ShapeConv: Shape-aware Convolutional Layer for Indoor RGB-D Semantic Segmentation

Jinming Cao[1] Hanchao Leng[1] Dani Lischinski[2] Danny Cohen-Or[3] Changhe Tu[1*] Yangyan Li[4*]
[1]Shandong University, China [2]The Hebrew University of Jerusalem, Israel
[3]Tel Aviv University, Israel [4]Alibaba Group, China
{jinming.ccao, hanchao.leng, danix3d, cohenor, changhe.tu, yangyan.lee}@gmail.com

Abstract

RGB-D semantic segmentation has attracted increasing attention over the past few years. Existing methods mostly employ homogeneous convolution operators to consume the RGB and depth features, ignoring their intrinsic differences. In fact, the RGB values capture the photometric appearance properties in the projected image space, while the depth feature encodes both the shape *of a local geometry as well as the* base *(whereabout) of it in a larger context. Compared with the* base*, the* shape *probably is more inherent and has a stronger connection to the semantics, and thus is more critical for segmentation accuracy. Inspired by this observation, we introduce a Shape-aware Convolutional layer (ShapeConv) for processing the depth feature, where the depth feature is firstly decomposed into a shape-component and a base-component, next two learnable weights are introduced to cooperate with them independently, and finally a convolution is applied on the re-weighted combination of these two components. ShapeConv is model-agnostic and can be easily integrated into most CNNs to replace vanilla convolutional layers for semantic segmentation. Extensive experiments on three challenging indoor RGB-D semantic segmentation benchmarks, i.e., NYU-Dv2(-13,-40), SUN RGB-D, and SID, demonstrate the effectiveness of our ShapeConv when employing it over five popular architectures. Moreover, the performance of CNNs with ShapeConv is boosted without introducing any computation and memory increase in the inference phase. The reason is that the learnt weights for balancing the importance between the shape and base components in ShapeConv become constants in the inference phase, and thus can be fused into the following convolution, resulting in a network that is identical to one with vanilla convolutional layers.*

1. Introduction

With the widespread use of depth sensors (such as Microsoft Kinect [31]), the availability of RGB-D data has boosted the advancement of RGB-D semantic segmentation, which contributes to an indispensable task in the computer vision community. Thanks to the flourishing of Convolutional Neural Networks (CNNs), recent studies mostly resort to CNNs for tackling this problem. Convolutional layers, deemed as the core building blocks of CNNs, are accordingly the key elements in RGB-D semantic segmentation models [6, 13, 15, 17, 21].

Figure 1. Visual demonstration of why the *shape* of an RGB-D image matters. Regarding the images on the top, lines with the same color share a same *shape*, yet with different *base*. The corresponding patches are shown on the bottom.

However, RGB and depth information are inherently different from each other. In particular, RGB values capture the photometric appearance properties in the projected image space, while the depth feature encodes both the *shape* of a local geometry as well as the *base* (whereabout) of it in a larger context. As a result, the convolution operator that is widely adopted for consuming RGB data might not be the optimal for processing the depth data. Taking Figure 1 as an example, we would expect the corresponding patches of the same chairs to have the same features, as they share

*Corresponding Author

the same *shape*. The shape is a more inherent property of the underlying object and has stronger connection to the semantics. We would expect to achieve shape invariance in the learning process. When a vanilla convolution operator is applied on these corresponding patches, the resulting features are different due to the differences in their *base* component, hindering the learning from achieving shape invariance. On the other hand, the *base* components cannot be simply discarded for pursuing the shape invariance in the current layer, as they form the *shape* in a followup layer with a larger context.

To address these problems, we propose a **Shape**-aware **Conv**lutional layer (ShapeConv), to learn the adaptive balance between the importance of *shape* and *base* information, giving the network the chance to focus more on the *shape* information whenever necessary for benefiting the RGB-D semantic segmentation task. We firstly decompose a patch[1] into two separate components, i.e., a *base-component* and a *shape-component*. The mean of patch values depicts the whereabout of the patch in a larger context, thus constitutes the base component, while the residual is the relative changes in the patch, which depicts the shape of the underlying geometry, thus constitutes to the shape component. Specifically, for an input patch (such as $\mathbb{P}_1$ in Figure 1), the *base* describes where the patch is, i.e., the distance from the observation point; while the *shape* expresses what the patch is, e.g., a chair corner. We then employ two operations, namely, base-product and shape-product, to respectively process these two components with two learnable weights, i.e., base-kernel and shape-kernel. The output from these two is then combined in an addition manner to form a shape-aware patch, which is further convolved with a normal convolutional kernel. In contrast to the original patch, the shape-aware one is capable of adaptively learning the shape characteristic with the shape-kernel, and the base-kernel serves to balance the contributions of the *shape* and the *base* for the final prediction.

In addition, since the base-kernel and shape-kernel become constants in the inference phase, we can fuse them into the following convolution kernel, resulting in a network that is identical to the one with vanilla convolutional layers. The proposed ShapeConv can be easily plugged into most CNNs as a replacement of the vanilla convolution in semantic segmentation without introducing any computation and memory increase in the inference phase. This simple replacement transforms CNNs designed for RGB data into ones better suited for consuming RGB-D data.

To validate the effectiveness of the proposed method, we conduct extensive experiments on three challenging RGB-D indoor semantic segmentation benchmarks: NYU-Dv2 [25](-13,-40), SUN RGBD [26], and SID [1]. We apply our ShapeConv to five popular semantic segmentation architectures and can observe promising performance improvements compared with baseline models. We found that ShapeConv can significantly improve the segmentation accuracy around the object boundaries (see Figure 5), which demonstrates the effective leveraging of the depth information[2].

[1]The operation unit of input features for the convolutional layer, whose spatial size is the same as the convolution kernel.

2. Related Work

CNNs have been widely used for semantic segmentation on RGB images [3, 4, 19, 18, 23, 33]. In general, existing segmentation architectures usually involve two stages: the backbone and the segmentation stage. The former stage is leveraged to extract features from RGB images, wherein popular models are ResNet [12], ResNeXt [29] which are pre-trained on the ImageNet dataset [24]. The latter stage aims to generate predictions based on the extracted features. Methods in this stage include Upsample [19], PPM [33] and ASPP [3, 4], etc. It is worth noting that both stages adopt the convolutional layers as the core building blocks.

