Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds

Francis Engelmann[†], Theodora Kontogianni[†], Alexander Hermans and Bastian Leibe
Computer Vision Group, Visual Computing Institute
RWTH Aachen University
{engelmann,kontogianni,hermans,leibe}@vision.rwth-aachen.de

Abstract

Deep learning approaches have made tremendous progress in the field of semantic segmentation over the past few years. However, most current approaches operate in the 2D image space. Direct semantic segmentation of unstructured 3D point clouds is still an open research problem. The recently proposed PointNet architecture presents an interesting step ahead in that it can operate on unstructured point clouds, achieving decent segmentation results. However, it subdivides the input points into a grid of blocks and processes each such block individually. In this paper, we investigate the question how such an architecture can be extended to incorporate larger-scale spatial context. We build upon PointNet and propose two extensions that enlarge the receptive field over the 3D scene. We evaluate the proposed strategies on challenging indoor and outdoor datasets and show improved results in both scenarios.

1. Introduction

Semantic segmentation is an important capability for intelligent vehicles, such as autonomous cars or mobile robots [7]. Identifying the semantic meaning of the observed 3D structure around the vehicle is a prerequisite for making complex navigation decisions or reconstruction [5, 6]. Consequently, the problem has attracted a lot of attention, and notable successes have been achieved with the help of deep learning techniques. However, most state-of-the-art semantic segmentation approaches operate on 2D images, which naturally lend themselves to processing with Convolutional Neural Networks (CNNs) [17, 3, 29, 23].

Processing unstructured 3D point clouds, such as those obtained from LiDAR or stereo sensors, is a much harder problem, and it is only recently that first successful deep learning approaches have been proposed for this task [19, 13, 32, 26]. Such point clouds can be obtained from LiDAR sensors mounted on top of a recording vehicle or they
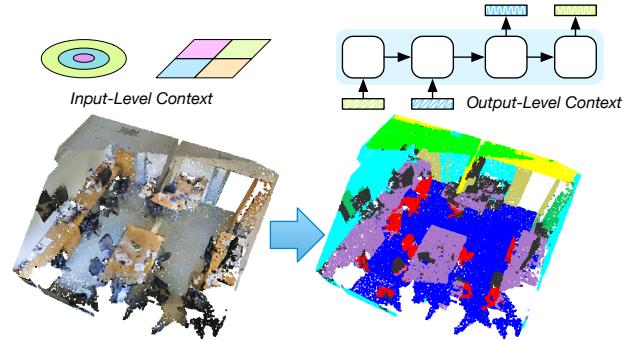

Figure 1: We explore mechanisms to extend the spatial context for 3D semantic segmentation of point clouds.

can be obtained from visual SLAM approaches operating on the vehicle's cameras [14]. Finding approaches that can directly operate on point cloud data is highly desirable, since it avoids costly preprocessing and format conversion steps. However, the question what is the best network architecture to process unstructured 3D point clouds is still largely open.

In this paper, we take inspiration from the recent PointNet work by Qi *et al*. [26], which currently defines the state of the art in 3D semantic segmentation. PointNet learns a higher dimensional spatial feature representation for each 3D point and then aggregates all the points within a small 3D volume (typically an occupancy grid cell) in order to bring in some form of 3D neighborhood context. However, this neighborhood context is very restricted, as each grid cell is processed independently.

In this paper, we investigate possible mechanisms to incorporate context into a point cloud processing architecture. We focus on spatial context, which has been identified as being very important for semantic segmentation [20, 25]. We introduce two mechanisms to add spatial context to an existing PointNet. The first mechanism incorporates neighborhood information by processing input data from multiple scales or multiple adjacent regions together (input-level context). The second mechanism operates on the estimated point descriptors and aims at consolidating them by exchanging information over a larger spatial neighborhood

---

[†]Both authors contributed equally.

(output-level context). For both mechanisms, we explore several possible realizations and compare them experimentally. As our results show, both mechanisms improve semantic segmentation quality.

Contributions. The key contributions of our work can be summarized as follows: (1) We present two mechanisms that can be used to incorporate spatial context into semantic 3D point cloud segmentation. (2) We show how these mechanisms can be incorporated into the PointNet pipeline. (3) We verify experimentally that our proposed extensions achieve improved results on challenging indoor and outdoor datasets.

2. Related Work

Unstructured Point Clouds. A varied number of sensors and setups exist which help to obtain unstructured point clouds: areal data from airborne laser scanners, laser scanners mounted on dynamic setups in a push-broom configuration [18], rotating lasers e.g. Velodyne [9], or static lasers [10]. Additionally, indoor spaces can be scanned using devices such as the Microsoft Kinect [22] or Matterport cameras [1]. All these devices produce point clouds of different quality and density. We apply our method to indoor data from [1] and to synthetic urban outdoor data from [8].

