A Review of panoptic segmentation for mobile mapping point clouds

Review Article

Author(s):
Xiang, Binbin; Yue, Yuanwen; Peters, Torben; Schindler, Konrad

A Review of panoptic segmentation for mobile mapping point clouds

Binbin Xiang *, Yuanwen Yue, Torben Peters, Konrad Schindler

Photogrammetry and Remote Sensing, ETH Zürich, 8093, Zürich, Switzerland

ARTICLE INFO

ABSTRACT

3D point cloud panoptic segmentation is the combined task to (i) assign each point to a semantic class and (ii) separate the points in each class into object instances. Recently there has been an increased interest in such comprehensive 3D scene understanding, building on the rapid advances of semantic segmentation due to the advent of deep 3D neural networks. Yet, to date there is very little work about panoptic segmentation of outdoor mobile-mapping data, and no systematic comparisons. The present paper tries to close that gap. It reviews the building blocks needed to assemble a panoptic segmentation pipeline and the related literature. Moreover, a modular pipeline is set up to perform comprehensive, systematic experiments to assess the state of panoptic segmentation in the context of street mapping. As a byproduct, we also provide the first public dataset for that task, by extending the NPM3D dataset to include instance labels. That dataset and our source code are publicly available.[1] We discuss which adaptations are need to adapt current panoptic segmentation methods to outdoor scenes and large objects. Our study finds that for mobile mapping data, KPConv performs best but is slower, while PointNet++ is fastest but performs significantly worse. Sparse CNNs are in between. Regardless of the backbone, instance segmentation by clustering embedding features is better than using shifted coordinates.

1. Introduction

Semantic segmentation and instance segmentation are two core tasks of scene understanding. Given densely sampled observations (e.g., pixels of an image, points of a point cloud), the goal of semantic segmentation is to assign a semantic category label to each observation. Instance segmentation aims to separate individual object instances, which only makes sense for categories that form clearly delineated objects. The joint task, to generate a complete, coherent scene interpretation in terms of semantic categories *and* individual objects, has been termed *panoptic segmentation* (Kirillov et al., 2019). In urban mobile mapping, panoptic segmentation has the natural application to structure raw 3D point clouds into semantically meaningful objects and surfaces, which are useful for higher-level tasks such as building inventories of street furniture, or enabling outdoor mobile robots. The aim of this paper is to establish a state of the art for the panoptic segmentation of mobile mapping point clouds, and to review the most promising algorithmic building blocks for that purpose.

In panoptic segmentation, the compact, countable objects (such as pedestrians and cars) are often called "things", whereas the amorphous and uncountable regions (such as the road surface) are called "stuff". For "stuff" categories panoptic segmentation is the same as semantic segmentation, as they do not form instances: each point is assigned a semantically defined category label. Whereas for "things" the model must assign a semantic label and, additionally, a unique instance label that distinguishes it from other instances, but does not have a semantic meaning — instance IDs are arbitrary and all perturbations of the labels are equally valid. Fig. 1 illustrates the difference between semantic, instance, and panoptic segmentation.

As shown in Fig. 2, panoptic segmentation is nowadays usually formulated as a combined classification and clustering task and solved with deep neural networks. The general architecture is to first extract a per-point feature encoding with a *backbone network*, then feed the encoding into two parallel branches (heads), one that predicts semantic class probabilities and another one that further transforms the points such that points on the same instance form compact clusters, achieving both task simultaneously based on shared features. Historically, instance segmentation has been developed in 2D image analysis, as an add-on to semantic segmentation. Consequently, the focus has been on the design of the instance branch, without questioning the backbone. No systematic studies exist that investigate how different 3D backbones support point cloud instance segmentation. Moreover, panoptic segmentation of outdoor scenes has almost exclusively been studied for sparse, sequential scan data from autonomous driving scenarios like SemanticKITTI (Behley et al., 2019, 2021) or nuScenes (Caesar et al.,
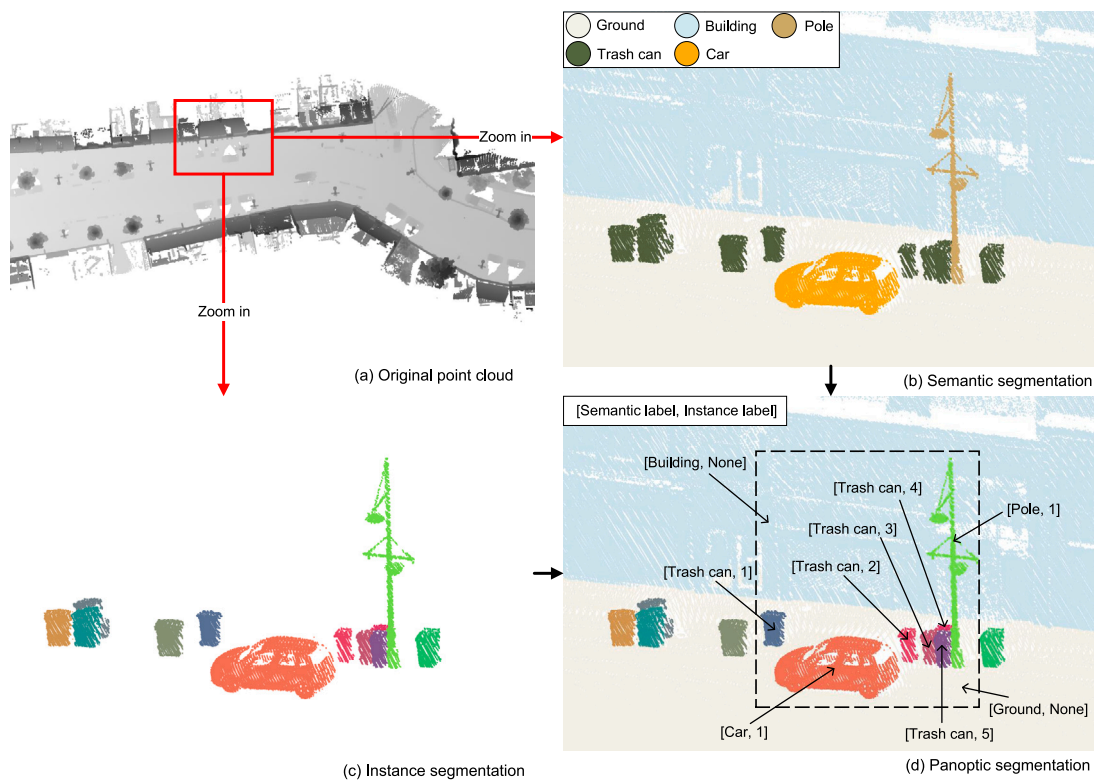

Fig. 1. An input point cloud (a), its semantic segmentation (b, per-point semantic labels), instance segmentation (c, per-point instance labels, each label is assigned a random color), and panoptic segmentation (d, per-point semantic labels and, where applicable, instance labels). In the example, the categories "ground" and "building" are considered "stuff" and not separated into instances.
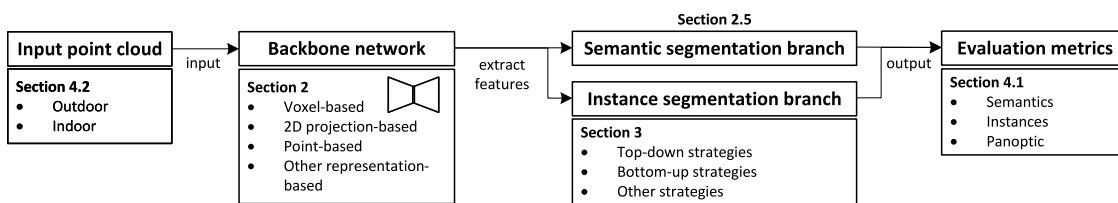
(a) High-level sketch of a typical deep neural network pipeline for panoptic segmentation, and references to the corresponding sections of the present review.
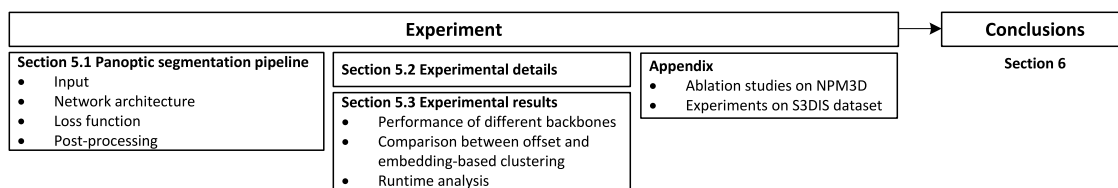


(b) Experiments and conclusions

Fig. 2. Complete overview of the paper.

2020; Fong et al., 2022), often in a real-time scenario. It appears that panoptic segmentation of dense, mapping-grade point clouds has not been investigated much.

There are a number of review papers about deep learning for 3D point clouds. Some review papers attempt a comprehensive overview of different point cloud analysis tasks (Ioannidou et al., 2017; Liu et al., 2019; Bello et al., 2020; Li et al., 2020; Lu and Shi, 2020; Guo et al., 2021), predominantly scene-level classification, segmentation, and object detection. Others concentrate on a single analysis task (Griffiths and Boehm, 2019; Wu et al., 2021; He et al., 2021b; Zamanakos et al., 2021; Zhao et al., 2021; Burume and Du, 2021; Jhaldiyal and Chaudhary, 2022; Diab et al., 2022; Alaba and Ball, 2022). For example, several papers review 3D object detection methods for autonomous driving (Wu et al., 2021; Zamanakos et al., 2021;

Alaba and Ball, 2022), and Burume and Du (2021) focus on instance segmentation in that scenario. Zhao et al. (2021) did a survey to explore the effects of different traditional point cloud clustering algorithms for LiDAR panoptic segmentation task. To date, there has not been a review that specifically addresses the panoptic segmentation task for dense, mapping-grade 3D LiDAR point clouds. The present work aims to fill that gap. Besides systematically introducing the task and discussing important representative works, it also provides a unified experimental test bed and comparative results for reference.

Following the literature about deep learning for point clouds (Griffiths and Boehm, 2019; Li et al., 2020; Wu et al., 2021; He et al., 2021b; Zamanakos et al., 2021; Diab et al., 2022; Alaba and Ball, 2022), 3D backbone networks are classified into four classes: voxel-based, point-based, 2D projection-based, and other presentations, which will be
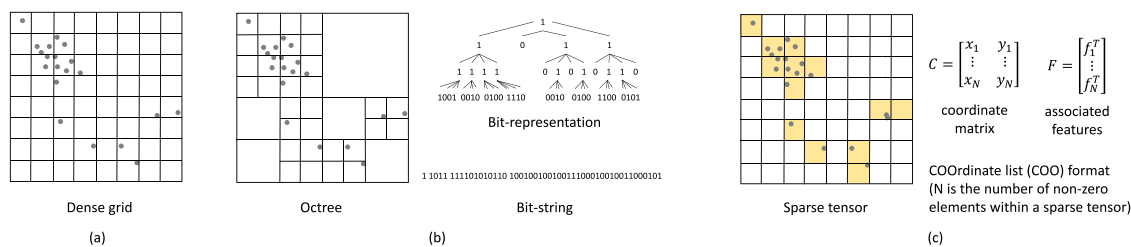
Fig. 3. Voxel-based representations. Three voxel-based data structures for point clouds are compared: dense grid, octree, and sparse tensor. For clarity, they are illustrated in 2D space. (a) Dense grids correspond to explicitly storing all voxels as a dense tensor (in 2D, a matrix). For the octree, like the one shown in (b), space is hierarchically subdivided, splitting only voxels that contain points. The resulting tree can be efficiently represented by a bit-string. The sparse tensor uses the COO format as shown in (c), with one matrix that stores the coordinates of occupied voxels and another one storing the feature values only for those voxels (yellow areas). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

discussed in Section 2. For the further investigation of dense outdoor point cloud panoptic segmentation, the paper focuses on three representative backbone networks: PointNet++ (Qi et al., 2017b), Sparse CNN (Choy et al., 2019), and KPConv (Thomas et al., 2019). The choice is motivated by the following considerations. First, PointNet++ and Sparse CNN are currently the most widely used point cloud backbones (see Table 2). Second, KPConv and Sparse CNN are particularly successful backbones for segmentation tasks, with KPConv achieving state-of-the-art performance for semantic segmentation on the NPM3D dataset (Roynard et al., 2018), while Sparse CNN is the most widely used backbone for state-of-the-art instance segmentation of indoor point clouds (Jiang et al., 2020b; Chen et al., 2021; Vu et al., 2022). The three types of backbones are described in detail in Section 2.

Section 3 of the paper distinguishes top-down and bottom-up strategies, like other articles about 3D point cloud instance segmentation (He et al., 2021b; Burume and Du, 2021). Additionally, the present paper introduces and summarizes the two most commonly used strategies for bottom-up methods (Table 2 and Section 5.1.3), and compares their performance in the context of outdoor panoptic segmentation (Section 5.3.2).

So far, there is no outdoor mobile mapping dataset tailored to panoptic segmentation, as existing datasets lack instance labels. Existing outdoor datasets of LiDAR point clouds with instance labels are dominated by the autonomous driving setting. I.e., the points are comparatively sparse, moreover instance labels are typically limited to pedestrians and vehicles (see Section 4.2.1), whereas for mapping a wider variety of object classes should be instantiated, such as trees, poles, street lamps and so on. For the present work, this meant on the one hand that a new dataset had to be generated, which we do by adding instance labels to an existing benchmark, see Section 4.2.1. On the other hand, the lack of a common dataset also meant that for the review it was not possible to collect the results of different methods from the literature. Instead, a common code base for different methods was implemented and applied on the new dataset to enable informative and fair comparisons. In summary, the *main contributions* of the present paper are:

- A detailed literature review about 3D point cloud panoptic segmentation, grouped into four sub-topics: datasets, 3D backbone networks for semantic segmentation, strategies for instance segmentation, and evaluation metrics. See also Fig. 2.
- An experimental evaluation and comparison in the outdoor mobile mapping setting. The evaluation concentrates on the bottom-up segmentation and grouping strategy (see Section 3), which so far proved to be the most successful approach and dominates the leaderboards for indoor datasets. To that end, a modular panoptic segmentation pipeline (Section 5.1) was created that allows one to combine different feature extraction and segmentation strategies. Extensive experiments are run on the NPM3D dataset (Roynard et al., 2018) with three different, representative backbones and two representative instance clustering approaches (Section 5.3).

- The instance-annotated dataset and all source code will be published, so as to expedite further research and promote the development of 3D panoptic segmentation. So far, there do not seem to be any other works that systematically investigate modern deep learning methods for panoptic segmentation of mobile mapping point clouds.

