UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

3DContextNet: K-d Tree Guided Hierarchical Learning of Point Clouds Using Local Contextual Cues

Zeng, W.; Gevers, T.

DOI
10.1007/978-3-030-11015-4_24

Publication date
2019

Document Version
Author accepted manuscript

Published in
Computer Vision – ECCV 2018 Workshops

Link to publication

Citation for published version (APA):
Zeng, W., & Gevers, T. (2019). 3DContextNet: K-d Tree Guided Hierarchical Learning of Point Clouds Using Local Contextual Cues. In L. Leal-Taixé, & S. Roth (Eds.), *Computer Vision – ECCV 2018 Workshops: Munich, Germany, September 8-14, 2018 : proceedings* (Vol. III, pp. 314-330). (Lecture Notes in Computer Science; Vol. 11131). Springer. https://doi.org/10.1007/978-3-030-11015-4_24

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:14 Nov 2026

This ECCV 2018 workshop paper, provided here by the Computer Vision Foundation, is the author-created version. The content of this paper is identical to the content of the officially published ECCV 2018 LNCS version of the paper as available on SpringerLink: https://link.springer.com/conference/eccv

3DContextNet: K-d Tree Guided Hierarchical Learning of Point Clouds Using Local and Global Contextual Cues

Wei Zeng, Theo Gevers

Computer Vision Lab, University of Amsterdam
w.zeng@uva.nl, th.gevers@uva.nl

Abstract. Classification and segmentation of 3D point clouds are important tasks in computer vision. Because of the irregular nature of point clouds, most of the existing methods convert point clouds into regular 3D voxel grids before they are used as input for ConvNets. Unfortunately, voxel representations are highly insensitive to the geometrical nature of 3D data. More recent methods encode point clouds to higher dimensional features to cover the global 3D space. However, these models are not able to sufficiently capture the local structures of point clouds.
Therefore, in this paper, we propose a method that exploits *both* local and global contextual cues imposed by the k-d tree. The method is designed to learn representation vectors progressively along the tree structure. Experiments on challenging benchmarks show that the proposed model provides discriminative point set features. For the task of 3D scene semantic segmentation, our method significantly outperforms the state-of-the-art on the Stanford Large-Scale 3D Indoor Spaces Dataset (S3DIS).

Keywords: Point Clouds · K-d Tree Structure · Contextual Cues · Hierarchical Learning

1 Introduction

Over the past few years, ConvNets have achieved excellent performance in different computer vision tasks such as image classification [17, 27, 16], object detection [11, 10, 25] and semantic segmentation [11, 3, 18, 21].

3D imaging technology has also experienced a major progress. In parallel, a number of annotated large-scale 3D datasets have become publicly available, which are crucial for supervised 3D deep learning models. For example, ModelNet [32] and ShapeNet [7] provide object-level man-made 3D models, whereas Stanford Large-Scale 3D Indoor Spaces Dataset [2] and ScanNet [8] are available as real 3D scene datasets.

Most of the traditional work convert the irregular 3D data (point clouds) to regular formats like 2D projection images [30, 29, 23] or 3D voxel grids [32, 23, 20] as a pre-processing step. Methods that employ 2D image projections of 3D models as their input, such as [30, 29], are well suited as inputs to 2D

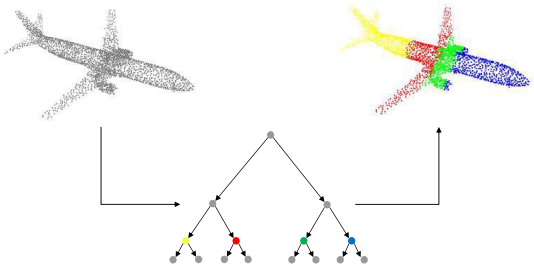

Fig. 1. Example of the implicit 3D space partition of a k-d tree. Colors of different local parts indicate different corresponding nodes in the k-d tree structure

ConvNet architectures. However, the intrinsic 3D geometrical information is distorted by the 3D-to-2D projection. Hence, this type of methods are limited by the exploitation of 3D spatial connections between regions. While it might seem straightforward to extend 2D CNNs to process 3D data by utilizing 3D convolutional kernels, data sparsity and computational complexity are the restrictive factors of this type of approaches [32, 20, 28, 5].

To fully exploit the 3D nature of point clouds, in this paper, the goal is to use the k-d tree structure [4] as the 3D data representation model, see Figure 1. Our method consists of two parts: *feature learning* and *aggregation*. The model exploits both local and global contextual information and aggregates point features to obtain discriminative 3D signatures in a hierarchical manner. In the feature learning stage, local patterns are identified by the use of an adaptive feature recalibration procedure, and global patterns are calculated as non-local responses of different regions at the same level. Then, in the feature aggregation stage, point features are merged hierarchically corresponding to the associated k-d tree structure in bottom-up fashion.

Our main contributions are as follows: (1) a novel 3D context-aware neural network is proposed for 3D point cloud feature learning by exploiting the implicit partition space of the k-d tree structure, (2) a novel method is presented to incorporate both local and global contextual information for point cloud feature learning, (3) for semantic segmentation, our method significantly outperforms the state-of-the-art on the challenging Stanford Large-Scale 3D Indoor Spaces Dataset(S3DIS) [2].