Traditional Methods. Hackel et al. [11] use traditional random forest classifiers with 3D features (without color). Their method is based on eigenvalues and eigenvectors of covariance tensors created by the nearest neighbors of the points. Their main contribution is an efficient approximate nearest neighbors computation at different scales. Munoz et al. [21] follow a similar approach but replace the random forest classifier with an associative markov network. Random forest classifiers are also used in [33] to classify data from 2D images and 3D point clouds, which they later fuse. Similarly, Xu et al. [31] fuse camera and LiDAR sensor data. Xiong et al. [30] propose a sequential parsing procedure that learns the spatial relationships of objects. Lai et al. [16] introduce a hierarchical sparse coding technique for learning features from synthetic data. Vosselman et al. [28] combine multiple segmentation and post-processing methods to achieve useful point cloud segmentations.

Deep-learning Methods. In a deep learning context, point clouds can be represented in a regular volumetric grid in order to apply 3D convolutions [19, 13]. Alternatively, 3D points can be mapped to a 2D representation followed by 2D convolutions [27]. In [2], the authors are performing 2D convolutions in 2D snapshots of a 3D point cloud and then project the labels back to 3D space. In [24] a deep learning framework learns semantic segmentation by tracking point clouds. Yi et al. [32] use spectral CNNs on 3D models represented as shape graphs for shape part segmentation. Recent methods operate directly on raw point clouds with kd-trees [15] or fully convolutional layers [26].
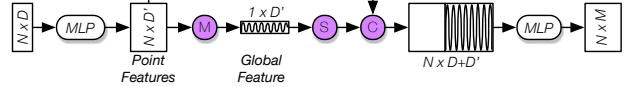

Figure 2: **Simplified PointNet Architecture.** In this work, we build upon the PointNet architecture for semantic segmentation. In short, it computes a global feature which summarizes a set of input points. Specifically, the network takes N points as input, applies a series of multi-layer-perceptrons transformations and aggregates the point features by max pooling them into a global feature. Global and local features are concatenated and the per point class scores are returned. (MLP): Multi-Layer-Perception, (M): Max-Pool, (S): Vertical Stack, (C): Concatenate. See text and Qi et al. [26] for more details.

3. Method

In this section we start by reviewing the PointNet model, then we introduce our mechanisms of extending context and finish by describing our two exemplary architectures.

3.1. PointNet

PointNet [26] is a deep neural network that, when used for semantic segmentation, takes as input a point cloud and outputs the per point semantic class labels. First, it splits a point cloud into 3D blocks, then it takes N points inside a block and after a series of Multi-Layer-Perceptrons (MLP) per point, the points are mapped into a higher dimensional space D', these are called local *point-features*. Max-pooling is applied to aggregate information from all the points resulting in a common *global-feature* invariant to input permutations. The global-feature is then concatenated with all the point-features. After another series of MLPs these combined features are used to predict the M output class scores. Figure 2 shows a simplified model.

Caveats. The global-features in PointNet summarize the context of a single block (block-feature), as a result the aggregated information is passed only among points inside the same block.

Context outside a block is equally important and could help make more informed class label predictions. Therefore we introduce two mechanisms to add context: **input-level context** – which operates directly on the input point clouds – and **output-level context** – which consolidates the output from the input-level context.

3.2. Input-Level Context

In this straightforward addition, we increase the context of the network by considering a group of blocks simultaneously instead of one individual block at a time as done in PointNet. Context is shared among all blocks in a group. These groups of blocks are selected either from
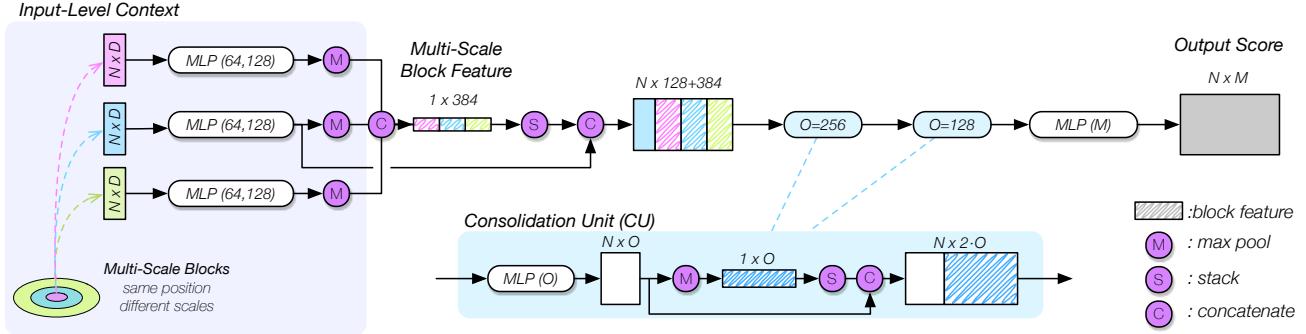
Figure 3: **Architecture with multi-scale input blocks and consolidation units (MS-CU).** The network takes as input three blocks from multiple scales, each one containing N D-dimensional points. Separately, for each scale, it learns a block-feature similarly to the PointNet mechanism. The concatenated block-features are appended to the input-features and then transformed by a sequence of consolidation units (see Section 3.3). The network outputs per point scores. Shaded fields represent block-features.

the same position but at multiple different scales (**Multi-Scale Blocks**, see Figure 3, left) or from neighboring cells in a regular grid (**Grid Blocks**, see Figure 4, left). For each input block, we compute a block-feature using the mechanism from PointNet. For the multi-scale version, we train a block-descriptor for each scale individually to obtain scale-dependent block-features. In the case of grid blocks, all block features are computed by a shared single-scale block-descriptor. In the end, both approaches output a set of block-features corresponding to the input blocks.

