Bidirectional Projection Network for Cross Dimension Scene Understanding

Wenbo Hu[1,3*] Hengshuang Zhao[2*] Li Jiang[1] Jiaya Jia[1] Tien-Tsin Wong[1,3†]
[1]The Chinese University of Hong Kong [2]University of Oxford
[3]Shenzhen Key Laboratory of Virtual Reality and Human Interaction Technology, SIAT, CAS
{wbhu, lijiang, leojia, ttwong}@cse.cuhk.edu.hk hengshuang.zhao@eng.ox.ac.uk

Abstract

2D image representations are in regular grids and can be processed efficiently, whereas 3D point clouds are unordered and scattered in 3D space. The information inside these two visual domains is well complementary, e.g., 2D images have fine-grained texture while 3D point clouds contain plentiful geometry information. However, most current visual recognition systems process them individually. In this paper, we present a bidirectional projection network (BPNet) *for joint 2D and 3D reasoning in an end-to-end manner. It contains 2D and 3D sub-networks with symmetric architectures, that are connected by our proposed* bidirectional projection module (BPM). *Via the* BPM*, complementary 2D and 3D information can interact with each other in multiple architectural levels, such that advantages in these two visual domains can be combined for better scene recognition. Extensive quantitative and qualitative experimental evaluations show that joint reasoning over 2D and 3D visual domains can benefit both 2D and 3D scene understanding simultaneously. Our* BPNet *achieves top performance on the ScanNetV2 benchmark for both 2D and 3D semantic segmentation. Code is available at* https://github.com/wbhu/BPNet.

1. Introduction

Scene understanding is a fundamental while challenging problem in computer vision. Multiple sensors are used to capture the scene information. The 2D camera is the most common sensor in our daily life. It projects the 3D space to image planes with plenty of fine-grained textures captured. 2D images with pixels densely arranged in regular grids can be processed efficiently with deep convolutional neural networks. We have witnessed remarkable improvements on 2D visual reasoning, *e.g.*, image classification [28, 53, 48, 17] and semantic segmentation [35, 4, 66, 72]. 3D sensors, on the other hand, can provide important geometry information of the scene. 3D data is usually represented in points that are unordered and irregularly scattered in 3D space. Conventional convolution that relies on ordered grids can not be directly adapted to 3D data. Hence, several tailor-made neural networks [41, 32, 12] have been proposed for 3D scene recognition and understanding.

We observe that the information inside 2D and 3D data is well complementary. 2D images provide detailed texture and color information while 3D point clouds contain strong shape and geometry knowledge. Although the techniques for individual 2D and 3D reasoning are studied a lot in the literature, the exploration of combining both 2D and 3D data for recognition is very limited. Existing methods that utilize both 2D and 3D data for recognition mostly adopt the unidirectional scheme. For example, to incorporate 3D information for 2D scene understanding, some methods either encode depth into geocentric inputs [13] or incorporate it into convolution operations [44, 60]. But depth map only contains limited geometry information as it is view-dependent. The occluded part under the viewpoint together with the global context is missing. In the other aspect, 3DMV [8] and MVPNet [24] utilize 2D information to assist the 3D recognition by first extracting multi-view image features and then lifting them into 3D space for fusing with the 3D features. However, we argue that the unidirectional scheme cannot fully leverage the complementary information inside the 2D and 3D data, bidirectionally interacting and fusing 2D and 3D features can better combine the advantages of these two visual domains as evidenced by our experiments.

In this paper, we present a *Bidirectional Projection Network (BPNet)* to enable information inside the 2D and 3D domains to flow bidirectionally at the network architectural level. Such that the complementary information can be well combined for joint 2D and 3D scene understanding in an end-to-end manner. Our method adopts two similar U-Net structures to process 2D and 3D data and introduces a *Bidirectional Projection Module (BPM)* to bidirectionally fuse the multi-view 2D and 3D features. In this way, both 2D and 3D sides can benefit from each other. The overall framework is shown in Figure 1. To be mentioned, we employ BPM at multiple pyramid levels, such that the features from

*Equal contribution.
†Corresponding author.

2D and 3D domains can be aggregated in a coarse-to-fine manner and the BPNet can harvest both low- and high-level complementary information. At each level, BPM builds the projection link matrix between 2D and 3D, and then transfers the features bidirectionally according to the link matrix, *i.e.*, 2D features are projected into 3D space for boosting the recognition of 3D and vice the verse.

We evaluated our model on ScanNetV2 [7] dataset for both 2D and 3D semantic segmentation tasks. BPNet achieves top performance on the benchmark in terms of mIoU and consistently outperforms the baseline with a single 2D/3D network. Also, the qualitative results show the effectiveness of combining 2D and 3D information. BPNet can distinguish objects without much shape difference (*e.g.*, "wall" and "picture") in 3D segmentation. Meanwhile, 2D objects are better segmented with sharper boundaries thanks to the underlying geometric clues provided by 3D features. Besides, we evaluated the generalization ability of BPNet for 2.5D data on the typical RGB-D dataset, NYUv2 [38], and the results show BPNet performs favorably against the typical RGB-D and joint 2D-3D baselines. We believe the proposed BPM is also advantageous to other tasks where both 2D and 3D resources are available, *e.g.*, classification, detection, and instance segmentation. Our contributions are summarized below.

- We argue that 2D and 3D information is complementary for the understanding of each other and joint optimization over both 2D and 3D scenes is applicable and proved to be beneficial.
- We propose a Bidirectional Projection Module (BPM) that enables information interacting between the 2D and 3D representations. And such bidirectional projection operation can be adopted at multiple levels in the decoder stage.
- We present a novel framework named Bidirectional Projection Network (BPNet) for jointly reasoning over 2D and 3D scenes. Our method achieves top performance on the challenging large-scale ScanNetv2 benchmark for both 2D and 3D semantic segmentation tasks, which demonstrates its effectiveness.

2. Related Work

2D semantic segmentation. Semantic segmentation on 2D images has been significantly improved with deep neural networks. By replacing the last fully-connected layers in classification frameworks with convolution operations, FCN [35] can produce the per-pixel predictions. Several encoder-decoder architectures [46, 39, 1] are proposed to utilize the information in low-level layers to help refine the segmentation outputs. The receptive field is vital for accurate scene understanding, thus, dilated convolutions [4, 66] are introduced to enlarge the receptive field. Contextual information is also proved to be effective as adopted in [34, 5, 72, 71, 65]. Meanwhile, attention models [73, 68, 23] are also utilized in semantic segmentation for their abilities to capture long-range contextual relationships. The capability of these 2D-driven approaches is limited by the lack of geometric information, and their performance would be further boosted with other domain features like 3D geometric representations.