2 Related Work

Previous work on ConvNets and volumetric models use different rasterization strategies. Wu et al. propose 3DShapeNets [32] using 3D binary voxel grids as input of a Convolutional Deep Belief Network. This is the first work to use

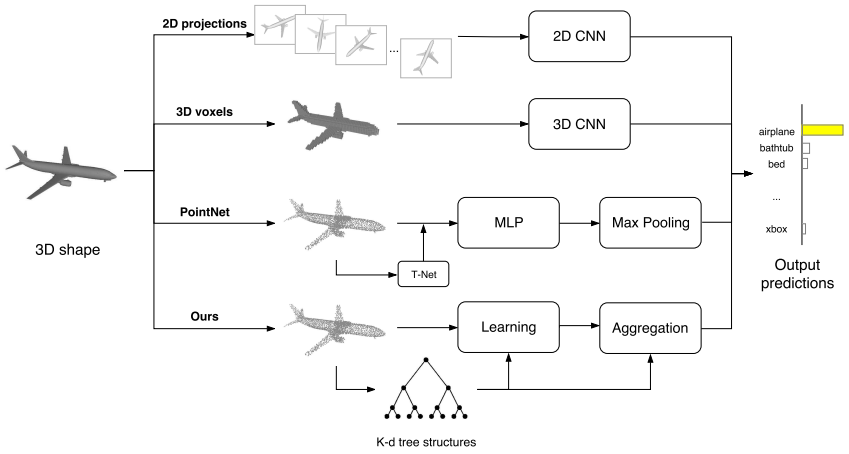

Fig. 2. Comparison to related work for the classification task. Our model is based on hierarchical feature learning and aggregation using the k-d tree structure

deep ConvNets for 3D data processing. VoxNet [20] proposes a 3D ConvNet architecture to integrate the 3D volumetric occupancy grid. ORION [28] exploits the 3D orientation to improve the results of voxel nets for 3D object recognition. Based on the ResNet [12] architecture, Voxception-ResNet (VRN) [5] proposes a very deep architecture. OctNet [26] exploits the sparsity in the input data by using a set of unbalanced octrees where each leaf node stores a pooled feature representation. However, most of the volumetric models are limited by their resolution, data sparsity, and computational cost of 3D convolutions.

Other methods rely on 2D projection images to represent the original 3D data and then apply 2D ConvNets to classify them. MVCNN [30] uses 2D rendered images of 3D shapes to learn representations of multiple views of a 3D model and then combines them to compute a compact descriptor. DeepPano [29] converts each 3D shape to a panoramic view and uses 2D ConvNets to build classifiers directly from these panoramas. With well-designed ConvNets, this type of methods (2D projections from 3D) performs successfully in different shape classification and retrieval tasks. However, due to the 3D-to-2D projection, these methods are limited in exploring the full 3D nature of the data. In addition, [6, 19] exploits ConvNets to process non-Euclidean geometries. Moreover, Geodesic Convolutional Neural Networks (GCNN) [19] apply linear and non-linear transformations to polar coordinates in a local geodesic system. However, these methods are limited to manifold meshes.

Only recently, a number of methods are proposed that apply deep learning directly to the raw 3D data (point clouds). PointNet [22] is the pioneering work that directly processes 3D point sets in a deep learning setting. Nonetheless, since every point is treated equally, this approach fails in retaining the full 3D information. The modified version of PointNet, PointNet++ [24], abstracts local patterns by sampling representative points and recursively applies Point-

Net [22] as a learning component to obtain the final representation. However, it directly discards the unselected points after each layer, and needs to sample points recursively at different scales which may yield relatively slow inference speed. Another recent work, Kd-Network [15] uses a 3D indexing structure to perform the computation. The method employs parameter sharing and calculates representations from the leaf nodes to the roots. However, this method needs to sample the point clouds and to construct k-d trees for every iteration. Further, the method employs multiple k-d trees to represent a single object. It is split-direction-dependent and is negatively influenced by a change in rotation (3D object classification) and viewpoint (3D scene semantic segmentation).

In contrast to previous methods, our model is based on a hierarchical feature learning and aggregation pipeline. Our neural network structure exploits the local and global contextual cues which are inferred by the implicit space partition of the k-d tree. In this way, our model learn features, and calculates the representation vectors progressively using the associated k-d tree. Figure 2 shows a comparison of related methods to our work for the classification task.

3 Method

In this section, we describe our architecture, *3DContextNet*, see Figure 3. First, the choice of the tree structure is motivated to subdivide the 3D space. Then, the feature learning stage is discussed that uses both local and global contextual cues to encode the point features. Finally, the feature aggregation stage is described that computes representation vectors progressively along the k-d trees.

3.1 K-d Tree Structure: Implicit 3D Space Partition

Our method is designed to capture both the local and global context by learning and aggregating point features progressively and hierarchically. Therefore, a representation model is required to partition 3D point clouds to encapsulate the latent relations between regions. To this end, the k-d tree structure [4] is chosen.