3.3. Output-Level Context

At this stage, we further consolidate the block-features obtained from the previous stage. Here, we differ between two consolidation approaches:

Consolidation Units (CU) consume a set of point features, transform them into a higher dimensional space using MLPs and apply max-pooling to generate a common block-feature which is again concatenated with each of the high dimensional input features (see Figure 3, blue box). This procedure is similar to the block-feature mechanism of PointNet. The key point is that CUs can be chained together into a sequence CUs forming a deeper network. The intuition behind this setup is as follows: In the beginning each point sees only its own features. After appending the block-features, each point is additionally informed about the features of its neighboring points. By applying CUs multiple times, this shared knowledge is reinforced.

Recurrent Consolidation Units (RCU) are the second type of context consolidation we employ. RCUs take as input a sequence of block-features originating from spatially nearby blocks and return a sequence of corresponding updated block-features. The core idea is to create block-features that take into consideration neighboring blocks as well. In more detail, RCUs are

implemented as RNNs, specifically GRUs [4], which are a simpler variation of standard LSTMs [12]. GRUs have the capability to learn long range dependencies. That range can either be over time (as in speech recognition) or over space as in our case. The cells of the unrolled GRU are connected in an unsynchronized many-to-many fashion (see Figure 4, blue box). This means that the updated block-features are returned only after the GRU has seen the whole input sequence of block-features. Intuitively, GRU retain relevant information about the scene in their internal memory and update it according to new observations. We use this memory mechanism to consolidate and share the information across all input blocks. For example, the decision about whether a point belongs to a chair is changed if the network remembers that it has seen a table further down in the room.

In the following, we describe two exemplary architectures which combine the previously introduced components. For those, we provide a detailed evaluation and report improved results in Section 4.

3.4. Multi-Scale (MS) Architecture

The full MS architecture is displayed in Figure 3. The learned block-features from the multi-scale blocks, (see Section 3.2) are concatenated into one **multi-scale block-feature**. This multi-scale block-feature is further concatenated with the transformed input point-features and passed through a series of CUs (see Section 3.3). Applying a final MLP results in output scores for each input point.

Specific for this architecture is the sampling procedure to select the positions of the multi-scale blocks: We randomly pick a D-dimensional point from the input point cloud as the center of the blocks and we group together N points that fall within a specified radius. This procedure is repeated at the same point for multiple radii.
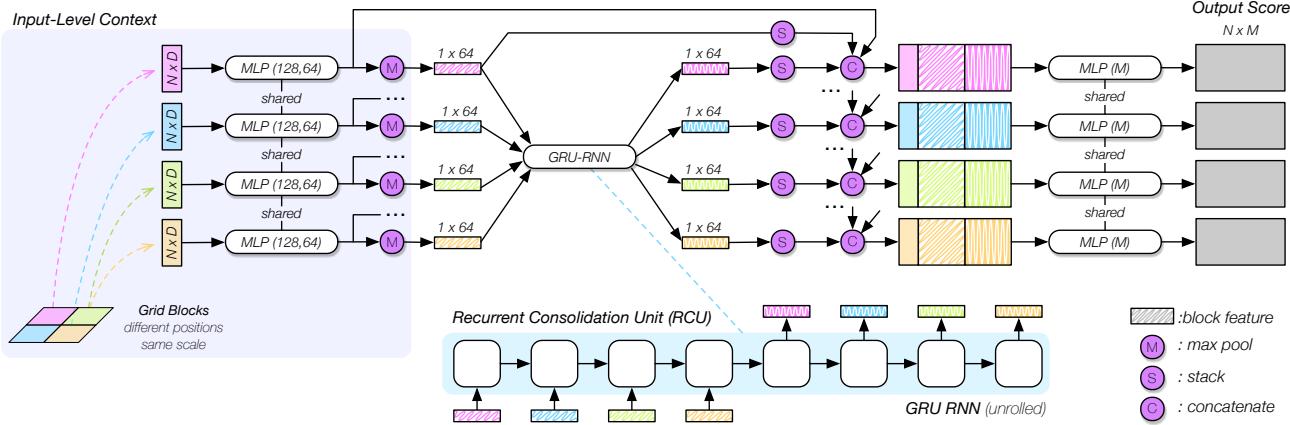
Figure 4: **Architecture with grid input blocks and a recurrent consolidation unit (GB-RCU).** The network takes as input four blocks from a grid structure, each one containing N D-dimensional points. It then learns the block-features using the same MLP weights for each block. All block-features are passed through a recurrent consolidation unit (see Section 3.3) which shares the spatial context among all blocks and returns updated block-features. The updated block-features are appended to the input-features together with the original block-features and used to compute the output per point scores. Shaded fields represent block-features. Some skip-connections are omitted for clarity.

