Know What Your Neighbors Do: 3D Semantic Segmentation of Point Clouds

Francis Engelmann(✉), Theodora Kontogianni, Jonas Schult, and Bastian Leibe

RWTH Aachen University, Aachen, Germany
{engelmann,kontogianni,schult,leibe}@vision.rwth-aachen.de

Abstract. In this paper, we present a deep learning architecture which addresses the problem of 3D semantic segmentation of unstructured point clouds (Fig. 1). Compared to previous work, we introduce grouping techniques which define *point neighborhoods* in the initial world space and the learned feature space. Neighborhoods are important as they allow to compute local or global point features depending on the spatial extend of the neighborhood. Additionally, we incorporate dedicated loss functions to further structure the learned point feature space: the *pairwise distance loss* and the *centroid loss*. We show how to apply these mechanisms to the task of 3D semantic segmentation of point clouds and report state-of-the-art performance on indoor and outdoor datasets.

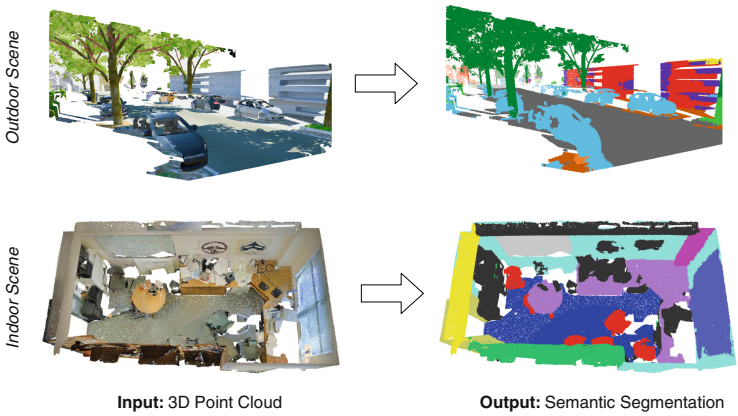

Fig. 1. We present a deep learning framework that predicts a semantic label for each point in a given 3D point cloud. The main components of our approach are *point neighborhoods* in different feature spaces and dedicated *loss functions* which help to refine the learned feature spaces. *Left*: point clouds from indoor and outdoor scenes. *Right:* semantic segmentation results produced by the presented method.

L. Leal-Taixé and S. Roth (Eds.): ECCV 2018 Workshops, LNCS 11131, pp. 395–409, 2019.
https://doi.org/10.1007/978-3-030-11015-4_29

1 Introduction

In the field of 3D scene understanding, semantic segmentation of 3D point clouds becomes increasingly relevant. Point cloud analysis has found its application in indoor scene understanding and more recently has become an essential component of outdoor applications [8]. This is due to the increasing availability and affordability of 3D sensors such as LiDAR or the Matterport scanner.

In the 2D image domain, for many tasks (including semantic segmentation) convolutional neural networks dominate the field. 2D convolutions allow processing large datasets with high resolution images by taking advantage of the locality of the convolutional operator. They reduce the number of model parameters allowing for deeper and more complex models while being efficient [3,17,19,30].

However point clouds have no inherent order, such as pixel neighborhoods. They are generally sparse in 3D space and the density varies with the distance to the sensor. Moreover, the number of points in a cloud can easily exceed the number of pixels in a high resolution image by multiple orders of magnitude. All these properties make it difficult to process point clouds directly with traditional convolutional neural networks.

Recently, a lot of effort has been put into bridging the success from 2D scene understanding into the 3D world [7,16,20–25,27]. In this work, we aim to further narrow down the gap between 2D and 3D semantic scene understanding. The straightforward approach of applying CNNs in the 3D space is implemented by preprocessing the point cloud into a voxel representation first in order to apply 3D convolutions on that new representation [21]. However 3D convolutions have drawbacks. Memory and computational time grows cubicly on the number of voxels, restricting approaches to use coarse voxels grids. However, by doing so, one then introduces discretization artifacts (especially for thin structures) and loose geometric information such as point density. Methods directly operating on the point cloud representation (e.g. [20,22]) produce promising results. However, in these methods, the point neighborhoods over which point features are aggregated are either global [20] or defined in a static coarse-to-fine approach [22]. Either way, the inherent way of convolutions to capture the local structure has only been transferred in a limited fashion.

In this work, we propose to define neighborhoods in an adaptive manner that is primarily sensitive to the local geometry by using K-means clustering on the input point cloud features in the world space and secondly defining dynamic neighborhoods in the learned feature space using k nearest neighbors (kNN). Next comes the observation that a well structured feature space is essential for learning on point clouds. Thus, we add dedicated loss functions which help shaping the feature space at multiple locations in the network.