2. Backbone networks for 3D point cloud analysis

Traditionally, 3D point cloud analysis has relied on manually designed features that capture the local point distribution (and thus, the surface shape) around a point (Niemeyer et al., 2012; Li et al., 2016; Zhu et al., 2017). This works well for categories with sufficiently obvious, distinctive features, but much time and effort are required to design features, select appropriate neighborhoods, etc. Such approaches no longer reach state-of-the-art performance, mostly because it is not clear how to manually design a good representation of contextual relations over larger contexts. Furthermore, the handcrafted features tuned for a point cloud with specific sensing characteristics typically fail to handle other types of scenes or other sensor settings, meaning that the tedious manual feature engineering must be repeated.

At present, deep learning is the standard technology to automatically extract discriminative, robust feature representations from raw data. Remarkable results have been achieved for 2D image classification, semantic segmentation, object detection, and more (LeCun et al., 2015). Motivated by those results and a growing number of public 3D point cloud datasets, researchers have developed powerful deep learning algorithms to also extract features for 3D point cloud analysis (Ioannidou et al., 2017). These existing 3D "backbone" networks can be grouped into three main schools, namely voxel-based networks, 2D projection-based networks, and point-based networks. Representative examples are summarized in Table 1.

2.1. Voxel-based networks

The points in a point cloud are generally unstructured and have irregular density. These characteristics are quite different from images, where relationships between pixels can be captured with regular, discrete convolution kernels. The simplest and most direct strategy to apply deep learning to 3D point clouds is to voxelise the underlying point data and use 3D Convolutional Neural Networks (CNNs) (Maturana and Scherer, 2015) (Figs. 3(a) and 4(a)). Perhaps the first attempt to use CNNs for semantic segmentation of outdoor point clouds was Huang and You (2016). However, when moving to a voxel representation the computational cost and, more importantly, the memory demand increases cubically with the resolution, thus limiting the practical usability of 3D CNNs (Riegler et al., 2017). To bring down the memory requirements, hierarchical and/or sparse voxel representations have been developed. Hierarchical volumetric models (Riegler et al., 2017; Wang et al., 2017) use data structures like octrees to focus memory allocation and computation on the relevant regions where there are
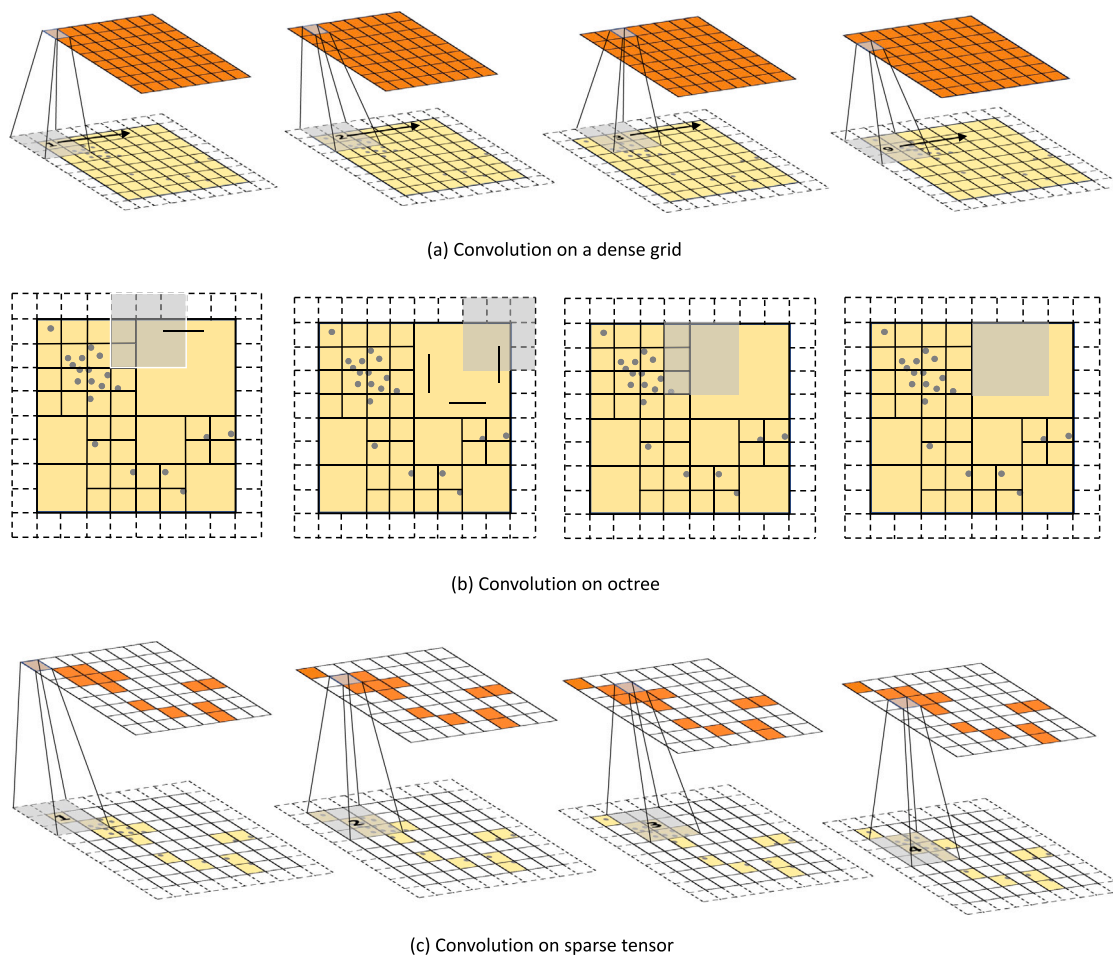
(a) Convolution on a dense grid

(b) Convolution on octree

(c) Convolution on sparse tensor

Fig. 4. Illustration of voxel-based convolutions in different formats. For clarity, they are shown in 2D space. Arrows indicate the direction in which the convolution kernel is moved, numbers denote the kernels' sequence order. (a) On a full, dense voxel grid standard discrete convolutions are applied in 3D. (b) The octree evaluates the kernel only along the borders of large voxels, where the support changes due to adjacent voxels. (c) On sparse tensors convolutions are evaluated only at non-empty locations.

points (Figs. 3(b) and 4(b)). The same principle, to omit the vast amount of empty 3D space, is also the basis of methods that employ sparse tensors (Graham et al., 2018; Choy et al., 2019; Tang et al., 2020) and associated functions that only perform calculations for non-empty voxels (Figs. 3(c) and 4(c)). The Minkowski Engine (Choy et al., 2019) and the SparseConvNet library (Graham et al., 2018) are two popular software frameworks that support all standard neural network layers like convolution, pooling and upsampling, for sparse tensors. Figs. 3 and 4 illustrate how dense voxel grids, octrees and sparse tensors, respectively, store point clouds and compute convolutions.

2.2. 2D projection-based networks

To avoid the complications of a 3D voxel space, another strategy are 2D projection-based methods. Their basic idea is to convert the original 3D point cloud into a regular 2D raster, which can then be treated with standard 2D CNNs. Variants include single- or multi-view perspective projections (Lawin et al., 2017; Kalogerakis et al., 2017), parallel (Tatarchenko et al., 2018) as well as spherical projections (Wu et al., 2018, 2019). This type of approach tends to work well when the underlying assumptions are met, e.g., well-behaved surfaces with an unambiguous normal/tangent (Tatarchenko et al., 2018) or individual scans (respectively, range images) that, in actual fact, induce a 2D parametrization.

2.3. Point-based networks

To side-step the representation change altogether, point-based networks directly work on irregular point clouds (Qi et al., 2017a,b; Hua et al., 2018; Wang et al., 2019a,c; Thomas et al., 2019). The earliest such method was PointNet (Qi et al., 2017a), where a per-point multiple-layer perceptron (MLP) is combined with max-pooling to achieve a global feature invariant to permutations of the points (Fig. 5(a)). PointNet++ (Qi et al., 2017b) builds upon PointNet and adds a step-wise hierarchical aggregation to retain more of the spatial layout (Fig. 5(b)). A different method, more in the spirit of CNNs, is the kernel point convolution (KPConv, Thomas et al., 2019), which approximates the 3D convolution in continuous space by interpolation and also hierarchically aggregates multi-scale features to capture both local and global information (Fig. 5(c)). It has shown excellent performance on outdoor data, like NPM3D (Roynard et al., 2018) and Semantic3D (Hackel et al., 2017).

2.4. Other representations

Besides the three main groups described so far, there have been a few other attempts to process 3D data with reasonable memory footprint. These include permutohedral lattices (Su et al., 2018; Rosu et al., 2019), offline grouping into irregular patches, combined with graph convolutional networks (Landrieu and Simonovsky, 2018), and hybrid models (Dai and Nießner, 2018; Jaritz et al., 2019).
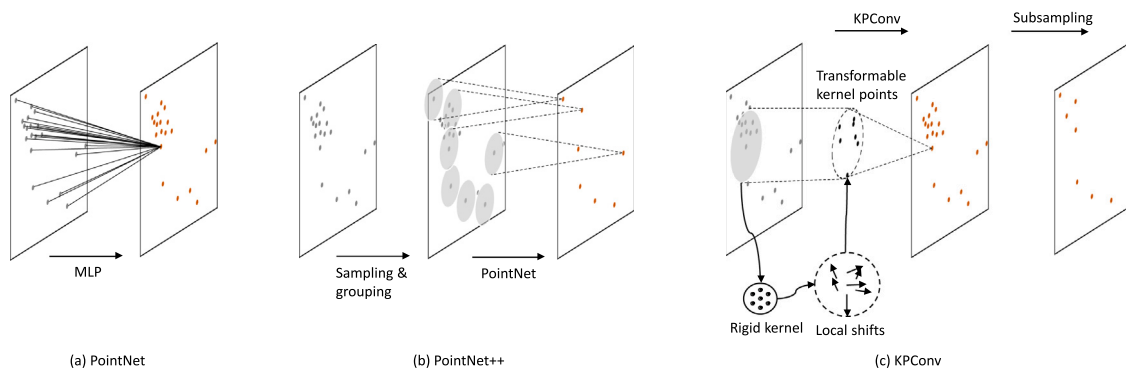
Fig. 5. Illustration of three point-based backbones. For clarity, the 2D versions are depicted. (1) In PointNet, each input point is processed separately through a shared MLP. All points are aggregated through global pooling. (2) PointNet++ improves upon PointNet with sampling and grouping operations to create local point sets and gradually aggregate their features. As a result, PointNet++ generates subsampled point clouds with enhanced representations, enabling efficient processing of large-scale data while preserving important details and global context. (3) KPConv extends traditional convolutions by defining the kernel weights at arbitrary support points, which additionally can be shifted locally to adapt to the local point distribution. The computation involves aggregating information from neighboring points within a fixed receptive field, enabling KPConv to extract local features. Additionally, KPConv incorporates subsampling methods to obtain multi-scale features.

2.5. Semantic segmentation of point clouds

Semantic segmentation assigns each point to a semantic class, without distinguishing different instances within the class. This is accomplished by adding a segmentation "head" to the backbone, which outputs a class distribution for each individual point. Overall, this results in an encoder–decoder structure: the backbone encodes the input into a latent representation optimized for classification, and the classification head decodes that representation into class scores. For voxel-based, projection-based and sparse tensor-based methods, the decoder consists of a sequence of transposed convolutions. For PointNet-style methods, the per-point features and the latent representation over a larger spatial context are simply concatenated and decoded with an MLP.

3. Strategies for instance segmentation

Once reliable backbones for 3D point cloud analysis were available, a natural next step was to also look into instance segmentation. This Section reviews the literature on that topic. At a conceptual level, existing strategies for instance segmentation can be grouped into two main types: top-down and bottom-up strategies. Fig. 6 illustrates the difference between the two strategies.

3.1. Top-down strategies

Top-down methods (Yi et al., 2019; Yang et al., 2019; Zhang et al., 2020b; Liu et al., 2020), sometimes also termed proposal-based methods, resemble the Mask R-CNN (He et al., 2017) approach to object detection (Fig. 6(a)). Instance segmentation is implemented as a sequence of two steps. First, localize a 3D bounding box of each instance, with a standard object detector that constructs candidate boxes and classifies them. Then predict per-point binary instance masks within each bounding box to obtain the final instance segmentation. Instead of directly regressing 3D bounding boxes, (Yi et al., 2019) design a generative model named Generated Shape Proposal Network (GSPN) to produce 3D proposals with a high probability of being compact objects. GSPN is integrated into an instance segmentation pipeline called Region-based PointNet (R-PointNet) that can flexibly refine proposals generated with GSPN into segmentation masks. Unlike R-PointNet, which needs two-stage training, 3D-BoNet (Yang et al., 2019) is an end-to-end network where bounding box detection and instance masking can be trained simultaneously. It also avoids intensive post-processing (non-maximum suppression) to prune dense object proposals. For large-scale outdoor point clouds, Zhang et al. (2020b) proposed a simple top-down instance segmentation that encodes a 2D grid-level feature

representation in a birds-eye view and predicts the planimetric object center and the height limits for each foreground grid location. Then it merges locations with similar object centers into instances.

Proposal-based methods are less prone to over-segmenting instances, thanks to the explicit object detector. The flip-side is that they cannot recover missed detections, and are challenged by overlapping detections. Typically their multi-stage inference and post-processing also lead to comparatively slow runtimes.

3.2. Bottom-up strategies

Bottom-up instance segmentation strategies (also called proposal-free or clustering-based methods, as shown in Fig. 6(b)) tend to outperform top-down methods, and have thus become popular (Wang et al., 2018; Liu and Furukawa, 2019; Liang et al., 2019; Wang et al., 2019b; Pham et al., 2019; Elich et al., 2019; Lahoud et al., 2019; Han et al., 2020; Zhang et al., 2020a; Wang et al., 2020). Their basic principle is to map points to a discriminative representation space where points from the same instance have similar features, whereas points on different objects have dissimilar features. Instances are then retrieved by clustering in that space (Comaniciu and Meer, 2002; Campello et al., 2013; Neven et al., 2019; Hong et al., 2021), as illustrated in Fig. 6(b). In the perhaps earliest attempt, SGPN (Wang et al., 2018) learns a similarity matrix that indicates pairwise similarities between points, then merges points with high similarity with a heuristic grouping algorithm. The size of the similarity matrix grows quadratically with the number of input points, which limits the size of point clouds that can be processed. In order to simultaneously segment instances and semantic object categories, several methods (Wang et al., 2019b; Pham et al., 2019; Elich et al., 2019; Lahoud et al., 2019; Han et al., 2020; He et al., 2020; Zhang and Wonka, 2021) employ parallel decoder branches for the point classification and the discriminative instance embedding. There are a number of variants of this general idea, e.g., Wang et al. (2019b) introduce a linking module between the two decoder branches to exploit synergies between semantic and instance segmentation. Pham et al. (2019) employ a multi-value conditional random field model to jointly optimize instance and semantic labels. Lahoud et al. (2019) estimate vectors pointing to the potential instance centers, as in the generalized Hough transform, to support the subsequent clustering step. In addition to the 3D offset vector, OccuSeg (Han et al., 2020) also learns occupancy signals, which can guide the subsequent graph-based clustering towards better instance segmentation. Elich et al. (2019) integrate 2D birds-eye-view information into a network for joint 3D semantic and instance segmentation, in order to better exploit global context. Zhang and Wonka (2021) introduce a probabilistic embedding instead of a deterministic one, and also propose a new loss function

Table 1
Summary of different types of representative backbone networks for 3D point clouds.