3D semantic segmentation. 3D point clouds are irregularly scattered in 3D space, resulting in difficulties in 3d scene understanding, as convolution operation generally work on regular grids [70]. Point based frameworks mainly adopt Multilayer Perception (MLP)-style networks [41, 43], including local region-feature enhancement [61, 63, 70, 20], kernel-based parametric convolution [59, 62, 55] and graph reasoning [57, 58, 27, 31, 30]. Apart from directly handling the irregular inputs, some methods transform the unordered point sets to ordered ones that can be processed by convolution operations, including voxelization, followed by 3D convolution approaches [36, 49, 54, 14] and the efficient generations [45, 12, 6]. Multi-view mechanisms [51, 52, 42, 29] are also widely adopted where 3D point clouds are projected into 2D images via different camera views. Besides, convolution on meshes [47], fused point-voxel representation [67], occupancy-aware design [14], and multi-task learning [21] are explored to improve the 3D scene understanding performance. However, the 3D source lacks detailed textures and color information, resulting in limited performance. And it would be improved with the help of 2D representations.

Recognition with combined 2D-3D data. There exist several approaches that unidirectionally fuse 2D and 3D information for improving the recognition performance in the 2D or 3D domain. To incorporate 3D information for 2D scene understanding, Gupta *et al.* [13] encode depth as HHA images to provide the geocentric information, 3DGNN [44] utilizes 3D graph neural networks for gradually refining the semantic representations and Depth-aware CNN [60] presents a depth-aware convolution to cooperate depth information. On the other hand, some methods are dedicated to introducing 2D information into 3D scene understanding, *e.g.*, back projecting multi-view 2D features to 3D volumes [8], or point clouds [24], then aggregating them with the original 3D features; and extracting features from texture patches [22] for the 3D semantic segmentation using surface parameterization. Different from them, we conduct cross-dimension reasoning which enables bidirectional 2D and 3D feature interaction, such that both 2D and 3D semantic segmentation can benefit from each other.

Besides, AutoContext [11] employs a decision tree to fuse up the hand-crafted features from images and point clouds for facade segmentation, xMUDA [25] presents cross-modal unsupervised domain adaption to improve performance on the image or point cloud domain. SurfaceNet [26] encodes

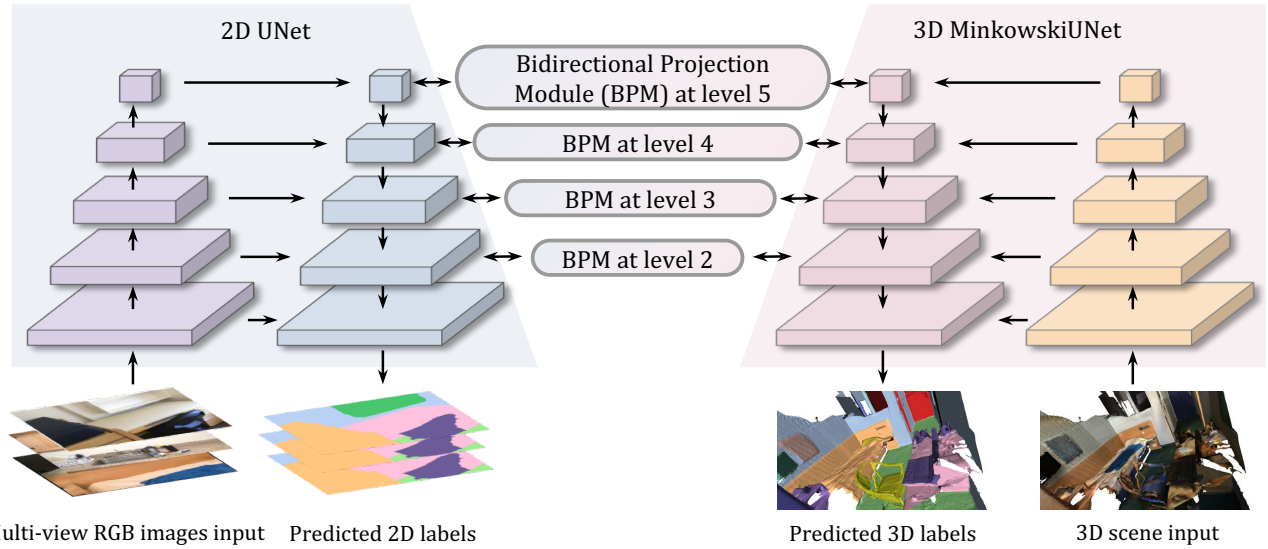

Figure 1. Overview of the Bidirectional Projection Network (BPNet). It consists of two symmetric sub-networks, 2D UNet [46] (in the left-hand side) and 3D MinkowskiUNet [6] (in the right-hand side). Both the 2D and 3D sub-networks are the U-shaped network with the same number of pyramid levels, but they are constructed based on conventional 2D convolution and 3D sparse convolution [12], respectively. At multiple pyramid levels in the decoder stage, the features interact bidirectionally between 2D and 3D sub-networks via the proposed Bidirectional Projection Module (BPM). Both 2D and 3D semantic labels will be predicted simultaneously by our BPNet.

camera parameters and images in a 3D voxel representation and adopts 3D CNN on it for multiview stereopsis. Moreover, SPLATNet [50] presents to represent point clouds in a high-dimensional lattice that enables fusing image and point features in the high-dimensional lattice for facade and 3D part segmentation. However, this 2D-3D fusion relies on the tailor-made bilateral convolution on the high-dimensional lattice. Different from them, our method can be directly applied to the conventional 2D and 3D CNNs for interacting 2D and 3D features bidirectionally in multi-levels.

3. Methodology

The goal of our method is to jointly predict the 2D and 3D semantics on real-world data, given 3D scenes and 2D image sequences along with corresponding camera matrix. In this section, we first present our Bidirectional Projection Module (BPM) (Figure 2) that enables cross-dimension feature interaction between 2D and 3D feature representations. Then we introduce the design details of our Bidirectional Projection Network (BPNet) (Figure 1). Finally, we give our implementation details of the proposed method.