3.5. Grid (G) Architecture

Figure 4 shows the pipeline of the architecture with grid input blocks. It consists of the following components: Input level context: a sequence of blocks $b_1, b_2, b_3 \ldots$ from Section 3.2 is fed into a series of MLPs that transform the point features, with weights shared among all blocks. These block-features are passed to an RCU that updates the individual block-features with common context from all neighboring blocks. The updated block-features are then concatenated with the original block-features. They are then used, along with the local features, for class predictions. After a series of fully connected layers the output of class scores is computed for each point.

4. Experiments

For experimental evaluation, we compare our two architectures with PointNet [26], the current state-of-the-art semantic segmentation method directly operating on point clouds. We produce quantitative results for our models and the baseline on two challenging datasets: *Stanford Large-Scale 3D Indoor Spaces* (S3DIS) [1] and on *Virtual KITTI* (vKITTI) [8]. Additionally, we provide qualitative results on point clouds obtained from a Velodyne HDL-64E LiDAR scanner from the *KITTI* dataset [9]. We will now describe these datasets in more detail.

Stanford Large Scale 3D Indoor Scenes. This dataset is composed of 6 different large scale indoor areas, mainly conference rooms, personal offices and open spaces. It contains dense 3D point clouds scanned using a *Matterport* camera. Each point is labeled with one of the 13 semantic

classes listed in Table 1. Using the reference implementation of PointNet, we were able to reproduce the results reported by Qi et al. [26], see Table 4. Throughout the paper, we follow the same evaluation protocol used in [26], which is a 6-fold cross validation over all the areas.

Virtual KITTI. Due to the lack of semantically annotated large-scale outdoor datasets, we rely on the photorealistic synthetic vKITTI dataset which closely mimics the real-world KITTI dataset. It consists of 5 different monocular video sequences in urban settings, fully annotated with depth and pixel-level semantic labels. In total there are 13 semantic class, listed in Table 2. For our purposes, we project the given 2D depth into 3D space to obtain semantically annotated 3D point clouds. Conveniently, this procedure results in point clouds that resemble the varying density of real world point clouds obtained by Velodyne LiDAR scanners (see Figure 5). For test and training, we split the original sequences into 6 non-overlapping subsequences. The final train-test sets are created by choosing point clouds from each subsequence at regular time-intervals. For evaluation, we also follow the 6-fold cross validation protocol.

4.1. Evaluation Measures

As in [26], we evaluate on the intersection over union (IoU), the average per class accuracy and overall accuracy. Intersection over union is computed as:

$$IoU = \frac{TP}{TP + FP + FN} \tag{1}$$

where TP is the number of true positives, FP the number of false positives and FN the number of false negatives.

4.2. Quantitative Results

In this section, we analyze the effectiveness of the input-block schemes and the consolidation units exemplary on the two previously introduced models. As input features, we differentiate between geometry (XYZ) and geometry with color (XYZ+RGB).

Geometry with Color. First, we compare the grid-blocks in combination with a recurrent consolidation block (G+RCU) to the original PointNet. Using the same evaluation setup as described in [26] we are able to show improved results over PointNet, see Table 4 and Table 1. This proves our hypothesis that RCU are able to convey context among blocks and thus improving results.

During training, each room is split into blocks of 1x1 m on the ground plane. Each block extends over the whole room height. Neighboring blocks are overlapping by 0.5 meters in both directions. We select four blocks simultaneously from a 2x2 neighborhood. Each block contains 4096 points. The unrolled GRU is 8 cells long (4 input, 4 output). It's memory size is 64. During testing, the room is split into non-overlapping blocks and evaluated on all 2x2 groups of blocks. Each block is evaluated only once.

Next, we take a look at the multi-scale input block with consolidation units (MS-CU) model. To build the multi-scale blocks, we follow the process described in Section 3.4. As radii, we choose [0.25, 0.5, 1.0] m. As distance metric we choose the *Chebyshev*-distance which generates axis-aligned rectangular blocks. The middle scale block is equal to the PointNet block regarding shape and size.

By using sampling (necessary for the multi-scale block construction), we diverge from the previous training procedure so we re-run all experiments under these new conditions.

We validate the influence of each of the architecture's components by adding them one-by-one to our pipeline and evaluating after each step, see Table 4 and Table 1. First, we only consider the input-level context i.e the multi-scale block feature (MS) as input to our pipeline while skipping the consolidation units. This shows some performance benefits over PointNet but not as much as one would expect
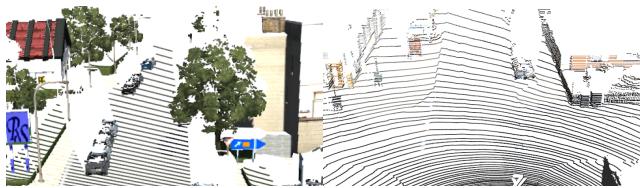

Figure 5: We train our network on synthetic point clouds generated from vKITTI [8] (left) and apply it onto real-world Velodyne LiDAR point clouds (right). The structure and the varying density are comparable.