	Representatives	Methodology	Advantages	Disadvantages
Voxel-based	3D CNN (Maturana and Scherer, 2015; Huang and You, 2016)	Encode 3D shapes by occupancy voxels.	- *Simplicity:* Direct application of 3D CNNs to 3D point clouds in voxel grids. - *Searching* for neighboring points takes only $\mathcal{O}(1)$ operations.	- *High computational cost* due to large number of voxels. Requirements increase cubically by $\mathcal{O}(n^3)$ with resolution. - *Low efficiency* due to high sparsity of the voxel grid. - *Quantization errors* due to voxelization of the point cloud.
	- OctNet (Riegler et al., 2017) - O-CNN (Wang et al., 2017)	Octree instead of grid representation. Hierarchically divide 3D space into a set of unbalanced octants based on point cloud density.	- *Much more efficient:* Storage and computational time reduces to $\mathcal{O}(n^2)$ compared to grid-based methods. - *Improved performance* for high-resolution 3D data when compared to dense voxel-based (3D CNN) models.	- *Slower search time* compared to voxel grids with $\mathcal{O}(log(n))$ due to superimposed tree structure. - *Input size limitation.* Due to the restriction of the maximal depth of an octree (i.e., three) (Xiang et al., 2019), hindering proper grid size for large-scale applications.
	Sparse tensor CNN (Graham et al., 2018; Choy et al., 2019)	An N-dimensional extension of a sparse matrix that only saves information on the non-empty region of the space.	- *Computational efficiency:* Fast computation due to reduced computations on unoccupied elements in the voxel grid. - *Memory efficiency:* 3D sparse tensors enable the representation of large sparse arrays in a compact format, reducing memory requirements. - *Highly optimized frameworks.* Minkowski Engine (Choy et al., 2019) and SparseConvNet, which support all standard CNN operations (Graham et al., 2018)	*Quantization errors.* All voxel-based methods lose some geometric properties of 3D objects, particularly the intrinsic characteristics of patterns and surfaces.
2D projection-based	- Multi views (Lawin et al., 2017) - Bird's-eye view (Zhang et al., 2018) - Spherical projection (Wu et al., 2018, 2019)	Projects 3D point clouds into images and applies 2D CNNs to them.	- *2D network utilization:*. Fully compatible with well-established 2D CNNs and pretrained networks on image datasets. - *Efficiency:* Same requirements as 2D CNNs with $\mathcal{O}(n^2)$ time and space complexity. Suitable for real-time applications.	- *Occlusions and Distortions:* Many projection-related errors appear, including various types of occlusions and projective distortions. - *Hyperparameters:* Adds additional hyperparameters for camera and poses.
Point-based	PointNet (Qi et al., 2017a)	Learns to project 3D points together with point attributes independently into a common feature space; Features are then aggregated using pooling for further processing.	- *No preprocessing required:* Works directly on point clouds and is invariant to point ordering. - *Efficiency:* Very low time and space complexity of $\mathcal{O}(n)$, which increases linear with the number of points.	*Inaccurate:* PointNet cannot capture fine-grained patterns and local structure, especially in large scenes. Independent feature representation of entire point cloud is aggregated into a single vector, resulting in a loss of information.

for the clustering step, with which they achieved good performance on the PartNet dataset (Mo et al., 2019). In order to mitigate imbalances in the data, which tend to harm the instance segmentation for rare categories, He et al. (2020) propose a memory-augmented network to memorize representative patterns.

Recently, a number of studies have applied the bottom-up approach to outdoor dataset (Milioto et al., 2020; Hong et al., 2021; Zhou et al., 2021; Zhao et al., 2021; Li et al., 2021). In general, these are variants of the two-branch architecture described above. For instance, Milioto et al. (2020) and Li et al. (2021) used spherical projection to implement a real-time, panoptic segmentation algorithm for the autonomous driving setting, while Panoptic-PolarNet (Zhou et al., 2021) used a polar bird's eye view. Their instance branch directly regresses the instance's center. DS-Net (Hong et al., 2021) utilizes a dynamic shift module that can automatically adjust the kernel function to different point densities and instance sizes. All these works target autonomous driving scenarios with sparse, vehicle-mounted panoramic LiDAR, in particular the popular SemanticKITTI (Behley et al., 2019) and nuScenes (Caesar et al., 2020) datasets. To our knowledge, none of them has been tuned to panoptic segmentation of dense, mobile-mapping type point clouds, neither have they been evaluated on such datasets, most prominently NPM3D (Roynard et al., 2018).

Compared to the 2D image case, 3D instance segmentation is not as mature, and there is still ample room for improvement. Even though discriminative per-point embedding features followed by clustering have emerged as the mainstream approach, there still are a number of open questions. One elementary issue is that object instances in 3D scans vary greatly in size and point density, leading to over- and/or under-segmentation when using fixed clustering parameters (e.g., the bandwidth Bw in the case of mean-shift Comaniciu and Meer, 2002). Therefore, there recently have been attempts to improve the clustering step (Engelmann et al., 2020; Jiang et al., 2020b,a; He et al., 2021a; Chen et al., 2021; Liang et al., 2021). The process usually starts by obtaining a sufficient number of instance candidates, which are then subject to various merging, pruning and de-duplicating operations to obtain the final instances. E.g., PointGroup (Jiang et al., 2020b) merges

Table 1 (*continued*).

|	Representatives	Methodology	Advantages	Disadvantages
Point-based	PointNet++ (Qi et al., 2017b)	Hierarchical application of mini PointNets on local 3D neighborhoods. Every layer applies PointNet on local 3D regions, which are then aggregated, grouped and passed to the following layer.	*Multi-scale features*. It extracts local features for points at different scales, thus improving the accuracy over PointNet.	- *Information loss*. Although it treats points independently at local scales in order to maintain permutation invariance, this independence ignores the geometric relationships between points and their neighbors, resulting in the absence of local features. - *Relative complexity*. It is more complex than PointNet and has more parameters. It is 3 times (or more) slower than PointNet.
	KPConv (Thomas et al., 2019)	KPconv is a point convolution in 3D. It first builds a radius graph pyramid on the 3D input point cloud by a combination of local neighborhood search and regular grid subsampling. Then the graph is processed by applying spatial convolution on the local *k* neighbors of each 3D point. It includes a version with a deformable kernels.	- *No preprocessing required:* Works directly on point clouds. - *Accuracy*. Outperforms traditional convolutional networks in 3D object classification and segmentation tasks. - *Density Robustness:* It is more robust to varying densities because of a regular subsampling strategy.	*Computational complexity*. KPConv has a higher computational complexity due to the local neighborhood search.
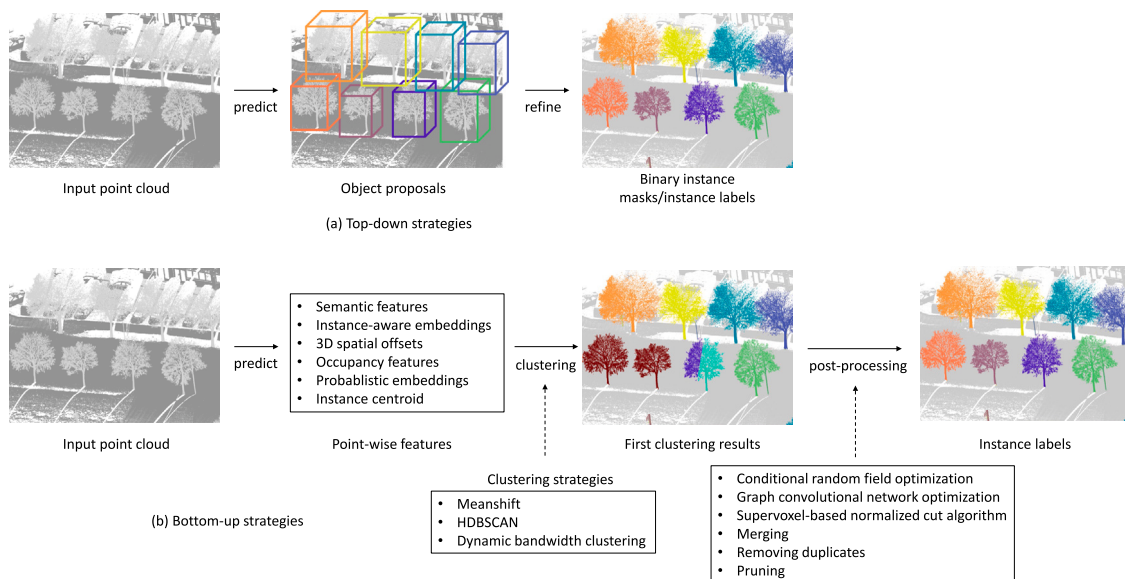
Fig. 6. Illustration of top-down (first row) and bottom-up (second row) strategies for instance segmentation.

clusters based on both their original and embedded coordinates to increase variety, scores the resulting instance candidates with a learned ScoreNet, then runs non-maximum suppression to prune overlapping candidates. 3D-MPA (Engelmann et al., 2020) and HAIS (Chen et al., 2021) favor aggregating candidates over non-maxima suppression. 3D-MPA generates multiple instance candidates by randomly sampling from the predicted instance centers, then constructs a graph convolutional network to allow for information exchange between adjacent candidates and find the best aggregation. Like the previously discussed methods, HAIS (Chen et al., 2021) clusters points into preliminary instance candidates based on semantics and location, then each candidate attempts to absorb nearby ones with a dynamic bandwidth that is proportional to its initial size. Finally, a neural network again acts as a score function to select instances. DyCo3D (He et al., 2021a) and SSTNet (Liang et al., 2021) also follow similar strategies. After obtaining the preliminary instance candidates, DyCo3D dynamically generates a filter to predict the binary instance mask for each candidate. Instead of directly clustering the points to generate preliminary instance candidates, SSTNet first generates geometrically homogeneous super-points, which are then linked into a tree. That tree serves as a basis for divisive grouping to find instance candidates, which are again refined by pruning and a scoring network. The SoftGroup network (Vu et al., 2022) also made improvements based on HAIS. In the clustering phase, each point is not first assigned one semantic class then clustered. Instead, a point may be associated with multiple possible semantic classes and, as a result, may belong to clusters in multiple classes. By doing so, the effects of semantic classification errors on subsequent instance segmentation will be mitigated. ASNet (Jiang et al., 2020a) converts the point clustering problem into an assignment problem, i.e., it samples *k* instance candidates and predicts, for each point, which of those candidates it belongs to. Note that at training time this involves bipartite matching (Kuhn, 1955) between the predicted and true instance centers, before one can compute their cross-entropy. The method also features a suppression module that learns to remove redundant candidates.

3.3. Other strategies

Panoster (Gasperini et al., 2021) outputs instance IDs directly, without having to sample or cluster points. The trick is to construct a softmax-based confusion matrix between real and predicted instances, and then use an impurity loss to boost the correlation between real and predicted values, together with a fragmentation loss to discourage instances with few points. ICM-3D (Chu et al., 2021) reformulates the instance segmentation problem into a classification problem inspired by SOLO (Wang et al., 2022), an effective 2D instance segmentation method. It eliminates additional clustering step to extract candidates, and avoids using non-maximum suppression to remove duplicate candidates. However, so far, this strategy is not competitive with clustering-based approaches in terms of segmentation quality.

Recently, the success of transformers for 2D segmentation has inspired a few works that employ transformers for 3D instance segmentation (Liu et al., 2022; Sun et al., 2022; Schult et al., 2023). They establish a query decoder that directly generates semantic classes, scores, and instance mask predictions. The predicted masks and ground truth masks are automatically associated using bipartite matching, enabling end-to-end training. Transformers achieve competitive performance, but have high computational overhead due to the complexity of the attention mechanism and the associated increase in the number of trainable parameters. How to more efficiently use the attention mechanism and find a good trade-off between efficiency and accuracy may be an interesting further direction for 3D instance segmentation.

4. Evaluation metrics and datasets

4.1. Evaluation metrics

Depending on the application, it may be more important to retrieve scene semantics or to correctly delineate instances. Different evaluation metrics are suited for the two subtasks of semantic segmentation and instance segmentation, and for the joint task of panoptic segmentation. For a comprehensive evaluation, all performance metrics are computed in the following experiments.