3.1. Bidirectional Projection Module

The Bidirectional Projection Module (BPM), as shown in Figure 2, is designed to construct skip connections between 2D and 3D sub-networks at the same decoder level for bidirectionally interacting features in 2D and 3D domains, such that the strengths of these two visual domains can be integrated to boost both 2D and 3D scene understanding. Specifically, given the 3D scene, and 2D image together with camera matrix $\mathcal{M}$, we first construct the link matrix $\mathcal{L}$ between voxels and pixels according to the perspective projection from 3D to 2D space. Then, at multiple levels in the decoding stage, we not only project the 3D features F_{3D} to 2D space but also back-project 2D features F_{2D} to 3D space according to the constructed link matrix. And finally, we concatenate the projected features with the original features followed by a 1×1 convolution to fuse them. We then feed the fused features into the next levels. The detailed procedures in BPM are discussed in the following paragraphs.

Link matrix construction. The camera imaging process can be seen as a perspective projection from 3D voxels to 2D pixels. Mathematically, we can formulate it as:

$$[u_i, v_i, 1]^{\mathrm{T}} = \mathcal{M}[x_i, y_i, z_i, 1]^{\mathrm{T}}, \tag{1}$$

where $[x_i, y_i, z_i, 1]^{\mathrm{T}}$ and $[u_i, v_i, 1]^{\mathrm{T}}$ are the homogeneous 3D coordinates of the i^{th} voxel and its projected 2D homogeneous coordinates, respectively; and $\mathcal{M}$ is the matrix that is the product of the intrinsic camera calibration matrix and the extrinsic camera pose matrix.

Based on it, we can create a link matrix $\mathcal{L}$ to store the corresponding 2D pixel coordinate $[u, v]^{\mathrm{T}}$ for each voxel $[x, y, z]^{\mathrm{T}}$, as shown in part (a) of Figure 2. But some voxels may have no corresponding pixels, since the image under a certain viewpoint only captures part of the whole 3D scene, like the k^{th} voxel in part (a) of Figure 2. Therefore, we further add a mask m with binary values to indicate whether this 3D voxel has a corresponding pixel. Due to the occlusion, another problem is the hidden surfaces (or voxels) that can be projected to certain pixels but have no relations with the pixels at all, like the j^{th} voxel in part (a) of Figure 2 is

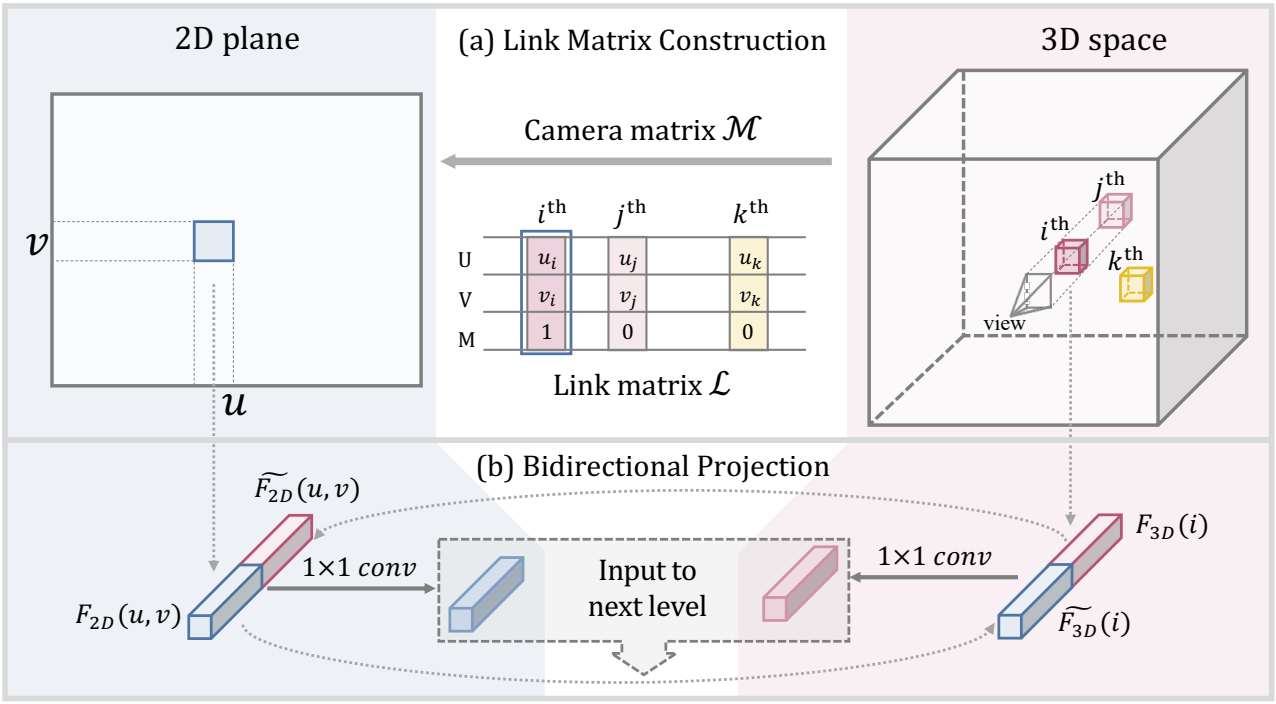

Figure 2. Bidirectional Projection Module (BPM). The link matrix construction procedure between one view and the 3D scene is illustrated in (a), and the bidirectional projection procedure is shown in (b). $\mathcal{M}$ is the camera matrix; the i^{th}, j^{th} and k^{th} voxels are typical examples of three kinds of voxels: voxels that have corresponding pixels under this view, voxels that are occluded by others, and voxels that are out of the view frustum. F_{2D} and F_{3D} are features in 2D and 3D spaces, respectively. $\widetilde{F_{2D}}$ and $\widetilde{F_{3D}}$ are the 2D and 3D bidirectionally projected features from the other domain, respectively.

occluded by the i^{th} voxel. In computer graphics, Z-Buffer algorithm [3] is commonly used to handle the hidden surface removal problem. It is also applicable to our case, but the depth map is available here, thus, we can efficiently determine whether a voxel is hidden or not by comparing the depth value and the z coordinate of the projected voxel. Compared with the Z-Buffer algorithm, doing so can speed up the computing, as we compare the projected z coordinate only once for each voxel.

