3DMV: Joint 3D-Multi-view Prediction for 3D Semantic Scene Segmentation

Angela Dai[1(✉)] and Matthias Nießner[2]

[1] Stanford University, Stanford, USA
adai@cs.stanford.edu
[2] Technical University of Munich, Munich, Germany

Abstract. We present 3DMV, a novel method for 3D semantic scene segmentation of RGB-D scans in indoor environments using a joint 3D-multi-view prediction network. In contrast to existing methods that either use geometry *or* RGB data as input for this task, we combine both data modalities in a joint, end-to-end network architecture. Rather than simply projecting color data into a volumetric grid and operating solely in 3D – which would result in insufficient detail – we first extract feature maps from associated RGB images. These features are then mapped into the volumetric feature grid of a 3D network using a differentiable back-projection layer. Since our target is 3D scanning scenarios with possibly many frames, we use a multi-view pooling approach in order to handle a varying number of RGB input views. This learned combination of RGB and geometric features with our joint 2D-3D architecture achieves significantly better results than existing baselines. For instance, our final result on the ScanNet 3D segmentation benchmark **increases from 52.8% to 75% accuracy** compared to existing volumetric architectures.

1 Introduction

Semantic scene segmentation is important for a large variety of applications as it enables understanding of visual data. In particular, deep learning-based approaches have led to remarkable results in this context, allowing prediction of accurate per-pixel labels in images [14,22]. Typically, these approaches operate on a single RGB image; however, one can easily formulate the analogous task in 3D on a per-voxel basis [5,13,21,34,40,41], which is a common scenario in the context of 3D scene reconstruction. In contrast to 2D, the third dimension offers a unique opportunity as it not only predicts semantics, but also provides a spatial semantic map of the scene content based on the underlying 3D representation. This is particularly relevant for robotics applications since a robot relies not only on information of *what* is in a scene but also needs to know *where* things are.

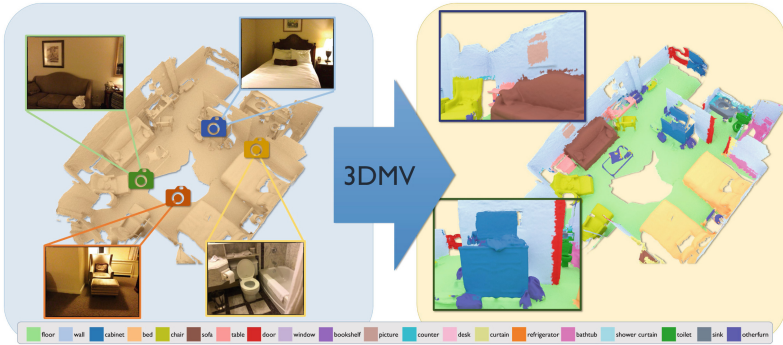
https://github.com/angeladai/3DMV.

Fig. 1. 3DMV takes as input a reconstruction of an RGB-D scan along with its color images (left), and predicts a 3D semantic segmentation in the form of per-voxel labels (mapped to the mesh, right). The core of our approach is a joint 3D-multi-view prediction network that leverages the synergies between geometric and color features. (Color figure online)

In 3D, the representation of a scene is typically obtained from RGB-D surface reconstruction methods [6,17,26,27] which often store scanned geometry in a 3D voxel grid where the surface is encoded by an implicit surface function such as a signed distance field [4]. One approach towards analyzing these reconstructions is to leverage a CNN with 3D convolutions, which has been used for shape classification [30,43], and recently also for predicting dense semantic 3D voxel maps [5,8,36]. In theory, one could simply add an additional color channel to the voxel grid in order to incorporate RGB information; however, the limited voxel resolution prevents encoding feature-rich image data (Fig. 1).

In this work, we specifically address this problem of how to incorporate RGB information for the 3D semantic segmentation task, and leverage the combined geometric and RGB signal in a joint, end-to-end approach. To this end, we propose a novel network architecture that takes as input the 3D scene representation as well as the input of nearby views in order to predict a dense semantic label set on the voxel grid. Instead of mapping color data directly on the voxel grid, the core idea is to first extract 2D feature maps from 2D images using the full-resolution RGB input. These features are then downsampled through convolutions in the 2D domain, and the resulting 2D feature map is subsequently backprojected into 3D space. In 3D, we leverage a 3D convolutional network architecture to learn from both the backprojected 2D features as well as 3D geometric features. This way, we can join the benefits of existing approaches and leverage all available information, significantly improving on existing approaches.

Our main contribution is the formulation of a joint, end-to-end convolutional neural network which learns to infer 3D semantics from both 3D geometry and 2D RGB input. In our evaluation, we provide a comprehensive analysis of the design choices of the joint 2D-3D architecture, and compare it with current state of the art methods. In the end, our approach increases 3D segmentation accuracy from 52.8% to 75% compared to the best existing volumetric architecture.

2   Related Work

Deep Learning in 3D. An important avenue for 3D scene understanding has been opened through recent advances in deep learning. Similar to the 2D domain, convolutional neural networks (CNNs) can operate in volumetric domains using an additional spatial dimension for the filter banks. 3D ShapeNets [2] was one of the first works in this context; they learn a 3D convolutional deep belief network from a shape database. Several works have followed, using 3D CNNs for object classification [23,30] or generative scene completion tasks [7,8,10]. In order to address the memory and compute requirements, hierarchical 3D CNNs have been proposed to more efficiently represent and process 3D volumes [10,12,32,33,38,42]. The spatial extent of a 3D CNN can also be increased with dilated convolutions [44], which have been used to predict missing voxels and infer semantic labels [36], or by using a fully-convolutional networks, in order to decouple the dimensions of training and test time [8]. Very recently, we have seen also network architectures that operate on an (unstructured) point-based representation [29,31].