Evaluation of semantic segmentation: Common metrics for semantic segmentation include overall accuracy (oAcc), which is however biased towards classes with many points; and the mean Intersection-over-Union (mIoU) over all categories. With N the number of total points, C the number of categories, TP_i the number of points correctly assigned to semantic category i, GT_i the number of points with true semantic label i, and PRE_i the number of points with predicted semantic label i, the metrics are defined as:

$$\mathrm{oAcc} = \frac{\sum_{i=1}^{C} \mathrm{TP}_i}{N}, \quad \mathrm{mIoU} = \frac{1}{C} \sum_{i=1}^{C} \frac{\mathrm{TP}_i}{\mathrm{GT}_i + \mathrm{PRE}_i - \mathrm{TP}_i}. \tag{1}$$

Evaluation of instance segmentation: To evaluate instance segmentation, one first removes all points assigned to "stuff" classes that do not have distinct instances. For the remaining "things" classes, common metrics are mean coverage (mCov), mean weighted coverage (mWCov), mean precision (mPrec), mean recall (mRec) and F1-score. First, one collects all points that share the same predicted instance label into an instance prediction. The semantic category of that instance is determined by majority voting and assigned to all its points. In each category i, a set of ground truth semantic instances $\{I_j^{\mathrm{gt}}, j \in \{1, \ldots, N_{\mathrm{gt}}^i\}\}$, and a set of predicted instances $\{I_k^{\mathrm{pre}}, k \in \{1, \ldots, N_{\mathrm{pre}}^i\}\}$ are given. Moreover, an operator that compares an instance from one set with all instances in the other and returns the highest intersection-over-union score is defined as:

$$\mathrm{maxIoU}(I_j^{\mathrm{gt}}) = \max_{k=1}^{N_{\mathrm{pre}}^i} \left(\mathrm{IoU}(I_j^{\mathrm{gt}}, I_k^{\mathrm{pre}}) \right),$$

$$\mathrm{maxIoU}(I_k^{\mathrm{pre}}) = \max_{j=1}^{N_{\mathrm{gt}}^i} \left(\mathrm{IoU}(I_j^{\mathrm{gt}}, I_k^{\mathrm{pre}}) \right). \tag{2}$$

The coverage in category i is then defined as:

$$\mathrm{Cov}_i = \frac{1}{N_{\mathrm{gt}}^i} \sum_{j=1}^{N_{\mathrm{gt}}^i} \mathrm{maxIoU}(I_j^{\mathrm{gt}}), \quad \mathrm{WCov}_i = \frac{1}{N_{\mathrm{gt}}^i} \sum_{j=1}^{N_{\mathrm{gt}}^i} w_j \cdot \mathrm{maxIoU}(I_j^{\mathrm{gt}}), \tag{3}$$

where $|I_j^{\mathrm{gt}}|$ is the number of points in instance I_j^{gt}, and the weight $w_j = |I_j^{\mathrm{gt}}| / \sum_{m=1}^{N_{\mathrm{gt}}^i} |I_m^{\mathrm{gt}}|$. For the further scores, one first discards all predicted instances that have $\mathrm{maxIoU}(I_k^{\mathrm{pre}}) < 0.5$, to obtain the set of valid predictions $\{I_l^{\mathrm{val}}, l \in [1, \ldots, N_{\mathrm{val}}^i]\}$. With the number N_{val}^i of valid predictions, the precision and recall in category i are defined as

$$\mathrm{Prec}_i = \frac{N_{\mathrm{val}}^i}{N_{\mathrm{pre}}^i}, \quad \mathrm{Rec}_i = \frac{N_{\mathrm{val}}^i}{N_{\mathrm{gt}}^i}. \tag{4}$$

The overall metrics are then found by computing the means across all categories:

$$\mathrm{mCov} = \frac{1}{C} \sum_{i=1}^{C} \mathrm{Cov}_i, \qquad \mathrm{mWCov} = \frac{1}{C} \sum_{i=1}^{C} \mathrm{WCov}_i, \tag{5}$$

$$\mathrm{mPrec} = \frac{1}{C} \sum_{i=1}^{C} \mathrm{Prec}_i, \qquad \mathrm{mRec} = \frac{1}{C} \sum_{i=1}^{C} \mathrm{Rec}_i, \tag{6}$$

and the F1-score is calculated in the usual manner as

$$\mathrm{F1} = \frac{2 \cdot \mathrm{mPrec} \cdot \mathrm{mRec}}{\mathrm{mPrec} + \mathrm{mRec}}. \tag{7}$$

Evaluation of panoptic segmentation: For panoptic segmentation, it is natural to also adopt the metrics proposed for the 2D image version of the task, namely panoptic quality (PQ), segmentation quality (SQ), recognition quality (RQ), and PQ† (Kirillov et al., 2019; Porzi et al., 2019). Again one first computes the scores separately per semantic category i, regarding all points as a single, big instance for the "stuff" categories. For the segmentation quality, one accumulates the IoU of all true positive instances:

$$\mathrm{SQ}_i = \frac{1}{N_{\mathrm{val}}^i} \sum_{l=1}^{N_{\mathrm{val}}^i} \mathrm{maxIoU}(I_l^{\mathrm{val}}). \tag{8}$$

The recognition quality is simply the per-category F1-score,

$$\mathrm{RQ}_i = \mathrm{F1}_i = \frac{2 \cdot \mathrm{Prec}_i \cdot \mathrm{Rec}_i}{\mathrm{Prec}_i + \mathrm{Rec}_i}, \tag{9}$$

and the panoptic quality combines the two previous metrics into a single number:

$$\mathrm{PQ}_i = \mathrm{SQ}_i \cdot \mathrm{RQ}_i. \tag{10}$$

Note that for each "stuff" class there can be at most 1 ground truth instance and 1 predicted instance in a point cloud. Hence the SQ, RQ and PQ metrics are not properly defined for "stuff" classes with IoU below 0.5, or penalize them very harshly if set to 0. Porzi et al. (2019) therefore proposed an improved metric PQ†, which uses PQ for "things" classes, but replaces it with the simple IoU for "stuff" classes.

$$\mathrm{PQ}\dagger_i = \begin{cases} \mathrm{IoU}(I^{\mathrm{gt}}, I^{\mathrm{pre}}) & \text{if class } i \text{ is "stuff"} \\ \mathrm{PQ}_i & \text{otherwise}. \end{cases} \tag{11}$$

In case the "stuff" classes have sufficient overlap to be valid, the difference between PQ and PQ† is small, which is always the case in our experiments. Nevertheless both are quoted for completeness.

As before, the final panoptic metrics are calculated by averaging the respective per-category scores over all semantic categories:

$$\mathrm{SQ} = \frac{1}{C} \sum_{i=1}^{C} \mathrm{SQ}_i, \quad \mathrm{RQ} = \frac{1}{C} \sum_{i=1}^{C} \mathrm{RQ}_i, \quad \mathrm{PQ} = \frac{1}{C} \sum_{i=1}^{C} \mathrm{PQ}_i,$$

$$\mathrm{PQ}\dagger = \frac{1}{C} \sum_{i=1}^{C} \mathrm{PQ}\dagger_i. \tag{12}$$

4.2. Dataset

This section briefly introduces a number of popular, publicly available 3D point cloud datasets suitable for training and testing panoptic segmentation. A distinction is made between indoor and outdoor datasets, which have rather different characteristics in terms of both scene content and sensor parameters.

4.2.1. Outdoor datasets

NPM3D (Roynard et al., 2018) is a public benchmark for point cloud semantic segmentation, with 10 classes including: ground, building, pole (road sign and traffic light), bollard, trash can, barrier, pedestrian, car, natural (vegetation) and unclassified. Results are evaluated only w.r.t. 9 classes, disregarding the "unclassified" label. The data has been captured with a mapping-grade mobile laser scanning system in different cities in France. There are 4 regions designated for training, all captured in Paris and Lille; and 3 regions for testing, captured in Dijon and Ajaccio. The standard 10-class version described above has actually been derived from a more fine-grained version of the dataset by keeping only the most frequent labels. The original annotations feature 50 different semantic classes (most of which are very rare), and also individual object instance labels for the training regions. For panoptic segmentation, a new version has been generated that still uses the 10 semantic category labels listed above, but also includes instance labels (available at https://doi.org/10.5281/zenodo.8188390). The classes ground, building and barrier are considered "stuff" and are not separated into instances. As no instance labels are available for the 3 test regions, our version for panoptic (or pure instance) segmentation only contains 4 different regions from Paris and Lille. Instead of a fixed training/test split all experiments therefore use 4-fold cross-validation.

Other outdoor datasets for panoptic segmentation exist, which target the rather specific situation of autonomous driving. They have a very different sensing geometry, namely sequences of individual, sparse 32-beam panorama scans, and are thus not as suitable for our target application of mobile mapping. **SemanticKITTI** (Behley et al., 2019, 2021) provides annotations for 22 scenes with 28 semantic classes, for a total of 23′201 scans for training, and 20′351 scans for testing. **nuScenes** (Caesar et al., 2020; Fong et al., 2022) is a similar dataset of much larger scale featuring 16 classes (10 "things" and 6 "stuff"), with 1000 scenes collected in Singapore and Boston, totaling 300k scans.

4.2.2. Indoor datasets

S3DIS (Armeni et al., 2016), the "Stanford 3D Indoor Scene Dataset", contains 6 large-scale indoor areas with a total of 271 rooms. Data were collected in three different office/university buildings using the Matterport scanner.[2] In total the colored 3D point clouds have ≈696M points. Each point is annotated with one out of 13 semantic categories and an instance ID. There are two common experimental protocols: one uses area 5 as a fixed test set, the other one employs 6-fold cross-validation. S3DIS is perhaps the most widely used indoor dataset for instance segmentation, and among the datasets that provide instance annotations arguably also the most relevant one for mobile mapping in terms of data quality. Hence, it is used for complementary indoor experiments, see the Appendix A.2.

Other indoor datasets for panoptic segmentation include **ScanNet** (Dai et al., 2017), a collection of labeled voxels (rather than points). The current version, ScanNet v2, has 1513 scans, annotated with 20 semantic classes (18 "things" and 2 "stuff") and instance IDs. There is a prescribed split into 1201 training scans, 312 validation scans and 100 test scans (with private labels). **SceneNN** (Hua et al., 2016) is a smaller indoor RGB-D dataset of 100 scenes. Of those, 76 scenes have been annotated with 40 semantic categories, split into 56 training scenes and 20 test scenes.

A number of recent papers have studied panoptic point cloud segmentation. Among those, the present review focuses on the most successful school, namely bottom-up approaches that directly assign instance labels to points and do not rely on dedicated bounding box detectors. Table 2 presents a summary. According to the motivation of the present paper (c.f. Section 1), the Table 2 looks at three fundamental attributes: (1) the 3D backbone architecture used for feature extraction, (2) the type of feature representation used for instance clustering, and (3) the dataset(s) used for evaluation. Methods based on 2D projection and subsequent feature extraction with 2D backbones have been omitted, as they have only been used for the autonomous driving setting, where a natural projection exists from 3D points to 2.5D scanner coordinates. One can readily see in Table 2 that PointNet++ and sparse voxel CNNs (Submanifold Sparse U-Net and Minkowski Engine) are the two most popular backbones. Since panoptic segmentation includes semantic segmentation, all methods extract a set of features optimized for semantic classification, which serve as input for the corresponding classifier. Additionally, most methods extract some dedicated representation for instance segmentation, most often either another set of features, optimized to discriminate instances, or an offset vector from each point to its associated instance center. Indoor panoptic segmentation is commonly trained and tested on S3DIS and/or ScanNet, SceneNN and other (synthetic) datasets are less widespread. Very few works have addressed outdoor settings, and those which have are restricted to the sparse, panoramic scans of the autonomous driving scenario. Instance segmentation of dense, full 3D mobile mapping data does not seem to have been attempted with modern, neural network-based methods.

5. Experiments

5.1. Panoptic segmentation pipeline

For the experiments, our study adopts the bottom-up instance segmentation strategy, due to its superior performance in recent experimental studies (Engelmann et al., 2020; Jiang et al., 2020b,a; He et al., 2021a; Chen et al., 2021; Liang et al., 2021). A detailed graphical illustration is given in Fig. 7. Notations and descriptions of all parameters appear in this paper can be found in Table 3.

5.1.1. Input

Mobile mapping point clouds of realistic extent and density are far too large for processing on current graphics hardware. To reduce the excessive point density in the near field of the scanner, the raw data are preprocessed with voxel-grid downsampling, keeping only one (randomly selected) point per $d \times d \times d$ voxel. Even after downsampling it is not possible to process larger regions in one piece. Hence, one must in practice work with local neighborhoods, potentially sacrificing some long-range context. In this study spherical neighborhoods of a fixed radius have been used. The set of points inside the sphere, denoted as $\mathbb{P}$, is transformed to relative coordinates w.r.t. the center point. If other per-point information shall be used (e.g., RGB color values or height in absolute scene coordinates), they are appended to the relative coordinates to form a K-dimensional input vector F per point. For example, with $K = 6$ the vector F contains the relative location of a 3D point and the RGB information. All possible feature combinations and hyperparameters are listed in Table 3. The features of the point set $\mathbb{P}$ are denoted $\mathbf{F} = \{F_i\} \in \mathbb{R}^{N \times K}$, with N the number of points.

The training stage randomly samples the spheres' center locations in the point cloud. Additionally, data augmentation is performed by random scaling and rotation around the vertical axis, as well as adding Gaussian noise to the point coordinates. When working with the PointNet++ backbone, which requires a fixed number of points per input sphere, random duplication or removal of points are applied to reach that number. At inference time a regular grid of overlapping spheres is used, with fixed stride s along the three coordinate axes.

---

[2] http://matterport.com/.

Table 2
Summary of existing work on bottom-up panoptic point cloud segmentation, underlying methods and datasets. Rows sorted in chronological order.