As a result, we construct link matrix $\mathcal{L}$ as an $N \times 3$ matrix, $[U, V, M]$, where N is the number of voxels in the 3D scene; U and V are the perspectively projected 2D coordinate according to Equation 1; and M is the binary mask to indicate whether this row of the link matrix is valid or not. Each row of the link matrix $\mathcal{L}$ can be represented as:

$$\mathcal{L}_i = \begin{cases} [u_i, v_i, 1], & \text{if } U_{min} \le u_i \le U_{max} \\ & \text{and } V_{min} \le v_i \le V_{max} \\ & \text{and } |d(u_i, v_i) - z_i'| \le \delta \\ [u_i, v_i, 0], & \text{otherwise} \end{cases} \quad (2)$$

where $i \in [1, N]$ is the index of the voxels; $[u_i, v_i]^T$ is the 2D coordinate; 1 or 0 is the binary mask value; $U_{min}, U_{max}, V_{min}$ and V_{max} are the boundaries of the view frustum; $d(\cdot)$ is the mapping from pixel coordinates to the depth; z_i' is the projected z coordinate of the voxel; and $\delta > 0$ is the threshold for depth matching (it is set to be the voxel size in our implementation.) Note that, this link matrix is bidirectional, it can be used both to transform 2D points to 3D points and vice versa.

Bidirectional projection. The link matrix is constructed for the original input 3D voxels and 2D images, while the 2D and 3D features have different spatial sizes compared with the original input data. To produce the link matrix for 2D and 3D features in certain decoder levels, we linearly remap the $[u, v]$ coordinates according to the down-sampling ratio to fit for the spatial size of features in the link vectors.

Having the remapped links for each level of features, we can transform the features bidirectionally between 2D and 3D domains in multiple levels, as shown in part (b) of Figure 2. At one of the decoder levels, we have the 2D feature F_{2D} with the shape of $H \times W \times C_{2D}$, and the 3D feature F_{3D} with the shape of $N \times C_{3D}$, where N is the number of voxels, C_{2D} and C_{3D} are the channel numbers of 2D and 3D, respectively. Meanwhile, the constructed link matrix $\mathcal{L}$ has the shape of $N \times 3$, where "3" indicates the corresponding u, v coordinate of 2D feature and validity mask m. To project 3D features to 2D space, we can formulate it as:

$$\widetilde{F_{2D}}(u_i, v_i) = \begin{cases} m_i \cdot F_{3D}(i), & \text{if } [u_i, v_i] \in \mathcal{L}_{uv} \\ \mathbf{0}, & \text{otherwise} \end{cases} \quad (3)$$

where $\widetilde{F_{2D}}$ is the projected 2D feature from 3D feature that has the shape of $H \times W \times C_{3D}$; $i \in [1, N]$ is the index of voxels or link matrix entries; $[u_i, v_i, m_i]$ is the i^{th} entry of $\mathcal{L}$; and $\mathcal{L}_{uv}$ is the first two columns of the link matrix $\mathcal{L}$. On the other hand, to back project 2D features to 3D space, we formulate it as:

$$\widetilde{F_{3D}}(i) = m_i \cdot F_{2D}(u_i, v_i), \tag{4}$$

where $\widetilde{F_{3D}}$ is the back-projected feature from 2D feature with the shape of $N \times C_{2D}$. These procedures can be implemented efficiently by the fanny indexing in PyTorch [40] via our constructed link matrix.

View fusion. The above discussion is based on a 3D scene with a single 2D view. For a 3D scene with multiple 2D views, we can directly project 3D features to each view with the corresponding link matrix. But to transform multi-view 2D features to 3D space, we need to fuse them after back-projecting them to 3D space. Different from 3DMV [8] that simply aggregates multi-view features by max-pooling, we use two-layer sparse convolutions to learn the impact factors for each view at every point and then weighted sum them up by the learned impact factors as following:

$$\widetilde{F_{3D}}' = \sum_{r=1}^{R} w_r \cdot \widetilde{F_{3D}^r}, \tag{5}$$

where R is the number of views and w_r is a vector of weights that learned from all the back-projected features.

3.2. Bidirectional Projection Network

Based on the proposed BPM, we give our full bidirectional projection network (BPNet) as illustrated in Figure 1. The 2D and 3D sub-networks in our BPNet are both U-shaped networks with residual blocks [17], but based on conventional 2D convolution and sparse 3D convolution [12], respectively. During the training phase, we randomly sample n 2D views to maintain the diversity of 2D data, while during the testing phase, we divide the 2D frames into n groups and select one central view from each group to reduce the overlap among 2D views. In our implementation, n is set to three and the ablation is to be discussed in Section 4.

We firstly voxelize the 3D point clouds into volumes, and then the voxels represented in the sparse tensor format are fed into the 3D MinkowskiUNet, while the multi-view 2D images are simultaneously fed into the 2D UNet. During the decoder stages, we bidirectionally interact the features between 2D and 3D sub-networks in multiple pyramid levels, *i.e.*, P2, P3, P4, and P5 levels, via the proposed BPM, thus harvesting both the low- and high-level features in these two domains. And finally, both 2D and 3D semantic labels are predicted from 2D and 3D parts in our BPNet, respectively.

3.3. Implementation

As our goal is simultaneously predicting the 2D and 3D semantic labels, the loss function $\mathbf{L}$ consists of two terms:

$$\mathbf{L} = \mathbf{H}_{3d} + \lambda \cdot \mathbf{H}_{2d}, \tag{6}$$

where $\mathbf{H}_{3d}$ and $\mathbf{H}_{2d}$ are the cross entropy losses for 3D and 2D predictions, respectively; and λ is a weight to balance the 2D and 3D losses, which is empirically set to 0.1 in our experiments.

We implement our algorithm based on the PyTorch [40] platform and MinkowskiEngine [6] sparse convolution library. We train our BPNet on the ScanNetV2 dataset [7] for 100 epochs. The 2D UNet part is initialized from the classification weights pretrained on ImageNet [9], while the 3D part is initialized from scratch. We use SGD solver [2] with a base learning rate of 0.01 and a mini-batch size of 16. And we employ a poly learning rate scheduler with the power set to 0.9. Momentum and weight decay are set to 0.9 and 0.0001, respectively.