As RGB semantic segmentation has been extensively studied in literature, a straightforward solution for RGB-D semantic segmentation is to adapt the well-developed architectures from the ones designed for RGB data. However, implementing such a idea is non-trivial due to the asymmetric modality problem between the RGB and the depth information. To tackle this, researchers have devoted efforts into two directions: designing dedicated architectures for RGB-D data [6, 8, 13, 15, 17, 21, 28], and presenting novel layers to enhance or replace the convolutional layers in RGB semantic segmentation [5, 27, 30]. Our method falls into the second category.

Methods in the first category propose to feed RGB and depth channels to two parallel CNNs streams, where the output features are fused with specific strategies. For example, [6] presents a gate-fusion method, [8, 13, 21] fuse the features in multi-levels of the backbone stages. Nevertheless, these methods mostly leverage separate networks to consume RGB and depth features, they are yet faced with two limitations: 1) it is hard to decide when is the best stage for the fusion to happen; and 2) the two-stream or multi-level way often results in large increase of computation.

In contrast, methods along the second direction target at designing novel layers based on the geometric characteristics of RGB-D data, which are more flexible and time-efficient. For instance, Wang *et al*. [27] proposed the depth-aware convolution to weight pixels based on a hand-crafted Gaussian function by leveraging the depth similarity between pixels. [30] presents a novel operator called malleable 2.5D convolution, to learn the receptive field along the depth-axis. [5] devises a S-Conv to infer the sampling offset of the convolution kernel guided by the 3D spatial

[2]Our code is released through https://github.com/hanchaoleng/ShapeConv.

information, enabling the convolutional layer to adjust the receptive field and geometric transformations. ShapeConv proposed a novel view of the content in each patch and a mechanism to leverage them adaptively with learnt weights. Moreover, ShapeConv can be converted into vanilla convolution in the inference phase, resulting in ZERO increase of memory and computation compared with the models with vanilla convolution.

3. Method

In this section, we first provide the basic formulation of the Shape-aware convolutional layer (ShapeConv) for RGB-D data, followed by its application in the training and inference phase. We end this section with the method architectures.

3.1. ShapeConv for RGB-D Data

Method Intuition. Given an input patch $\mathbb{P} \in R^{K_h \times K_w \times C_{in}}$, K_h and K_w are the spatial dimensions of the kernel; C_{in} represents the channel numbers in the input feature map, the output features from the vanilla convolution layer are obtained by,

$$\mathbb{F} = Conv(\mathbb{K}, \mathbb{P}), \tag{1}$$

where $\mathbb{K} \in R^{K_h \times K_w \times C_{in} \times C_{out}}$ denotes the learnable weights of kernels in a convolutional layer (The bias terms are not included for simplicity.); C_{out} represents the channel numbers in the output feature map. Each element of $\mathbb{F} \in R^{C_{out}}$ is calculated as,

$$\mathbb{F}_{c_{out}} = \sum_{i}^{K_h \times K_w \times C_{in}} (\mathbb{K}_{i,c_{out}} \times \mathbb{P}_i).$$

It can be easily recognized that $\mathbb{F}$ usually changes with respect to different values of $\mathbb{P}$. Take the two patches in the Figure 1, $\mathbb{P}_1$ and $\mathbb{P}_2$, as an example. The corresponding output features, $\mathbb{F}_1$ and $\mathbb{F}_2$ from the vanilla convolution layer are learned by: $\mathbb{F}_1 = Conv(\mathbb{K}, \mathbb{P}_1)$, $\mathbb{F}_2 = Conv(\mathbb{K}, \mathbb{P}_2)$. Since $\mathbb{P}_1$ and $\mathbb{P}_2$ are not identical (different distances from the observation points), accordingly, their features are usually different, and this may lead to distinct prediction results.

Nevertheless, $\mathbb{P}_1$ and $\mathbb{P}_2$, corresponding to the red regions in Figure 1, actually belong to the same class - chair. And vanilla convolutional layers cannot well handle such situations. In fact, there exists some invariants of these two patches, namely, the *shape*. It refers to the relative depth correlation under local features, which is however, unexpectedly ignored by the existing methods. In view of this, we propose to fill this gap via effectively modeling the *shape* for RGB-D semantic segmentation.

ShapeConv Formulation. Based on the aforementioned analysis, in this paper, we offer to decompose an input patch into two components: a base-component $\mathbb{P}_B$ describing where the patch is, and a shape-component $\mathbb{P}_S$ expressing what the patch is. Therefore, we refer the mean[3] of patch values to be $\mathbb{P}_B$, and its relative values to be as $\mathbb{P}_S$:

$$\begin{aligned} \mathbb{P}_B &= m(\mathbb{P}), \\ \mathbb{P}_S &= \mathbb{P} - m(\mathbb{P}), \end{aligned}$$

where $m(\mathbb{P})$ is the mean function on $\mathbb{P}$ (over the $K_h \times K_w$ dimensions), and $\mathbb{P}_B \in R^{1 \times 1 \times C_{in}}$, and $\mathbb{P}_S \in R^{K_h \times K_w \times C_{in}}$.

Note that directly convolved $\mathbb{P}_S$ with $\mathbb{K}$ in Equation 1 is sub-optimal, as the values from $\mathbb{P}_B$ contributes the class discrimination across patches. Thus, our ShapeConv instead leverages two learnable weights, $\mathbb{W}_B \in R^1$ and $\mathbb{W}_S \in R^{K_h \times K_w \times K_h \times K_w \times C_{in}}$, to separately consume the above two components. The outputted features are then combined in an element-wise addition manner, which forms a new shape-aware patch with the same size as the original one $\mathbb{P}$. The formulation of ShapeConv is given as,

$$\begin{aligned} \mathbb{F} &= ShapeConv(\mathbb{K}, \mathbb{W}_B, \mathbb{W}_S, \mathbb{P}) \\ &= Conv(\mathbb{K}, \mathbb{W}_B \diamond \mathbb{P}_B + \mathbb{W}_S * \mathbb{P}_S) \\ &= Conv(\mathbb{K}, \mathbf{P_B} + \mathbf{P_S}) \\ &= Conv((\mathbb{K}, \mathbf{P_{BS}}), \end{aligned} \tag{2}$$