Methods	Backbone	Features extracted by semantic and instance segmentation branch				Dataset				
		Semantic features	Embedding features	Offsets	Others	S3DIS	Scan-net	Se-manticKITTI	nuScenes	Others
ASIS (Wang et al., 2019b)	PointNet (Qi et al., 2017a)/PointNet++ (Qi et al., 2017b)	✓	✓			✓				ShapeNet (Yi et al., 2016)
JSIS3D (Pham et al., 2019)	PointNet (Qi et al., 2017a)	✓	✓			✓				SceneNN (Hua et al., 2016)
MTML (Lahoud et al., 2019)	3D convolution network based on the SSCNet (Song et al., 2017)	✓	✓		Direction embedding features		✓			
MPNet (He et al., 2020)	PointNet++ (Qi et al., 2017b)	✓	✓			✓	✓			PartNet (Mo et al., 2019)
3D-MPA (Engelmann et al., 2020)	Minkowski CNN (Choy et al., 2019)	✓		✓		✓	✓			
PointGroup (Jiang et al., 2020b)	Submanifold sparse U-Net (Graham et al., 2018)	✓		✓		✓	✓			
OccuSeg (Han et al., 2020)	Submanifold sparse U-Net (Graham et al., 2018)	✓	✓	✓	Occupancy features	✓	✓			SceneNN (Hua et al., 2016)
ASNet (Jiang et al., 2020a)	PointNet (Qi et al., 2017a)/PointNet++ (Qi et al., 2017b)	✓	✓		3D instance centroid and centroid distance	✓				SceneNN (Hua et al., 2016)
DS-Net (Hong et al., 2021)	Cylinder 3D (Zhou et al., 2020)	✓		✓				✓	✓	
Panoster (Gasperini et al., 2021)	KPConv (Thomas et al., 2019)/SalsaNext (Cortinhal et al., 2020)	✓						✓		
DyCo3D (He et al., 2021a)	Submanifold sparse U-Net (Graham et al., 2018)	✓		✓		✓	✓			
Zhang and Wonka (2021)	PointNet++ (Qi et al., 2017b)	✓			Probabilistic embedding features		✓			PartNet (Mo et al., 2019)
HAIS (Chen et al., 2021)	Submanifold sparse U-Net (Graham et al., 2018)	✓		✓		✓	✓			
SSTNet (Liang et al., 2021)	Submanifold sparse U-Net (Graham et al., 2018)	✓		✓		✓	✓			
SoftGroup (Vu et al., 2022)	Submanifold sparse U-Net (Graham et al., 2018)	✓		✓		✓	✓			

5.1.2. Network architecture

Out of the numerous neural network architectures that are nowadays available to encode 3D point coordinates (and associated input features) into a predictive latent representation, our study evaluates three representative ones as backbones for the experimental pipeline, see Table 2: PointNet++, denoted as $PN(x)$, is a popular and widely used architecture, and arguably the best-performing representative from the first generation of point cloud networks, based on per-point multi-layer perceptrons (MLPs) with shared weights, rather than convolutions. Sparse CNN (as implemented in the Minkowski Engine), denoted as $MC(x)$, is a recent high-performance variant of standard, discrete convolution on sparse 3D voxel grids. And KPConv, denoted as $KP(x)$, is the perhaps most elaborate approximation of convolution in continuous space, sparsified by only sampling the (continuous) output at the original point locations. In our pipeline the backbones are interchangeable and operate as a black box module that turns the input into a per-point feature representation of fixed size (channel depth). That representation is further fed into two parallel heads responsible for semantic segmentation and instance segmentation (Fig. 7). The semantic segmentation head $H^1(\cdot)$ predicts, for every point, the membership probabilities **CP** for all semantic categories. For the instance segmentation head there are again two variants: $H^2(\cdot)$ transforms the input representation into an embedding feature space **EF** such that points on the same instance form compact clusters; whereas $H^3(\cdot)$ instead estimates per-point 3D offset vectors **O** pointing to the center of its associated instance. All three heads are three-layer MLPs with the same structure. Their first two layers are linear transformations with the same number of input and output channels, followed by Batch Normalization (Ioffe and Szegedy, 2015) and LeakyReLU activations (Maas et al., 2013). The output size of the third layer differs between the three heads: for H^1 it is the number C of semantic categories, for H^2 it is the (tunable) dimension of the embedding space, and for H^3 it is a 3D displacement vector.

5.1.3. Loss functions

Semantic segmentation head: The output of this network branch are class probabilities $\mathbf{CP} = \{CP_i\} \in \mathbb{R}^{N \times C}$ for N points and C semantic classes, with $i \in \{1, \ldots, N\}$. The corresponding loss function L_s is the standard cross-entropy between the rows of **CP** and the ground truth
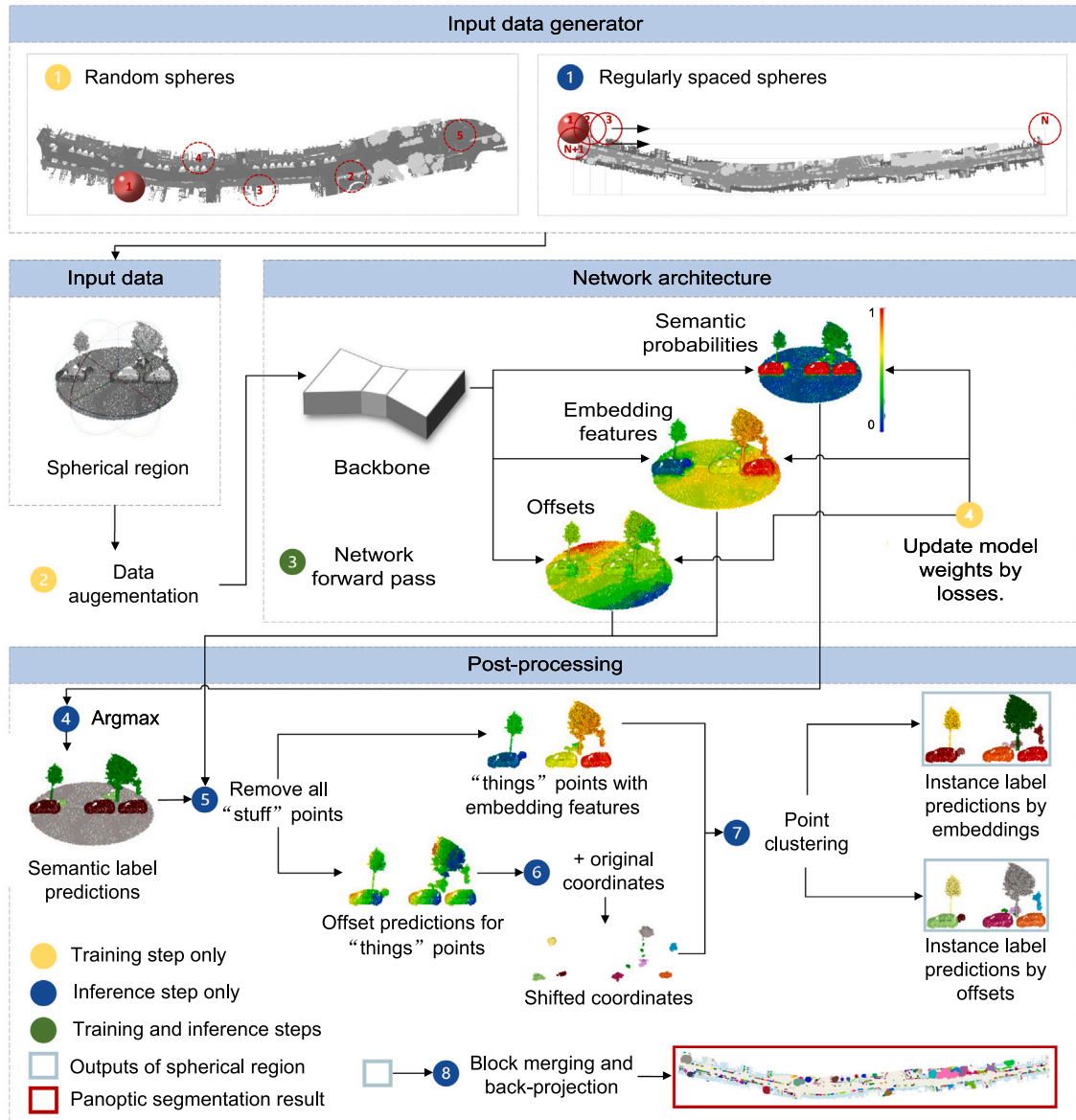
Fig. 7. Proposed panoptic segmentation pipeline for mobile mapping point clouds.

labels y:

$$L_s = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{ij} log(CP_{ij}). \tag{13}$$

Instance embedding head: The output of this branch $\mathbf{EF} = \{EF_i\} \in \mathbb{R}^{N \times T}$ is an embedding of the points in a T-dimensional feature space, optimized such that points on the same object are close to each other whereas points on different objects are far apart. Following our own preliminary experiments and recent literature (Wang et al., 2019b; Engelmann et al., 2020; He et al., 2020), the dimension is set to $T = 5$. The associated loss L_e is the discriminative loss function (De Brabandere et al., 2017) prevalent in bottom-up instance segmentation (see

Table 2):

$$L_e = L_{e\_var} + L_{e\_dist} + 0.001 \cdot L_{e\_reg}, \quad \text{with}$$

$$L_{e\_var} = \frac{1}{N_{gt}} \sum_{i=1}^{N_{gt}} \frac{1}{|I_i^{gt}|} \sum_{j=1}^{|I_i^{gt}|} [\|\mu_i - EF_j\|_1 - \delta_v]_+^2 ,$$

$$L_{e\_dist} = \frac{1}{N_{gt}(N_{gt}-1)} \sum_{\substack{i_A=1 \\ i_A \neq i_B}}^{N_{gt}} \sum_{i_B=1}^{N_{gt}} [2\delta_d - \|\mu_{i_A} - \mu_{i_B}\|_1]_+^2 , \tag{14}$$

$$L_{e\_reg} = \frac{1}{N_{gt}} \sum_{i=1}^{N_{gt}} \|\mu_i\|_1 .$$

N_{gt} denotes the total number of ground truth instances, $|I_i^{gt}|$ is the number of points in the ith ground truth instance I_i^{gt}, μ_i is the mean

Table 3
Notation and description of parameters.

Category	Parameter	Description
Input	d	Voxel grid size for down-sampling
	R	Radius of input spheres
	s	Sphere sampling stride during inference
	k	Fixed number of points/sphere if using PointNet++
	K	Dimension of input features for vector F: K = 3 … input features are relative (X, Y, Z) coordinates, K = 4 … relative (X, Y, Z) and original Z, K = 6 … relative (X, Y, Z) and RGB, K = 7 … relative (X, Y, Z) and original Z and RGB.
Training	W_e	Weight of embedding loss
	W_o	Weight of offset loss
	W_r	Weight of regularizer
Thresholds	Th_d	Threshold for instance clustering
	Th_n	Minimum point number for a valid instance
	Th_{bm}	Threshold for block merging
	Bw	Mean-shift kernel bandwidth

Table 4
Best parameter settings for the NPM3D dataset.

Category	Parameter	Best values/default values (unit)
Input	d	0.12 (m)
	R	8 (m)
	s	8 (m)
	k	17 500
	K	3 for PointNet++ 4 for KPConv and sparse CNN
Training	W_e	1
	W_o	0.1
	W_r	0
Thresholds	Th_d	$1.5 \cdot d$
	Th_n	10
	Th_{bm}	0.01
	Bandwidth	0.6

embedding of all points in instance I_i^{gt}, and EF_j is the predicted embedding of point j in instance I_i^{gt}. The operators are $\| \cdot \|_1$ for the L^1 (Manhattan) distance and $[x]_+ = \max(0, x)$. The hyper-parameters are chosen as $\delta_v = 0.5$ and $\delta_d = 1.5$.

Instance offset head: Instead of a discriminative embedding, this branch directly regresses offset vectors $\mathbf{O} = \{O_i\} \in \mathbb{R}^{N \times 3}$ from each 3D point's location to the centroid of the instance the point belongs to. As loss function L_o the one proposed by PointGroup (Jiang et al., 2020b) is adopted. Besides a standard L^1 regression loss (L_{o_reg}) between the predicted and ground truth offsets, the loss also utilizes the cosine similarity (L_{o_dir}) to better constrain the direction of the offset:

$$L_o = L_{o\_reg} + L_{o\_dir}, \quad \text{with}$$

$$L_{o\_reg} = \frac{1}{\sum_{i=1}^{N_{gt}} |I_i^{gt}|} \sum_{i=1}^{N_{gt}} \sum_{j=1}^{|I_i^{gt}|} \|O_j - (C_i - P_j)\|_1, \quad (15)$$

$$L_{o\_dir} = -\frac{1}{\sum_{i=1}^{N_{gt}} |I_i^{gt}|} \sum_{i=1}^{N_{gt}} \sum_{j=1}^{|I_i^{gt}|} \frac{O_j}{\|O_j\|_2} \cdot \frac{C_i - P_j}{\|C_i - P_j\|_2}.$$

Again, N_{gt} is the number of ground truth instances and $|I_i^{gt}|$ is the number of points in instance I_i^{gt}. Furthermore, O_j is the predicted offset from point j to the centroid of instance I_i^{gt}, P_j are the original 3D coordinates of point j, C_i is the centroid of instance I_i^{gt} and $\| \cdot \|_2$ is the L^2 (Euclidean) distance.

The entire neural network, including the backbone and the prediction heads, is trained from scratch by minimizing a joint loss function

$$L = L_s + W_e \cdot L_e + W_o \cdot L_o + W_r \cdot L_r, \quad (16)$$

where L_r is a standard L^2 regularizer on the network weights. W_e, W_o and W_r denote the weight for each loss term, respectively.

5.1.4. Post-processing

During inference, a sliding-window scheme is employed, with regularly-spaced spheres with a stride s, chosen such that adjacent spheres overlap and every point in the test set is processed at least once. The semantic class probabilities from overlapping spheres are averaged, and the final semantic segmentation is obtained as the point-wise argmax over those average scores. Points assigned to a "stuff" class are removed before instance segmentation. All points assigned to "things" classes are embedded according to the predictions of the instance segmentation head, i.e., either by reading out their feature embedding, or by shifting them in 3D space according to the offset

vectors. The embedded points are then clustered into instances. In line with other authors (Zhao et al., 2021) our results suggest that conventional clustering as a post-process performs better than learned clustering within an end-to-end architecture. Like several others (Wang et al., 2019b; Lahoud et al., 2019; He et al., 2021a) our study finds mean-shift to be an efficient and reliable clustering algorithm for the learned instance embedding. On the contrary, and in line with the literature, connected components search works better to cluster points based on their shifted coordinates (offset predictions add original coordinates). To that end, one defines two thresholds Th_d and Th_n. A set of points is considered an instance if *(i)* all points have the same semantic label, *(ii)* the Euclidean distances between shifted coordinates of those points are $< Th_d$, and *(iii)* the number of points in the instance is $> Th_n$.

Clusters with too few points are not assigned to any instance at this stage, their instance labels will be determined later when moving back from the subsampled to the original point cloud (see below). The Block-Merging procedure of Wang et al. (2018) is used to reconcile instance labels between overlapping spheres, except that a threshold Th_{bm} on the IoU turned out to be a better merging criterion than the absolute number of common points. The three thresholds (hyper-parameters) are fixed, with the exception of an obvious (linear) dependency between Th_d and the voxel grid size, see Table 4.

Finally, to obtain a complete panoptic segmentation of the full, raw point cloud (i.e., undo the voxel-grid sampling), the semantic labels and instance labels are mapped back to the original points with nearest-neighbor assignment.

5.2. Experiment details

The segmentation pipeline has been implemented based on the Torch-Points3D library (Chaton et al., 2020). 4-fold cross-validation experiments were conducted on the NPM3D dataset. For each backbone, the hyper-parameters are tuned to yield the best performance, see Table 4. The corresponding ablation studies are available in Appendix A.1.

In all experiments, the network is trained on a single NVIDIA TITAN RTX GPU for 500 epochs, with SGD with momentum as the optimizer and a base learning rate of 0.01. Each epoch comprises 3000 randomly sampled spheres, using mini-batches of 8 spheres.

5.3. Experimental results

Following the pipeline described in Section 5.1 and the experimental details in Section 5.2, this research averaged the results for 4 cross-validation experiments to obtain final results for each backbone.