Figure 6: **Qualitative results on laser point clouds.** Dataset: Velodyne HDL-64E laser scans from KITTI Raw [9]. We trained our model on vKITTI point clouds without color and applied it on real-world laser point clouds. So far, only classes like road, building and car give decent results.

considering the enlarged input context. Next, we take only single-scale input blocks and add one consolidation unit (SS+CU(1)). The results show that the CU outperforms the MS input blocks. It also shows that CUs provide a simple technique to boost the network's performance. Finally, we combine both the MS blocks and the CU while appending another CU to the network (MS+CC(2)). This full model is depicted in Figure 3.

Geometry only. Until now, each input point was described by a 9-dimensional feature vector $[X, Y, Z, R, G, B, X', Y', Z']$ where $[X, Y, Z]$ are the spatial coordinates of a point, $[R, G, B]$ its color and $[X', Y', Z']$ the normalized coordinated based on the size of the environment, see [26] for further details. Without doubt, color is a very strong input feature in the context of semantic segmentation. In this section, we pose the question what will happen if no color information is available like it is the case with point clouds obtained from laser scanners. To simulate the missing colors, we simply discard the color information from the input feature and re-run the experiments. Table 3 and 2 show the obtained results. See caption for discussion of the results.

4.3. Qualitative Results

We present qualitative results of our models applied to indoor scenarios in Figure 7 and outdoor results in Figure 8 along with a short discussion. Additionally, we applied our pre-trained geometry-only model (vKITTI) to real-world laser data. The results are shown in Figure 6 and Figure 9.

5. Conclusion

In this work, we investigated the question how to incorporate spatial context into a neural network architecture for 3D semantic segmentation. Building upon PointNet, we proposed two extension (Input-level context and Output-level context) which we successfully applied onto indoor and outdoor datasets. Still, numerous other combinations remain possible. The full exploration of the design space is left for future work.

S3DIS Dataset [1]	mean IoU	Ceiling	Floor	Wall	Beam	Column	Window	Door	Table	Chair	Sofa	Bookcase	Board	Clutter
*PointNet [26]	43.5	81.5	86.7	64.8	29.4	16.3	39.1	48.1	52.5	42.5	5.4	37.6	30.4	31.4
*MS	44.4	82.2	86.9	64.2	33.8	22.8	43.3	52.0	51.0	38.6	9.2	36.1	23.6	33.7
*MS + RCU	45.5	83.6	86.9	**67.5**	**40.5**	17.1	37.0	48.8	**53.9**	42.3	6.8	**39.7**	**32.8**	34.2
*SS + CU(1)	45.9	**88.6**	92.6	66.3	36.2	23.6	47.1	51.2	50.2	36.9	**12.6**	33.7	22.7	35.3
*MS + CU(2)	**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**
PointNet [26]	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
G + RCU	**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**

Table 1: **IoU per semantic class. S3DIS dataset with XYZ-RGB input features.** We compare our models with different components against the original PointNet baseline. By adding different components, we can see an improvement of mean IoU. We obtain state-of-the-art results in mean IoU and all individual class IoU. *Entries marked with * use random sampling for input block selection instead of discrete positions on a regular grid.*

S3DIS Dataset [1]	mean IoU	Ceiling	Floor	Wall	Beam	Column	Window	Door	Table	Chair	Sofa	Bookcase	Board	Clutter
*PointNet [26]	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)	**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**

vKITTI Dataset [8]	mean IoU	Terrain	Tree	Vegetation	Building	Road	GuardRail	TrafficSign	TrafficLight	Pole	Misc	Truck	Car	Van
*PointNet [26]	17.9	32.9	76.4	11.9	17.7	49.9	3.6	2.8	3.7	3.5	0.7	1.5	25.1	3.4
*MS + CU(2)	**26.4**	**38.9**	**87.1**	**14.6**	**44.0**	**58.4**	**12.4**	**9.4**	**10.6**	**5.3**	**2.2**	**3.6**	**43.0**	**13.3**

Table 2: **IoU per semantic class. S3DIS and vKITTI datasets both with XYZ input features (no color).** Our methods not only outperform PointNet consistently on two datasets, the improvements in mean IoU are also more considerable when no color is available. This suggests that our network architectures are able to learn improved geometric features and are more robust to varying point densities as they occur in the outdoor vKITTI dataset.

	mean IoU	overall accuracy	avg. class accuracy
S3DIS Dataset [1] – no RGB			
*PointNet [26]	40.0	72.1	52.9
*MS + CU(2)	**43.0**	**75.4**	**55.2**
vKITTI Dataset [8] – no RGB			
*PointNet [26]	17.9	63.3	29.9
*MS + CU(2)	**26.4**	**73.2**	**40.9**

Table 3: **S3DIS and vKITTI datasets with only XYZ input features, without RGB.** We show improved results on indoor (S3DIS) and outdoor (vKITTI) datasets. Our presented mechanisms are even more important when no color is available.