where $\diamond$ and $*$ denote the base-product and shape-product operator, respectively, which are defined as,

$$\begin{cases} \mathbf{P_B} = \mathbb{W}_B \diamond \mathbb{P}_B \\ \mathbf{P_{B}}_{_{1,1,c_{in}}} = \mathbb{W}_B \times \mathbb{P}_{B_{1,1,c_{in}}}, \end{cases} \tag{3}$$

$$\begin{cases} \mathbf{P_S} = \mathbb{W}_S * \mathbb{P}_S \\ \mathbf{P_S}_{k_h,k_w,c_{in}} = \sum_i^{K_h \times K_w} (\mathbb{W}_{S_{i,k_h,k_w,c_{in}}} \times \mathbb{P}_{S_{i,c_{in}}}), \end{cases} \tag{4}$$

where c_{in}, k_h, k_w are the indices of the elements in C_{in}, K_h, K_w dimensions, respectively.

We reconstruct the shape-aware patch $\mathbf{P_{BS}}$ from the addition of $\mathbf{P_B}$ and $\mathbf{P_S}$, and $\mathbf{P_{BS}} \in R^{K_h \times K_w \times C_{in}}$, which enables it to be smoothly convolved by the kernel $\mathbb{K}$ of vanilla convolutional layer. Nevertheless, the $\mathbf{P_{BS}}$ is equipped with the important shape information which is learned by the two additional weights, making the convolutional layer to focus on the situations when merely using depth values fails.

3.2. ShapeConv in Training and Inference

Training phase. The proposed ShapeConv in Section 3.1 can effective leverage the *shape* information of patches. However, replacing vanilla convolutional layer with ShapeConv in CNNs introduces more computational

[3] As the depth values are obtained from a fixed observation point, we notice that the rotational transformations cannot be addressed due to the angle of view limitation. As a result, we focus more on the translational transformations in this paper.

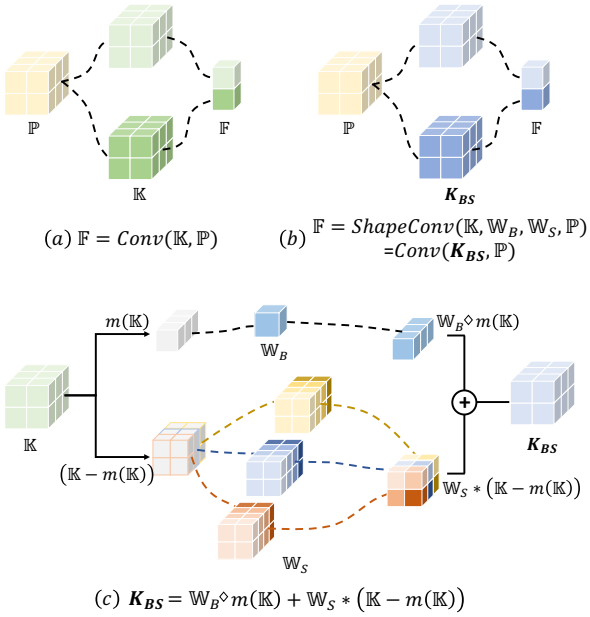

Figure 2. Comparison of vanilla convolution and ShapeConv within a patch $\mathbb{P}$. In this figure, $K_h = K_w = 2$, $C_{in} = 3$, and $C_{out} = 2$, "+" denotes element-wise addition. (a) Vanilla convolution with kernel $\mathbb{K}$; (b) ShapeConv with folding the $\mathbb{W}_B$ and $\mathbb{W}_S$ into $\mathbf{K_{BS}}$; (c) The computation of $\mathbf{K_{BS}}$ from $\mathbb{K}$, $\mathbb{W}_B$ and $\mathbb{W}_S$.

cost due to the two *product* operation in Equation 3 and 4. To tackle this problem, we propose to shift these two operations from patches to kernels,

$$\begin{cases} \mathbf{K_B} = \mathbb{W}_B \diamond \mathbb{K}_B \\ \mathbf{K_B}_{1,1,c_{in},c_{out}} = \mathbb{W}_B \times \mathbb{K}_{B_{1,1,c_{in},c_{out}}}, \end{cases}$$

$$\begin{cases} \mathbf{K_S} = \mathbb{W}_S * \mathbb{K}_S \\ \mathbf{K_S}_{k_h,k_w,c_{in},c_{out}} = \sum_i^{K_h \times K_w} (\mathbb{W}_{S_{i,k_h,k_w,c_{in}}} \times \mathbb{K}_{S_{i,c_{in},c_{out}}}), \end{cases}$$

where $\mathbb{K}_B \in R^{1\times1\times C_{in}\times C_{out}}$ and $\mathbb{K}_S \in R^{K_h\times K_w\times C_{in}\times C_{out}}$ denote the base-component of kernels and shape-component, respectively, and $\mathbb{K} = \mathbb{K}_B + \mathbb{K}_S$.

We therefore re-formalize ShapeConv the Equation 2 to following:

$$\begin{aligned} \mathbb{F} &= ShapeConv(\mathbb{K}, \mathbb{W}_B, \mathbb{W}_S, \mathbb{P}) \\ &= Conv(\mathbb{W}_B \diamond m(\mathbb{K}) + \mathbb{W}_S * (\mathbb{K} - m(\mathbb{K})), \mathbb{P}) \\ &= Conv(\mathbb{W}_B \diamond \mathbb{K}_B + \mathbb{W}_S * \mathbb{K}_S, \mathbb{P}) \\ &= Conv(\mathbf{K_B} + \mathbf{K_S}, \mathbb{P}) \\ &= Conv(\mathbf{K_{BS}}, \mathbb{P}), \end{aligned} \tag{5}$$

where $m(\mathbb{K})$ is the mean function on $\mathbb{K}$ (over the $K_h \times K_w$ dimensions). And we require $\mathbf{K_{BS}} = \mathbf{K_B} + \mathbf{K_S}$, $\mathbf{K_{BS}} \in R^{K_h\times K_w\times C_{in}\times C_{out}}$.