Multi-view Deep Networks. An alternative way of learning a classifier on 3D input is to render the geometry, run a 2D feature extractor, and combine the extracted features using max pooling. The multi-view CNN approach by Su et al. [37] was one of the first to propose such an architecture for object classification. However, since the output is a classification score, this architecture does not spatially correlate the accumulated 2D features. Very recently, a multi-view network has been proposed for part-based mesh segmentation [18]. Here, 2D confidence maps of each part label are projected on top of ShapeNet [2] models, where a mesh-based CRF accumulates inputs of multiple images to predict the part labels on the mesh geometry. This approach handles only relatively small label sets (e.g., 2–6 part labels), and its input is 2D renderings of the 3D meshes; i.e., the multi-view input is meant as a replacement input for 3D geometry. Although these methods are not designed for 3D semantic segmentation, we consider them as the main inspiration for our multi-view component.

Multi-view networks have also been proposed in the context of stereo reconstruction. For instance, Choy et al. [3] use an RNN to accumulate features from different views and Tulsiani et al. [39] propose an unsupervised approach that takes multi-view input to learn a latent 3D space for 3D reconstruction. Multi-view networks have also been used in the context of stereo reconstruction [19,20], leveraging feature projection into 3D to produce consistent reconstruction. An alternative way to combine several input views with 3D, is by projecting colors directly into the voxels, maintaining one channel for each input view per voxel [16]. However, due to memory requirements, this becomes impractical for a large number of input views.

3D Semantic Segmentation. Semantic segmentation on 2D images is a popular task and has been heavily explored using cutting-edge neural network approaches [14,22]. The analog task can be formulated in 3D, where the goal is to predict

semantic labels on a per-voxel level [40,41]. Although this is a relatively recent task, it is extremely relevant to a large range of applications, in particular, robotics, where a spatial understanding of the inferred semantics is essential. For the 3D semantic segmentation task, several datasets and benchmarks have recently been developed. The ScanNet [5] dataset introduced a 3D semantic segmentation task on approx. 1.5k RGB-D scans and reconstructions obtained with a Structure Sensor. It provides ground truth annotations for training, validation, and testing directly on the 3D reconstructions; it also includes approx. 2.5 mio RGB-D frames whose 2D annotations are derived using rendered 3D-to-2D projections. Matterport3D [1] is another recent dataset of about 90 building-scale scenes in the same spirit as ScanNet; it includes fewer RGB-D frames (approx. 194,400) but has more complete reconstructions.

3   Overview

The goal of our method is to predict a 3D semantic segmentation based on the input of commodity RGB-D scans. More specifically, we want to infer semantic class labels on per-voxel level of the grid of a 3D reconstruction. To this end, we propose a joint 2D-3D neural network that leverages both RGB and geometric information obtained from a 3D scans. For the geometry, we consider a regular volumetric grid whose voxels encode a ternary state (known-occupied, known-free, unknown). To perform semantic segmentation on full 3D scenes of varying sizes, our network operates on a per-chunk basis; i.e., predicting columns of a scene in sliding-window fashion through the xy-plane at test time. For a given xy-location in a scene, the network takes as input the volumetric grid of the surrounding area (chunks of $31 \times 31 \times 62$ voxels). The network then extracts geometric features using a series of 3D convolutions, and predicts per-voxel class labels for the center column at the current xy-location. In addition to the geometry, we select nearby RGB views at the current xy-location that overlap with the associated chunk. For all of these 2D views, we run the respective images through a 2D neural network that extracts their corresponding features. Note that these 2D networks all have the same architecture and share the same weights.

In order to combine the 2D and 3D features, we introduce a differentiable backprojection layer that maps 2D features onto the 3D grid. These projected features are then merged with the 3D geometric information through a 3D convolutional part of the network. In addition to the projection, we add a voxel pooling layer that enables handling a variable number of RGB views associated with a 3D chunk; the pooling is performed on a per-voxel basis. In order to run 3D semantic segmentation for entire scans, this network is run for each xy-location of a scene, taking as input the corresponding local chunks.

In the following, we will first introduce the details of our network architecture (see Sect. 4) and then show how we train and implement our method (see Sect. 5).
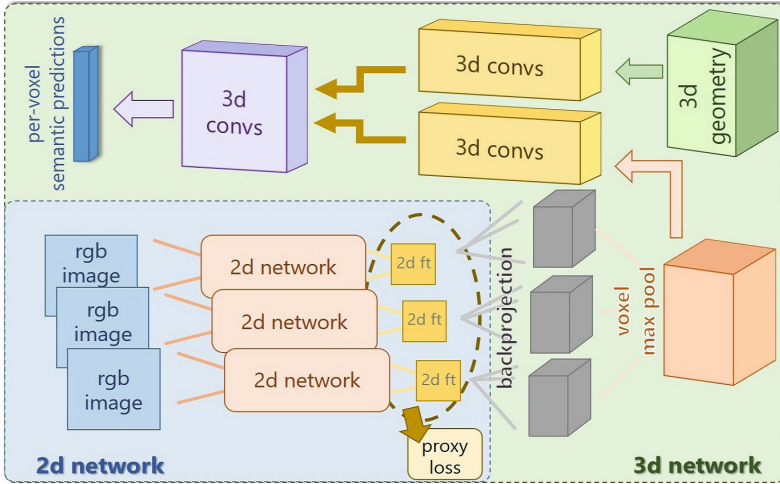
Fig. 2. Network overview: our architecture is composed of a 2D and a 3D part. The 2D side takes as input several aligned RGB images from which features are learned with a proxy loss. These are mapped to 3D space using a differentiable backprojection layer. Features from multiple views are max-pooled on a per-voxel basis and fed into a stream of 3D convolutions. At the same time, we input the 3D geometry into another 3D convolution stream. Then, both 3D streams are joined and the 3D per-voxel labels are predicted. The whole network is trained in an end-to-end fashion.

4   Network Architecture

Our network is composed of a 3D stream and several 2D streams that are combined in a joint 2D-3D network architecture. The 3D part takes as input a volumetric grid representing the geometry of a 3D scan, and the 2D streams take as input the associated RGB images. To this end, we assume that the 3D scan is composed of a sequence of RGB-D images obtained from a commodity RGB-D camera, such as a Kinect or a Structure Sensor; although note that our method generalizes to other sensor types. We further assume that the RGB-D images are aligned with respect to their world coordinate system using an RGB-D reconstruction framework; in the case of ScanNet [5] scenes, the BundleFusion [6] method is used. Finally, the RGB-D images are fused together in a volumetric grid, which is commonly done by using an implicit signed distance function [4]. An overview of the network architecture is provided in Fig. 2.