S3DIS Dataset [1] XYZ-RGB	mean IoU	overall accuracy	avg. class accuracy
*PointNet [26]	43.5	75.0	55.5
*MS	44.4	75.5	57.6
*MS + RCU	45.5	77.2	57.2
*SS + CU(1)	45.9	77.8	57.7
*MS + CU(2)	**47.8**	**79.2**	**59.7**
PointNet [26]	47.6	78.5	66.2
G + RCU	**49.7**	**81.1**	**66.4**

Table 4: **S3DIS Dataset with XYZ-RGB input features.** Comparison of different context expansion techniques on input- and output-level (see Sections 3.2–3.3). MS: Multi-Scale, SS: Single-Scale, G: Grid, CU: Consolidation Unit, RCU: Recurrent Consolidation Unit. *Entries marked with * use random sampling for input block selection instead of discrete positions on a regular grid.*
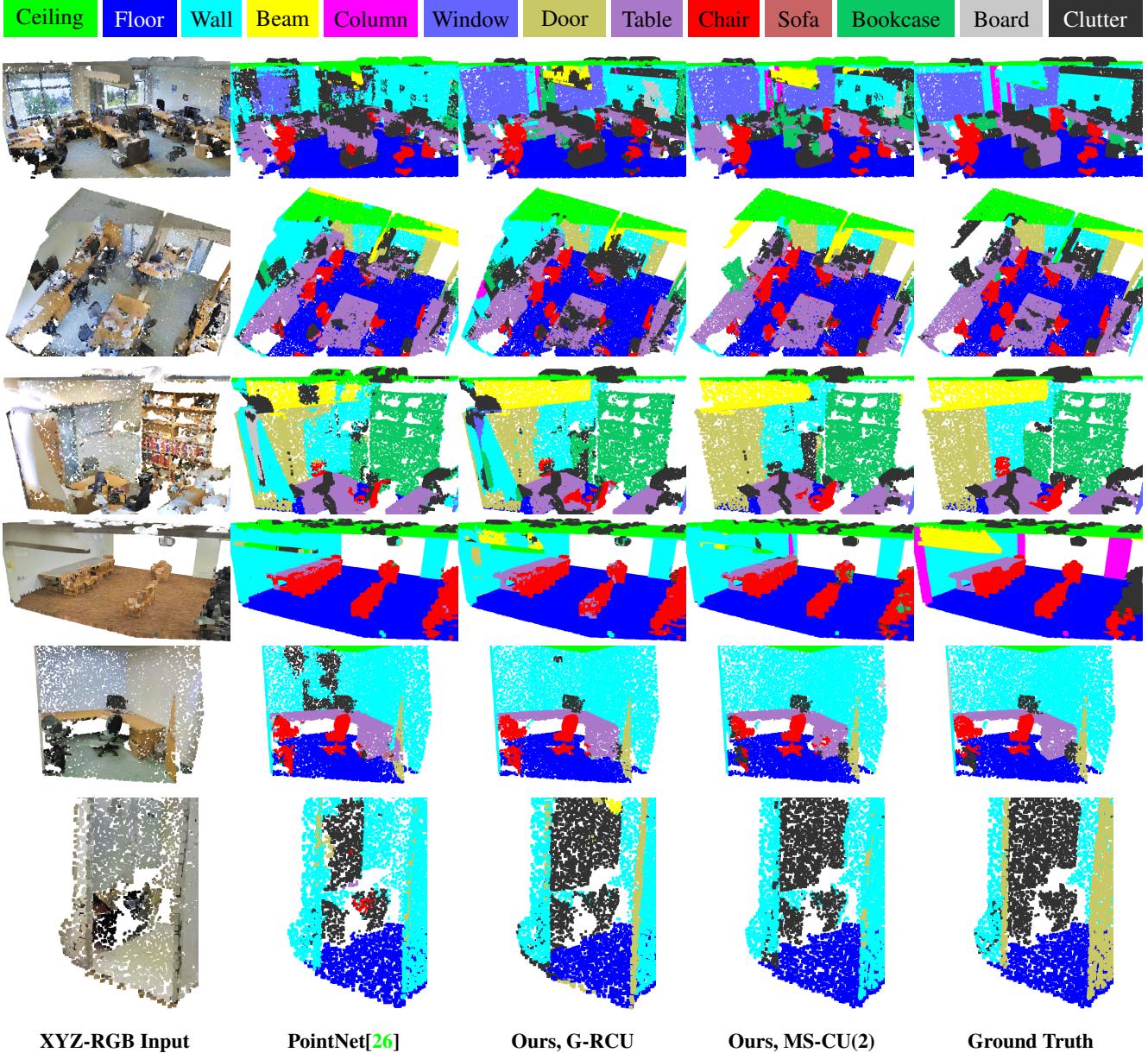
| Ceiling | Floor | Wall | Beam | Column | Window | Door | Table | Chair | Sofa | Bookcase | Board | Clutter |