4. Experimental Evaluation

4.1. Dataset and Metric

ScanNetV2 [7] contains indoor scenes like offices and living rooms, with 2.5 million RGB-D frames in more than 1500 scans, annotated with 3D camera poses, surface reconstructions, and semantic segmentation. It is officially split into 1201 training and 312 validation scans that were taken from 706 different scenes, which means each scene was captured around one to three times, and 100 scans test set with hidden ground truth, used for the benchmark. For the 3D input, as done in [6], we extract points cloud from the reconstructed surfaces, and voxelize them into 3D volumes, each voxel is associate with a 3D feature (R, G, B), while for 2D input, we use the RGB images. Following MinkowskiNet [6], we set the voxel size to 2cm for the benchmark results, and set it to 5cm in the ablation study for efficient training. For the evaluation metrics, we use the mean of class-wise intersection over union (mIoU).

4.2. Comparison on the ScanNet Benchmark

3D semantic segmentation. To evaluate the effectiveness of our BPNet for 3D semantic segmentation, we compare our method with typical streams of methods on the test set of ScnaNetV2 in Table 1, including point-based methods: PointNet++ [43], MCCNN [19], PointASNL [64] and KP-FCNN [55]; sparse convolution based method: MinkowskiNet [6]; and joint 2D-3D-input based methods: 3DMV [8], MVPNet [24] and SPLATNet [50]. We can see that BPNet outperforms the point-based methods, *e.g.*, PointNet++, MCCNN, and KP-FCNN, by a large margin (≥ 6.5 mIoU), due to their limited receptive field and less effectiveness on feature extraction. For the strong sparse convolution based method, MinkowskiNet, although its network receptive field is very large and global context is rich, our method also outperforms it by 1.3 mIoU, since our method leverages the extra 2D features' information. And importantly, our method significantly outperforms other joint 2D-3D-input based methods (≥ 10.5 mIoU), since their methods adopt

Method	mIoU	bath	bed	bkshf	cab	chair	cntr	curt	desk	door	floor	other	pic	fridge	shower	sink	sofa	table	toilet	wall	window
PointNet++ [43]	33.9	58.4	47.8	45.8	25.6	36.0	25.0	24.7	27.8	26.1	67.7	18.3	11.7	21.2	14.5	36.4	34.6	23.2	54.8	52.3	25.2
SPLATNet† [50]	39.3	47.2	51.1	60.6	31.1	65.6	24.5	40.5	32.8	19.7	92.7	22.7	00.0	00.1	24.9	27.1	51.0	38.3	59.3	69.9	26.7
3DMV† [8]	48.4	48.4	53.8	64.3	42.4	60.6	31.0	57.4	43.3	37.8	79.6	30.1	21.4	53.7	20.8	47.2	50.7	41.3	69.3	60.2	53.9
FAConv[69]	63.0	60.4	74.1	76.6	59.0	74.7	50.1	73.4	50.3	52.7	91.9	45.4	32.3	55.0	42.0	67.8	68.8	54.4	89.6	79.5	62.7
MCCNN [19]	63.3	86.6	73.1	77.1	57.6	80.9	41.0	68.4	49.7	49.1	94.9	46.6	10.5	58.1	64.6	62.0	68.0	54.2	81.7	79.5	61.8
FPConv [33]	63.9	78.5	76.0	71.3	60.3	79.8	39.2	53.4	60.3	52.4	94.8	45.7	25.0	53.8	72.3	59.8	69.6	61.4	87.2	79.9	56.7
MVPNet† [24]	64.1	83.1	71.5	67.1	59.0	78.1	39.4	67.9	64.2	55.3	93.7	46.2	25.6	64.9	40.6	62.6	69.1	66.6	87.7	79.2	60.8
DCM-Net [47]	65.8	77.8	70.2	80.6	61.9	81.3	46.8	69.3	49.4	52.4	94.1	44.9	29.8	51.0	82.1	67.5	72.7	56.8	82.6	80.3	63.7
PointConv [62]	66.6	78.1	75.9	69.9	64.4	82.2	47.5	77.9	56.4	50.4	95.3	42.8	20.3	58.6	75.4	66.1	75.3	58.8	90.2	81.3	64.2
PointASNL [64]	66.6	70.3	78.1	75.1	65.5	83.0	47.1	76.9	47.4	53.7	95.1	47.5	27.9	63.5	69.8	67.5	75.1	55.3	81.6	80.6	70.3
KP-FCNN [55]	68.4	84.7	75.8	78.4	64.7	81.4	47.3	77.2	60.5	59.4	93.5	45.0	18.1	58.7	80.5	69.0	78.5	61.4	88.2	81.9	63.2
MinkowskiNet [6]	73.6	85.9	81.8	83.2	70.9	84.0	52.1	85.3	66.0	64.3	95.1	54.4	28.6	73.1	89.3	67.5	77.2	68.3	87.4	85.2	72.7
BPNet (Ours)†	74.9	90.9	81.8	81.1	75.2	83.9	48.5	84.2	67.3	64.4	95.7	52.8	30.5	77.3	85.9	78.8	81.8	69.3	91.6	85.6	72.3

Table 1. Comparison with the typical streams of methods on ScanNetV2 3D Semantic label benchmark, including point cloud based, sparse convolution based, and joint 2D-3D-input (marked with †) based methods.