We showcase the effectiveness of our approach on the task of semantic segmentation of 3D point clouds. We present a comprehensive ablation study evaluating all introduced mechanisms. We apply our method in different scenarios: indoor data from the Stanford 3D Indoor Scene dataset [14] and ScanNet dataset [5] as well as outdoor data from the Virtual KITTI 3D dataset [7,9].

2 Related Work

Before the introduction of deep learning methods, there have been numerous traditional approaches [10,15,18,32] applied to the task of semantically labelling 3D point clouds. Since then, methods relying on deep learning can be roughly split into two groups: methods that impose structure on the unstructured 3D point cloud (by voxelization or projection) followed by standard convolutions, and methods that operate directly on the 3D point clouds:

Voxelized Methods. Up until recently, the standard method to perform semantic segmentation of 3D data involved *voxelization*. Voxelization approaches transform the unstructured 3D point clouds into regular volumetric 3D grids (*voxels*). By doing so, 3D convolutions can be directly applied to the voxels [6,29]. Alternatively, *projection* approaches map the 3D points into 2D images as seen by virtual cameras. Then, 2D convolutions are applied on the projections [2,21]. These methods suffer from major disadvantages: the mapping from a sparse representation to a dense one leads to an increased memory footprint. Moreover, the fixed grid resolution results in discretization artifacts and loss of information.

Point Cloud Methods. A new set of methods started with the work of PointNet [20]. PointNet operates directly on 3D points. The key idea is the extraction of point features through a sequence of MLPs processing the points individually (point features) followed by a max-pooling operation that describes the points globally (global features). Point- and global-representations are fused (concatenation + MLP) before making the final label predictions. Many methods followed the PointNet paradigm to operate directly on point clouds. Where PointNet partitions the space into cuboidal blocks of fixed arbitrary size, others use octrees [26] or kd-trees [24] to partition the space in a more meaningful way. Furthermore, PointNet does not take into consideration the local geometry and surface information. Clustering has been used in many classical approaches as a way of imposing structure, mostly as a prepossessing step [31,32]. So [22,24] were introduced trying to apply hierarchical grouping of the points and incorporate local structure information. The former used farthest point sampling and the latter kd-trees. The authors of [25] generalize the convolution operator on a spatial neighborhood. Taking it further from local neighborhoods, [16] organizes the points into superpoints of homogeneous elements and defines relationships between them with the use of graph neural networks on their so-called superpoint graph. In [7] also cuboidal blocks are used, which act as superpoints and update their respective global features based on the surrounding blocks in space or scale using GRUs.

We now compare our method to the recent PointNet++ [22] which is an hierarchical extension of the original PointNet [22]. PointNet (PN) globally aggregates point features using max-pooling. PN++ forms local groups (or neighborhoods) based on the metric world space and collapses each group onto a single

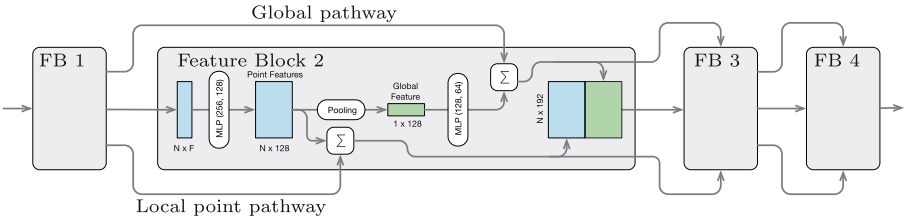

Fig. 2. The first component of our model is a *feature network*, depicted above. It learns high-dimensional features using a series of *feature blocks*. The details of a feature block are shown for feature block 2, the others are equal. Further details and a motivation for the architecture are given in Sect. 3.1. Our complete model is shown in Fig. 5.

representative point. This technique is repeated to increase the receptive field in each iteration. In our work, we follow a similar approach by iteratively applying feature-space neighborhoods (N_F-modules, introduced later): In every $\mathcal{N}_F$ iteration, each point is updated with the aggregated feature information from its kNNs. Repeating this procedure allows the information to flow over many points, one hop per iteration. Unlike [22], we build the neighborhood based on the feature space, this allows the network to learn the grouping. In PN++, neighborhoods are statically defined by a metric distance.

Feature Networks. As a first step on our network, we learn strong features using a dedicated feature network. The idea of extracting initial strong features is prominent in the field: In [23], features are learned in 2D image space using CNN for the task of 2.5 semantic segmentation. For the task of object detection, VoxelNet [33] uses a cascade of PointNet like architectures named *Voxel Feature Encoding* to obtain a more meaningful feature representation.

3 Our Approach

In the following, we describe the main components of our network. Starting from the initial point features (e.g. position and color), we learn more powerful feature representations using a new *Feature Network* as described in Sect. 3.1. Next, we define two kinds of neighborhoods (Sect. 3.2) within the point cloud, one defined on the learned feature space and one on the input world space. Based on these groupings, we learn regional descriptors which we use to inform the feature points about their neighborhood. Finally, we further enforce structure on the learned feature space by defining two dedicated loss functions (Sect. 3.3).