A k-d tree is a space partitioning structure which is constructed by recursively computing axis-aligned hyperplanes to divide point sets. In this paper, we choose the standard k-d tree construction to obtain balanced k-d trees from the 3D input point clouds/sets. The latent region subdivisions of the constructed k-d tree are used to capture the local and global contextual information of point sets. Each node, at a certain level, represents a local region at the same scale, whereas nodes at different levels represent subdivisions at corresponding scales. In contrast to the k-d network of [15], splitting directions and positions are not used for the tree construction. In this way, our method is more robust to jittering and rotation than [15] which trains different affine transformations depending on the splitting directions of the nodes.

The k-d tree structure can be used to search for k-nearest neighbors for each point to determine the local point adjacency and neighbor connectivity. Our

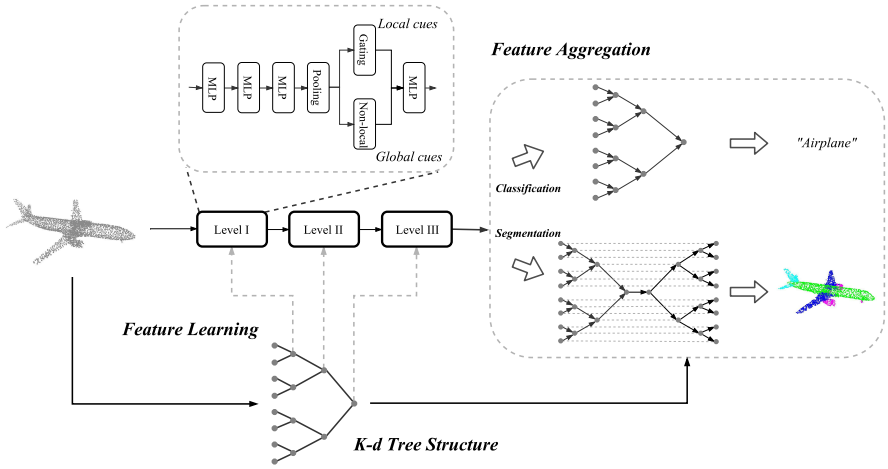

Fig. 3. 3DContextNet architecture. 3D object point clouds are used to illustrate that our method is suitable for both 3D classification and segmentation tasks. The corresponding nodes of the k-d tree determine the receptive fields at different levels. For feature learning, both local and global contextual information is encoded for each level. The associated k-d tree forms the computational graph to compute the representation vectors progressively for feature aggregation

approach uses the implicit local partitioning obtained by the k-d tree structure to determine the point adjacency and neighbor connectivity.

In general, conventional ConvNets learn and merge nearby features at the same time enlarging the receptive fields of the network. Because of the non-overlapping partitioning of the k-d tree structure, in our method, learning and merging at the same time would decrease the size of the remaining points too fast. This may lead to a lack of fine geometrical cues which are factored out during the early merging stages. To this end, our approach divides the network architecture into two parts: *feature learning* and *aggregation*.

3.2 Feature Learning Stage

Given as input is a 3D point set with the corresponding k-d tree. The tree leaves contain the individual (raw) 3D points with their representation vectors, denoted by $X = \{x_1, \ldots, x_n\} \subseteq R^F$. For example, $F = 3$ denotes the initial vectors containing the 3D point coordinates. Features are directly learned from the raw point clouds without any pre-processing step. According to [36], a function $S(X)$ is permutation invariant to the elements in X, if and only if it can be decomposed in the form of $\rho(\sum_{x \in X} \varphi(x))$, for a suitable transformation of ρ and φ. We follow PointNet [22], where a point set is mapped to a discriminative vector as follows:

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)), \tag{1}$$

where $f : 2^{\mathbb{R}^N} \rightarrow \mathbb{R}$, $h : \mathbb{R}^N \rightarrow \mathbb{R}^K$ and $g : \underbrace{\mathbb{R}^K \times \ldots \times \mathbb{R}^K}_{n} \rightarrow \mathbb{R}$ is a symmetric function.

In the feature learning stage, point features are computed at different levels hierarchically. For a certain level, we first process each point using shared multi-layer perceptron networks (MLP) as function h in equation (1). Then, different local region representations are computed by a symmetric function, max pooling in our work, for the subdivision regions at the same level, as function g in equation (1). Then, local and global contextual cues are calculated in parallel based on the local region representations. Note that both the local and global features are concatenated with the corresponding points to retain the number of points.

Local Contextual Cues: Adaptive Feature Recalibration To model the inter-dependencies between point features in the same region, we use the local region representations obtained from the symmetric function to perform adaptive feature recalibration [13]. All operations are adaptive to each local region, represented by a certain node in the k-d tree. The local region representation obtained by the symmetric function can be interpreted as a feature descriptor for the corresponding local region. A gating function is used with a sigmoid activation to capture the feature-wise dependencies. Point features in this local region are then rescaled by the activations to obtain the adaptive recalibrated output:

$$\tilde{y}_i = \sigma(g(Y)) \cdot y_i, \qquad i = 1, ..., m \tag{2}$$

where σ denotes the sigmoid activation and g is the symmetric function to obtain the local region representation. $Y = \{y_1, \ldots, y_m\}$ is the point feature set of the local region and m is the number of points in that region. In this way, feature dependencies are consolidated for each local region by enhancing informative features. As a result, we can obtaion more discriminative local patterns. Note that the activations act as feature weights and adaptively recalibrate point features for different local regions.