4.1   3D Network

Our 3D network part is composed of a series of 3D convolutions operating on a regular volumetric gird. The volumetric grid is a subvolume of the voxelized 3D representation of the scene. Each subvolume is centered around a specific xy-location at a size of $31 \times 31 \times 62$ voxels, with a voxel size of $4.8\,cm$. Hence, we

consider a spatial neighborhood of $1.5\,\mathrm{m} \times 1.5\,\mathrm{m}$ and $3\,\mathrm{m}$ in height. Note that we use a height of $3\,\mathrm{m}$ in order to cover the height of most indoor environments, such that we only need to train the network to operate in varying xy-space. The 3D network takes these subvolumes as input, and predicts the semantic labels for the center columns of the respective subvolume at a resolution of $1 \times 1 \times 62$ voxels; i.e., it simultaneously predicts labels for 62 voxels. For each voxel, we encode the corresponding value of the scene reconstruction state: known-occupied (i.e., on the surface), known-free space (i.e., based on empty space carving [4]), or unknown space (i.e., we have no knowledge about the voxel). We represent this through a 2-channel volumetric grid, the first a binary encoding of the occupancy, and the second a binary encoding of the known/unknown space. The 3D network then processes these subvolumes with a series of nine 3D convolutions which expand the feature dimension and reduce the spatial dimensions, along with dropout regularization during training, before a final set of fully connected layers which predict the classification scores for each voxel.

In the following, we show how to incorporate learned 2D features from associated 2D RGB views.

4.2 2D Network

The aim of the 2D part of the network is to extract features from each of the input RGB images. To this end, we use a 2D network architecture based on ENet [28] to learn those features. Note that although we can use a variable of number of 2D input views, all 2D networks share the same weights as they are jointly trained. Our choice to use ENet is due to its simplicity as it is both fast to run and memory-efficient to train. In particular, the low memory requirements are critical since it allows us to jointly train our 2D-3D network in an end-to-end fashion with multiple input images per train sample. Although our aim is 2D-3D end-to-end training, we additionally use a 2D proxy loss for each image that allows us to make the training more stable; i.e., each 2D stream is asked to predict meaningful semantic features for an RGB image segmentation task. Here, we use semantic labels of the 2D images as ground truth; in the case of ScanNet [5], these are derived from the original 3D annotations by rendering the annotated 3D mesh from the camera points of the respective RGB image poses. The final goal of the 2D network is to obtain the features in the last layer before the proxy loss per-pixel classification scores; these features maps are then backprojected into 3D to join with the 3D network, using a differentiable backprojection layer. In particular, from an input RGB image of size 328×256, we obtain a 2D feature map of size $(128\times)41 \times 32$, which is then backprojected into the space of the corresponding 3D volume, obtaining a 3D representation of the feature map of size $(128\times)31 \times 31 \times 62$.

4.3 Backprojection Layer

In order to connect the learned 2D features from each of the input RGB views with the 3D network, we use a differentiable backprojection layer. Since we

assume known 6-DoF pose alignments for the input RGB images with respect to each other and the 3D reconstruction, we can compute 2D-3D associations on-the-fly. The layer is essentially a loop over every voxel in 3D subvolume where a given image is associated to. For every voxel, we compute the 3D-to-2D projection based on the corresponding camera pose, the camera intrinsics, and the world-to-grid transformation matrix. We use the depth data from the RGB-D images in order to prune projected voxels beyond a threshold of the voxel size of 4.8 cm; i.e., we compute only associations for voxels close to the geometry of the depth map. We compute the correspondences from 3D voxels to 2D pixels since this allows us to obtain a unique voxel-to-pixel mapping. Although one could pre-compute these voxel-to-pixel associations, we simply compute this mapping on-the-fly in the layer as these computations are already highly memory bound on the GPU; in addition, it saves significant disk storage since this it would involve a large amount of index data for full scenes.

Once we have computed voxel-to-pixel correspondences, we can project the features of the last layer of the 2D network to the voxel grid:

$$n_{feat} \times w_{2d} \times h_{2d} \to n_{feat} \times w_{3d} \times h_{3d} \times d_{3d}$$

For the backward pass, we use the inverse mapping of the forward pass, which we store in a temporary index map. We use 2D feature maps (feature dim. of 128) of size $(128\times)41 \times 31$ and project them to a grid of size $(128\times)31 \times 31 \times 62$.

In order to handle multiple 2D input streams, we compute voxel-to-pixel associations with respect to each input view. As a result, some voxels will be associated with multiple pixels from different views. In order to combine projected features from multiple input views, we use a voxel max-pooling operation that computes the maximum response on a per feature channel basis. Since the max pooling operation is invariant to the number of inputs, it enables selecting for the features of interest from an arbitrary number of input images.

4.4 Joint 2D-3D Network

The joint 2D-3D network combines 2D RGB features and 3D geometric features using the mapping from the backprojection layer. These two inputs are processed with a series of 3D convolutions, and then concatenated together; the joined feature is then further processed with a set of 3D convolutions. We have experimented with several options as to where to join these two parts: at the beginning (i.e., directly concatenated together without independent 3D processing), approximately 1/3 or 2/3 through the 3D network, and at the end (i.e., directly before the classifier). We use the variant that provided the best results, fusing the 2D and 3D features together at 2/3 of the architectures (i.e., after the 6th 3D convolution of 9); see Table 5 for the corresponding ablation study. Note that the entire network, as shown in Fig. 2, is trained in an end-to-end fashion, which is feasible since all components are differentiable. Table 1 shows an overview of the distribution of learnable parameters of our 3DMV model.

Table 1. Distribution of learnable parameters of our 3DMV model. Note that the majority of the network weights are part of the combined 3D stream just before the per-voxel predictions where we rely on strong feature maps; see top left of Fig. 2.