3.1 Feature Network

In this section, we describe our simple yet powerful architecture to learn point features. The goal of this component is to transform input features - such as

position and color - into stronger learned features. It can be seen as the distilation of important elements from previous works, in particular *PointNet* [20] and *ConsolidationUnits* [7]. A schematic visualization is shown in Fig. 2.

The network is built from a sequence of *feature blocks*. Each feature block performs the following tasks: Starting from a set of N points, each with feature dimension F, it produces refined *point features* by passing the incoming features F through a multi layer perceptron (MLP). Then, a global representation is computed by aggregating all point features using max-pooling. This *global feature* is again passed through an MLP. Finally, after vertically stacking the global feature N times, it is concatenated with the point features. Thus, a single feature block corresponds to a simplified PointNet. An important distinction is that feature blocks can be stacked to arbitrary depth.

In addition to the feature blocks, we introduce pathway connections which allow the individual feature blocks to consult features from previous layers. We distinguish between the point features (local point pathway) and global features (global pathway). Inspired by DenseNet [12] and ResNet [11], these features can be combined either by concatenation or summation. Our findings are that concatenation gives slightly inferior results over addition with the cost of a higher memory footprint. At the same time, increasing the number of feature blocks in the network is even more important. Thus, in the interest of scalability, in our final feature network we prefer addition over concatenation and use 17 *feature blocks*. Our experiments on different number of feature blocks and aggregation functions are summarized in Table 6. As a result, the feature network provides us with strong features required for the subsequent components.

3.2 Neighborhoods

We employ two different grouping mechanism to define neighborhoods over the point cloud: The *feature space neighborhood* $\mathcal{N}_F$ is obtained by computing the k nearest neighbors (kNN) for each point in the learned feature space, and the *world space neighborhood* $\mathcal{N}_W$ is obtained by clustering points using K-means in the world space. In this context, the world space corresponds to the features of the input point cloud, such as position and color. In the following, we explain the two neighborhoods in more detail and show how they are used to update each point feature.

Feature Space Neighborhood $\mathcal{N}_F$ (See Fig. 3)**.** From the set of N input features of dimensionality F, we compute an $N \times N$ similarity matrix based on the pairwise L_1-distance between the feature points $\mathbf{x}$. We concatenate the features of each point with the features of its k nearest neighbors to construct a kNN tensor. Each slice in the tensor corresponds to an $\mathcal{N}_F$-neighborhood of a feature point $\mathbf{x}_i$. Next, we learn a representation of this neighborhood using an MLP and we generate the updated feature point by applying max-pooling. This procedure is equivalent for each input feature and can be efficiently implemented using convolutions. We will refer to this architecture as an $\mathcal{N}_F$-module.

As such, an $\mathcal{N}_F$-module updates the local feature of a point based on its neighborhood in the feature space. By concatenating multiple $\mathcal{N}_F$-modules, we

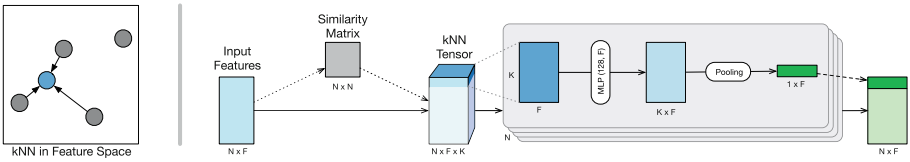

Fig. 3. The *feature space neighborhood* $\mathcal{N}_F(\mathbf{x})$ of a point $\mathbf{x} \in \mathbb{R}^F$ in a F-dimensional feature space is defined as the k *nearest neighbors* (kNN) in the feature space. *Left:* Example for $k = 3$. The point features $\mathbf{x}$ (blue) are updated based on the point features in the $\mathcal{N}_F$ neighborhood. *Right:* Details of a $\mathcal{N}_F$-module for learning point features. (Color figure online)

can increase the receptive field of the operation, one hop at a time, which is comparable to applying multiple convolutions in the image space.

World Space Neighborhood $\mathcal{N}_W$. Unlike kNN, K-means assigns a variable number of points to a neighborhood. K-means clustering is an iterative method, it alternatively assigns points to the nearest mean which represents the cluster center. Then it recomputes the means based on the assigned points. When applied to the world space, K-means can be seen as a pooling operation which reduces the input space and increases the receptive field by capturing long-range dependencies. Additionally, we are offered a feature point representative per cluster by averaging over all cluster members in the feature space.

We use this functionality in the $\mathcal{N}_W$-module: we perform K-means clustering in the world space, and represent each cluster by the average over all feature points in the cluster. Next, we concatenate this average to all the feature points within the same cluster. We then again apply max-pooling which produces a regional descriptor for this cluster. A visualization is shown in Fig. 5.