Global Contextual Cues: Non-local Responses Global contextual cues are based on the non-local responses to capture a greater range of dependencies. Intuitively, a non-local operation computes the response for one position as a weighted sum over the features for all positions in the input feature maps. A generic non-local operation [31] in deep neural networks is calculated by:

$$z_i = \frac{1}{C(x)} \sum_{\forall j} G(x_i, x_j) H(x_j), \tag{3}$$

where i is the index of the output position and j is the index that enumerates all possible positions. In our case, i represents a local region at a certain level and j enumerates the number of local regions at the same level. Function G denotes the relationships between i and j. Further, function H computes a representation

of the input signal at position j. Then, the response is normalized by a factor $C(x)$.

The k-d tree divides the input point set into different local regions. These are represented by different nodes of the tree. Larger range dependencies for different local regions at the same level are computed as non-local responses of the corresponding nodes of the tree. We consider H as an MLP, and the pairwise function G as an embedded Gaussian function:

$$G(x_i, x_j) = e^{\theta(x_i)^T \phi(x_j)}, \tag{4}$$

where $\theta(x_i)$ and $\phi(x_j)$ are two MLPs representing two embeddings. In this paper, the relationships between different nodes at the same level should be undirected, and hence $G(x_i, x_j) = G(x_j, x_i)$. Therefore, the two embeddings are the same i.e. $\theta = \phi$. The normalization factor is calculated by $C(x) = \sum_{\forall j} G(x_i, x_j)$. Note that this operation is different from a fully-connected layer. The non-local responses are based on the connections between different local regions, whereas fully-connected layers use learned weights.

Due to our input format and architecture, the receptive fields of the convolutional kernels are always 1×1 in the feature learning stage. Following DenseNet [14], to strengthen the information flow between layers, layers at the same level are connected (in the feature learning stage) with each other by concatenating all corresponding point features together. Such connections also lead to an implicit deep supervision which makes the network easier to train. The output of the feature learning stage has the same number of points as the input point set.

3.3 Feature Aggregation Stage

In the feature aggregation stage, the associated k-d tree structure is used to form the computational graph to progressively abstract over larger regions. For the classification task, the global signature is computed for the entire 3D model. For the semantic segmentation task, the outputs are the point labels. Instead of aggregating the information once over all points, the more discriminative features are computed in a bottom-up manner. The representation vector of a non-leaf node at a certain level is computed from its children nodes by MLPs and the symmetric function. To that end, max pooling is used as the symmetric function.

For classification, by using this bottom-up and hierarchical approach, more discriminative global signatures are obtained. This procedure corresponds to a ConvNet in which the representation of a certain location is computed from the representations of nearby locations at the previous layers by a series of convolutions and pooling operations. Our architecture is able to progressively capture features at increasingly larger scales. Features at lower levels have smaller receptive fields, whereas features at higher levels have larger receptive fields. That is due to the data-dependent partition of the k-d tree structure. Additionally, our model is invariant to the input order of the point sets, because the aggregating direction is along the k-d tree structure, which is invariant to input permutations.

For the semantic segmentation task, the k-d tree structure is used to represent an encoder-decoder architecture with skip connections to link the related layers. The input of the feature aggregation stage is the point feature set in which the representation of each point encapsulates both local and global contextual information at different scales. The output is a semantic label for each point.

In conclusion, our architecture fully utilizes the local and global contextual cues in the feature learning stage. It calculates the representation vectors hierarchically in the feature aggregation stage. Hence, with k-d tree guided hierarchical learning, our 3DContextNet can obtain discriminative features for point clouds.

3.4 Discussion

Our method is related to PointNet [22] which encodes the coordinates of each point to higher dimensional features. However, by its design, this method is not able to sufficiently capture the local patterns in 3D space. More recently, PointNet++ [24] is proposed which abstracts local patterns by selecting representative points in a metric space and recursively applies PointNet as a local feature learner to obtain features of the whole point set. In fact, the method handles the non-uniform point sampling problem. However, the set of abstraction layers need to sample the point sets multiple times at different scales which leads to a relative slow inference speed. Further, only the selected points are preserved. Others are directly discarded after each layer which causes the loss of fine geometric details. Another recent work, K-d network [15] performs linear and non-linear transformations and share the transformation parameters corresponding to the splitting directions of each node in the k-d tree. The input of this method is the constructed k-d trees. It needs to calculate the representation vectors for all the nodes of the associated tree structure. For each node at a certain level, the input is the representation vectors of the two previous nodes. The method heavily depends on the splitting direction of each node to train different multiplicative transformations at each level. Hence, the method is not invariant to rotation. Furthermore, point cloud sampling and k-d tree fitting during every iteration lead to slow training and inference speed.