Table 5

Comparison of three backbones in terms of panoptic segmentation metrics and mIoU for semantic segmentation. For metrics PQ† and mIoU, also the standard deviation (std) over 4 cross-validation runs is given. **Blue** indicates which backbone performed best.

Backbone	Panoptic segmentation				Panoptic segmentation ("things")			Panoptic segmentation ("stuff")			Semantic segmentation
	PQ	PQ† ± std	RQ	SQ	PQ	RQ	SQ	PQ	RQ	SQ	mIoU ± std
KPConv	66.4	67.0 ± 2.3	74.4	86.7	60.4	65.8	90.9	78.4	91.6	78.4	74.3 ± 5.2
Sparse CNN	63.0	65.1 ± 2.2	69.9	84.7	57.9	63.2	90.4	73.3	83.3	73.3	73.2 ± 4.0
PointNet++	40.1	43.3 ± 5.4	45.7	74.5	29.2	35.3	80.7	62.1	66.6	62.1	61.9 ± 6.9

5.3.1. Performance of different backbones

Table 5 shows the quantitative results of the cross-validation experiments on NPM3D, mainly focusing on semantic segmentation mIoU and panoptic segmentation metrics. The table also records the standard deviation (std) values of two main metrics: PQ† for panoptic segmentation and mIoU for semantic segmentation. As illustrated in Table 5, KPConv outperforms the other two backbones in all metrics. Sparse CNN has the smallest standard deviations for PQ† and mIoU, meaning that it is particularly stable across the four cross-validation folds. The PointNet++ backbone, on the other hand, shows comparatively poor performance, in particular when it comes to separating instances of "things". The experimental results validate the observation from Table 1, that PointNet++ ignores the geometric relationships between neighboring points (see Fig. 5), resulting in poor generalization performance for large, complex scenes compared to the other two methods.

Figs. 9(a) and 9(b) show the IoU and PQ† as well as the standard deviation for each semantic class. Looking at the results, it is clear that KPConv achieves the highest IoU and PQ† in most semantic classes. Sparse CNN reached the highest IoU for the pole class and the highest PQ† for the pedestrian class. This can be visually confirmed by the corresponding qualitative results for semantic and instance segmentation shown in Figs. 8 and 10. In the third row of Fig. 8, the points inside the black circle marker that should belong to a pole are accurately segmented only by sparse CNN. A possible explanation is that KPConv is able to capture more complex structures, while being more sensitive to noise. For instance, in the example inside the black circle marker in the third row of Fig. 8, when a street sign with a short pole is close to the tree crown, KPConv might mistake the street sign for a protruding part of the tree crown. On the other hand, sparse CNN is regularized more strictly by the native voxel resolution and therefore potentially less sensitive (Table 1), so it achieves a better segmentation in this particular case. Moreover, for moving pedestrians, the collected data often has motion artifacts (elongated ghosting or tail-like features in the point cloud). Also these can lead to over-segmentation with KPConv, due to its higher sensitivity. PointNet++ is relatively poor for all classes. In the second and third rows, marked by red circles in Fig. 8, the PointNet++ backbone fails to correctly classify pedestrians, which is much worse than the other two backbones. In addition, the PointNet++ backbone often confuses barriers with buildings, see Fig. 8.

The largest standard deviation was observed for the trash can class, Fig. 9(a), which is due to the poor results of all three backbones for area 4. It is noted that Area 4 was collected in the city of Paris, while data for the other three areas was collected in Lille. This causes a larger domain gap (e.g., differently sized and/or shaped street furniture), and consequently poorer semantic segmentation in Area 4. As shown by the black circle markers in the first row of Fig. 8, none of the three backbones can correctly classify the points belonging to trash cans. The black circle marker in the second line of Fig. 8 shows a case that can only be segmented well by KPConv. The natural class gets satisfactory IoU but very low PQ† values, for all backbones. This is because the vegetation instances are poorly segmented, especially when trees are close to each other and overlap, see first and second lines of Fig. 10.

5.3.2. Comparison between offset and embedding-based clustering

Table 6 shows a comparison between instance (and panoptic) segmentation obtained with either the offset or the embedding head. In general, instance segmentation results based on embedding features are significantly better than those based on shifted coordinates for the KPConv and sparse CNN backbones. In the case of the PointNet++, performance based on embedding features is only slightly better than the one with shifted coordinates (2.4% higher PQ†).

Figs. 11(a) and 11(b) show the precision and recall per backbone and instantiation head. Clustering based on embedding features generally achieves better instance segmentation precision than shifted coordinates, except for the bollard class. For the KPConv and sparse CNN backbones, instance segmentation recall based on embedding features is also higher or approximately equal to the one based on shifted coordinates. For the PointNet++ backbone its the opposite, here the recall is higher if shifted coordinates are used. Fig. 12 compares both clustering schemes on two example scenes based on the KPConv backbone. In the first row (Area 1), black circle marker points out two trash cans that are very close to each other. Here the learned embedding features can better distinguish them, while the shifted coordinates are not able to separate them (under-segmentation). The second row (Area 2) shows a streetlight inside the black circle marker. It can be seen that the embedding features belonging to this instance are very compact, and thus the instance can be easily and completely segmented from other instances. However, clustering based on shifted coordinates splits the lamp head and the lamp pole into two instances (over-segmentation). For more visual examples see Fig. 13.

5.3.3. Runtime analysis

Runtime was measured for training and testing to compare the computational complexity of the different backbones. Table 7 shows the time required to load the data and to perform the actual computation for one training epoch. The computation includes the following operations: forward pass through the complete network, calculation of losses and gradients, and update of network weights. Data loading involves unpacking the input data from the data loader and performing the necessary preprocessing steps. As shown in Table 7, the KPConv backbone has a longer data loading time because, in addition to the data augmentation, it requires time to down-sample the point cloud and precompute neighborhoods on the CPU to speed up the forward pass.

Table 8 shows the runtime for different inference stages. The time for the forward pass depends on the backbone network, where PointNet++ takes the shortest time and Sparse CNN takes the longest. Instance clustering and block merging are independent of the backbone, so the time for these two steps given in Table 8 is the average over the 4-fold cross-validation experiments for all three backbones. Clustering time is measured for embedding features and shifted coordinates. Clustering of embedding features is done with the mean-shift implementation of scikit-learn (Pedregosa et al., 2011) which runs on CPU. Connected-components clustering of shifted coordinates employs the Torch-Points3D (Chaton et al., 2020) implementation, which uses the algorithm proposed in PointGroup (Jiang et al., 2020b). Neighborhood search is done on the GPU, while cluster assignment is done on the CPU. Overall the time required for clustering based on feature embedding is almost five times longer than for shifted coordinates.
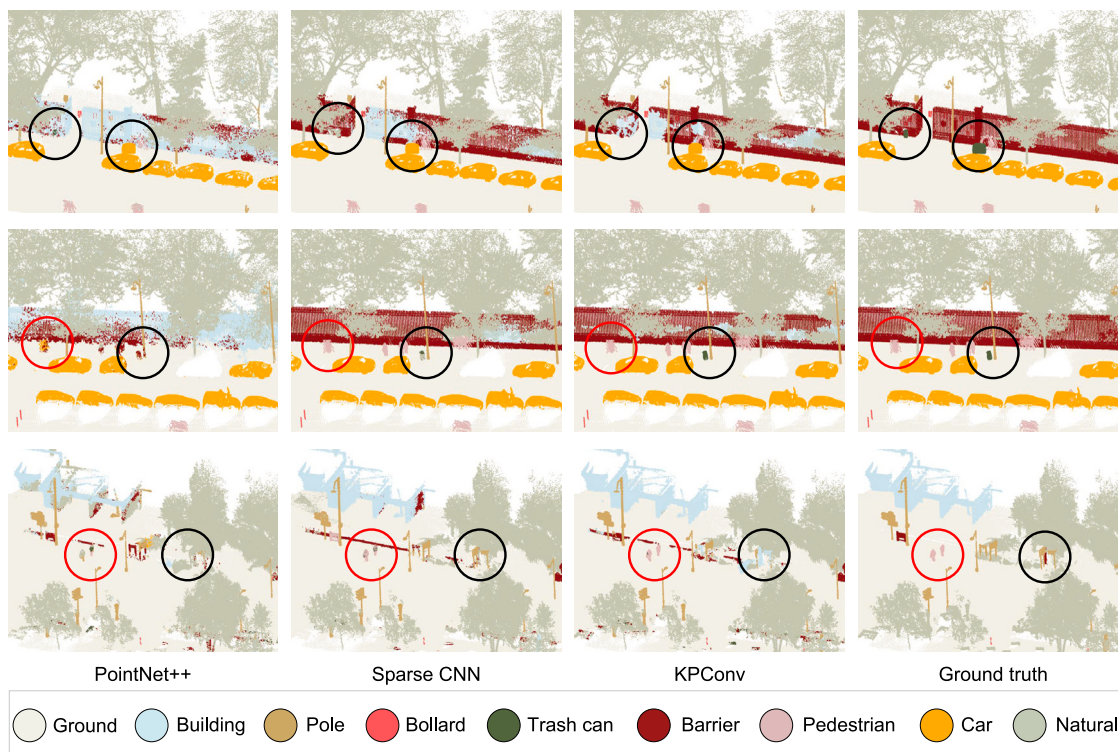
Fig. 8. Close-up semantic segmentation results with different backbones, and ground truth. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
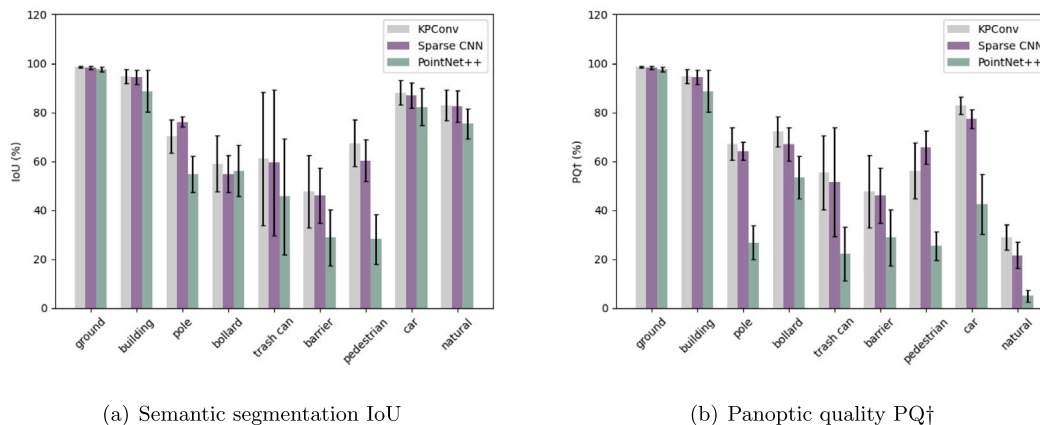

(a) Semantic segmentation IoU

(b) Panoptic quality PQ†

Fig. 9. Per-class results with 3 different backbones (mean and standard deviation of 4 cross-validation folds).

Table 6
To compare 2 different kinds of features for instance clustering based on 3 backbones on NPM3D dataset by using panoptic segmentation metrics and instance segmentation metrics. **Blue** indicates which instance strategy (embedding or offset prediction) performed better, **bold** font marks the best result per column.

Backbone	Instance	Instance segmentation					Panoptic segmentation				Panoptic segmentation ("things")		
		mCov	mWCov	mPrec	mRec	F1	PQ	PQ†	RQ	SQ	PQ	RQ	SQ
KPConv	Offset	70.5	73.2	48.8	72.6	58.2	60.0	60.7	67.7	85.9	50.9	55.8	89.7
	Embed	72.8	76.1	**60.3**	76.2	**67.2**	**66.4**	**67.0**	**74.4**	**86.7**	60.4	**65.8**	**90.9**
Sparse CNN	Offset	70.1	73.5	52.0	72.9	60.5	59.7	61.8	66.7	84.0	52.9	58.3	89.4
	Embed	**73.5**	**77.3**	55.7	**77.3**	64.6	63.0	65.1	69.9	84.7	57.9	63.2	90.4
PointNet++	Offset	54.9	53.9	27.5	58.0	36.3	37.7	40.9	42.6	75.5	25.6	30.6	82.2
	Embed	47.8	51.7	30.8	47.5	37.0	40.1	43.3	45.7	74.5	29.2	35.3	80.7

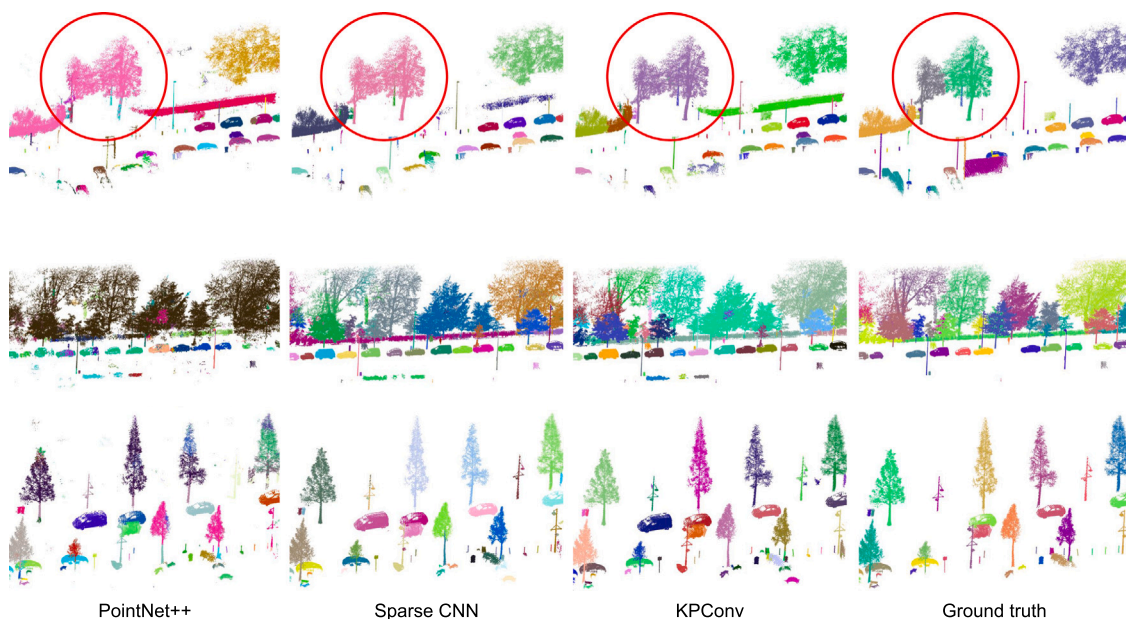PointNet++   Sparse CNN   KPConv   Ground truth

Fig. 10. Some close-up instance segmentation qualitative results based on 3 backbones and ground truth on NPM3D dataset. Different colors represent different instance, and the colors are randomly generated. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
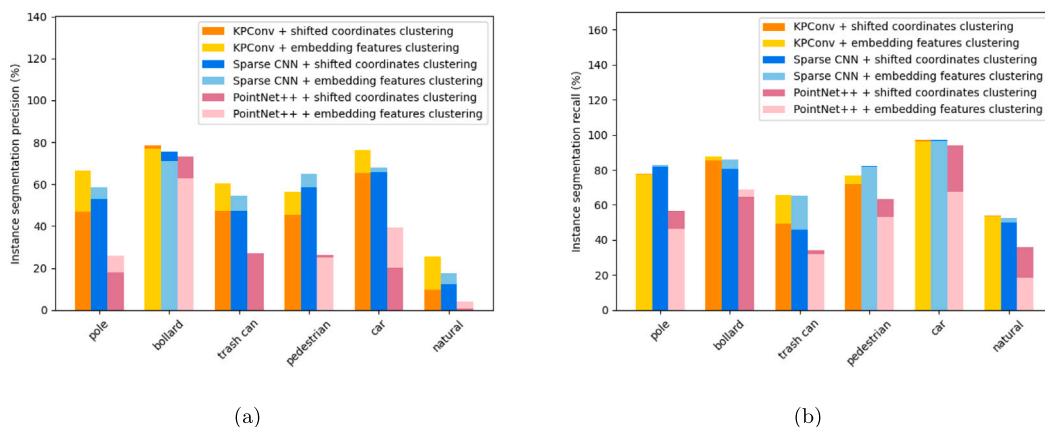

(a)   (b)

Fig. 11. Comparison of instance segmentation precision and recall for each "things" class, based on 3 backbones and 2 types of clustering features.