	2D only	3D (2D input only)	3D (3D geo only)	3D (fused 2D/3D)
# trainable params	146,176	379,744	87,136	10,224,300

4.5   Evaluation in Sliding Window Mode

Our joint 2D-3D network operates on a per-chunk basis; i.e., it takes fixed sub-volumes of a 3D scene as input (along with associated RGB views), and predicts labels for the voxels in the center column of the given chunk. In order to perform a semantic segmentation of large 3D environments, we slide the subvolume through the 3D grid of the underlying reconstruction. Since the height of the subvolume (3 m) is sufficient for most indoor environments, we only need to slide over the xy-domain of the scene. Note, however, that for training, the training samples do not need to be spatially connected, which allows us to train on a random set of subvolumes. This de-coupling of training and test extents is particularly important since it allows us to provide a good label and data distribution of training samples (e.g., chunks with sufficient coverage and variety).

5   Training

5.1   Training Data

We train our joint 2D-3D network architecture in an end-to-end fashion. To this end, we prepare correlated 3D and RGB input to the network for the training process. The 3D geometry is encoded in a ternary occupancy grid that encodes known-occupied, known-free, and unknown states for each voxel. The ternary information is split upon 2 channels, where the first channel encodes occupancy and the second channel encodes the known vs. unknown state. To select train subvolumes from a 3D scene, we randomly sample subvolumes as potential training samples. For each potential train sample, we check its label distribution and discard samples containing only structural elements (i.e., wall/floor) with 95% probability. In addition, all samples with empty center columns are discarded as well as samples with less than 70% of the center column geometry annotated.

For each subvolume, we then associate k nearby RGB images whose alignment is known from the 6-DoF camera pose information. We select images greedily based on maximum coverage; i.e., we first pick the image covering the most voxels in the subvolume, and subsequently take each next image which covers the most number of voxels not covered by current set. We typically select 3–5 images since additional gains in coverage become smaller with each added image. For each sampled subvolume, we augment it with 8 random rotations for a total of 1,316,080 train samples. Since existing 3D datasets, such as ScanNet [5] or Matterport3D [1] contain unannotated regions in the ground truth (see Fig. 3, right), we mask out these regions in both our 3D loss and 2D proxy loss. Note that this strategy still allows for making predictions for all voxels at test time.

5.2   Implementation

We implement our approach in PyTorch. While 2D and 3D conv layers are already provided by the PyTorch API, we implement a custom layer for the backprojection layer. We implement this backprojection in python, as a custom PyTorch layer, representing the projection as series of matrix multiplications in order to exploit PyTorch parallelization, and run the backprojection on the GPU through the PyTorch API. For training, we have tried only training parts of the network; however, we found that the end-to-end version that jointly optimizes both 2D and 3D performed best. In the training processes, we use an SGD optimizer with a learning rate of 0.001 and a momentum of 0.9; we set the batch size to 8. Note that our training set is quite biased towards structural classes (e.g., wall, floor), even when discarding most structural-only samples, as these elements are vastly dominant in indoor scenes. In order to account for this data imbalance, we use the histogram of classes represented in the train set to weight the loss during training. We train our network for 200,000 iterations; for our network trained on 3 views, this takes $\approx$24 h, and for 5 views, $\approx$48 h.

6   Results

In this section, we provide an evaluation of our proposed method with a comparison to existing approaches. We evaluate on the ScanNet dataset [5], which contains 1513 RGB-D scans composed of 2.5M RGB-D images. We use the public train/val/test split of 1045, 156, 312 scenes, respectively, and follow the 20-class semantic segmentation task defined in the original ScanNet benchmark. We evaluate our results with per-voxel class accuracies, following the evaluations of previous work [5,8,31]. Additionally, we visualize our results qualitatively and in comparison to previous work in Fig. 3, with close-ups shown in Fig. 4. Note that we map the predictions from all methods back onto the mesh reconstruction for ease of visualization.

Comparison to State of the Art. Our main results are shown in Table 2, where we compare to several state-of-the-art volumetric (ScanNet [5], ScanComplete [8]) and point-based approaches (PointNet++[31]) on the ScanNet test set. Additionally, we show an ablation study regarding our design choices in Table 3.

The best variant of our 3DMV network achieves 75% average classification accuracy which is quite significant considering the difficulty of the task and the performance of existing approaches. That is, we improve 22.2% over existing volumetric and 14.8% over the state-of-the-art PointNet++ architecture.

How Much Does RGB Input Help? Table 3 includes a direct comparison between our 3D network architecture when using RGB features against the exact same 3D network without the RGB input. Performance improves from 54.4% to 70.1% with RGB input, even with just a single RGB view. In addition, we tried out the naive alternative of using per-voxel colors rather than a 2D feature extractor. Here, we see only a marginal difference compared to the purely geometric baseline

(54.4% vs. 55.9%). We attribute this relatively small gain to the limited grid resolution ($\approx$5 cm voxels), which is insufficient to capture rich RGB features. Overall, we can clearly see the benefits of RGB input, as well as the design choice to first extract features in the 2D domain.

How Much Does Geometric Input Help? Another important question is whether we actually need the 3D geometric input, or whether geometric information is a redundant subset of the RGB input; see Table 3. The first experiment we conduct in this context is simply a projection of the predicted 2D labels on top of the geometry. If we only use the labels from a single RGB view, we obtain 27% average accuracy (vs. 70.1% with 1 view + geometry); for 3 views, this label backprojection achieves 44.2% (vs. 73.0% with 3 views + geometry). Note that this is related to the limited coverage of the RGB backprojections (see Table 4).

However, the interesting experiment now is what happens if we still run a series of 3D convolutions after the backprojection of the 2D labels. Again, we omit inputting the scene geometry, but we now learn how to combine and propagate the backprojected features in the 3D grid; essentially, we ignore the first part of our 3D network; cf. Fig. 2. For 3 RGB views, this results in an accuracy of 58.2%; this is higher than the 54.4% of geometry only; however, it is much lower than our final 3-view result of 73.0% from the joint network. Overall, this shows that the combination of RGB and geometric information aptly complements each other, and that the synergies allow for an improvement over the individual inputs by 14.8% and 18.6%, respectively (for 3 views).