3.5 Implementation Details

Our 3DContextNet model deals with point clouds of a fixed size $N = 2^D$ where D is the depth of the corresponding balanced k-d tree. Point clouds of different sizes can be converted to the same size using sub- or oversampling. In our experiments, not all the levels of the k-d tree are used. For simplicity and efficiency reasons, this number is $L = 3$ for both the feature learning and aggregation stage. The receptive fields (number of points) for each level in the feature learning stage are 32 - 64 - 128 for the classification tasks and 32 - 128 - 512 for the segmentation tasks.

In the feature learning stage, the sizes of the shared MLPs are (64, 64, 128, 128) - (64, 64, 256, 256) - (64, 64, 512, 512) for the three levels, respectively. The size of MLPs for θ and H are 64 - 128 - 256 and 128 - 256 - 512, respectively.

Table 1. 3D semantic segmentation results on the Stanford Large-Scale 3D Indoor Spaces Dataset (S3DIS). Our method outperforms previous state-of-the-art methods by a large margin

	mean IoU	overall accuracy	avg. class accuracy
Baseline [22]	20.1	53.2	-
PointNet [22]	47.6	78.5	66.2
MS + CU(2) [9]	47.8	79.2	59.7
G + RCU [9]	49.7	81.1	66.4
PointNet++ [24]	53.2	83.0	70.5
Ours	**55.6**	**84.9**	**74.5**

Dense connections are applied within each level before the max-pooling layer. In the feature aggregation stage, the MLPs and pooling operations are used recursively to progressively abstract the discriminative representations. For the classification task, the sizes of the MLPs are (1024) - (512) - (256), respectively. For the segmentation task, like the hourglass shape, the sizes of the MLPs are (1024) - (512) - (256) - (256) - (512) - (1024), respectively. The output is then processed by two fully-connected layers with size 256. Dropout is applied after each fully-connected layer with a ratio of 0.5.

4 Experiments

In this section, we evaluate our 3DContextNet on different 3D point cloud datasets. First, it is shown that our model significantly outperforms state-of-the-art methods for the task of semantic segmentation on the Stanford Large-Scale 3D Indoor Spaces Dataset [2]. Then, it is shown that our model provides competitive results for the task of 3D object classification on the ModelNet40 dataset [32] and the task of 3D object part segmentation on the ShapeNet part dataset [7].

4.1 3D Semantic Segmentation of Scenes

Our network is evaluated on the *Stanford Large-Scale 3D Indoor Spaces* (S3DIS) dataset [2, 1] for 3D semantic segmentation task. The dataset contains 6 large scale indoor areas and each point is labeled with one of the 13 semantic categories, including 5 types of furniture (*board*, *bookcase*, *chair*, *sofa* and *table*) and 7 building elements (*ceiling*, *beam*, *door*, *wall*, *window*, *column* and *floor*) plus *clutter*. We follow the same setting as in [22] and use a 6-fold cross validation over all the areas.

Our method is compared with the baseline by PointNet [22] and the recently introduced MS+CU and G+RCU models [9]. We also produce the results of PointNet++ [24] for this dataset. During training, we use the same preprocessing as in [22]. We first split rooms into blocks of $1m \times 1m$ and represent

Table 2. IoU per semantic class for the S3DIS dataset with $XYZ - RGB$ as input. It can be derived that our method obtains the state-of-the-art results in mean IoU and for most of the individual classes

	mean IoU	Ceiling	Floor	Wall	Beam	Column	Window	Door	Table	Chair	Sofa	Bookcase	Board	clutter
PointNet [22]	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) [9]	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
G + RCU [9]	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
PointNet++ [24]	53.2	90.2	91.7	73.1	42.7	21.2	49.7	42.3	62.7	**59.0**	19.6	**45.8**	**48.2**	45.6
Ours	**55.6**	**92.6**	93.1	**73.9**	**52.9**	**35.0**	**55.8**	**57.5**	**62.9**	49.0	**22.0**	42.8	39.8	**45.8**

Table 3. IoU per semantic class for the S3DIS dataset using only XYZ input features (no color/appearance). It is shown that our method provides comparable results in mean IoU and for all individual classes even without color/appearance information

	mean IoU	Ceiling	Floor	Wall	Beam	Column	Window	Door	Table	Chair	Sofa	Bookcase	Board	clutter
PointNet [22]	40.0	84.0	87.2	57.9	37.0	19.6	29.3	35.3	51.6	42.4	11.6	26.4	12.5	25.5
MS + CU(2) [9]	43.0	86.5	**94.9**	58.8	37.7	25.6	28.8	36.7	47.2	46.1	18.7	30.0	16.8	31.2
PointNet++ [24]	47.0	88.0	92.4	**64.7**	37.7	16.8	31.0	**41.1**	**59.6**	**52.0**	**29.4**	**42.2**	**19.2**	36.9
Ours	**48.6**	**90.5**	92.8	63.6	**49.4**	**31.2**	**44.2**	37.8	**59.6**	50.6	17.7	38.7	17.3	**37.9**