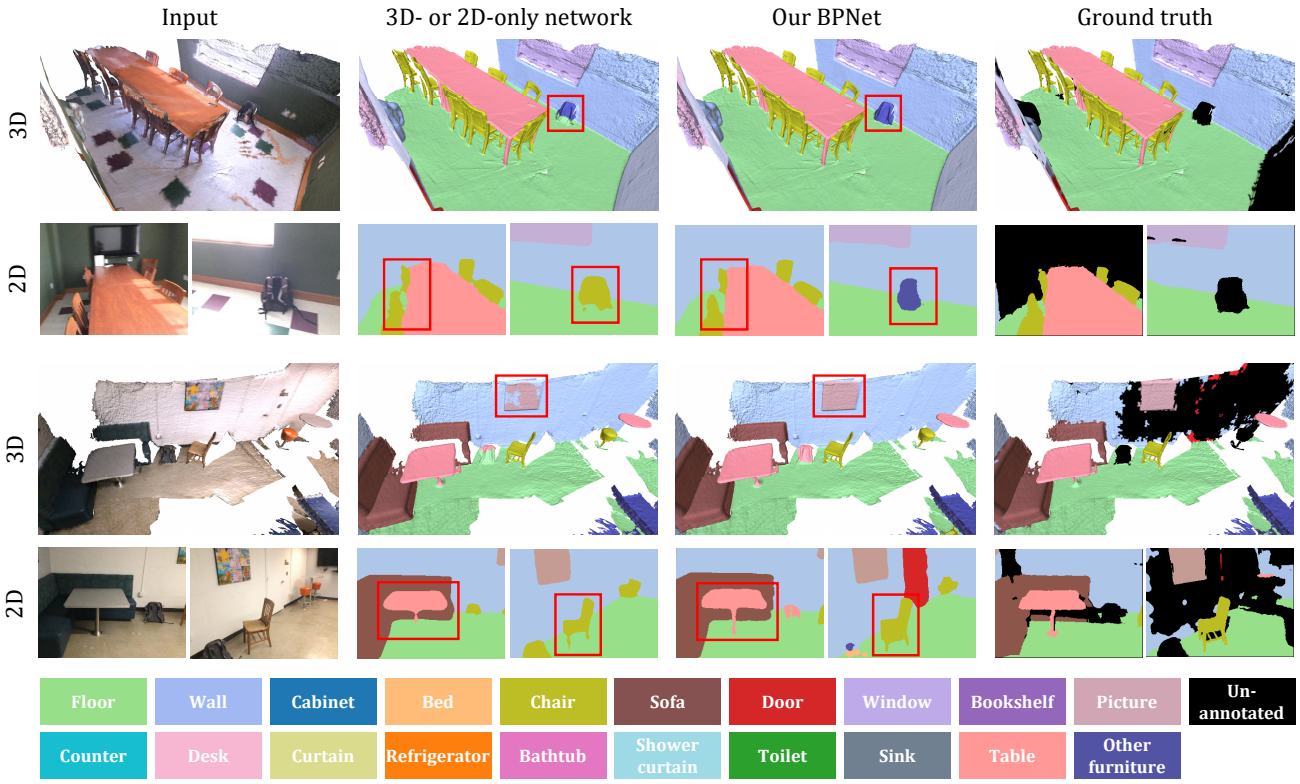

Figure 3. Qualitative 3D and 2D result examples of the 3D-only network, MinkowskiNet [6], 2D-only network, UNet34, our BPNet, and the ground truths. Different semantics are labeled as corresponding colors as shown in the bottom color palette. We highlight the differences between the results of BPNet and others by red boxes.

the unidirectional scheme to combine the 2D and 3D information, that means they only regard the 2D CNN as a pre-feature-extractor without bidirectionally interacting features between 2D and 3D CNNs. Besides, other information may further be utilized to boost the performance, like the instance annotation [14].

To qualitatively evaluate the 3D semantic segmentation results, we compare our results with the strong 3D-only method, MinkowskiNet [6], and also the ground truth in "3D" rows part in Figure 3. As shown in the red boxes, MinkowskiNet cannot correctly distinguish the boundaries in "backpack/floor" and "picture/wall", while our BPNet performs well. This is because the textures of input 3D data are too coarse, that makes it challenging to correctly predict

Method	mIoU	bath	bed	bkshf	cab	chair	cntr	curt	desk	door	floor	other	pic	fridge	shower	sink	sofa	table	toilet	wall	window
PSPNet [72]	47.5	49.0	58.1	28.9	50.7	6.7	37.9	61.0	41.7	43.5	82.2	27.8	26.7	50.3	22.8	61.6	53.3	37.5	82.0	72.9	56.0
UNet34 [46]	48.9	55.3	62.6	26.6	50.3	23.5	37.9	52.4	49.8	41.6	84.5	28.6	32.1	54.0	12.8	60.8	55.3	38.5	81.6	73.6	56.6
3DMV [8]	49.8	48.1	61.2	57.9	45.6	34.3	38.4	62.3	52.5	38.1	84.5	25.4	26.4	55.7	18.2	58.1	59.8	42.9	76.0	66.1	44.6
FuseNet† [16]	53.5	57.0	68.1	18.2	51.2	29.0	43.1	65.9	50.4	49.5	90.3	30.8	42.8	52.3	36.5	67.6	62.1	47.0	76.2	77.9	54.1
SSMA† [56]	57.7	69.5	71.6	43.9	56.3	31.4	44.4	71.9	55.1	50.3	88.7	34.6	34.8	60.3	35.3	70.9	60.0	45.7	90.1	78.6	59.9
RFBNet† [10]	59.2	61.6	75.8	65.9	58.1	33.0	46.9	65.5	54.3	52.4	92.4	35.5	33.6	57.2	47.9	67.1	64.8	48.0	81.4	81.4	61.4
BPNet (Ours)†	67.0	82.2	79.5	83.6	65.9	48.1	45.1	76.9	65.6	56.7	93.1	39.5	39.0	70.0	53.4	68.9	77.0	57.4	86.5	83.1	67.5

Table 2. Comparison with the top methods on ScanNetV2 2D Semantic label benchmark, including 2D-only, 3D predictions projection, and joint 2D-3D-input based methods (marked with †).

semantics only depend on the 3D input, while our method can additionally leverage the advantages in 2D data to integrate the high-quality texture in images for better predicting semantics. It is worth noting that, obtaining corresponding 2D input is not expensive as cameras are usually equipped with the 3D sensor in the common applications, *e.g.*, indoor/outdoor robotics and autonomous driving.

2D semantic segmentation. We then evaluate the effectiveness of BPNet for 2D semantic segmentation. We compare BPNet with top methods on the ScanNet benchmark, including 2D-only methods: PSPNet [72] and UNet [46]; method of projecting 3D predictions to 2D images: 3DMV [8]; and joint 2D-3D-input based methods: FuseNet [16], SSMA [56] and RFBNet [10], as shown in Table 2. Note that, the 3D and 2D results of our BPNet in Table 1 and 2 are produced by the same model. We can see that BPNet outperforms the 2D-only methods by a large margin ($\geq$ 18.1 mIoU). We admit that this comparison is not completely fair since BPNet also leverages 3D information, but such comparison gives evidence that 3D information can boost 2D semantic segmentation. For the method of projecting 3D predictions to 2D images, like 3DMV, although the 3D information is utilized, it still cannot perform as well as BPNet as it is not tailor-made for 2D semantic segmentation. For the joint 2D-3D-input based methods, like FuseNet, SSMA, and RFBNet, both 2D and 3D information are employed, BPNet still outperforms them a lot ($\geq$ 7.8 mIoU), thanks to our bidirectional projection module (BPM) that effectively interacts with 2D and 3D features to integrate the strengths of these two visual domains.