Table 2. Comparison of our final trained model (5 views, end-to-end) against other state-of-the-art methods on the ScanNet dataset [5]. We can see that our approach makes significant improvements, 22.2% over existing volumetric and approx. 14.8% over state-of-the-art PointNet++ architectures.

	wall	floor	cab	bed	chair	sofa	table	door	wind	bkshf	pic	cntr	desk	curt	fridg	show	toil	sink	bath	other	avg
ScanNet [5]	70.1	90.3	49.8	62.4	69.3	75.7	**68.4**	48.9	20.1	64.6	3.4	32.1	36.8	7.0	66.4	46.8	69.9	39.4	74.3	19.5	50.8
ScanComplete [8]	87.2	96.9	44.5	65.7	75.1	72.1	63.8	13.6	16.9	70.5	10.4	31.4	40.9	49.8	38.7	46.8	72.2	47.4	85.1	26.9	52.8
PointNet++ [31]	**89.5**	**97.8**	39.8	69.7	**86.0**	68.3	59.6	27.5	23.7	84.3	0.0	**37.6**	66.7	48.7	54.7	**85.0**	84.8	62.8	86.1	30.7	60.2
3DMV (ours)	73.9	95.6	**69.9**	**80.7**	85.9	**75.8**	67.8	**86.6**	**61.2**	**88.1**	**55.8**	31.9	**73.2**	**82.4**	**74.8**	82.6	**88.3**	**72.8**	**94.7**	**58.5**	**75.0**

How to Feed 2D Features into the 3D Network? An interesting question is where to join 2D and 3D features; i.e., at which layer of the 3D network do we fuse together the features originating from the RGB images with the features from the 3D geometry. On the one hand, one could argue that it makes more sense to feed the 2D part early into the 3D network in order to have more capacity for learning the joint 2D-3D combination. On the other hand, it might make more sense to keep the two streams separate for as long as possible to first extract strong independent features before combining them.

To this end, we conduct an experiment with different 2D-3D network combinations (for simplicity, always using a single RGB view without end-to-end training); see Table 5. We tried four combinations, where we fused the 2D and
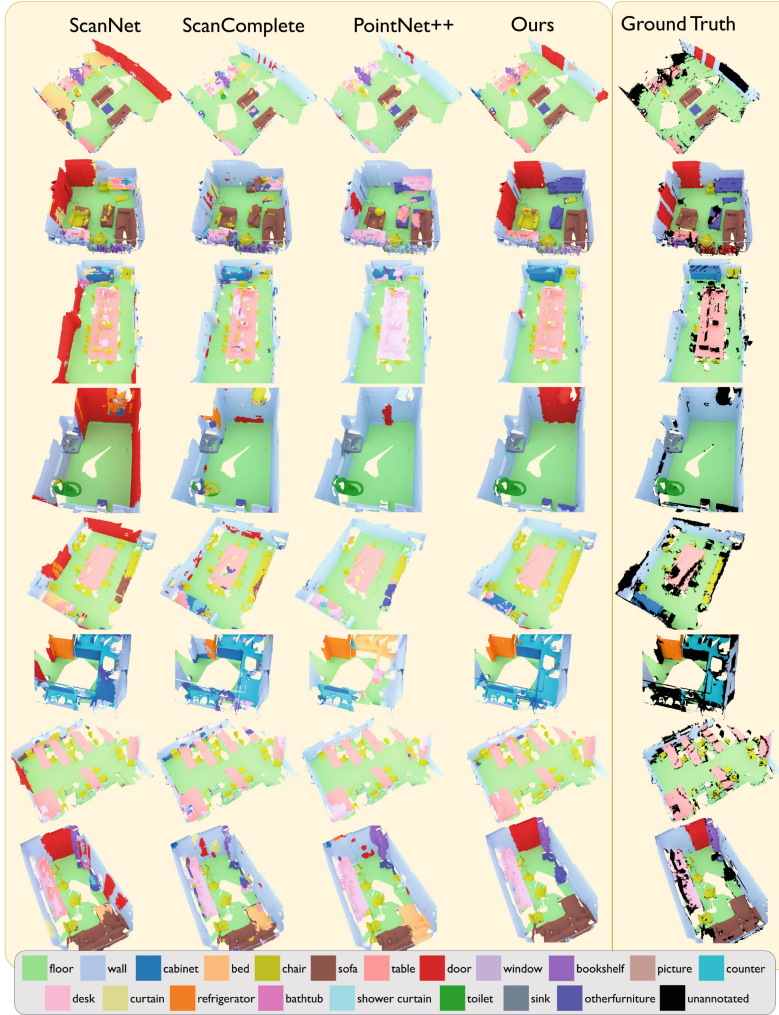
Fig. 3. Qualitative semantic segmentation results on the ScanNet [5] test set. We compare with the 3D-based approaches of ScanNet [5], ScanComplete [8], PointNet++ [31]. Note that the ground truth scenes contain some unannotated regions, denoted in black. Our joint 3D-multi-view approach achieves more accurate semantic predictions.

3D features at the beginning, after the first third of the network, after the second third, and at the very end into the 3D network. Interestingly, the results are relatively similar ranging from 67.6%, 65.4% to 69.1% and 67.5% suggesting that the 3D network can adapt quite well to the 2D features. Across these experiments, the second third option turned out to be a few percentage points higher than the alternatives; hence, we use that as a default in all other experiments.

How Much Do Additional Views Help? In Table 3, we also examine the effect of each additional view on classification performance. For geometry only, we obtain an average classification accuracy of 54.4%; adding only a single view per chunk increases to 70.1% (+15.7%); for 3 views, it increases to 73.1% (+3.0%); for 5 views, it reaches 75.0% (+1.9%). Hence, for every additional view the incremental gains become smaller; this is somewhat expected as a large part of the benefits are attributed to additional coverage of the 3D volume with 2D features. If we already use a substantial number of views, each additional added feature shares redundancy with previous views, as shown in Table 4.