3.3 Loss Functions

In this section, we define the loss function $\mathcal{L}$ that is minimized during the training of our network. The classification loss $\mathcal{L}_{class}$ at the end of our network is realized as the cross entropy between predicted per-class probabilities and one-hot encoded ground truth semantic labels. Beside the classification loss, we introduce two additional losses $\mathcal{L}_{pair}$ and $\mathcal{L}_{cent}$ which further help to shape the feature space. The final loss is computed as the sum: $\mathcal{L} = \mathcal{L}_{class} + \mathcal{L}_{pair} + \mathcal{L}_{cent}$.

Pairwise Similarity Loss $\mathcal{L}_{pair}$. So far, we assumed that points from the same semantic class are likely to be nearby in the feature space. The *pairwise similarity loss*, described in this section, explicitly enforces this assumption. Similar to [4], we notice that semantic similarity can be measured directly as a distance in the feature space. By minimizing pairwise distances, we can learn an embedding where two points sampled from the same object produce nearby points in the feature space. Equivalently, two points originating from different objects have a large pairwise distance in the feature space. This goal is illustrated with a 2D embedding in Fig. 4.

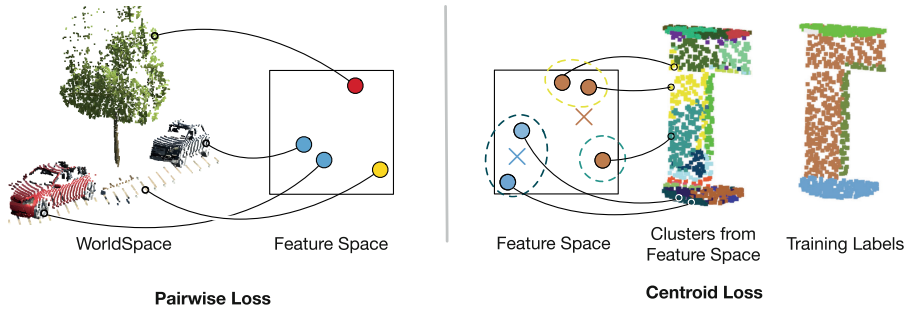

Fig. 4. *Left:* The pairwise distance loss $\mathcal{L}_{pair}$ minimizes the distance in the feature space between points of the same semantic class while it increases the distance between points of different classes. *Right:* The centroid loss $\mathcal{L}_{cent}$ minimizes the distance between features and their corresponding centroids, shown as crosses. The feature space is sketched as a 2D embedding. The point colors in the feature space represent training labels. To demonstrate the quality of our embedding, we further show clustering results (dashed lines) and their projection into world space (middle). See Sect. 3.3 for details. (Color figure online)

All we need is a pairwise distance matrix, which we already compute in the $\mathcal{N}_F$-module (Sect. 3.2). Hence, the distance loss is a natural extension of our network and comes at almost no additional memory cost. For a pair of points (i, j) with features $\mathbf{x}_i$ and $\mathbf{x}_j$, the loss is defined as follows:

$$\ell_{i,j} = \begin{cases} \max(||\mathbf{x}_i - \mathbf{x}_j|| - \tau_{\text{near}}, 0) & \text{if } \mathcal{C}_i = \mathcal{C}_j \\ \max(\tau_{\text{far}} - ||\mathbf{x}_i - \mathbf{x}_j||, 0) & \text{if } \mathcal{C}_i \neq \mathcal{C}_j \end{cases} \tag{1}$$

where τ_{near} and τ_{far} are threshold values and $\mathcal{C}_i$ is the semantic class of point i. Finally, the loss $\mathcal{L}_{pair}$ is computed as the sum over all pairwise losses $\ell_{i,j}$.

Centroid Loss $\mathcal{L}_{cent}$. This loss reduces the within-class distance by minimizing the distance between point features $\mathbf{x}_i$ and a corresponding representative feature $\overline{\mathbf{x}}_i$ (centroid). It makes the features in the feature space more compact. During training, the representative feature can be computed as the mean feature over all points from the same semantic class. An illustration is shown in Fig. 4. We define the centroid loss as the sum over all $(\mathbf{x}_i, \overline{\mathbf{x}}_i)$ pairs:

$$\mathcal{L}_{cent} = \sum_{i \in [1..N]} ||\mathbf{x}_i - \overline{\mathbf{x}}_i|| \tag{2}$$

where N is the total number of points. As distance measure $||\cdot||$, we found the cosine distance to be more effective than the L_1 or L_2 distance measures.