XYZ-RGB Input **PointNet[26]** **Ours, G-RCU** **Ours, MS-CU(2)** **Ground Truth**

Figure 7: **Indoor qualitative results.** Dataset: S3DIS [1] with XYZ-RGB input features. From left to right: input point cloud, baseline method PointNet, our results using the G-RCU model (see Figure 4), our results using the MS-CU(2) model (see Figure 3), ground truth semantic labels. Our models produce more consistent and less noisy labels.
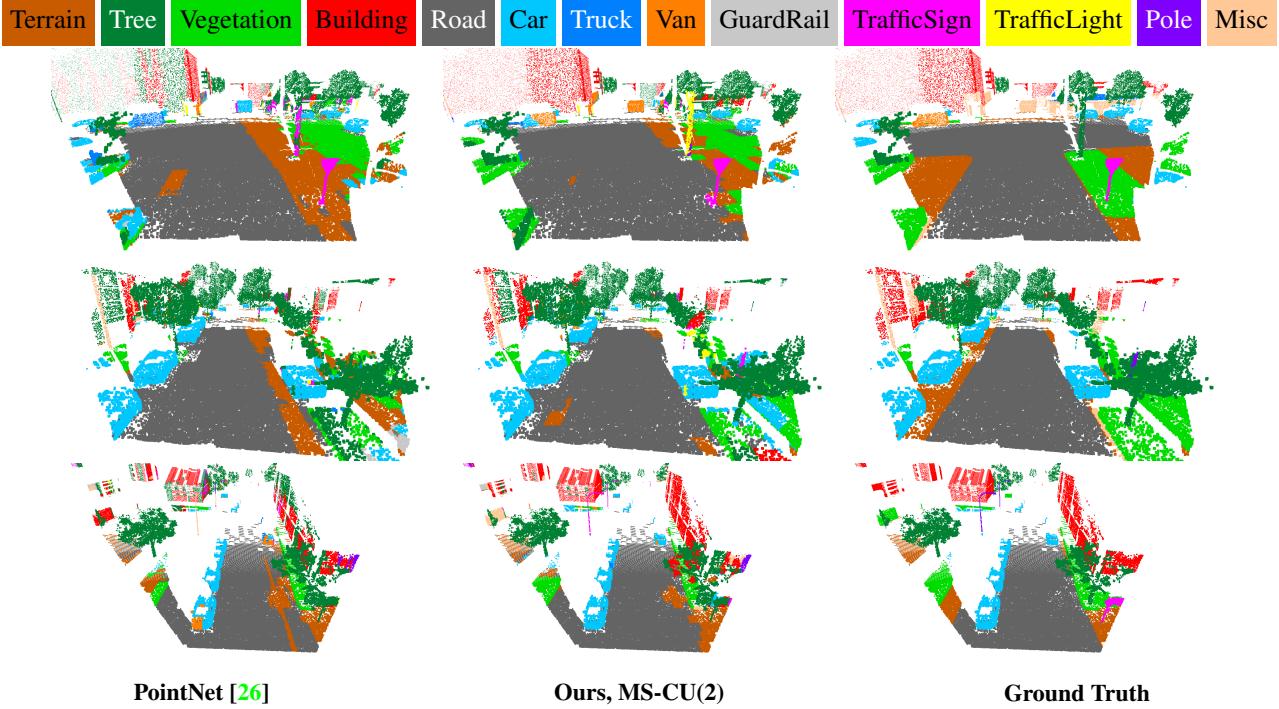
Terrain | **Tree** | **Vegetation** | **Building** | **Road** | **Car** | **Truck** | **Van** | **GuardRail** | **TrafficSign** | **TrafficLight** | **Pole** | **Misc**

PointNet [26]   Ours, MS-CU(2)   Ground Truth

Figure 8: **Outdoor qualitative results.** Dataset: Virtual KITTI [8]. Results were obtained using only XYZ coordinates as input, no color information was used. Left: baseline method PointNet. Center: our results using the MS-CU model as illustrated in Figure 3. Right: ground truth semantic labels. The outputs of our method are less fragmented (cars, houses) and finer structures like street lights and poles are recognized better.
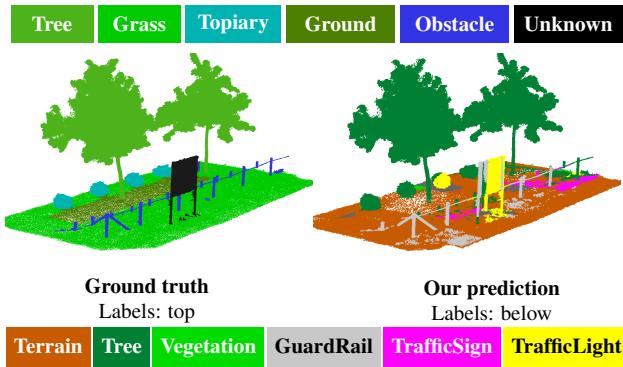

Tree | **Grass** | **Topiary** | **Ground** | **Obstacle** | **Unknown**

Ground truth   Our prediction
Labels: top   Labels: below

Terrain | **Tree** | **Vegetation** | **GuardRail** | **TrafficSign** | **TrafficLight**

Figure 9: **Qualitative results on 3DRMS'17 Challenge.** We trained our model on vKITTI point clouds without color and applied it to the 3DRMS laser data. Training and test datasets do not have the same semantic labels. Despite that, common classes like trees are successfully segmented and plausible ones are given otherwise (e.g. terrain instead of grass, guardrail instead of obstacle).

