seems to learn to not trust entirely the earlier levels' prediction but instead to focus on the whole feature vector. In the end, the integration of the two types of inherited features results in better predictions than those of the pure-geometric approaches. The runtime of the models having less than 20 features is usually around 30–50 ms, and accordingly the best model has a runtime of 49 ms. Instead, we have noticed that the more features, the higher the runtime. In particular, the *Geom-A* methods can reach runtimes of more than 200 ms. In all experiments, we use the *Geom-L* variant for all levels $l > 0$.

5.1. Evaluation of the a-priori positive classifier

To evaluate the A-Priori Positive Classifier described in Section 3.5, we report in Table 8 the results of the best-performing SVM method

Table 8
Average results on SemanticKITTI Scenario 08 by P-SVM using the A-Priori technique described in Section 5.1. Runtime is improved (43 ms) at the cost of a slight drop in performance.

Level	Method	Acc	IoUN	IoUT	F1	Co	TPR	TNR
		%	%	%	%	%	%	%
0	*Geom*	97.5	97.4	56.1	71.9	70.6	66.7	99.0
1	*Geom-L*	94.2	93.4	67.7	80.8	77.3	80.6	96.7
2	*Geom-L*	91.5	87.0	80.0	88.9	82.0	89.3	92.9

when also applying this technique. Such a technique obviously requires well-performing coarse classifiers since a false positive cell has an avalanche effect on every finer cell. Results show a small degradation of the performance with respect to the full method with results reported in Table 4, and a runtime reduced to 43 ms.

6. Discussion

Our method has proven to be competitive against state-of-the-art deep learning approaches: it gives slightly worse results but with a lower runtime and power consumption. In relation to navigation risk, our approach yields slightly lower results in terms of the True Negative Rate compared to deep learning-based methods. However, as depicted qualitatively in Figs. 4 and 5, it exhibits minimal drawbacks while maintaining a high level of segmentation quality. Typically, critical regions are accurately classified, and any errors that occur can be corrected through post-processing techniques such as outlier removal filters or filters that leverage successive grid integration.

The scalability to multi-class segmentation represents the main limitation of our approach. While our method performs well in segmenting two classes, specifically *Traversable* and *Non-traversable*, it is not suitable for general semantic segmentation tasks with several target classes. In contrast, deep learning methods can be easily adapted to multi-class segmentation by simply adjusting the number of classes in the outer layer, while our method would require a new design process for handcrafted feature engineering.

7. Conclusions

In this paper, a real-time machine learning-based method for traversability analysis has been proposed. The method runs completely on CPU and combines geometric features and a pyramid-polar space representation that fuses multiple levels of features for fast and robust execution. An extensive evaluation on public datasets of urban and city-like scenarios has been done. The evaluation has been conducted against state-of-the-art methods that can be run on both CPUs and GPUs, and execution times were compared for both execution modes. The results demonstrate that the proposed method gets close to the state-of-the-art in terms of traversability analysis performance, while being faster and cheaper in terms of hardware resources since it runs completely on CPU without the need for high-end GPUs like the other methods. Power consumption measurements demonstrated that our method consumes over 70% less power compared to the deep learning approaches it was compared against. Additionally, performance evaluations done on a Jetson Xavier board have demonstrated that our method is well-suited for embedded devices with limited CPU capabilities, enabling quasi-real-time execution. In contrast, other methods fail to meet real-time constraints even when exploiting the provided embedded GPU. The paper proposes an ablation study to support the choices we made to select a suitable feature set. As a first step in future work, we plan to use also the remission values of the LiDAR scans, designing suitable features for them. Further improvements can be obtained with the integration of self-learning subsystems, in particular integrating image-based scene understanding techniques, to better utilize the color information, when available.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data used for the research is public. Our code has been shared and the link is inside the paper document.

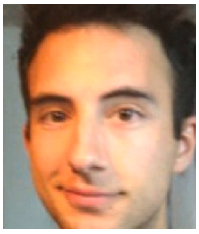
Acknowledgment

This project has been partially supported by FlexSight Srl company, Italy.