4 Implementation Details

In this section, we describe the integration of the aforementioned components into a deep learning architecture. The complete model is depicted in Fig. 5. We

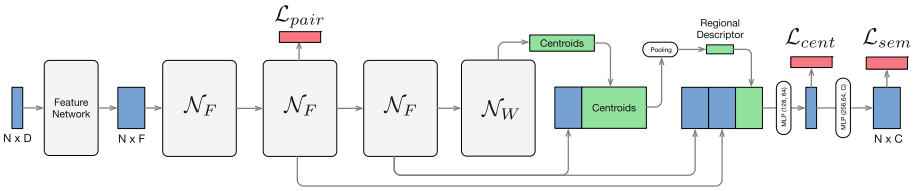

Fig. 5. Our complete network architecture. It consists of a *Feature Network* (Sect. 3.1), followed by three $\mathcal{N}_F$-modules and one $\mathcal{F}_W$-module (Sect. 3.2). Point features are represented by blue rectangles, losses are shown in red (Sect. 3.3). Green blocks represent features computed over clusters in the world space. (Color figure online)

start by learning strong features using our *Feature Network* (Sect. 3.1) producing F-dimensional features. See Table 6 for an evaluation and discussion on the architecture. We then feed these features into three stacked $\mathcal{N}_F$-modules. The subsequent $\mathcal{N}_W$-module computes a regional descriptors for each cluster (based on world space with descriptors form the feature space). We concatenate the regional descriptors to its corresponding feature points of the second and third $\mathcal{N}_F$-module. The concatenated features are passed through another MLP after which we compute the centroid loss. Finally, we reduce the feature points to 13 dimensions corresponding to the number of semantic classes in our datasets. The pairwise distance loss is computed in the beginning in the network. This informs the networks as early as possible which points should have similar features. This provides early layers a stronger signal of what should be learned and simplifies gradient propagation as the gradient is passed through fewer layers [1]. Although the distance loss could be appended at each point where a similarity matrix is computed, we found it most effective to add it to the second $\mathcal{N}_F$-module. An ablation study is provided in Table 1 and shows the contribution of each component in the performance.

Table 1. Ablation study highlighting the contribution of the individual components of our pipeline. The reported numbers refer to our validation set

Components	oAcc	mAcc	mIoU
Feature Network (FN) (Sect. 3.1)	82.43	56.96	47.25
$\mathcal{N}_F$*3 (Sect. 3.2)	81.70	55.14	47.37
$\mathcal{N}_F$*3 + $\mathcal{L}_{pair}$ (Sect. 3.3)	82.51	58.20	49.41
FN + $\mathcal{N}_F$*3 + $\mathcal{L}_{pair}$	83.84	58.29	50.56
FN + $\mathcal{N}_F$*3 + $\mathcal{N}_W$ + $\mathcal{L}_{pair}$ (Sect. 3.2)	84.31	59.18	50.95
FN + $\mathcal{N}_F$*3 + $\mathcal{N}_W$ + $\mathcal{L}_{pair}$ + $\mathcal{L}_{cent}$ (Sect. 3.3)	84.19	60.59	51.56

Table 2. Stanford Large-Scale 3D Indoor Spaces. 6-fold cross validation results. We can present state-of-the-art results in the more difficult CV and slightly inferior results on Area 5.

S3DIS [14]: 6-fold CV	oAcc	mAcc	mIoU
PointNet [20]	78.62	–	47.71
G+RCU [7]	81.1	66.4	49.7
SPG [16]	82.90	64.45	54.06
DGCNN [28]	**84.1**	–	56.1
PointNet++ [22]	81.03	67.05	54.49
RSN [13]	-	66.45	56.47
Ours	83.95	**67.77**	**58.27**

5 Evaluation

In this section, we evaluate the performance of our approach on the task of 3D semantic segmentation. We show qualitative and quantitative results and compare them to previous methods. We evaluate our method on multiple datasets: two indoor and one outdoor dataset showing the versatility of our approach. For each dataset, we report the *overall accuracy* (oAcc), the *mean class accuracy* (mAcc) and the *mean class intersection-over-union* (mIoU).

5.1 Indoor Datasets

We evaluate our model on the *Stanford Large-Scale 3D Indoor Spaces* (S3DIS) dataset [14] and the *ScanNet* dataset [5]. Both datasets have recently become popular to evaluate 3D semantic segmentation methods. The S3DIS dataset consists of 6 different indoor areas, totaling to 272 rooms. Each point is labeled as one of 13 semantic classes as shown in Fig. 6. The ScanNet dataset contains 1523 RGB-D scans labeled with 20 different semantic classes.

5.2 Outdoor Dataset

We apply our approach to the vKITTI3D dataset, a large-scale outdoor data set in an autonomous driving setting. Introduced in [7], this dataset is an adaptation of the synthetic *Virtual KITTI* dataset [9] for the task of semantic segmentation of 3D point clouds. It is split in 6 different sequences containing 13 semantic classes listed in Fig. 7.