each point by a 9-dimensional vector containing coordinates (x, y, z), the color information RGB and the normalized position (x', y', z'). The baseline extracts the same 9-dim local features and three additional ones: local point density, local curvature and normals. The standard MLP is used as the classifier. PointNet [22] computes the global point cloud signature and feeds it back to per point features. In this way, each point representation incorporates both local and global information. Recent work by [9] proposes two models that enlarge the receptive field over the 3D scene. The motivation is to incorporate both the input-level context and the output-level context. MS+CU represents the multi-scale input block with a consolidation unit model, while G+RCU stands for the grid-blocks in combination with a recurrent consolidation block model. PointNet++ [24] exploits metric space distances to build a hierarchical grouping of points and abstracts the features progressively. Results are shown in Table 1. A significance test is conducted between our results and the state-of-the-art results obtained by PointNet++ [24]. The p-value equals to 0.0122 in favor of our method.

We also compare the mean IoU for each semantic class with $XYZ-RGB$ and only with XYZ as input, see Table 2 and Table 3 respectively. We obtain state-of-the-art results in mean IoU and for most of the individual classes for both $XYZ - RGB$ and XYZ input. The reason of obtaining comparable results with PointNet++ [24] for furnitures is that the k-d tree structure is computed along the axes. Therefore, it may be inefficient for precise prediction near the splitting boundaries, especially for relatively small objects. Note that our model using only geometry information (i.e. XYZ) achieves better results than the original PointNet method using both geometry and color/appearance information.

A number of qualitative results are presented in Figure 4 for the 3D indoor semantic segmentation task. It can be derived that our method provides more

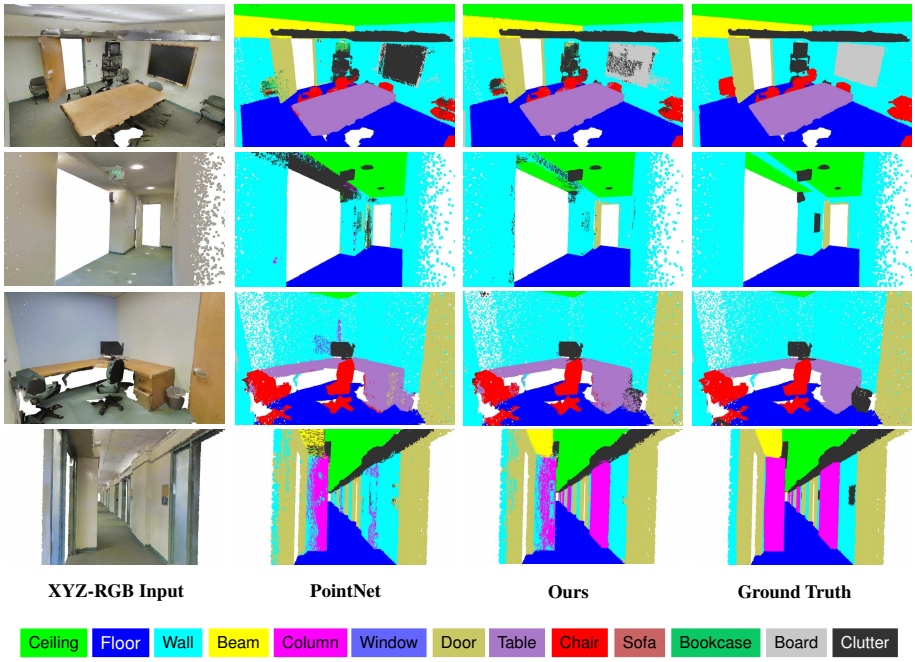

Fig. 4. Qualitative results for 3D indoor semantic segmentation. Results for the S3DIS dataset with $XYZ-RGB$ as input. From left to right: the input point cloud, the results of PointNet, our results, and the ground truth semantic labels. Our model obtains more consistent and less noisy predictions

precise predictions for local structures. It shows that our model exploits both local and global contextual cues to learn discriminative features to achieve proper semantic segmentation. Moreover, our model size is less than 160 MB and average inference time is less than 70 ms per block, which makes our method suitable for large scale point cloud analysis.

Ablation Study In this section, experiments are conducted to validate the effects of the different components of our proposed architecture for 3D semantic segmentation task. The baseline is the model corresponding to the vanilla PointNet, but utilizing the k-d tree partitioning to guide the feature learning stage. For a certain level, max-pooling is used to obtain different local region representations which are concatenated with the corresponding point features. We also trained models with different sets of components to test the effectiveness of our approach. We use the sixth fold setting of [24] for S3DIS as our experiment setting (i.e. we test on Area 6 and train on the rest). Results are reported in Table 4. Experimental results show that: (1) with k-d tree guided hierarchical feature learning, the baseline obtains better results than PointNet. Hence,

Table 4. Effectiveness of different components of our architecture. We use the sixth fold setting of [24] for S3DIS as our training/testing split

	mean IoU	overall accuracy
PointNet	63.9	86.0
PointNet++	69.1	90.1
Baseline with k-d tree guided	68.1	89.2
Only Progressively Aggregation	68.3	88.9
Only Global Cues	68.7	88.9
Only Local Cues	69.9	89.8
Global Cues and Progressively Aggregation	69.8	89.9
Local Cues and Progressively Aggregation	71.2	90.4
Local and Global Cues	71.5	90.1
All	**72.0**	**90.6**

local structures do help, (2) local contextual cues boost the performance the most, indicating that local neighborhoods of points contain fine-grained structure information, (3) any single combination of two components increases the performance and combining all of them provides state-of-the-art 3D semantic segmentation results.