References

[1] E. Bekiaris, A.J. Amditis, M.C. Panou, DRIVABILITY: a new concept for modelling driving performance, Cogn. Technol. Work 5 (2003) 152–161.
[2] P. Papadakis, Terrain traversability analysis methods for unmanned ground vehicles: A survey, Eng. Appl. Artif. Intell. 26 (4) (2013) 1373–1385.
[3] H. Lee, W. Chung, A self-training approach-based traversability analysis for mobile robots in urban environments, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 3389–3394, http://dx.doi.org/10.1109/ICRA48506.2021.9561394.
[4] J.J. Leonard, J.P. How, S.J. Teller, M. Berger, S. Campbell, G.A. Fiore, L. Fletcher, E. Frazzoli, A.S. Huang, S. Karaman, O. Koch, Y. Kuwata, D.C. Moore, E. Olson, S.C. Peters, J. Teo, R. Truax, M.R. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M.E. Antone, R. Galejs, S. Krishnamurthy, J. Williams, A perception-driven autonomous urban vehicle, in: The DARPA Urban Challenge, 2009, pp. 163–230.
[5] S. Sivaraman, M.M. Trivedi, Dynamic probabilistic drivability maps for lane change and merge driver assistance, IEEE Trans. Intell. Transp. Syst. 15 (2014) 2063–2073.
[6] K. Kim, B. Kim, K.-J. Lee, B. Ko, K. Yi, Design of integrated risk management-based dynamic driving control of automated vehicles, IEEE Intell. Transp. Syst. Mag. 9 (2017) 57–73.
[7] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, P. Mahoney, Stanley: The robot that won the DARPA Grand challenge, J. Field Robotics 23 (2006) 661–692.
[8] L. Chen, J. Yang, H. Kong, Lidar-histogram for fast road and obstacle detection, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 1343–1348.
[9] D. Wolf, J. Prankl, M. Vincze, Fast semantic segmentation of 3D point clouds using a dense CRF with learned parameters, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 4867–4873, http://dx.doi.org/10.1109/ICRA.2015.7139875.
[10] B. Suger, B. Steder, W. Burgard, Traversability analysis for mobile robots in outdoor environments: A semi-supervised learning approach based on 3D-lidar data, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 3941–3946, http://dx.doi.org/10.1109/ICRA.2015.7139749.
[11] T. Shan, J. Wang, B. Englot, K. Doherty, Bayesian generalized kernel inference for terrain traversability mapping, in: A. Billard, A. Dragan, J. Peters, J. Morimoto (Eds.), Proceedings of the 2nd Conference on Robot Learning, in: Proceedings of Machine Learning Research, vol. 87, PMLR, 2018, pp. 829–838.
[12] M. Bellone, G. Reina, L. Caltagirone, M. Wahde, Learning traversability from point clouds in challenging scenarios, IEEE Trans. Intell. Transp. Syst. 19 (2018) 296–305.
[13] K.S. Viswanath, K. Singh, P. Jiang, P.B. Sujit, S. Saripalli, OFFSEG: A semantic segmentation framework for off-road driving, in: 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), 2021, pp. 354–359.
[14] N. Hirose, A. Sadeghian, M. Vázquez, P. Goebel, S. Savarese, Gonet: A semi-supervised deep learning approach for traversability estimation, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 3044–3051.
[15] N. Hirose, A. Sadeghian, F. Xia, R. Martin-Martin, S. Savarese, VUNet: Dynamic scene view synthesis for traversability estimation using an RGB camera, IEEE Robot. Autom. Lett. PP (2019) 1, http://dx.doi.org/10.1109/LRA.2019.2894869.
[16] S. Hosseinpoor, J. Torresen, M. Mantelli, D. Pitto, M. Kolberg, R. Maffei, E. Prestes, Traversability analysis by semantic terrain segmentation for mobile robots, in: 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), IEEE, 2021, pp. 1407–1413.
[17] E. Goh, J. Chen, B. Wilson, Mars terrain segmentation with less labels, in: 2022 IEEE Aerospace Conference (AERO), IEEE, 2022, pp. 1–10.