In fact, the two formulations of ShpeConv, i.e., Equation 2 and Equation 5 are mathematically equivalent, i.e.,

$$\begin{aligned} \mathbb{F} &= ShapeConv(\mathbb{K}, \mathbb{W}_B, \mathbb{W}_S, \mathbb{P}) \\ &= Conv(\mathbb{K}, \mathbf{P_{BS}}) \\ &= Conv(\mathbf{K_{BS}}, \mathbb{P}), \end{aligned} \tag{6}$$

$$\begin{aligned} \mathbb{F}_{c_{out}} &= \sum_i^{K_h\times K_w\times C_{in}} (\mathbb{K}_{i,c_{out}} \times \mathbf{P_{BS}}_i) \\ &= \sum_i^{K_h\times K_w\times C_{in}} (\mathbf{K_{BS}}_{i,c_{out}} \times \mathbb{P}_i), \end{aligned} \tag{7}$$

please refer to the Supp. for detailed proof. In this way, we utilize the ShapeConv in Equation 5 in our implementation as illustrated in Figure 2(b) and (c).

Inference phase. During inference, since the two additional weights i.e. $\mathbb{W}_B$ and $\mathbb{W}_S$, become constants, we can fuse them into $\mathbf{K_{BS}}$ as shown in Figure 2(c) with $\mathbf{K_{BS}} = \mathbb{W}_B \diamond \mathbb{K}_B + \mathbb{W}_S * \mathbb{K}_S$. And $\mathbf{K_{BS}}$ shares the same tensor size with $\mathbb{K}$ in Equation 1, thus, our ShapeConv is actually the same as the vanilla convolutional layer in Figure 2(a). In other words, when replacing vanilla convolution with ShapeConv, there would introduce zero additional inference time.

3.3. ShapeConv-enhanced Network Architecture

Different from devising specially dedicated architectures for RGB-D segmentation [21, 22, 17], the proposed ShapeConv is a more generalized approach that can be easily plugged into most CNNs as a replacement for the vanilla convolution in semantic segmentation, which is then transformed for adapting the RGB-D data.

Figure 3 depicts an example of the overall method architecture. In order to leverage the advanced backbones in semantic segmentation, we firstly require to convert the input features from RGB images to RGB-D data via the concatenation of the RGB and D information. In practice, D can be depth values [11, 20] or HHA[4] images [10, 19, 16, 6]. We then replace the vanilla convolution layer with the ShapeConv in both the backbone and segmentation stages. It is worth noting that, $\mathbb{W}_B$ is initialized to one, $\mathbb{W}_S$ can be viewed as C_{in} square $(K_h \times K_w) \times (K_h \times K_w)$ matrices, which are initialized to the identity matrix. In this way, ShapeConv is equivalent to the vanilla convolution at the beginning of training since $\mathbf{K_{BS}} = \mathbb{K}$. This initialization approach offers two advantages: 1) It makes the ShapeConv-enhanced networks do not interfere with the RGB data, i.e., the RGB features are processed in the same way as before. 2) It facilitates ShapeConv to reuse the parameters from pre-trained models.

Thus, with this approach, future advances in RGB semantic segmentation architectures can be easily transferred

[4]Horizontal disparity, Height above ground and normal Angle to the vertical axis.

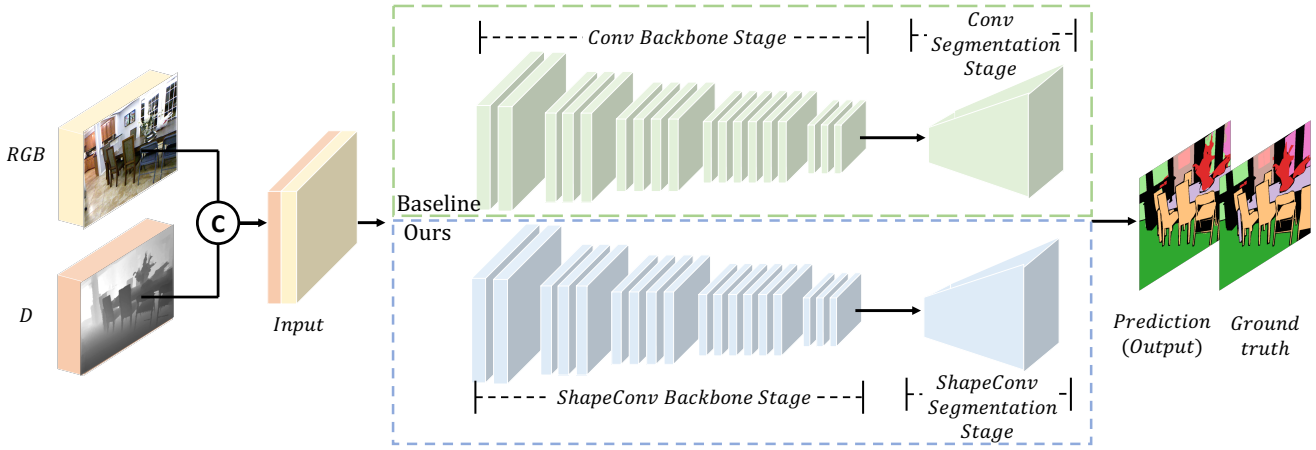

Figure 3. The overall semantic segmentation network architecture. In this figure, yellow and orange cube denote the RGB and D inputs; "C" denotes channel-wise concatenation; Green and blue boxes denote architectures consisting of vanilla convolutional layers and ShapeConv layers, respectively.

to consuming the RGB-D data, greatly reducing the effort that would otherwise be spent on designing dedicated networks for RGB-D semantic segmentation. We have shown the results of building RGB-D segmentation networks with this style using several popular architectures [3, 4, 18, 23, 33] in Sec 4.2.