5.3 Training Details

For the experiments on the S3DIS dataset, we follow a similar training procedure as [20] i.e. we split the rooms in blocks of 1 m^2 on the ground plane. From each block we randomly sample 4096 points. During evaluation, we predict class labels

Table 3. Stanford Large-Scale 3D Indoor Spaces. Results on Area 5

S3DIS [14]: Area 5	oAcc	mAcc	mIoU
PointNet [20]	–	48.98	41.09
MS + CU(2) [7]	-	52.11	43.02
G + RCU [7]	-	54.06	45.14
SEGCloud [27]	–	57.35	48.92
SPG [16]	**85.14**	**61.75**	**54.67**
Ours	84.15	59.10	52.17

Table 4. IoU per semantic class on the S3DIS dataset. We compare our model against the original PointNet and other recent methods. On average our method outperforms the current state-of-the-art by a large margin, specifically on 'bookcase' and 'board' while being slightly worse on 'beam' and 'sofa'

Method	mIoU	Ceiling	Floor	Wall	Beam	Column	Window	Door	Table	Chair	Sofa	Bookcase	Board	Clutter
PointNet [20]	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	54.1	42.0	9.6	38.2	29.4	35.2
MS+CU(2) [7]	47.8	88.6	95.8	67.3	36.9	24.9	48.6	52.3	51.9	45.1	10.6	36.8	24.7	37.5
SegCloud [27]	48.9	90.1	**96.1**	69.9	0.0	18.4	38.4	23.1	**75.9**	**70.4**	**58.4**	40.9	13.0	42.0
G+RCU [7]	49.7	90.3	92.1	67.9	**44.7**	24.2	**52.3**	51.2	58.1	47.4	6.9	39.0	30.0	41.9
SPG [16]	54.1	**92.2**	95.0	72.0	33.5	15.0	46.5	60.9	65.1	69.5	56.8	38.2	6.9	51.3
Ours	**58.3**	92.1	90.4	**78.5**	37.8	**35.7**	51.2	**65.4**	64.0	61.6	25.6	**51.6**	**49.9**	**53.7**

for all points. Additionally, we add translation augmentation to the block positions. Each point is represented by a 9D feature vector $[x, y, z, r, g, b, x', y', z']$ consisting of the position $[x, y, z]$, color $[r, g, b]$ and normalized coordinates $[x', y', z']$ as in [20]. The hyperparameters of our method are set as follows: for the kNN-clustering, we set $k = 30$ and use the L_1-distance measure. For K-means we dynamically set $\mathrm{K} = \lfloor \mathrm{N}/52 \rfloor$ where N is the number of points per block. We report scores on a 6-fold cross validation across all areas in Table 2 along with the detailed scores of per class IoU in Table 4. Additionally, we provide scores for Area 5 in Table 3 to compare ourself to [16,27].

On the ScanNet dataset [5], we use the reference implementation of PointNet++ to train and evaluate our model. This approach allows us to focus on the comparison of the models while abstracting from the training procedures. All hyperparameters remain the same. The results are shown in Table 5.

Table 5. ScanNet. Overall point accuracy (oAcc), mean semantic class accuracy (mAcc), mean Intersection-over-Union (mIoU). ScanNet dataset using the official training and test split from [5], scores are shown on a per-point basis as computed by the PN++ reference implementation. To train on our hardware, we set the batch size to 32 and number of points to 1024

ScanNet [5]	oAcc	mAcc
PointNet++ [22]	71.40	24.51
Our method	**75.53**	**25.39**

Table 6. Study on the feature network. We evaluate the number of layers and compare feature fusion using concatenation or addition. Deeper networks perform better, in general feature addition is slightly stronger while being more memory efficient than concatenation

# Layers	Fusion	mIoU
3	additive	42.15
12	additive	44.46
17	additive	**45.15**
17	concat	44.23
12	concat	43.35
3	concat	41.73