Is End-to-End Training of the Joint 2D-3D Network Useful? Here, we examine the benefits of training the 2D-3D network in an end-to-end fashion, rather than simply using a pre-trained 2D network. We conduct this experiment with 1, 3, and 5 views. The end-to-end variant consistently outperforms the fixed version, improving the respective accuracies by 1.0%, 0.2%, and 0.5%. Although the end-to-end variants are strictly better, the increments are smaller than we initially hoped for. We also tried removing the 2D proxy loss that enforces good 2D predictions, which led to a slightly lower performance. Overall, end-to-end training with a proxy loss always performed best and we use it as our default.

Table 3. Ablation study for different design choices of our approach on ScanNet [5]. We first test simple baselines where we backproject 2D labels from 1 and 3 views (rows 1–2), then run set of 3D convs after the backprojections (row 3). We then test a 3D-geometry-only network (row 4). Augmenting the 3D-only version with per-voxel colors shows only small gains (row 5). In rows 6–11, we test our joint 2D-3D architecture with varying number of views, and the effect of end-to-end training. Our 5-view, end-to-end variant performs best.

	wall	floor	cab	bed	chair	sofa	table	door	wind	bkshf	pic	cntr	desk	curt	fridg	show	toil	sink	bath	other	avg
2D only (1 view)	37.1	39.1	26.7	33.1	22.7	38.8	17.5	38.7	13.5	32.6	14.9	7.8	19.1	34.4	33.2	13.3	32.7	29.2	36.3	20.4	27.1
2D only (3 views)	58.6	62.5	40.8	51.6	38.6	59.7	31.1	55.9	25.9	52.9	25.1	14.2	35.0	51.2	57.3	36.0	47.1	44.7	61.5	34.3	44.2
Ours (no geo input)	76.2	92.9	59.3	65.6	80.6	73.9	63.3	75.1	22.6	80.2	13.3	31.8	43.4	56.5	53.4	43.2	82.1	55.0	80.8	9.3	58.2
Ours (3D geo only)	60.4	95.0	54.4	69.5	79.5	70.6	71.3	65.9	20.7	71.4	4.2	20.0	38.5	15.2	59.9	57.3	78.7	48.8	87.0	20.6	54.4
Ours (3D geo+voxel color)	58.8	94.7	55.5	64.3	72.1	80.1	65.5	**70.7**	33.1	69.0	2.9	31.2	49.5	37.2	49.1	54.1	75.9	48.4	85.4	20.5	55.9
Ours (1 view, fixed 2D)	77.3	96.8	**70.0**	78.2	82.6	**85.0**	68.5	88.8	36.0	82.8	15.7	32.6	60.3	71.0	76.7	82.2	74.8	57.6	87.0	58.5	69.1
Ours (1 view)	70.7	96.8	61.4	76.4	84.4	80.3	70.4	83.9	57.9	85.3	41.7	35.0	64.5	75.6	81.3	58.2	85.0	60.5	81.6	51.7	70.1
Ours (3 view, fixed 2D)	**81.1**	96.4	58.0	77.3	84.7	85.2	**74.9**	87.3	51.2	86.3	33.5	**47.0**	52.4	79.5	79.0	72.3	80.8	**76.1**	92.5	**60.7**	72.8
Ours (3 view)	75.2	**97.1**	66.4	77.6	80.6	84.5	66.5	85.8	61.8	87.1	47.6	24.7	68.2	75.2	78.9	73.6	86.9	**76.1**	89.9	57.2	73.0
Ours (5 view, fixed 2D)	77.3	95.7	68.9	**81.7**	**89.6**	84.2	74.8	83.1	**62.0**	87.4	36.0	40.5	55.9	**83.1**	**81.6**	77.0	87.8	70.7	93.5	59.6	74.5
Ours (5 view)	73.9	95.6	69.9	80.7	85.9	75.8	67.8	86.6	61.2	**88.1**	**55.8**	31.9	**73.2**	82.4	74.8	**82.6**	**88.3**	72.8	**94.7**	58.5	**75.0**

Evaluation in 2D Domains Using NYUv2. Although we are predicting 3D per-voxel labels, we can also project the obtained voxel labels into the 2D images. In Table 6, we show such an evaluation on the NYUv2 [35] dataset. For this task, we train our network on both ScanNet data as well as the NYUv2 train annotations projected into 3D. Although this is not the actual task of our method, it can be seen as an efficient way to accumulate semantic information from multiple RGB-D frames by using the 3D geometry as a proxy for the learning framework. Overall, our joint 2D-3D architecture compares favorably against the respective baselines on this 13-class task.

Table 4. Amount of coverage from varying number of views over the annotated ground truth voxels of the ScanNet [5] test scenes.

	1 view	3 views	5 views
Coverage	40.3%	64.4%	72.3%

Table 5. Evaluation of various network combinations for joining the 2D and 3D streams in the 3D architecture (cf. Fig. 2, top). We use the single view variant with a fixed 2D network here for simplicity. Interestingly, performance only changes slightly; however, the 2/3 version performed the best, which is our default for all other experiments.