4. Experiments

Datasets and metrics. Among the existing RGB-D segmentation problems, the indoor semantic segmentation is rather challenging, as the objects are often complex and with severe occlusions [5]. Thus, in order to validate the effectiveness of the proposed method, we conducted experiments on three indoor RGB-D benchmarks: NYU-Depth-V2 (NYUDv2-13 and -40) [25], SUN-RGBD [26] and Stanford Indoor Dataset (SID) [1]. NYUDv2 contains 1,449 RGB-D scene images, where 795 images are split for training and 654 images for testing. We adopted two popular settings for this dataset, i.e., 13-class [25] and 40-class [9], where all pixels are labeled with 13 and 40 classes, respectively. SUN-RGBD is composed of 10,355 RGB-D indoor images with 37 categories for each pixel label. We followed the widely used setting in [26] to split the dataset into a training set of 5285 images and a testing set of 5050 images. SID contains 70, 496 RGB-D images with 13 object categories. In particular, areas 1, 2, 3, 4, and 6 used for the training and Area 5 is for testing following [27].

We reported the results using the same evaluation protocol and metrics as FCN [19], i.e., Pixel Accuracy (*Pixel Acc.*), Mean Accuracy (*Mean Acc.*), Mean Region Intersection Over Union (*Mean IoU*), and Frequency Weighted Intersection Over Union (*f.w. IoU*).

Comparison protocol. We adopted several popular architectures with different backbones as our baseline methods to demonstrate the effectiveness and generalization capability of ShapeConv. For all the baseline methods, we only replaced the vanilla convolutional layers with our ShapeConv, without any change to other settings. This guarantees that the obtained performance improvements is due to the application of ShapeConv, but not other factors.

Table 1. Performance comparison with baselines on NYUDv2-13 dataset. Deeplabv3+ is the adopted architecture.

Back bone	Setting	Pixel Acc.(%)	Mean Acc.(%)	Mean IoU.(%)	f.w. IoU.(%)
ResNet 50 [12]	Baseline	80.0	72.5	60.8	67.6
	Baseline★	80.6	72.7	61.6	68.5
	Ours	80.4	73.0	61.8	68.1
	Ours★	81.1	73.4	62.7	69.1
	+	0.4	0.5	1.0	0.5
	+★	0.5	0.7	1.1	0.6
ResNet 101 [12]	Baseline	80.0	73.4	61.3	67.6
	Baseline★	81.0	74.3	63.1	68.9
	Ours	81.2	74.9	62.9	69.1
	Ours★	81.9	75.7	64.0	70.1
	+	1.2	1.5	1.6	1.5
	+★	0.9	1.4	0.9	1.2
ResNext 101_32x8d [29]	Baseline	81.8	73.9	63.2	70.1
	Baseline★	82.2	74.4	63.7	70.6
	Ours	82.6	75.7	65.1	71.2
	Ours★	82.9	76.0	65.6	71.6
	+	0.8	1.8	1.9	1.1
	+★	0.7	1.6	1.9	1.0

Implementation Details. We used the ResNet [12] and ResNeXt [29] initialized with the pre-trained model on ImageNet [24] in the backbone stage. If not otherwise noted, the inputs of both the baseline and ours are the concatenation of RGB and HHA images. We adopted both single-scale and multi-scale testing strategies during inference. For the latter one, left-right flipped images and five scales are exploited: [0.5, 0.75, 1.0, 1.25, 1.5, 1.75]. ★ in tables of this section denotes the multi-scale strategy. Note that, no post-processing tricks like CRF [2] is used in our experiments.

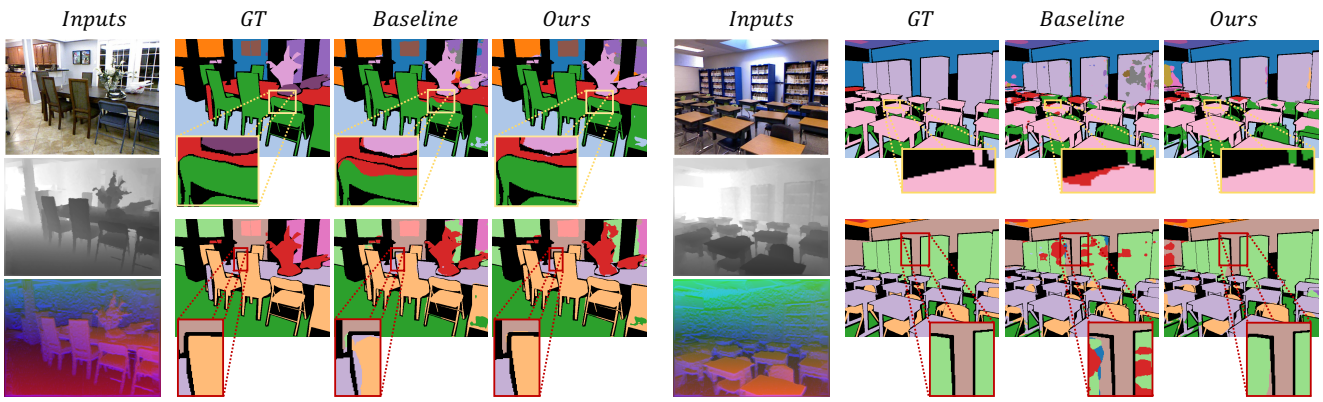

Figure 4. Visualization results from NYUDv2 dataset. Input column denotes RGB, Depth, HHA images from top to bottom; the black regions in the GT, Baseline and Ours indicate the ignored category. The upper and lower cases are from NYUDv2-40 and NYUDv2-13, respectively.

Table 2. Performance comparison with baselines on NYUDv2-40 dataset. Deeplabv3+ is the adopted architecture.

Back bone	Setting	Pixel Acc.(%)	Mean Acc.(%)	Mean IoU.(%)	f.w. IoU.(%)
ResNet 50 [12]	Baseline	73.1	57.7	45.6	59.2
	Baseline★	74.2	59.0	47.1	60.2
	Ours	74.1	59.1	47.3	60.5
	Ours★	75.0	60.4	48.8	61.4
	+	1.0	1.4	1.7	1.3
	+★	0.8	1.4	1.7	1.2
ResNet 101 [12]	Baseline	73.4	58.9	45.9	59.7
	Baseline★	74.4	60.2	47.6	60.7
	Ours	74.5	59.5	47.4	60.8
	Ours★	75.5	60.7	49.0	61.7
	+	1.1	0.6	1.59	1.1
	+★	1.1	0.5	1.4	1.0
ResNext 101_32x8d [29]	Baseline	74.7	61.5	48.9	61.5
	Baseline★	75.4	62.6	50.3	62.2
	Ours	75.8	62.8	50.2	62.6
	Ours★	76.4	63.5	51.3	63.0
	+	1.1	1.3	1.3	1.1
	+★	1.0	0.9	1.0	0.8

4.1. Experiments on Different Datasets

NYUDv2 Dataset. We adopted two popular settings for this dataset, i.e., 13-class [25] and 40-class [9], and show the results of baseline and our method with different backbones on NYUDv2-13 and NYUDv2-40 in Table 1 and Table 2, respectively. It can be seen that architectures with ShapeConv outperform the baselines with a large margin under all settings.