Also, we qualitatively compare the results of 2D-only network, UNet34, BPNet, and ground truth in "2D" rows part in Figure 3. We can see that BPNet not only correctly distinguishes backpack from the floor but also segments sharper and more accurate boundaries for the "chair" and "table". It is because of the underlying geometric clues provided by 3D features.

4.3. Ablation Study

To explore the effectiveness of different configurations for BPNet, we conduct the following ablation experiments on the validation set of ScanNetV2. To save training time, we use MinkowskiUNet18A for the 3D part and UNet34 for the 2D part in our BPNet, and the voxel size is set to 5cm in the first three ablations. Since there are too many 2D frames in the validation sequences, we follow the benchmark method to sub-sample a frame from every 100 frames to form the new validation set for 2D semantic segmentation.

Method	Projection Level	mIoU	
		2D	3D
UNet34	—	61.5	—
MinkowskiUNet18A	—	—	68.0
Ours W/ BPM	P2	63.5	70.3
Ours W/ BPM	P3	64.1	70.5
Ours W/ BPM	P4	62.3	69.7
Ours W/ BPM	P5	61.8	68.5
Ours W/ BPM	P2, P3, P4, P5	**65.1**	**70.6**
Ours W/ $\text{UPM}_{\text{2D}\rightarrow\text{3D}}$	P2, P3, P4, P5	62.2	69.7
Ours W/ $\text{UPM}_{\text{2D}\leftarrow\text{3D}}$	P2, P3, P4, P5	65.0	68.8

Table 3. 2D and 3D semantic segmentation results of different projection levels and directions on the validation set of ScanNetV2.

Ablation for projection level. As shown in Figure 1, the BPM is applied in all the four pyramid levels (P2, P3, P4, and P5) for our full method. For ablation, we apply BPM at a certain level and compare the results of baseline methods (UNet34 and MinkowskiUNet18A), our full method, and its variants. From the first seven rows in Table 3, we can see that our framework with BPM at every single level performs better than the 2D-only network, UNet34, or the 3D-only network, MinkowskiUNet18A. It means bidirectionally interacting 2D and 3D features even only at a single level can improve both 2D and 3D semantic segmentation, since useful information from the other domains is introduced. More importantly, our framework with BPM at all the four levels outperforms all the other variants, which shows bidirectionally interacting 2D and 3D features at both low- and high-levels can better integrate the advantages in the 2D and 3D domains.

Unidirectional projection *vs.* Bidirectional projection. To evaluate the effectiveness of projection directions, we

Number of Views	1	2	3	4	5
2D mIoU	**66.5**	65.8	65.1	63.9	63.6
3D mIoU	68.1	68.6	**70.6**	70.5	70.5

Table 4. 2D and 3D semantic segmentation results of different view numbers on the validation set of ScanNetV2.

Method	Voxel Size	mIoU	
		2D	3D
UNet34	—	61.5	—
MinkowskiUNet18A	5cm	—	68.0
MinkowskiUNet18A	2cm	—	72.1
Ours	5cm	65.1	70.6
Ours	2cm	**71.9**	**73.9**

Table 5. 2D and 3D semantic segmentation results of different voxel sizes on the validation set of ScanNetV2.

replace the bidirectional projection module (BPM) with the unidirectional projection module (UPM), *i.e.*, $\text{UPM}_{\text{2D}\rightarrow\text{3D}}$ and $\text{UPM}_{\text{2D}\leftarrow\text{3D}}$, and compare the results in Table 3. We can see that our method with $\text{UPM}_{\text{2D}\rightarrow\text{3D}}$ and $\text{UPM}_{\text{2D}\leftarrow\text{3D}}$ outperforms the 2D- or 3D-only baselines, but our method with BPM performs best for both 2D and 3D results. It evidences that 2D and 3D semantic segmentation can better benefit each other by bidirectionally interacting features.

Ablation for the number of 2D views. We then explore the influence of the number of 2D views, and the results are shown in Table 4. For 2D semantic segmentation, BPNet with one 2D view performs best while with five views perform worst. This is because when there is only one view, the 3D to 2D information transformation can be best focused. On the other hand, when the number of views increasing from one to three, the 3D semantic segmentation result becomes better and better, but it decreases slightly when the view number further increasing to five. It may because too few views cannot provide sufficient 2D information while too many views can hinder the network from extracting useful information while discarding the redundant information.

Ablation for voxel size. Finally, we evaluate the influence of voxel size for both 3D and 2D semantic segmentation, as shown in Table 5. Comparing the second and third rows, we can see that more fine-grained voxels can greatly improve the 3D semantic segmentation for the 3D-only network. And comparing the last two rows and others, we can see that decreasing voxel size can not only improve the performance of 3D semantic segmentation but also greatly improve the 2D semantic segmentation interestingly. It evidences that the 3D information is transformed into 2D CNNs and the higher quality 3D information can better boost the 2D semantic segmentation.

4.4. BPNet on NYUv2

Although our BPNet is designed for the 3D scene with 2D images, we can also convert the RGB-D data into the 3D scene to perform semantic segmentation. Note that, RGB-D data is known as 2.5D data rather than 3D data as the depth is view-dependent. To evaluate the generality of BPNet on 2.5D data, we apply it to the popular RGB-D semantic segmentation dataset, NYUv2 [38]. It contains 1,449 densely labeled pairs of aligned RGB and depth images. We followed the official training and testing set splits, which use 795 instances for training and 654 for testing. We converted the depth to point clouds according to the depth camera's matrix and back-projected the 2D annotations for the generated point clouds as our 3D labels.

Method	Accuracy
SceneNet [15]	52.5
Hermans *et al.* [18]	54.3
SemanticFusion [37]	59.2
Dai *et al.* [7]	60.7
3DMV [8]	71.2
BPNet	**73.5**

Table 6. Semantic segmentation results (13-class task) on NYUv2 [38] using dense pixel classification accuracy metric. Note that the reported result of Dai *et al.* is on the 11-class task.

Followed 3DMV [8], we adopt the configuration of the 13-class label and report the dense pixel classification accuracy compared with typical RGB-D based methods and joint 2D-3D method 3DMV in Table 6. Overall, our BPNet performs favorably against the typical RGB-D and joint 2D-3D baselines. The results on ScanNetV2 and NYUv2 demonstrate the generality of BPNet for different types of datasets.