|	wall	floor	cab	bed	chair	sofa	table	door	wind	bkshf	pic	cntr	desk	curt	fridg	show	toil	sink	bath	other	avg
begin	78.8	96.3	63.7	72.8	83.3	81.9	**74.5**	81.6	39.5	89.6	**24.8**	33.9	52.6	**74.8**	76.0	47.5	80.1	**65.4**	85.9	49.4	67.6
1/3	79.3	95.5	65.1	75.2	80.3	81.5	73.8	86.0	30.5	**91.7**	11.3	**35.5**	46.4	66.6	67.9	44.1	**81.7**	55.5	85.9	53.3	65.4
2/3	77.3	**96.8**	**70.0**	**78.2**	82.6	**85.0**	68.5	**88.8**	36.0	82.8	15.7	32.6	**60.3**	71.0	**76.7**	**82.2**	74.8	57.6	87.0	**58.5**	**69.1**
end	**82.7**	96.3	67.1	77.8	**83.2**	80.1	66.0	80.3	**41.0**	83.9	24.3	32.4	57.7	70.1	71.5	58.5	79.6	65.1	**87.2**	45.8	67.5
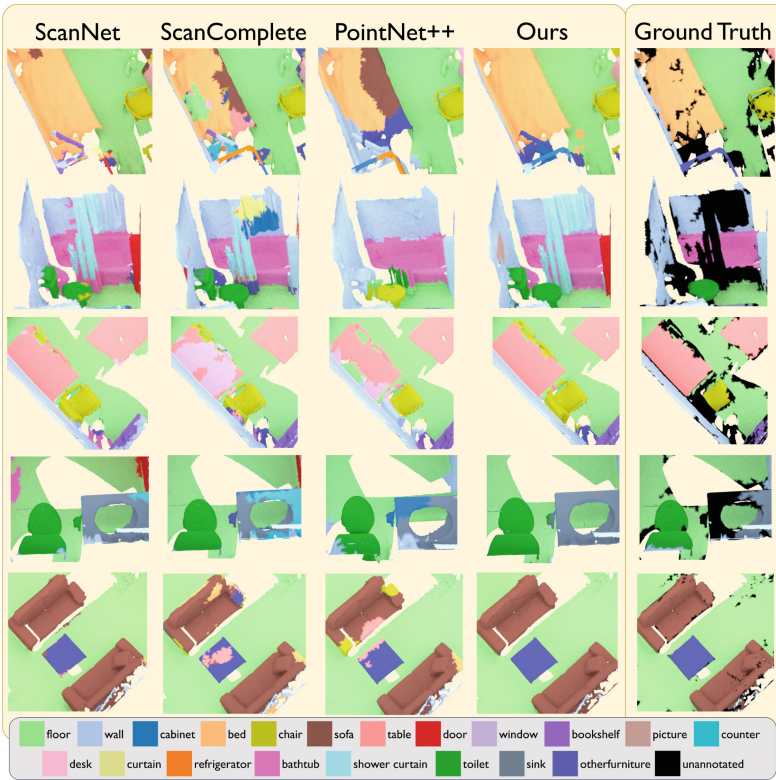
Fig. 4. Additional qualitative semantic segmentation results (close ups) on the Scan-Net [5] test set. Note the consistency of our predictions compared to the other baselines.

Table 6. We can also evaluate our method on 2D semantic segmentation tasks by projecting the predicted 3D labels into the respective RGB-D frames. Here, we show a comparison on dense pixel classification accuracy on NYU2 [25]. Note that the reported ScanNet classification is on the 11-class task.

	bed	books	ceil.	chair	floor	furn.	obj.	pic.	sofa	table	tv	wall	wind.	avg.
SceneNet [11]	70.8	5.5	76.2	59.6	95.9	62.3	50.0	18.0	61.3	42.2	22.2	86.1	32.1	52.5
Hermans et al. [15]	68.4	45.4	**83.4**	41.9	91.5	37.1	8.6	35.8	58.5	27.7	38.4	71.8	48.0	54.3
ENet [28]	79.2	35.5	31.6	60.2	82.3	61.8	50.9	43.0	61.2	42.7	30.1	84.1	67.4	56.2
SemanticFusion [24] (RGBD+CRF)	62.0	**58.4**	43.3	59.5	92.7	64.4	**58.3**	65.8	48.7	34.3	34.3	86.3	62.3	59.2
SemanticFusion [24,9] (Eigen+CRF)	48.3	51.5	79.0	74.7	90.8	63.5	46.9	63.6	46.5	45.9	**71.5**	**89.4**	55.6	63.6
ScanNet [5]	81.4	-	46.2	67.6	**99.0**	65.6	34.6	-	67.2	50.9	35.8	55.8	63.1	60.7
3DMV (ours)	**84.3**	44.0	43.4	**77.4**	92.5	**76.8**	54.6	**70.5**	**86.3**	**58.6**	67.3	84.5	**85.3**	**71.2**

Summary Evaluation.

– RGB and geometric features are orthogonal and help each other.
– More views help, but increments get smaller with every view.
– End-to-end training is strictly better, but the improvement is not that big.
– Variations of where to join the 2D and 3D features change performance to some degree; 2/3 performed best in our tests.
– Our results are significantly better than the best volumetric or PointNet baseline (+22.2% and +14.8%, respectively).

Limitations. While our joint 3D-multi-view approach achieves significant performance gains over previous state of the art in 3D semantic segmentation, there are still several important limitations. Our approach operates on dense volumetric grids, which become quickly impractical for high resolutions; e.g., RGB-D scanning approaches typically produce reconstructions with sub-centimeter voxel resolution; sparse approaches, such as OctNet [33], might be a good remedy. Additionally, we currently predict only the voxels of each column of a scene jointly, while each column is predicted independently, which can give rise to some label inconsistencies in the final predictions since different RGB views might be selected; note, however, that due to the convolutional nature of the 3D networks, the geometry remains spatially coherent.

7   Conclusion and Future Work

We presented 3DMV, a joint 3D-multi-view approach built on the core idea of combining geometric and RGB features in a joint network architecture. We show that our joint approach can achieve significantly better accuracy for semantic 3D scene segmentation. In a series of evaluations, we carefully examine our design choices; for instance, we demonstrate that the 2D and 3D features complement each other rather than being redundant; we also show that our method can successfully take advantage of using several input views from an RGB-D sequence to gain higher coverage, thus resulting in better performance. In the end, we are able to show results at more than **14% higher classification accuracy**

than the best existing 3D segmentation approach. Overall, we believe that these improvements will open up new possibilities where not only the semantic content, but also the spatial 3D layout plays an important role.

For the future, we still see many open questions in this area. First, the 3D semantic segmentation problem is far from solved, and semantic instance segmentation in 3D is still at its infancy. Second, there are many fundamental questions about the scene representation for realizing 3D convolutional neural networks, and how to handle mixed sparse-dense data representations. And third, we also see tremendous potential for combining multi-modal features for generative tasks in 3D reconstruction, such as scan completion and texturing.