We also compare the performance of our ShapeConv with several recently developed methods in Table 3 and Table 4. As illustrated in Table 3, ShapeConv achieves the best over all the four metrics on NYUDv2-13. Compared to the recently proposed method [32], our approach yields around 6.3% improvements on Mean IOU which is the most commonly used metric for semantic segmentation. In addition, our method also achieves a competitive performance on NYUDv2-40 in Table 4.

Table 3. Performance comparison with other methods on NYUDv2-13 dataset.

Method	Pixel Acc.(%)	Mean Acc.(%)	Mean IoU.(%)	f.w. IoU.(%)
Eigen [7]	75.4	66.9	-	-
MVCNet [20]	77.8	69.5	57.3	-
Ours	**82.6**	**75.7**	**65.1**	**71.2**
MVCNet [20]★	79.1	70.6	59.1	-
PVNet [32]★	82.5	74.4	59.3	-
Ours★	**82.9**	**76.0**	**65.6**	**71.6**

Table 4. Performance comparison with other methods on NYUDv2-40 dataset.

Method	Pixel Acc.(%)	Mean Acc.(%)	Mean IoU.(%)	f.w. IoU.(%)
FCN [19]	65.4	46.1	34.0	49.5
LSD-GF [6]	71.9	60.7	45.9	59.3
D-CNN [27]	-	61.1	48.4	-
MMAF-Net [8]	72.2	59.2	44.8	-
ACNet [13]	-	-	48.3	-
Ours	**75.8**	**62.8**	**50.2**	**62.6**
CFN [17]★	-	-	47.7	-
3DGNN [22]★	-	55.7	43.1	-
RDF [21]★	76.0	62.8	50.1	-
M2.5D [30]★	**76.9**	-	50.9	-
SGNet [5]★	76.8	63.3	51.1	-
Ours★	76.4	**63.5**	**51.3**	**63.0**

SUN-RGBD Dataset. The comparison results between baseline and ours with SUN-RGBD dataset are reported in Table 5. It can be observed that our ShapeConv also produces a positive effect under all settings. We also compared the performance of ours with several recently developed methods in Table 6. It is worth noting that the performance of the ShapeConv-enhanced Network with backbone of ResNet-50 in Table 5 has already achieved better results than several methods in Table 6, such as 3DGNN-101 [22]

Table 5. Performance comparison with baselines on SUN-RGBD dataset. The architectures adopted in this table is deeplabv3+ with different backbones.

Backbone	Setting	Pixel Acc.(%)	Mean Acc.(%)	Mean IoU.(%)	f.w. IoU.(%)
ResNet 50 [12]	Baseline	81.1	56.5	45.5	69.7
	Baseline★	81.4	57.5	46.6	70.0
	Ours	81.6	56.8	46.3	70.3
	Ours★	81.9	57.9	47.7	70.6
	+	0.5	0.3	0.8	0.6
	+★	0.5	0.4	1.1	0.6
ResNet 101 [12]	Baseline	81.6	57.8	46.9	70.4
	Baseline★	81.6	58.4	47.6	70.5
	Ours	82.0	58.5	47.6	71.2
	Ours★	82.2	59.2	48.6	71.3
	+	0.4	0.7	0.7	0.8
	+★	0.6	0.8	1.0	0.8

and RDF-152 [21] which take the ResNet-101 and -152 as backbone, respectively.

Table 6. Performance comparison on SUN-RGBD dataset.

Method	Pixel Acc.(%)	Mean Acc.(%)	Mean IoU.(%)	f.w. IoU.(%)
3DGNN-101 [22]	-	55.7	44.1	-
D-CNN-50 [27]	-	53.5	42.0	-
MMAF-Net-152 [8]	81.0	58.2	47.0	-
SGNet-101 [5]	81.0	**59.8**	47.5	-
Ours-101	**82.0**	58.5	**47.6**	**71.2**
CFN-101 [17]★	-	-	48.1	-
3DGNN-101 [22]★	-	57.0	45.9	-
RDF-152 [21]★	81.5	60.1	47.7	-
SGNet-101 [5]★	82.0	**60.7**	48.6	-
Ours-101★	**82.2**	59.2	**48.6**	**71.3**

SID Dataset. Note that SID dataset is much larger than the other two datasets, contributing to a better testbed for evaluating RGB-D semantic segmentation model capabilities. The results on SID dataset between the baseline with ours and the state-of-the-art methods are reported in Table 7. We can observe that our ShapeConv surpasses these methods with a large margin. Note that even though we utilized a strong baseline (ResNet-101 backbone) which surpasses MMAF-Net-152 (ResNet-152 backbone) with 1.7% Mean IoU, our ShapeConv can still achieves a 6% Mean IoU improvement. This highlights the effectiveness of our method.

Table 7. Performance comparison on SID dataset. The architectures of baseline and ours adopted in this table is deeplabv3+ with ResNet-101 backbone and the "+" denote the deltas relative to the baseline method.

Method	Pixel Acc.(%)	Mean Acc.(%)	Mean IoU.(%)	f.w. IoU.(%)
D-CNN [27]	65.4	55.5	39.5	49.9
MMAF-Net-152 [8]	76.5	62.3	52.9	-
Baseline-101	78.7	63.2	54.6	65.6
Ours-101	**82.7**	**70.0**	**60.6**	**71.2**
+	4.0	6.8	6.0	5.6

4.2. Experiments on Different Architectures

Our proposed ShapeConv is a general layer for RGB-D semantic segmentation which can be easily plugged into most CNNs as a replacement for the vanilla convolution in semantic segmentation. To verify its generalization properties, we also evaluated the effectiveness of our method in several representative semantic segmentation architectures: Deeplabv3+ [4], Deeplabv3 [3], UNet [23], PSPNet [33] and FPN [18] with different backbones (ResNet-50 [12], ResNet-101 [12]) on NYUDv2-40 dataset, and reported the performance in Table 8. We can see that ShapeConv brings significant performance improvements under all settings, demonstrating the generalization capability of our method.