4.2 3D Object Classification and Part Segmentation

We evaluate our method on the ModelNet40 shape classification benchmark [32]. The dataset contains a collection of 3D CAD models of 40 categories. We use the official split consisting of 9843 examples for training and 2468 for testing. Using the same experimental settings of [22], we convert the CAD models to point sets by uniformly sampling (1024 points in our case) over the mesh faces. Then, these points are normalized to have zero mean and unit sphere. We also randomly rotate the point sets along the z-axis and jitter the coordinates of each point by Gaussian noise for data augmentation during training.

It can be derived from Table 5, that our model outperforms PointNet [22]. Our model has competitive performance compared to PointNet++. However, our method is much faster in inference time. Table 6 summarizes the comparison of time and space computations between PointNet, PointNet++ and our proposed method. We measure forward pass time with a batch size of 8 using TensorFlow 1.1. PointNet has the best time efficiency, but our model is faster than PointNet++ while keeping a comparable classification performance.

We also evaluate our method on the ShapeNet part dataset [7]. The dataset contains 16881 CAD models of 16 categories. Each category is annotated with 2 to 6 parts. There are 50 different parts annotated in total. We use the official split for training and testing. In this dataset, both the number of shapes and the parts within categories are highly imbalanced. Therefore, many previous methods train their network on every category separately. Our network is trained across categories.

Table 5. 3D object classification results on ModelNet40. The result of our model outperforms PointNet and is comparable to PointNet++

Method	Input	Accuracy (%)
DeepPano [29]	image	77.6
MVCNN [30]	image	90.1
MVCNN-MultiRes [23]	image	91.4
3DShapeNets [32]	voxel	77
VoxNet [20]	voxel	83
Subvolume [23]	voxel	89.2
PointNet (vanilla) [22]	point cloud	87.2
PointNet [22]	point cloud	89.2
K-d network [15]	point cloud	90.6
PointNet++ [24]	point cloud	90.7
PointNet++ (with normal) [24]	point cloud	91.9
Ours	point cloud	90.2
Ours (with normal)	point cloud	91.1

Table 6. Comparison of the model sizes and the inference time for the classification task. Our model is faster than PointNet++ while keeping comparable classification performance

	PointNet [22]	PointNet++(SSG) [24]	PointNet++(MSG) [24]	PointNet++(MRG) [24]	3DContextNet
Model size (MB)	40	8.7	12	24	56.8
Forward time (ms)	25.3	82.4	163.2	87.0	45.9

We compare our model with two traditional learning based techniques Wu [33] and Yi [34], the volumetric deep learning baseline (3DCNN) in PointNet [22], as well as state-of-the-art approaches of SSCNN [35] and PointNet++ [24], see Table 7. The point intersection over union for each category as well as the mean IoU are reported. In comparison to PointNet, our approach performs better on most of the categories, which proves the importance of local and global contextual information. See Figure 5 for a number of qualitative results for the 3D object part segmentation task.

Table 7. 3D object part segmentation results on ShapeNet part dataset

	mean	airplane	bag	cap	car	chair	earphone	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skateboard	table
#shapes		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
Wu [33]	-	63.2	-	-	-	73.5	-	-	-	74.4	-	-	-	-	-	-	74.8
K-d Networks [15]	77.2	79.9	71.2	80.9	68.8	88.0	72.4	88.9	86.4	79.8	94.9	55.8	86.5	79.3	50.4	71.1	80.2
3DCNN [22]	79.4	75.1	72.8	73.3	70.0	87.2	63.5	88.4	79.6	74.4	93.9	58.7	91.8	76.4	51.2	65.3	77.1
Yi [34]	81.4	81.0	78.4	77.7	75.7	87.6	61.9	92.0	85.4	82.5	95.7	70.6	91.9	85.9	53.1	69.8	75.3
PointNet [22]	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
SSCNN [35]	84.7	81.6	81.7	81.9	75.2	90.2	74.9	93.0	86.1	84.7	95.6	66.7	92.7	81.6	60.6	82.9	82.1
PointNet++ [24]	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
Ours	84.3	83.3	78.0	84.2	77.2	90.1	73.1	91.6	85.9	81.4	95.4	69.1	92.3	81.7	60.8	71.8	81.4