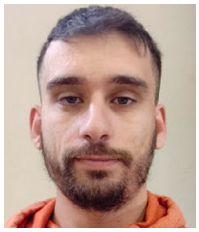
[18] K.S. Sikand, S. Rabiee, A. Uccello, X. Xiao, G. Warnell, J. Biswas, Visual representation learning for preference-aware path planning, in: 2022 International Conference on Robotics and Automation (ICRA), IEEE, 2022, pp. 11303–11309.
[19] B. Wu, A. Wan, X. Yue, K. Keutzer, Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 1887–1893.
[20] B. Wu, X. Zhou, S. Zhao, X. Yue, K. Keutzer, Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 4376–4382.
[21] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, M. Tomizuka, Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation, in: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Eds.), Computer Vision – ECCV 2020, Springer International Publishing, Cham, 2020, pp. 1–19.
[22] L. Caltagirone, S. Scheidegger, L. Svensson, M. Wahde, Fast LIDAR-based road detection using fully convolutional neural networks, in: 2017 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2017, pp. 1019–1024.
[23] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, H. Foroosh, Polarnet: An improved grid representation for online lidar point clouds semantic segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9601–9610.
[24] E.E. Aksoy, S. Baci, S. Cavdar, Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving, in: 2020 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2020, pp. 926–932.
[25] T. Cortinhal, G. Tzelepis, E. Erdal Aksoy, SalsaNext: Fast, uncertainty-aware semantic segmentation of LiDAR point clouds, in: International Symposium on Visual Computing, Springer, 2020, pp. 207–222.
[26] A. Milioto, I. Vizzo, J. Behley, C. Stachniss, RangeNet ++: Fast and accurate LiDAR semantic segmentation, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 4213–4220.
[27] C. Qi, H. Su, K. Mo, L.J. Guibas, PointNet: Deep learning on point sets for 3D classification and segmentation, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 77–85.
[28] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, A. Markham, Randla-net: Efficient semantic segmentation of large-scale point clouds, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11108–11117.
[29] H. Thomas, C.R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, L.J. Guibas, Kpconv: Flexible and deformable convolution for point clouds, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6411–6420.
[30] R. Cheng, R. Razani, E. Taghavi, E. Li, B. Liu, 2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 12547–12556.
[31] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, S. Savarese, Segcloud: Semantic segmentation of 3d point clouds, in: 2017 International Conference on 3D Vision (3DV), IEEE, 2017, pp. 537–547.
[32] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, D. Lin, Cylindrical and asymmetrical 3d convolution networks for lidar segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 9939–9948.
[33] D. Fusaro, E. Olivastri, D. Evangelista, P. Iob, A. Pretto, An hybrid approach to improve the performance of encoder-decoder architectures for traversability analysis in urban environments, in: 2022 IEEE Intelligent Vehicles Symposium (IV), 2022, pp. 1745–1750, http://dx.doi.org/10.1109/IV51971.2022.9827248.
[34] L. Landrieu, M. Simonovsky, Large-scale point cloud semantic segmentation with superpoint graphs, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4558–4567.
[35] Z. Liang, M. Yang, H. Li, C. Wang, 3D instance embedding learning with a structure-aware loss function for point cloud segmentation, IEEE Robotics Autom. Lett. 5 (3) (2020) 4915–4922, http://dx.doi.org/10.1109/lra.2020.3004802.