Table 8. Performance comparison with different baseline methods on NYUDv2-40 dataset.

Architecture	Back bone	Setting	Pixel Acc.(%)	Mean Acc.(%)	Mean IoU.(%)	f.w. IoU.(%)
Deeplabv3+ [4]	Res Net 101	Baseline	73.4	58.9	45.9	59.7
		Ours	74.5	59.5	47.4	60.8
		+	1.1	0.6	1.5	1.1
	Res Net 50	Baseline	73.1	57.7	45.6	59.2
		Ours	74.1	59.1	47.3	60.5
		+	1.0	1.4	1.7	1.3
Deeplabv3 [3]	Res Net 101	Baseline	73.3	57.3	45.1	59.2
		Ours	73.6	58.5	46.4	59.7
		+	0.3	1.2	1.3	0.5
	Res Net 50	Baseline	71.6	55.5	43.2	57.2
		Ours	72.8	56.6	44.9	58.5
		+	1.2	1.1	1.7	1.3
UNet [23]	Res Net 101	Baseline	70.9	54.7	42.1	57.7
		Ours	72.3	56.5	43.9	58.8
		+	1.4	1.8	1.8	1.1
	Res Net 50	Baseline	70.0	51.7	39.7	55.5
		Ours	70.8	54.1	42.0	56.9
		+	0.8	2.4	2.3	1.4
PSPNet [33]	Res Net 101	Baseline	72.8	56.8	44.2	58.9
		Ours	73.3	59.2	46.3	59.6
		+	0.5	2.4	2.1	0.7
	Res Net 50	Baseline	71.1	53.6	42.0	56.7
		Ours	72.0	56.2	44.0	57.7
		+	0.9	2.6	2.0	1.0
FPN [18]	Res Net 101	Baseline	72.8	57.3	44.7	59.1
		Ours	73.6	58.4	45.9	60.0
		+	0.8	1.1	1.2	0.9
	Res Net 50	Baseline	70.3	52.8	40.9	56.0
		Ours	71.5	54.9	42.8	57.5
		+	1.2	2.1	1.9	1.5

4.3. Visualization

Figure 4 illustrates the qualitative results on NYUDv2-13 and -40, more results can be found in the Supp. As shown in this figure, the depth information, especially the detailed one, can be well utilized by ShapeConv to extract the object features. For instance, the chair and table regions in the top example of Figure 4(a) are with gradually changed colors, making it hard to predict accurate segmentation boundaries of the baseline method. The shape features learned by ShapeConv makes the accurate cut following the geometric hints compare with the conventional convolutional layer. For other two cases, i.e., the chair in the bottom example of Figure 4(a) and the desk in the top example of Figure 4(b), the ShapeConv can also significantly improve the segmentation results in edge areas compared with

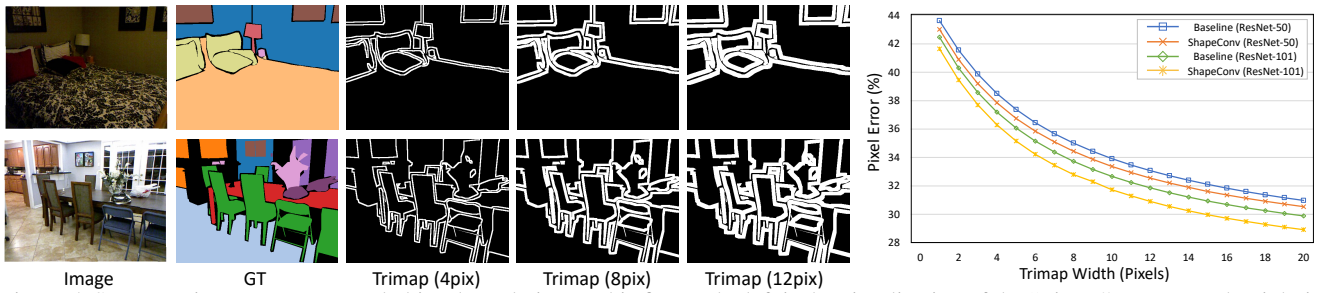

Figure 5. Segmentation accuracy around object boundaries. In this figure, the left is the visualization of the "trimap" measure; The right is the percent of misclassified pixels within trimaps of different widths.

the baseline. It is worth noting that for the multiple bookshelves in the bottom example of Figure 4(b), ShapeConv achieves more consistent predictions. This is because our ShapeConv yields a positive tendency for smoothing neighborhood regions within same classes.

To validate the effectiveness of our method on modeling the depth information, we adopted the comparison strategy proposed by Kohli *et al*. [14]. Specifically, we counted the relative number of misclassified pixels within a narrow band ("trimap") surrounding ground-truth object boundaries. As shown in Figure 5, our method outperforms the baseline across all trimap widths. This further demonstrates the segmentation effectiveness of our method on edge areas, where the shape information matters.

4.4. Ablation Study

We conducted ablation experiments to validate the indispensability of the two introduced weights in Equation 5. As can be observed in Table 9, the model performance degrades when removing either $\mathbb{W}_B$ or $\mathbb{W}_S$, or both. This proves that both the base-kernel and shape-kernel are essential for the final performance improvement, and combing these two achieves the best results.

Table 9. Performance comparison with and without $\mathbb{W}_B$ and $\mathbb{W}_S$ in ShapeConv on NYUDv2-40. The architecture adopted in this table is deeplabv3+ with ResNet-101 as backbone.