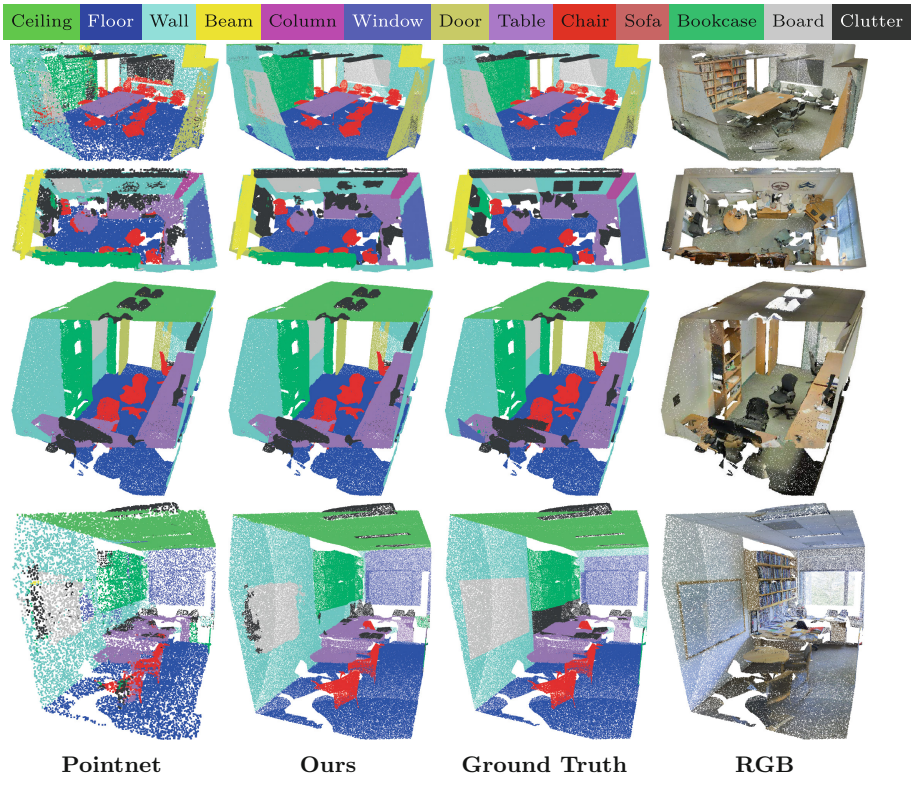

Fig. 6. Qualitative results on S3DIS dataset. We show three exemplary rooms. Our method provides segmentations of objects with minimal noise and clear boundaries. As pointed out in the qualitative results, our method performs quite well in challenging objects like 'board' and 'bookcase'. (Color figure online)

On the VKITTI3D dataset, we follow again the same training procedure as on the S3DIS dataset. The scenes are split into blocks of 9 m^2 on the ground plane. Since the dataset is much sparser, we set the number of points sampled per block to N = 256. Training on a 6-fold cross validation is performed as in [7]. We use the same input features as in the indoor dataset and additionally, we analyze how well our method performs if we take into consideration only geometric features (xyz-position) while leaving out color information. This is an interesting experiment, as color is not always imminently available e.g. point clouds from laser scanners. We show quantitative results in Table 7. Qualitative results are shown in Fig. 7.

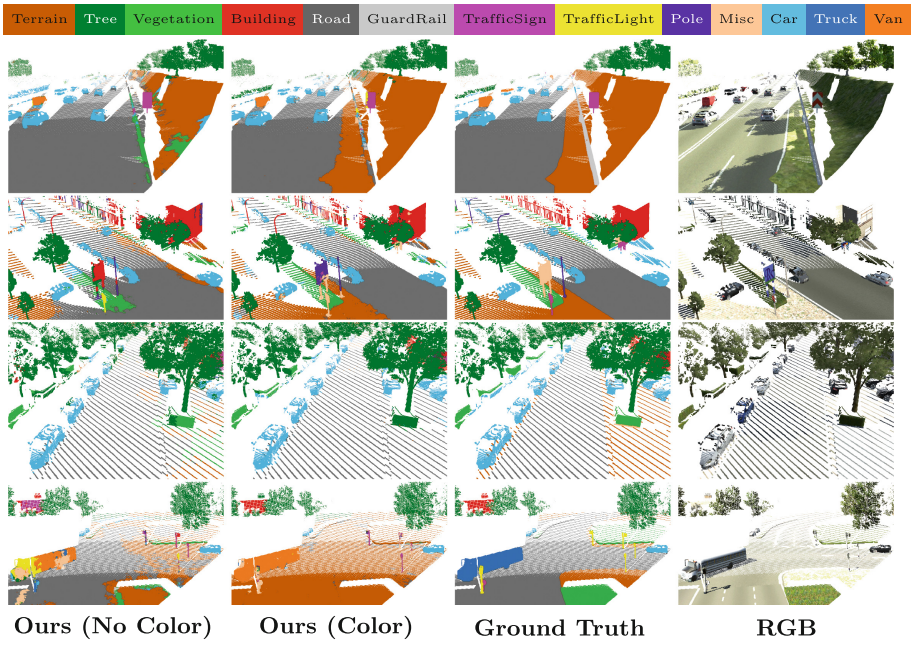

Fig. 7. Qualitative results on VKITTI3D dataset. In general, color is an important attribute to distinguish between shapes that have similar structure e.g. 'terrain' and 'road'. The last row shows a failure case, during training our model was not able to differentiate between 'Van' and 'Truck', and between 'Terrain' and 'Road'.