(a) Embedding features   (b) Instance segmentation   (c) Shifted coordinates   (d) Instance segmentation   (e) Ground truth
                              based on (a)   based on (c)
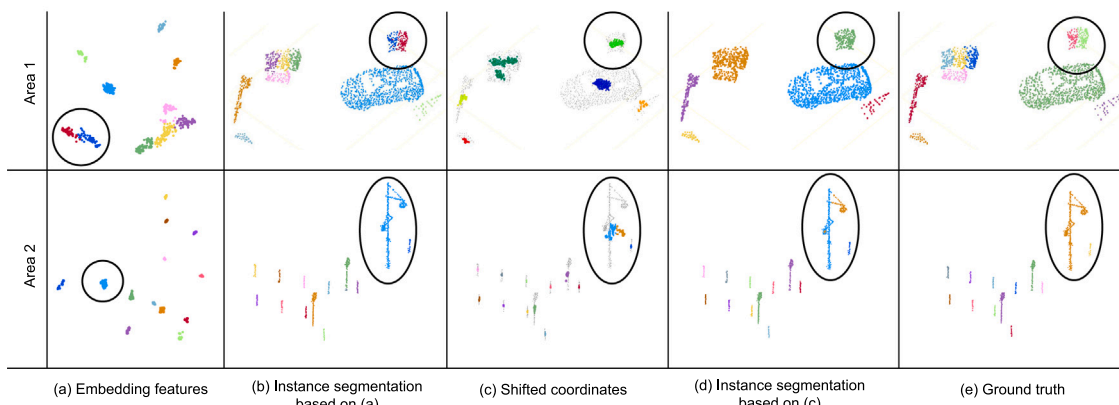
Fig. 12. Take 2 close-ups of instance segmentation based on KPConv backbone as examples. Column (a) displays the embedding features in 2D space by PCA algorithm, where the colors are randomly generated and the points of different instances have different colors. (b) Instance segmentation based on the clustering of embedding features, where the colors of points keep correspondence with column (a). (c) The gray points are the original points, and the colored points indicate the shifted coordinates, where the colors are randomly generated and the points of different instances have different colors. (d) Instance segmentation based on shifted coordinates clustering. (e) shows the ground truth of instance segmentation, where the colors are randomly generated and the points of different instances have different colors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
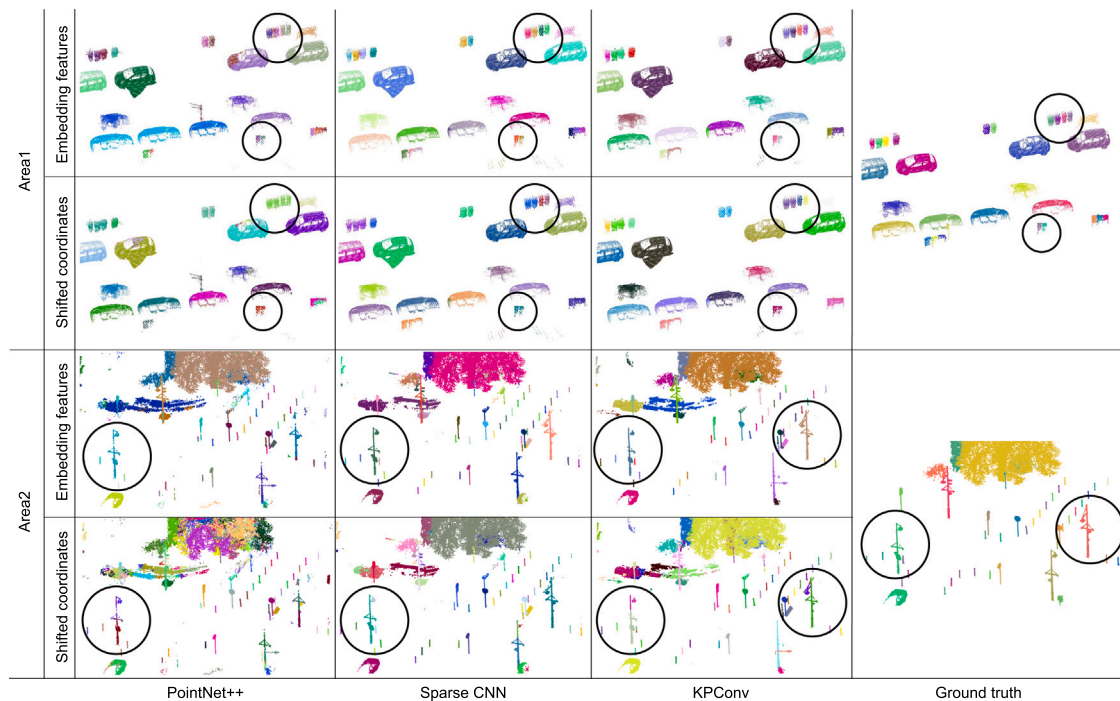
Fig. 13. Qualitative results of instance segmentation based on 2 different features and 3 backbone networks. Different colors indicate different instances, and the colors are randomly generated. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 7
Training time of main stages for three different backbones. All times were measured on the same hardware (8-core CPU with 4GB of memory per processor core and a single Nvidia Titan RTX GPU). The table shows the average duration of one epoch.

Backbone	Training time (s/epoch)		
	Computation	Data loading	Total
KPConv	99.9	755.8	855.6
Sparse CNN	84.6	173.4	258.0
PointNet++	83.8	142.7	226.5

Table 8
Inference time of main stages for three different backbones on NPM3D dataset. All times were measured with the same hardware (8-core CPU with 4GB of memory per processor core and a single Nvidia Titan RTX GPU).

Backbone	Inference times (s/million points)					
	Backbone fwd pass (GPU)	Instance clustering (GPU/CPU)		Block merging (CPU)	Total	
		Embed	Offset		Embed	Offset
KPConv	5.0	39.1	8.2	1.2	45.3	14.4
Sparse CNN	9.8	39.1	8.2	1.2	50.1	19.2
PointNet++	3.2	39.1	8.2	1.2	43.5	12.6

6. Conclusions

A comprehensive review has been presented for the task of 3D panoptic segmentation, including suitable 3D backbone networks, instance segmentation strategies, existing datasets and evaluation metrics. So far, a systematic comparison of panoptic segmentation methods for outdoor mobile mapping point clouds has been lacking. Therefore, instances have been annotated in the NPM3D dataset, and a modular panoptic segmentation pipeline has been implemented for such mobile mapping point clouds, with a choice of three different backbones representative of different schools of point cloud processing, and two representative instance segmentation strategies. In a comprehensive experimental evaluation, it was found that the KPConv backbone achieved the highest panoptic segmentation performance, but also has the highest runtime. PointNet++ has the lowest computational cost for training and inference, but yields relatively poor results

compared to the others. The sparse voxel CNN offers a good trade-off between computation time and segmentation performance. Moreover, our experiments showed that for our mobile mapping dataset instance segmentation by clustering embedding features gives better results than clustering with shifted coordinates, regardless of the backbone used.

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.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.isprsjprs.2023.08.008.

References

Alaba, S.Y., Ball, J.E., 2022. A survey on deep-learning-based LiDAR 3D object detection for autonomous driving. Sensors 22 (24), 9577. http://dx.doi.org/10.3390/s22249577.

Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S., 2016. 3D semantic parsing of large-scale indoor spaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 1534–1543.

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Gall, J., Stachniss, C., 2021. Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The semantickitti dataset. Int. J. Robot. Res. 40 (8–9), 959–967. http://dx.doi.org/10.1177/02783649211006735.

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J., 2019. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. ICCV, pp. 9297–9307.

Bello, S.A., Yu, S., Wang, C., Adam, J.M., Li, J., 2020. Review: Deep learning on 3D point clouds. Remote Sens. 12 (11), 1729. http://dx.doi.org/10.3390/rs12111729.

Burume, D., Du, S., 2021. Deep learning methods applied to 3D point clouds based instance segmentation: A review. http://dx.doi.org/10.20944/preprints202111.0228.v1.

Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O., 2020. nuScenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 11621–11631, arXiv:arXiv:1903.11027.

Campello, R.J.G.B., Moulavi, D., Sander, J., 2013. Density-based clustering based on hierarchical density estimates. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (Eds.), Advances in Knowledge Discovery and Data Mining. pp. 160–172.

Chaton, T., Chaulet, N., Horache, S., Landrieu, L., 2020. Torch-Points3D: A modular multi-task framework for reproducible deep learning on 3D point clouds. In: International Conference on 3D Vision (3DV). pp. 1–10. http://dx.doi.org/10.1109/3DV50981.2020.00029.

Chen, S., Fang, J., Zhang, Q., Liu, W., Wang, X., 2021. Hierarchical aggregation for 3D instance segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. ICCV, pp. 15467–15476.

Choy, C., Gwak, J., Savarese, S., 2019. 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 3075–3084.

Chu, R., Chen, Y., Kong, T., Qi, L., Li, L., 2021. ICM-3D: Instantiated category modeling for 3D instance segmentation. IEEE Robot. Autom. Lett. 7 (1), 57–64. http://dx.doi.org/10.1109/LRA.2021.3108483.

Comaniciu, D., Meer, P., 2002. Mean shift: a robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. 24 (5), 603–619. http://dx.doi.org/10.1109/34.1000236.

Cortinhal, T., Tzelepis, G., Aksoy, E.E., 2020. SalsaNext: fast, uncertainty-aware semantic segmentation of LiDAR point clouds for autonomous driving. arXiv:arXiv:2003.03653.

Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M., 2017. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In: Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition. CVPR, pp. 5828–5839.

Dai, A., Nießner, M., 2018. 3DMV: Joint 3D-multi-view prediction for 3D semantic scene segmentation. In: Proceedings of the European Conference on Computer Vision. ECCV, pp. 452–468.

De Brabandere, B., Neven, D., Van Gool, L., 2017. Semantic instance segmentation with a discriminative loss function. arXiv:arXiv:1708.02551.

Diab, A., Kashef, R., Shaker, A., 2022. Deep learning for LiDAR point cloud classification in remote sensing. Sensors 22 (20), 7868. http://dx.doi.org/10.3390/s22207868.

Elich, C., Engelmann, F., Schult, J., Kontogianni, T., Leibe, B., 2019. 3D-BEVIS: Bird's-eye-view instance segmentation. In: Fink, G.A., Frintrop, S., Jiang, X. (Eds.), Pattern Recognition. Springer International Publishing, Cham, pp. 48–61. http://dx.doi.org/10.1007/978-3-030-33676-9_4.

Engelmann, F., Bokeloh, M., Fathi, A., Leibe, B., Nießner, M., 2020. 3D-MPA: Multi proposal aggregation for 3D semantic instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 9031–9040.

Fong, W.K., Mohan, R., Hurtado, J.V., Zhou, L., Caesar, H., Beijbom, O., Valada, A., 2022. Panoptic nuScenes: A large-scale benchmark for LiDAR panoptic segmentation and tracking. IEEE Robot. Autom. Lett. 7 (2), 3795–3802. http://dx.doi.org/10.1109/LRA.2022.3148457.

Gasperini, S., Mahani, M.-A.N., Marcos-Ramiro, A., Navab, N., Tombari, F., 2021. Panoster: End-to-end panoptic segmentation of LiDAR point clouds. IEEE Robot. Autom. Lett. 6 (2), 3216–3223. http://dx.doi.org/10.1109/LRA.2021.3060405.

Graham, B., Engelcke, M., van der Maaten, L., 2018. 3D semantic segmentation with submanifold sparse convolutional networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 9224–9232.

Griffiths, D., Boehm, J., 2019. A review on deep learning techniques for 3D sensed data classification. Remote Sens. 11 (12), 1499. http://dx.doi.org/10.3390/rs11121499.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M., 2021. Deep learning for 3D point clouds: A survey. IEEE Trans. Pattern Anal. Mach. Intell. 43 (12), 4338–4364. http://dx.doi.org/10.1109/TPAMI.2020.3005434.

Hackel, T., Savinov, N., Ladicky, L., Wegner, J.D., Schindler, K., Pollefeys, M., 2017. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. IV-1-W1, pp. 91–98.

Han, L., Zheng, T., Xu, L., Fang, L., 2020. OccuSeg: Occupancy-aware 3D instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 2940–2949.

He, K., Gkioxari, G., Dollár, P., Girshick, R.B., 2017. Mask R-CNN. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. ICCV, pp. 2980–2988.

He, T., Gong, D., Tian, Z., Shen, C., 2020. Learning and memorizing representative prototypes for 3D point cloud semantic and instance segmentation. In: European Conference on Computer Vision. ECCV, vol. 12363, pp. 564–580. http://dx.doi.org/10.1007/978-3-030-58523-5_33.