$\mathbb{W}_B$	$\mathbb{W}_S$	Pixel Acc.(%)	Mean Acc.(%)	Mean IoU.(%)	f.w. IoU.(%)
		73.4	58.9	45.9	59.7
✓		73.9	59.4	47.0	60.1
	✓	74.1	59.2	46.3	60.1
✓	✓	74.5	59.5	47.4	60.8

To provide a more in-depth analysis of ShapeConv, we conducted detailed ablation studies on the NYUDv2-40 dataset with deeplabv3+ and ResNet-101 as baseline and backbone, respectively. Results on more datasets can be found at the Supp. Table 10 illustrates the results and the key observations from this table are as follows: 1) The input features with HHA outperform the Depth images for the baseline and ours; 2) Replacing the vanilla convolution with ShapeConv leads to considerable performance improvements on both Depth and HHA; 3) The multi-scale setting in testing phase brings more performance gains; 4) Cascading the ShapeConv with HHA and multi-scale testing can achieve the best result.

Table 10. Ablation study of the proposed ShapeConv on the NYUDv2-40 dataset. RGB, Detph and HHA denote the inputs consisting of RGB images, depth images and HHA images.

Setting	Pixel Acc.(%)	Mean Acc.(%)	Mean IoU.(%)	f.w. IoU.(%)
a.RGB	71.8	56.9	43.9	57.3
b.RGB+Depth	72.8	58.9	44.9	57.7
c.RGB+Depth★	73.9	59.1	46.8	60.0
d.RGB+HHA	73.4	58.9	45.9	59.7
e.RGB+HHA★	74.4	60.2	47.6	60.7
f.RGB+Depth+ShapeConv	73.9	58.2	46.2	60.0
g.RGB+Depth+ShapeConv★	74.8	59.2	47.5	60.8
h.RGB+HHA+ShapeConv	74.5	59.5	47.4	60.8
i.RGB+HHA+ShapeConv★	75.5	60.7	49.0	61.7

5. Conclusion

In this paper, we propose a ShapeConv layer to effectively leverage the depth information for RGB-D semantic segmentation. In particular, an input patch is firstly decomposed into two components, i.e., *shape* and *base*, which are then decorated with two corresponding learnable weights before the convolution is applied. We have conducted extensive experiments on several challenging indoor RGB-D semantic segmentation benchmarks and promising experimental results can be observed. Moreover, it is worth noting that our ShapeConv introducing no additional computation or memory in comparison with the vanilla convolution during inference, yet with superior performance.

In fact, the shape-component is inherent in the local geometry and highly relevant to the semantics in images. In the future, we plan to expand the application scope to other geometry entities, such as point clouds, where the shape-base decomposition is more challenging due to the additional degree of freedom.

Acknowledgments. This work is supported by the National Key Research and Development Program of China grant No.2017YFB1002603, the National Science Foundation of China General Program grant No.61772317, 61772318 and 62072284, "Qilu" Young Talent Program of Shandong University, and the Research Intern Program of Alibaba Group.

References

[1] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

[2] 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. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 801–818, 2018.

[5] Lin-Zhuo Chen, Zheng Lin, Ziqin Wang, Yong-Liang Yang, and Ming-Ming Cheng. Spatial information guided convolution for real-time rgbd semantic segmentation. *IEEE Transactions on Image Processing*, 30:2313–2324, 2021.

[6] Yanhua Cheng, Rui Cai, Zhiwei Li, Xin Zhao, and Kaiqi Huang. Locality-sensitive deconvolution networks with gated fusion for rgb-d indoor semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3029–3037, 2017.

[7] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[8] Fahimeh Fooladgar and Shohreh Kasaei. Multi-modal attention-based fusion model for semantic segmentation of rgb-depth images. *arXiv preprint arXiv:1912.11691*, 2019.

[9] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 564–571, 2013.

[10] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[11] Caner Hazirbas, Lingni Ma, Csaba Domokos, and Daniel Cremers. Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *Asian Conference on Computer Vision*, pages 213–228. Springer, 2016.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[13] Xinxin Hu, Kailun Yang, Lei Fei, and Kaiwei Wang. Acnet: Attention based network to exploit complementary features for rgbd semantic segmentation. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1440–1444. IEEE, 2019.

[14] Pushmeet Kohli, Philip HS Torr, et al. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

[15] Siqi Li, Changqing Zou, Yipeng Li, Xibin Zhao, and Yue Gao. Attention-based multi-modal fusion network for semantic scene completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11402–11409, 2020.

[16] Zhen Li, Yukang Gan, Xiaodan Liang, Yizhou Yu, Hui Cheng, and Liang Lin. Lstm-cf: Unifying context modeling and fusion with lstms for rgb-d scene labeling. In *Proceedings of the European Conference on Computer Vision*, pages 541–557. Springer, 2016.

[17] Di Lin, Guangyong Chen, Daniel Cohen-Or, Pheng-Ann Heng, and Hui Huang. Cascaded feature network for semantic segmentation of rgb-d images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1311–1319, 2017.

[18] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[20] Lingni Ma, Jörg Stückler, Christian Kerl, and Daniel Cremers. Multi-view deep learning for consistent semantic mapping with rgb-d cameras. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 598–605. IEEE, 2017.

[21] Seong-Jin Park, Ki-Sang Hong, and Seungyong Lee. Rdfnet: Rgb-d multi-level residual feature fusion for indoor semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4980–4989, 2017.

[22] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 3d graph neural networks for rgbd semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5199–5208, 2017.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 234–241. Springer, 2015.

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[25] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of the European Conference on Computer Vision*, pages 746–760. Springer, 2012.

[26] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 567–576, 2015.

[27] Weiyue Wang and Ulrich Neumann. Depth-aware cnn for rgb-d segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 135–150, 2018.

[28] Yikai Wang, Wenbing Huang, Fuchun Sun, Tingyang Xu, Yu Rong, and Junzhou Huang. Deep multimodal fusion by channel exchanging. *Advances in Neural Information Processing Systems*, 33, 2020.

[29] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017.

[30] Yajie Xing, Jingbo Wang, and Gang Zeng. Malleable 2.5 d convolution: Learning receptive fields along the depth-axis for rgb-d scene parsing. *arXiv preprint arXiv:2007.09365*, 2020.

[31] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10, 2012.

[32] Cheng Zhao, Li Sun, Pulak Purkait, Tom Duckett, and Rustam Stolkin. Dense rgb-d semantic mapping with pixel-voxel neural network. *Sensors*, 18(9):3099, 2018.

[33] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2881–2890, 2017.