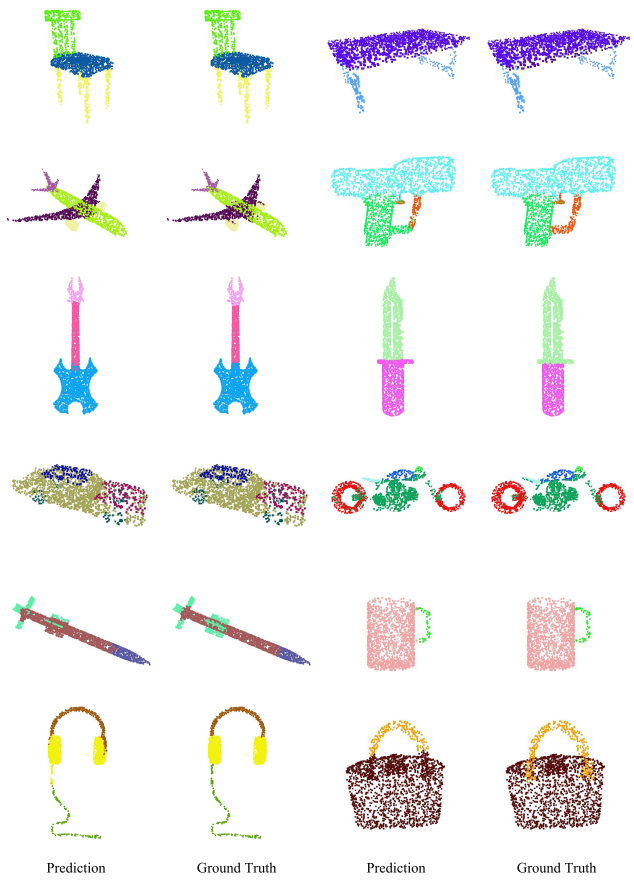

Fig. 5. Qualitative results for the 3D object part segmentation task. For each group from left to right: the prediction and the ground truth

5 Conclusion

In this paper, we proposed a deep learning architecture that exploits the local and global contextual cues imposed by the implicit space partition of the k-d tree for feature learning, and calculate the representation vectors progressively along the associated k-d tree for feature aggregation. Large scale experiments showed that our model outperformed existing state-of-the-art methods for semantic segmentation task. Further, the model obtained comparable results for 3D object classification and 3D part segmentation.

In the future, other hierarchical 3D space partition structures can be studied as the underlying structure for the deep net computation and the non-uniform point sampling issue needs to be taken into consideration.

References

1. Armeni, I., Sax, S., Zamir, A.R., Savarese, S.: Joint 2d-3d-semantic data for indoor scene understanding. arXiv preprint arXiv:1702.01105 (2017)
2. Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3d semantic parsing of large-scale indoor spaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1534–1543 (2016)
3. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. arXiv preprint arXiv:1511.00561 (2015)
4. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Communications of the ACM **18**(9), 509–517 (1975)
5. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Generative and discriminative voxel modeling with convolutional neural networks. arXiv preprint arXiv:1608.04236 (2016)
6. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
7. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
8. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. arXiv preprint arXiv:1702.04405 (2017)
9. Engelmann, F., Kontogianni, T., Hermans, A., Leibe, B.: Exploring spatial context for 3d semantic segmentation of point clouds
10. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 1440–1448 (2015)
11. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition pp. 770–778 (2016)
13. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507 (2017)
14. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. arXiv preprint arXiv:1608.06993 (2016)
15. Klokov, R., Lempitsky, V.: Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. arXiv preprint arXiv:1704.01222 (2017)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
17. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
18. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3431–3440 (2015)
19. Masci, J., Boscaini, D., Bronstein, M., Vandergheynst, P.: Geodesic convolutional neural networks on riemannian manifolds. In: Proceedings of the IEEE international conference on computer vision workshops. pp. 37–45 (2015)

20. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. pp. 922–928. IEEE (2015)
21. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1520–1528 (2015)
22. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. arXiv preprint arXiv:1612.00593 (2016)
23. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view cnns for object classification on 3d data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5648–5656 (2016)
24. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems. pp. 5099–5108 (2017)
25. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
26. Riegler, G., Ulusoy, A.O., Geiger, A.: Octnet: Learning deep 3d representations at high resolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. vol. 3 (2017)
27. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**(3), 211–252 (2015)
28. Sedaghat, N., Zolfaghari, M., Brox, T.: Orientation-boosted voxel nets for 3d object recognition. arXiv preprint arXiv:1604.03351 (2016)
29. Shi, B., Bai, S., Zhou, Z., Bai, X.: Deeppano: Deep panoramic representation for 3-d shape recognition. IEEE Signal Processing Letters **22**(12), 2339–2343 (2015)
30. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE international conference on computer vision. pp. 945–953 (2015)
31. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks (2017)
32. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1912–1920 (2015)
33. Wu, Z., Shou, R., Wang, Y., Liu, X.: Interactive shape co-segmentation via label propagation. Computers & Graphics **38**, 248–254 (2014)
34. Yi, L., Kim, V.G., Ceylan, D., Shen, I., Yan, M., Su, H., Lu, A., Huang, Q., Sheffer, A., Guibas, L., et al.: A scalable active framework for region annotation in 3d shape collections. ACM Transactions on Graphics (TOG) **35**(6), 210 (2016)
35. Yi, L., Su, H., Guo, X., Guibas, L.: Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In: Computer Vision and Pattern Recognition (CVPR) (2017)
36. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., Smola, A.: Deep sets (2017)