He, T., Shen, C., van den Hengel, A., 2021a. DyCo3D: Robust instance segmentation of 3D point clouds through dynamic convolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 354–363.

He, Y., Yu, H., Liu, X.-Y., Yang, Z., Sun, W., Wang, Y., Fu, Q., Zou, Y., Mian, A.S., 2021b. Deep learning based 3D segmentation: A survey. arXiv:arXiv:2103.05423.

Hong, F., Zhou, H., Zhu, X., Li, H., Liu, Z., 2021. LiDAR-based panoptic segmentation via dynamic shifting network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 13090–13099.

Hua, B.-S., Pham, Q.-H., Nguyen, D.T., Tran, M.-K., Yu, L.-F., Yeung, S.-K., 2016. SceneNN: A scene meshes dataset with aNNotations. In: International Conference on 3D Vision (3DV). pp. 92–101. http://dx.doi.org/10.1109/3DV.2016.18.

Hua, B.S., Tran, M.K., Yeung, S.K., 2018. Pointwise convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 984–993.

Huang, J., You, S., 2016. Point cloud labeling using 3D convolutional neural network. In: Proceedings of the International Conference on Pattern Recognition. ICPR, pp. 2670–2675.

Ioannidou, A., Chatzilari, E., Nikolopoulos, S., Kompatsiaris, Y., 2017. Deep learning advances in computer vision with 3D data: A survey. ACM Comput. Surv. 50 (2), 1–38. http://dx.doi.org/10.1145/3042064.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceeedings of the International Conference on Machine Learning. ICML, PMLR, pp. 448–456.

Jaritz, M., Gu, J., Su, H., 2019. Multi-view pointnet for 3D scene understanding. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops.

Jhaldiyal, A., Chaudhary, N., 2022. Semantic segmentation of 3D LiDAR data using deep learning: a review of projection-based methods. Appl. Intell. http://dx.doi.org/10.1007/s10489-022-03930-5.

Jiang, H., Yan, F., Cai, J., Zheng, J., Xiao, J., 2020a. End-to-end 3D point cloud instance segmentation without detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 12796–12805.

Jiang, L., Zhao, H., Shi, S., Liu, S., Fu, C.-W., Jia, J., 2020b. PointGroup: Dual-set point grouping for 3D instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 4867–4876.

Kalogerakis, E., Averkiou, M., Maji, S., Chaudhuri, S., 2017. 3D shape segmentation with projective convolutional networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 6630–6639.

Kirillov, A., He, K., Girshick, R.B., Rother, C., Dollár, P., 2019. Panoptic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 9396–9405.

Kuhn, H.W., 1955. The Hungarian method for the assignment problem. Nav. Res. Logist. Q. 2 (1–2), 83–97. http://dx.doi.org/10.1002/nav.3800020109.

Lahoud, J., Ghanem, B., Pollefeys, M., Oswald, M., 2019. 3D instance segmentation via multi-task metric learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. ICCV, pp. 9255–9265.

Landrieu, L., Simonovsky, M., 2018. Large-scale point cloud semantic segmentation with superpoint graphs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 4558–4567.

Lawin, F.J., Danelljan, M., Tosteberg, P., Bhat, G., Khan, F., Felsberg, M., 2017. Deep projective 3D semantic segmentation. In: Proceedings of the International Conference on Computer Analysis of Images and Patterns. CAIP, vol. 10424, pp. 95–107. http://dx.doi.org/10.1007/978-3-319-64689-3_8.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444. http://dx.doi.org/10.1038/nature14539.

Li, Y., Ma, L., Zhong, Z., Liu, F., Cao, D., Li, J., Chapman, M.A., 2020. Deep learning for LiDAR point clouds in autonomous driving: A review. IEEE Trans. Neural Netw. Learn. Syst. 32 (8), 3412–3432. http://dx.doi.org/10.1109/TNNLS.2020.3015992.

Li, E., Razani, R., Xu, Y., Liu, B., 2021. CPSeg: Cluster-free panoptic segmentation of 3D LiDAR point clouds. arXiv:arXiv:2111.01723.

Li, Z., Zhang, L., Tong, X., Du, B., Wang, Y., Zhang, Z., Liu, H., Mei, J., Xing, X., Mathiopoulos, P., 2016. A three-step approach for TLS point cloud classification. IEEE Trans. Geosci. Remote Sens. 54 (9), 5412–5424. http://dx.doi.org/10.1109/TGRS.2016.2564501.

Liang, Z., Li, Z., Xu, S., Tan, M., Jia, K., 2021. Instance segmentation in 3D scenes using semantic superpoint tree networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. ICCV, pp. 2783–2792.

Liang, Z., Yang, M., Wang, C., 2019. 3D graph embedding learning with a structure-aware loss function for point cloud semantic instance segmentation. arXiv:arXiv:1902.05247.

Liu, C., Furukawa, Y., 2019. MASC: Multi-scale affinity with sparse convolution for 3D instance segmentation. arXiv:arXiv:1902.04478.

Liu, J., He, T., Yang, H., Su, R., Tian, J., Wu, J., Guo, H., Xu, K., Ouyang, W., 2022. 3D-QueryIS: A query-based framework for 3D instance segmentation. arXiv:arXiv:2211.09375.

Liu, W., Sun, J., Li, W., Hu, T., Wang, P., 2019. Deep learning on point clouds and its application: A survey. Sensors 19 (19), 4188. http://dx.doi.org/10.3390/s19194188.

Liu, S.-H., Yu, S.-Y., Wu, S.-C., Chen, H.-T., Liu, T.-L., 2020. Learning Gaussian instance segmentation in point clouds. arXiv:arXiv:2007.09860.

Lu, H., Shi, H., 2020. Deep learning for 3D point cloud understanding: A survey. arXiv:arXiv:2009.08920.

Maas, A.L., Hannun, A.Y., Ng, A.Y., et al., 2013. Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of the International Conference on Machine Learning. ICML, vol. 30, Citeseer, p. 3, (1).

Maturana, D., Scherer, S., 2015. VoxNet: A 3D convolutional neural network for real-time object recognition. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 922–928.

Milioto, A., Behley, J., McCool, C., Stachniss, C., 2020. LiDAR panoptic segmentation for autonomous driving. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 8505–8512.

Mo, K., Zhu, S., Chang, A.X., Yi, L., Tripathi, S., Guibas, L., Su, H., 2019. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 909–918.

Neven, D., Brabandere, B.D., Proesmans, M., Gool, L.V., 2019. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 8837–8845.

Niemeyer, J., Rottensteiner, F., Soergel, U., 2012. Conditional random fields for LiDAR point cloud classification in complex urban areas. ISPRS Ann. Photogr. Remote Sens. Spatial Inf. Sci. 1 (3), 263–268.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in python. J. Mach. Learn. Res. 12, 2825–2830.

Pham, Q.H., Nguyen, D., Hua, B.-S., Roig, G., Yeung, S., 2019. JSIS3D: Joint semantic-instance segmentation of 3D point clouds with multi-task pointwise networks and multi-value conditional random fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 8819–8828.

Porzi, L., Bulo, S.R., Colovic, A., Kontschieder, P., 2019. Seamless scene segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 8277–8286.

Qi, C.R., Su, H., Mo, K., Guibas, L., 2017a. PointNet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 77–85.

Qi, C.R., Yi, L., Su, H., Guibas, L., 2017b. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). pp. 5105–5114.

Riegler, G., Ulusoy, A.O., Geiger, A., 2017. OctNet: Learning deep 3D representations at high resolutions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 6620–6629.

Rosu, R.A., Schütt, P., Quenzel, J., Behnke, S., 2019. LatticeNet: Fast point cloud segmentation using permutohedral lattices. arXiv:arXiv:1912.05905.

Roynard, X., Deschaud, J.-E., Goulette, F., 2018. Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. Int. J. Robot. Res. 37 (6), 545–557. http://dx.doi.org/10.1177/0278364918767506.

Schult, J., Engelmann, F., Hermans, A., Litany, O., Tang, S., Leibe, B., 2023. Mask3D for 3D semantic instance segmentation. In: International Conference on Robotics and Automation. ICRA.

Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T., 2017. Semantic scene completion from a single depth image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 1746–1754.

Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J., 2018. SPLATNet: Sparse lattice networks for point cloud processing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 2530–2539.

Sun, J., Qing, C., Tan, J., Xu, X., 2022. Superpoint transformer for 3D scene instance segmentation. arXiv:arXiv:2211.15766.

Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., Han, S., 2020. Searching efficient 3D architectures with sparse point-voxel convolution. In: Proceedings of the European Conference on Computer Vision. ECCV, pp. 685–702. http://dx.doi.org/10.1007/978-3-030-58604-1_41.

Tatarchenko, M., Park, J., Koltun, V., Zhou, Q.-Y., 2018. Tangent convolutions for dense prediction in 3D. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 3887–3896.

Thomas, H., Qi, C.R., Deschaud, J.-E., Marcotegui, B., Goulette, F., Guibas, L., 2019. KPConv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. ICCV, pp. 6410–6419.

Vu, T., Kim, K., Luu, T.M., Nguyen, T., Yoo, C.D., 2022. SoftGroup for 3D instance segmentation on point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 2708–2717.

Wang, L., Huang, Y., Hou, Y., Zhang, S., Shan, J., 2019a. Graph attention convolution for point cloud semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 10296–10305.

Wang, Y., Jiang, T., Liu, J., Li, X., Liang, C., 2020. Hierarchical instance recognition of individual roadside trees in environmentally complex urban areas from uav laser scanning point clouds. ISPRS Int. J. Geo-Inf. 9, 595. http://dx.doi.org/10.3390/ijgi9100595.

Wang, P.-S., Liu, Y., Guo, Y.-X., Sun, C.-Y., Tong, X., 2017. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. ACM Trans. Graph. 36 (4), 1–11. http://dx.doi.org/10.1145/3072959.3073608.

Wang, X., Liu, S., Shen, X., Shen, C., Jia, J., 2019b. Associatively segmenting instances and semantics in point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 4091–4100.

Wang, Y., Sun, Y., Liu, Z., Sarma, S., Bronstein, M., Solomon, J., 2019c. Dynamic graph CNN for learning on point clouds. ACM Trans. Graph. 38 (5), 1–12. http://dx.doi.org/10.1145/3326362.

Wang, W., Yu, R., Huang, Q., Neumann, U., 2018. SGPN: Similarity group proposal network for 3D point cloud instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)). pp. 2569–2578.

Wang, X., Zhang, R., Shen, C., Kong, T., Li, L., 2022. SOLO: A simple framework for instance segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 44 (11), 8587–8601. http://dx.doi.org/10.1109/TPAMI.2021.3111116.

Wu, B., Wan, A., Yue, X., Keutzer, K., 2018. SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR point cloud. In: Proceedings of the IEEE International Conference on Robotics and Automation. ICRA, pp. 1887–1893.

Wu, Y., Wang, Y., Zhang, S., Ogai, H., 2021. Deep 3D object detection networks using LiDAR data: A review. IEEE Sens. J. 21 (2), 1152–1171. http://dx.doi.org/10.1109/JSEN.2020.3020626.

Wu, B., Zhou, X., Zhao, S., Yue, X., Keutzer, K., 2019. SqueezeSegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a LiDAR point cloud. In: Proceedings of the International Conference on Robotics and Automation. ICRA, pp. 4376–4382.

Xiang, B., Tu, J., Yao, J., Li, L., 2019. A novel octree-based 3D fully convolutional neural network for point cloud classification in road environment. IEEE Trans. Geosci. Remote Sens. 57 (10), 7799–7818. http://dx.doi.org/10.1109/TGRS.2019.2916625.

Yang, B., Wang, J., Clark, R., Hu, Q., Wang, S., Markham, A., Trigoni, N., 2019. Learning object bounding boxes for 3D instance segmentation on point clouds. In: Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems (NeurIPS). vol. 32, Curran Associates, Inc., URL https://proceedings.neurips.cc/paper/2019/file/d0aa518d4d3bfc721aa0b8ab4ef32269-Paper.pdf.

Yi, L., Kim, V.G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L., 2016. A scalable active framework for region annotation in 3D shape collections. ACM Trans. Graph. (ToG) 35 (6), 1–12. http://dx.doi.org/10.1145/2980179.2980238.

Yi, L., Zhao, W., Wang, H., Sung, M., Guibas, L., 2019. GSPN: Generative shape proposal network for 3D instance segmentation in point cloud. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 3942–3951.

Zamanakos, G., Tsochatzidis, L., Amanatiadis, A., Pratikakis, I., 2021. A comprehensive survey of LIDAR-based 3D object detection methods with deep learning for autonomous driving. Comput. Graph. 99, 153–181. http://dx.doi.org/10.1016/j.cag.2021.07.003.

Zhang, D., Chun, J., Cha, S., Kim, Y.M., 2020a. Spatial semantic embedding network: Fast 3D instance segmentation with deep metric learning. arXiv:arXiv:2007.03169.

Zhang, F., Guan, C., Fang, J., Bai, S., Yang, R., Torr, P.H.S., Prisacariu, V., 2020b. Instance segmentation of LiDAR point clouds. In: Proceedings of the IEEE International Conference on Robotics and Automation. ICRA, pp. 9448–9455.

Zhang, C., Luo, W., Urtasun, R., 2018. Efficient convolutions for real-time semantic segmentation of 3D point clouds. In: International Conference on 3D Vision (3DV). pp. 399–408. http://dx.doi.org/10.1109/3DV.2018.00053.

Zhang, B., Wonka, P., 2021. Point cloud instance segmentation using probabilistic embeddings. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 8883–8892.

Zhao, Y., Zhang, X., Huang, X., 2021. A technical survey and evaluation of traditional point cloud clustering methods for LiDAR panoptic segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. ICCVW, pp. 2464–2473.

Zhou, Z., Zhang, Y., Foroosh, H., 2021. Panoptic-PolarNet: Proposal-free LiDAR point cloud panoptic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 13194–13203.

Zhou, H., Zhu, X., Song, X., Ma, Y., Wang, Z., Li, H., Lin, D., 2020. Cylinder3D: An effective 3D framework for driving-scene LiDAR semantic segmentation. arXiv: arXiv:2008.01550.

Zhu, Q., Li, Y., Hu, H., Wu, B., 2017. Robust point cloud classification based on multi-level semantic relationships for urban scenes. ISPRS J. Photogramm. Remote Sens. 129, 86–102. http://dx.doi.org/10.1016/j.isprsjprs.2017.04.022.