[36] Z. Liang, M. Yang, L. Deng, C. Wang, B. Wang, Hierarchical depthwise graph convolutional neural network for 3d semantic segmentation of point clouds, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 8152–8158.
[37] D. Paz, H. Zhang, Q. Li, H. Xiang, H.I. Christensen, Probabilistic semantic mapping for urban autonomous driving applications, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 2059–2064.
[38] T. Guan, Z. He, D. Manocha, L. Zhang, TTM: Terrain traversability mapping for autonomous excavator navigation in unstructured environments, 2021, ArXiv abs/2109.06250.
[39] W. Zhang, S. Lyu, C. Yao, F. Xue, Z. Zhu, Z. Jia, Analysis of robot traversability over unstructured terrain using information fusion, in: 2022 International Conference on Advanced Robotics and Mechatronics (ICARM), IEEE, 2022, pp. 413–418.
[40] X. Yan, J. Gao, C. Zheng, C. Zheng, R. Zhang, S. Cui, Z. Li, 2Dpass: 2d priors assisted semantic segmentation on lidar point clouds, in: European Conference on Computer Vision, Springer, 2022, pp. 677–695.
[41] L. Deng, M. Yang, Z. Liang, Y. He, C. Wang, Fusing geometrical and visual information via superpoints for the semantic segmentation of 3D road scenes, Tsinghua Sci. Technol. 25 (4) (2020) 498–507, http://dx.doi.org/10.26599/TST.2019.9010038.
[42] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, S. Pu, Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 16024–16033.
[43] Z. Zhu, N. Li, R. Sun, H. Zhao, D. Xu, Off-road autonomous vehicles traversability analysis and trajectory planning based on deep inverse reinforcement learning, in: 2020 IEEE Intelligent Vehicles Symposium (IV), 2020, pp. 971–977.
[44] C. Jung, D.H. Shim, Incorporating multi-context into the traversability map for urban autonomous driving using deep inverse reinforcement learning, IEEE Robot. Autom. Lett. 6 (2) (2021) 1662–1669, http://dx.doi.org/10.1109/LRA.2021.3059628.
[45] D. Fusaro, E. Olivastri, D. Evangelista, M. Imperoli, E. Menegatti, A. Pretto, Pushing the limits of learning-based traversability analysis for autonomous driving on CPU, in: I. Petrovic, E. Menegatti, I. Marković (Eds.), Intelligent Autonomous Systems, vol. 17, Springer Nature Switzerland, Cham, 2023, pp. 529–545.
[46] C.M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag, Berlin, Heidelberg, 2006.
[47] F. Fan, J. Xiong, M. Li, G. Wang, On interpretability of artificial neural networks: A survey, IEEE Trans. Radiat. Plasma Med. Sci. 5 (2021) 741–760.
[48] N. Cristianini, J. Shawe-Taylor, An Introduction To Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000, http://dx.doi.org/10.1017/CBO9780511801389.
[49] I.T. Jolliffe, J. Cadima, Principal component analysis: a review and recent developments, Phil. Trans. R. Soc. A 374 (2016).
[50] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, J. Gall, Semantickitti: A dataset for semantic scene understanding of lidar sequences, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 9296–9306.
[51] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? The KITTI vision benchmark suite, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3354–3361.
[52] Y. Hou, X. Zhu, Y. Ma, C.C. Loy, Y. Li, Point-to-voxel knowledge distillation for LiDAR semantic segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 8479–8488.
[53] P. Xiao, Z. Shao, S. Hao, Z. Zhang, X. Chai, J. Jiao, Z. Li, J. Wu, K. Sun, K. Jiang, Y. Wang, D. Yang, PandaSet: Advanced sensor suite dataset for autonomous driving, in: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), 2021, pp. 3095–3101.
[54] N. Corporate, Jetson AGX Xavier Series, 2023, https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/. (Accessed: 2023-01-17).