5. Conclusion

In this work, we propose a bidirectional projection network, BPNet, to jointly perform 2D and 3D semantic segmentation, that leverages the complementary advantages in 2D and 3D data. We enable the bidirectional feature interacting between 2D and 3D CNNs in multiple pyramid levels via the proposed bidirectional projection module (BPM), such that the strengths of these two visual domains can be integrated for better scene recognition. BPNet achieves top performance on the ScanNetV2 benchmark and consistently outperforms our baseline with a single 2D/3D network. Also, with our BPNet, we can get more consistent 2D and 3D results. We believe the insight behind the BPM can also benefit other visual recognition tasks where 2D and 3D observations are both available, and it would advance related techniques in the community.

Acknowledgments. This project is supported by Shenzhen Science and Technology Program (No.JCYJ20180507182410327) and The Science and Technology Plan Project of Guangzhou (No.201704020141).

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*, 2017.

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTA*. 2010.

[3] Edwin Catmull. A subdivision algorithm for computer display of curved surfaces. Technical report, UTAH UNIV SALT LAKE CITY SCHOOL OF COMPUTING, 1974.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. *ICLR*, 2015.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2018.

[6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019.

[7] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017.

[8] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *ECCV*, 2018.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[10] Liuyuan Deng, Ming Yang, Tianyi Li, Yuesheng He, and Chunxiang Wang. Rfbnet: deep multimodal networks with residual fusion blocks for rgb-d semantic segmentation. *arXiv:1907.00135*, 2019.

[11] Raghudeep Gadde, Varun Jampani, Renaud Marlet, and Peter V Gehler. Efficient 2d and 3d facade segmentation using auto-context. *TPAMI*, 2017.

[12] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018.

[13] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *ECCV*, 2014.

[14] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3d instance segmentation. In *CVPR*, 2020.

[15] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Understanding real world indoor scenes with synthetic data. In *CVPR*, 2016.

[16] Caner Hazirbas, Lingni Ma, Csaba Domokos, and Daniel Cremers. Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *ACCV*, 2016.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[18] Alexander Hermans, Georgios Floros, and Bastian Leibe. Dense 3d semantic mapping of indoor scenes from rgb-d images. In *ICRA*, 2014.

[19] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. *TOG*, 2018.

[20] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, 2020.

[21] Zeyu Hu, Mingmin Zhen, Xuyang Bai, Hongbo Fu, and Chiew-lan Tai. Jsenet: Joint semantic segmentation and edge detection network for 3d point clouds. In *ECCV*, 2020.

[22] Jingwei Huang, Haotian Zhang, Li Yi, Thomas Funkhouser, Matthias Nießner, and Leonidas J Guibas. Texturenet: Consistent local parametrizations for learning from high-resolution signals on meshes. In *CVPR*, 2019.

[23] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019.

[24] Maximilian Jaritz, Jiayuan Gu, and Hao Su. Multi-view pointnet for 3d scene understanding. In *ICCVW*, 2019.

[25] Maximilian Jaritz, Tuan-Hung Vu, Raoul de Charette, Emilie Wirbel, and Patrick Pérez. xmuda: Cross-modal unsupervised domain adaptation for 3d semantic segmentation. In *CVPR*, 2020.

[26] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In *ICCV*, 2017.

[27] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *ICCV*, 2019.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[29] Abhijit Kundu, Xiaoqi Yin, Alireza Fathi, David Ross, Brian Brewington, Thomas Funkhouser, and Caroline Pantofaru. Virtual multi-view fusion for 3d semantic segmentation. In *ECCV*, 2020.

[30] Huan Lei, Naveed Akhtar, and Ajmal Mian. Seggcn: Efficient 3d point cloud segmentation with fuzzy spherical kernel. In *CVPR*, 2020.

[31] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *ICCV*, 2019.

[32] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NIPS*, 2018.

[33] Yiqun Lin, Zizheng Yan, Haibin Huang, Dong Du, Ligang Liu, Shuguang Cui, and Xiaoguang Han. Fpconv: Learning local flattening for point convolution. In *CVPR*, 2020.

[34] Wei Liu, Andrew Rabinovich, and Alexander C. Berg. Parsenet: Looking wider to see better. In *ICLRW*, 2016.

[35] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[36] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015.

[37] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *ICRA*, 2017.

[38] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[39] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NIPS*, 2019.

[41] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[42] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3D data. In *CVPR*, 2016.

[43] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017.

[44] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 3d graph neural networks for rgbd semantic segmentation. In *ICCV*, 2017.

[45] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, 2017.

[46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[47] Jonas Schult, Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. Dualconvmesh-net: Joint geodesic and euclidean convolutions on 3d meshes. In *CVPR*, 2020.

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[49] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017.

[50] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *CVPR*, 2018.

[51] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*, 2015.

[52] Hao Su, Fan Wang, Eric Yi, and Leonidas J. Guibas. 3D-assisted feature synthesis for novel views of an object. *ICCV*, 2015.

[53] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[54] Lyne P. Tchapmi, Christopher B. Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. SEGCloud: Semantic segmentation of 3D point clouds. In *3DV*, 2017.

[55] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019.

[56] Abhinav Valada, Rohit Mohan, and Wolfram Burgard. Self-supervised model adaptation for multimodal semantic segmentation. *IJCV*, 2020.

[57] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *ECCV*, 2018.

[58] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *CVPR*, 2019.

[59] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *CVPR*, 2018.

[60] Weiyue Wang and Ulrich Neumann. Depth-aware cnn for rgb-d segmentation. In *ECCV*, 2018.

[61] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *TOG*, 2019.

[62] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, 2019.

[63] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *ECCV*, 2018.

[64] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *CVPR*, 2020.

[65] Maoke Yang, Kun Yu, Chi Zhang, Zhiwei Li, and Kuiyuan Yang. Denseaspp for semantic segmentation in street scenes. In *CVPR*, 2018.

[66] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2016.

[67] Feihu Zhang, Jin Fang, Benjamin Wah, and Philip Torr. Deep fusionnet for point cloud semantic segmentation. In *ECCV*, 2020.

[68] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *CVPR*, 2018.

[69] Jiazhao Zhang, Chenyang Zhu, Lintao Zheng, and Kai Xu. Fusion-aware point convolution for online semantic 3d scene segmentation. In *CVPR*, 2020.

[70] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019.

[71] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. ICNet for real-time semantic segmentation on high-resolution images. In *ECCV*, 2018.

[72] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[73] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In *ECCV*, 2018.