Table 7. Virtual KITTI 3D. The upper part of the tables shows results trained on position only. In the lower part, we additionally trained with color. Geometric features alone are quite powerful. Adding color helps to differ between geometric similar classes

VKITTI3D [7]: 6-fold CV	oAcc	mAcc	mIoU
PointNet [20] from [7]	63.3	29.9	17.9
MS+CU(2) [7]	73.2	40.9	26.4
Ours	**78.19**	**56.43**	**33.36**
Ours (+ color)	**79.69**	**57.59**	**35.59**

6 Conclusion

We have presented a deep learning framework for 3D semantic segmentation of point clouds. Its main components are $\mathcal{N}_F$- and $\mathcal{N}_W$-modules. They allow to incorporate neighborhood information from the feature space and from the world space. We have also introduced the pairwise distance loss $\mathcal{L}_{pair}$ and the centroid loss $\mathcal{L}_{cent}$ in the context of 3D semantic segmentation. The presented method produces state-of-the-art results on current indoor and outdoor datasets.

Acknowledgement. This project was funded by the ERC Consolidator Grant DeeViSe (ERC-2017-CoG-773161).

References

1. Harley, A.W., Konstantinos, G., Derpanis, I.K.: Segmentation-aware convolutional networks using local attention masks. In: IEEE International Conference on Computer Vision (ICCV) (2017)
2. Boulch, A., Saux, B.L., Audebert, N.: Unstructured point cloud semantic labeling using deep segmentation networks. In: Eurographics Workshop on 3D Object Retrieval (2017)
3. Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. arXiv preprint arXiv:1606.00915 (2016)
4. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005)
5. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: richly-annotated 3D reconstructions of indoor scenes. In: Proceding of Computer Vision and Pattern Recognition (CVPR). IEEE (2017)
6. Maturana, D., Scherer, S.: VoxNet: a 3D convolutional neural network for real-time object recognition. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2015)
7. Engelmann, F., Kontogianni, T., Hermans, A., Leibe, B.: Exploring spatial context for 3D semantic segmentation of point clouds. In: IEEE International Conference on Computer Vision, 3DRMS (ICCV) Workshop (2017)
8. Engelmann, F., Stückler, J., Leibe, B.: SAMP: shape and motion priors for 4d vehicle reconstruction. In: IEEE Winter Conference on Applications of Computer Vision, WACV (2017)
9. Gaidon, A., Wang, Q., Cabon, Y., Vig, E.: Virtual worlds as proxy for multi-object tracking analysis. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
10. Hackel, T., Wegner, J.D., Schindler, K.: Fast semantic segmentation of 3D points clouds with strongly varying density. ISPRS **3**(3), 177–184 (2016)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
12. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)

13. Huang, Q., Wang, W., Neumann, U.: Recurrent slice networks for 3D segmentation on point clouds. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
14. Iro, A., et al.: 3D Semantic Parsing of Large-Scale Indoor Spaces. In: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
15. Lai, K., Bo, L., Fox, D.: Unsupervised feature learning for 3D scene labeling. In: IEEE International Conference on Robotics and Automation (ICRA) (2014)
16. landrieu, L., Simonovsky, M.: Large-scale point cloud semantic segmentation with superpoint graphs. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
17. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
18. Munoz, D., Vandapel, N., Hebert, M.: Directional associative Markov network for 3-D point cloud classification. In: International Symposium on 3D Data Processing, Visualization and Transmission (2008)
19. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: IEEE International Conference on Computer Vision (ICCV) (2015)
20. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: deep learning on point sets for 3D classification and segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
21. Qi, C.R., Su, H., Niener, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view CNNs for object classification on 3D data. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
22. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: deep hierarchical feature learning on point sets in a metric space. In: Conference on Neural Information Processing Systems (NIPS) (2017)
23. Qi, X., Liao, R., Ya, J., Fidler, S., Urtasun, R.: 3D graph neural networks for RGBD semantic segmentation. In: IEEE International Conference on Computer Vision (ICCV) (2017)
24. Roman, K., Victor, L.: Escape from cells: deep Kd-networks for the recognition of 3D point cloud models. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
25. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
26. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Octree generating networks: efficient convolutional architectures for high-resolution 3D outputs (2017)
27. Tchapmi, L.P., Choy, C.B., Armeni, I., Gwak, J., Savarese, S.: SEGCloud: semantic segmentation of 3D point clouds. In: International Conference on 3D Vision (3DV) (2017)
28. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. arXiv preprint arXiv:1801.07829 (2018)
29. Wu, Z., et al.: 3D ShapeNets: a deep representation for volumetric shape modeling. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
30. Wu, Z., Shen, C., van den Hengel, A.: High-performance semantic segmentation using very deep fully convolutional networks. arXiv preprint arXiv:1604.04339 (2016)
31. Xiong, X., Munoz, D., Andrew, J., Hebert, B.M.: 3-D scene analysis via sequenced predictions over points and regions. In: IEEE International Conference on Robotics and Automation (ICRA) (2011)

32. Xiong, X., Munoz, D., Bagnell, J.A., Hebert, M.: 3-D scene analysis via sequenced predictions over points and regions. In: IEEE International Conference on Robotics and Automation (ICRA) (2011)
33. Zhou, Y., Tuzel, O.: VoxelNet: end-to-end learning for point cloud based 3D object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)