Daniel Fusaro is a Ph.D. Student in Robotics and Artificial Intelligence at the University of Padua, Dept. of Information Engineering. He received his Master degree in Computer Engineering from University of Padua in October 2021 with the thesis "Machine Learning-based Traversability Analysis for Autonomous Driving". He has been involved in several robotics projects collaborating also with private companies, including a research scholarship for the development of an Underwater Pose Estimation system. His main research covers Computer Vision, Autonomous Driving and Deep Learning.

Emilio Olivastri is Ph.D. Candidate in Robotics and Artificial Intelligence at the University of Padova, Dept. of Information Engineering. After his bachelor degree in Information Engineering at the University of Padova in 2019 he entered the master degree in Computer Engineering and graduated in 2021 with a master thesis on "Multi-sensor State Estimation for Autonomous Underwater Vehicles". Olivastri's 's main research topic is Simultaneous Localization and Mapping(SLAM), with focus on robust pose graph optimization and outlier detection.

Ivano Donadi is a research fellow in Robotics and Artificial Intelligence at the University of Padova, Dept. of Information Engineering. After his bachelor degree in Information Engineering At the University of Padova in 2020 he entered the master degree in Computer Engineering and graduated in 2022 with a master thesis on "A Multi-view Pixel-wise Voting Network for 6DoF Pose Estimation". His main research focus is visual pose estimation for stereo cameras setups.



Daniele Evangelista is Ph.D. Candidate in Robotics and Artificial Intelligence at the University of Padova, Dept. of Information Engineering. After his bachelor degree in Informatics Engineering at the University of Cassino in 2014 he entered the master degree in Robotics and Artificial Intelligence at the University of Sapienza in Rome and graduated in 2018 with a master thesis on object detection and localization techniques for industrial robotics. He has been team leader of the RoboCup@Work team of Sapienza from 2016 to 2018. In 2019 he joined the Ph.D. programme at the University of Padova and he is currently candidate for the final dissertation of his Ph.D. Thesis that is focused on inspection software tools and calibration techniques for industrial robotics. During the Ph.D. programme Daniele has been involved in several European projects, among the others, SPIRIT a H2020 project for which he served as main research developer for the University of Padova. He is currently author and co-author of 17 international scientific publications. Daniele has been CEO of Alibabyte Software House from 2013 to 2016. Daniele is currently CEO of FlexSight, a startup company operating in the field of machine vision for robotics, multi-view 3D reconstruction and autonomous driving for vehicles and mobile robots.

Emanuele Menegatti is Full Professor of the School of Engineering at Dept. of Information Engineering of University of Padova since 2017. After his graduation in Physics in 1998, he received his MsC in AI & Robotics from the University of Edinburgh (UK) in 2000 and his Ph.D. in Computer Engineering in 2003 from Univ. of Padua. Menegatti's main research interests are in the field of Robot Perception. In particular, he is working on neurorobotics, RGB-D people tracking for camera network, and service robotics. He is teaching master courses on "Intelligent Robotics" and bachelor course in "Computer Architecture" and a course for school teachers on "Educational Robotics". He was coordinator of the FP7 FoF-EU project "Thermobot" and local principal investigator for the European Projects "3DComplete", "FibreMap", "Focus", "eCraft2Learn" and "SPIRIT". Menegatti also served as Project Reviewer for the European Commission in FP7 and H2020. He is author of more than 50 publications in international journals and more than 120 publications in international conferences. In 2005, Menegatti founded IT+Robotics, a Spin-off company of the Univ. of Padua, active in the field of industrial robot vision, machine vision for quality inspection, automatic off-line robot programming. In 2014, he founded EXiMotion a startup company active in the field of educational robotics and service robotics. His team, the WHI Team (University of Padova), won the Cybathlon BCI Series in September 2019.

Alberto Pretto is an associate professor at the Department of Information Engineering of the University of Padua. Pretto's main research topics are autonomous robotics and perception. He received a M.Sc. and a Ph.D. degree in Computer Engineering from the University of Padua in 2003 and 2009, respectively. His career includes both academic and professional experiences in IT (Information Technology) companies. Between 2004 and 2005 he worked as an analyst programmer at Padova Ricerche Scpa, a company operating in the field of university to industry technology transfer. In 2005 he was one of the founders of IT+Robotics Srl, a spin-off of the University of Padua that works in the fields of industrial robotics and machine vision, where he served as CEO until 2008. Between 2010 and 2013, Pretto was a Post-Doc at the Intelligent Autonomous Systems Laboratory of the University of Padua, and between 2013 and 2018 he was a fixed-term assistant professor at the Department of Computer, Automation and Management Engineering, Sapienza University of Rome. During his academic career, he spent a total of 14 months as a visiting researcher in three research laboratories: Autonomous Intelligent Systems laboratory of the University of Freiburg (Germany), UCLA Vision Lab of the University of California, Los Angeles (USA), and Photogrammetry and Robotics laboratory of the University of Bonn (Germany). In 2019 with other partners he founded FlexSight Srl, a start-up operating in the field of computer vision and artificial intelligence on embedded devices, where he served as CEO until August 2020.