6. Acknowledgment

References

[1] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3D Semantic Parsing of Large-Scale Indoor Spaces. In *CVPR*, 2016. 2, 4, 6, 7

[2] A. Boulch, B. L. Saux, and N. Audebert. Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks. In *Eurographics Workshop on 3D Object Retrieval*, 2017. 2

[3] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *arXiv preprint arXiv:1606.00915*, 2016. 1

[4] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*, 2014. 3

[5] F. Engelmann, J. Stückler, and B. Leibe. Joint object pose estimation and shape reconstruction in urban street scenes using 3D shape priors. In *Proc. of the German Conference on Pattern Recognition (GCPR)*, 2016. 1

[6] F. Engelmann, J. Stückler, and B. Leibe. SAMP: shape and motion priors for 4d vehicle reconstruction. In *WACV*, 2017. 1

[7] G. Engelmann, M. Kowal, and R. W. D. Doncker. A highly integrated drive inverter using directfets and ceramic dc-link capacitors for open-end winding machines in electric vehicles. 2015. 1

[8] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In *CVPR*, 2016. 2, 4, 5, 6, 8

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets Robotics: The KITTI Dataset. *IJRR*, 32(11), 2013. 2, 4, 5

[10] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys. Semantic3D.net: A new Large-scale Point Cloud Classification Benchmark. *arXiv preprint arXiv:1704.03847*, 2017. 2

[11] T. Hackel, J. D. Wegner, and K. Schindler. Fast Semantic Segmentation of 3D Points Clouds with Strongly Varying Density. *ISPRS*, 3(3), 2016. 2

[12] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8), 1997. 3

[13] J. Huang and S. You. Point cloud labeling using 3D Convolutional Neural Network. In *ICPR*, 2016. 1, 2

[14] A. Kasyanov, F. Engelmann, J. Stückler, and B. Leibe. Keyframe-Based Visual-Inertial Online SLAM with Relocalization. In *IROS*, 2017. 1

[15] R. Klokov and V. S. Lempitsky. Escape from Cells: Deep Kd-Networks for The Recognition of 3D Point Cloud Models. *arXiv preprint arXiv:1704.01222*, 2017. 2

[16] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3D scene labeling. In *ICRA*, 2014. 2

[17] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *CVPR*, 2015. 1

[18] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *IJRR*, 36(1), 2017. 2

[19] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *IROS*, 2015. 1, 2

[20] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The Role of Context for Object Detection and Semantic Segmentation in the Wild. In *CVPR*, 2014. 1

[21] D. Munoz, N. Vandapel, and M. Hebert. Directional Associative Markov Network for 3-D Point Cloud Classification. In *3DPVT*, 2008. 2

[22] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2

[23] H. Noh, S. Hong, and B. Han. Learning Deconvolution Network for Semantic Segmentation. In *ICCV*, 2015. 1

[24] P. Ondruska, J. Dequaire, D. Zeng Wang, and I. Posner. End-to-End Tracking and Semantic Segmentation Using Recurrent Neural Networks. In *RSS, Workshop on Limits and Potentials of Deep Learning in Robotics*, 2016. 2

[25] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes. In *CVPR*, 2017. 1

[26] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, 2017. 1, 2, 4, 5, 6, 7, 8

[27] C. R. Qi, H. Su, M. Niener, A. Dai, M. Yan, and L. J. Guibas. Volumetric and Multi-View CNNs for Object Classification on 3D Data. In *CVPR*, 2016. 2

[28] G. Vosselman. Point Cloud Segmentation for Urban Scene Classification. *ISPRS*, 1, 2013. 2

[29] Z. Wu, C. Shen, and A. van den Hengel. High-performance Semantic Segmentation Using Very Deep Fully Convolutional Networks. *arXiv preprint arXiv:1604.04339*, 2016. 1

[30] X. Xiong, D. Munoz, J. A. Bagnell, and M. Hebert. 3-D Scene Analysis via Sequenced Predictions over Points and Regions. In *ICRA*, 2011. 2

[31] P. Xu, F. Davoine, J. Bordes, H. Zhao, and T. Denoeux. Information Fusion on Oversegmented Images: An Application for Urban Scene Understanding. In *MVA*, 2013. 2

[32] L. Yi, H. Su, X. Guo, and L. J. Guibas. SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation. *arXiv preprint arXiv:1612.00606*, 2016. 1, 2

[33] R. Zhang, S. A. Candra, K. Vetter, and A. Zakhor. Sensor Fusion for Semantic Segmentation of Urban Scenes. In *ICRA*, 2015. 2