References

1. Chang, A., et al.: Matterport3D: learning from RGB-D data in indoor environments. In: International Conference on 3D Vision (3DV) (2017)
2. Chang, A.X., et al.: ShapeNet: an information-rich 3D model repository. Technical report, Stanford University – Princeton University – Toyota Technological Institute at Chicago. arXiv:1512.03012 [cs.GR] (2015)
3. Choy, C.B., Xu, D., Gwak, J.Y., Chen, K., Savarese, S.: 3D-R2N2: a unified approach for single and multi-view 3D object reconstruction. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9912, pp. 628–644. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46484-8_38
4. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 303–312. ACM (1996)
5. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: richly-annotated 3D reconstructions of indoor scenes. In: Proceedings of Computer Vision and Pattern Recognition (CVPR). IEEE (2017)
6. Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., Theobalt, C.: BundleFusion: real-time globally consistent 3D reconstruction using on-the-fly surface reintegration. ACM Trans. Gr. (TOG) **36**(3), 24 (2017)
7. Dai, A., Qi, C.R., Nießner, M.: Shape completion using 3D-encoder-predictor CNNs and shape synthesis. In: Proceedings of Computer Vision and Pattern Recognition (CVPR). IEEE (2017)
8. Dai, A., Ritchie, D., Bokeloh, M., Reed, S., Sturm, J., Nießner, M.: ScanComplete: large-scale scene completion and semantic segmentation for 3D scans. arXiv preprint arXiv:1712.10215 (2018)
9. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2650–2658 (2015)
10. Han, X., Li, Z., Huang, H., Kalogerakis, E., Yu, Y.: High resolution shape completion using deep neural networks for global structure and local geometry inference. In: IEEE International Conference on Computer Vision (ICCV) (2017)
11. Handa, A., Patraucean, V., Badrinarayanan, V., Stent, S., Cipolla, R.: SceneNet: understanding real world indoor scenes with synthetic data. arXiv preprint arXiv:1511.07041 (2015)
12. Häne, C., Tulsiani, S., Malik, J.: Hierarchical surface prediction for 3D object reconstruction. arXiv preprint arXiv:1704.00710 (2017)

13. Hane, C., Zach, C., Cohen, A., Angst, R., Pollefeys, M.: Joint 3D scene reconstruction and class segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 97–104 (2013)
14. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988. IEEE (2017)
15. Hermans, A., Floros, G., Leibe, B.: Dense 3D semantic mapping of indoor scenes from RGB-D images. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 2631–2638. IEEE (2014)
16. Ji, M., Gall, J., Zheng, H., Liu, Y., Fang, L.: SurfaceNet: an end-to-end 3D neural network for multiview stereopsis. arXiv preprint arXiv:1708.01749 (2017)
17. Kähler, O., Prisacariu, V.A., Ren, C.Y., Sun, X., Torr, P., Murray, D.: Very high frame rate volumetric integration of depth images on mobile devices. IEEE Trans. Vis. Comput. Gr. **21**(11), 1241–1250 (2015)
18. Kalogerakis, E., Averkiou, M., Maji, S., Chaudhuri, S.: 3D shape segmentation with projective convolutional networks. In: Proceedings of CVPR. IEEE 2 (2017)
19. Kar, A., Häne, C., Malik, J.: Learning a multi-view stereo machine. In: Advances in Neural Information Processing Systems, pp. 364–375 (2017)
20. Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., Bry, A.: End-to-end learning of geometry and context for deep stereo regression. CoRR, abs/1703.04309 (2017)
21. Kundu, A., Li, Y., Dellaert, F., Li, F., Rehg, J.M.: Joint semantic segmentation and 3D reconstruction from monocular video. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8694, pp. 703–718. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10599-4_45
22. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431–3440 (2015)
23. Maturana, D., Scherer, S.: VoxNet: A 3D convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 922–928. IEEE (2015)
24. McCormac, J., Handa, A., Davison, A., Leutenegger, S.: SemanticFusion: dense 3D semantic mapping with convolutional neural networks. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 4628–4635. IEEE (2017)
25. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7576, pp. 746–760. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33715-4_54
26. Newcombe, R.A., et al.: KinectFusion: real-time dense surface mapping and tracking. In: 2011 10th IEEE international symposium on Mixed and Augmented Reality (ISMAR), pp. 127–136. IEEE (2011)
27. Nießner, M., Zollhöfer, M., Izadi, S., Stamminger, M.: Real-time 3D reconstruction at scale using voxel hashing. ACM Trans. Gr. (TOG) **32**, 169 (2013)
28. Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: ENet: a deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147 (2016)
29. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: deep learning on point sets for 3D classification and segmentation. In: Proceedings of Computer Vision and Pattern Recognition (CVPR), vol. 1, no. 2, p. 4. IEEE (2017)
30. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.: Volumetric and multi-view CNNs for object classification on 3D data. In: Proceedings of Computer Vision and Pattern Recognition (CVPR). IEEE (2016)

31. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems, pp. 5105–5114 (2017)
32. Riegler, G., Ulusoy, A.O., Bischof, H., Geiger, A.: OctNetFusion: learning depth fusion from data. arXiv preprint arXiv:1704.01047 (2017)
33. Riegler, G., Ulusoy, A.O., Geiger, A.: OctNet: learning deep 3D representations at high resolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
34. Savinov, N., Ladicky, L., Hane, C., Pollefeys, M.: Discrete optimization of ray potentials for semantic 3D reconstruction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5511–5518 (2015)
35. Silberman, N., Fergus, R.: Indoor scene segmentation using a structured light sensor. In: Proceedings of the International Conference on Computer Vision - Workshop on 3D Representation and Recognition (2011)
36. Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T.: Semantic scene completion from a single depth image. In: Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition (2017)
37. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3D shape recognition. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 945–953 (2015)
38. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Octree generating networks: efficient convolutional architectures for high-resolution 3D outputs. arXiv preprint arXiv:1703.09438 (2017)
39. Tulsiani, S., Zhou, T., Efros, A.A., Malik, J.: Multi-view supervision for single-view reconstruction via differentiable ray consistency. In: CVPR. vol. 1, p. 3 (2017)
40. Valentin, J., et al.: SemanticPaint: interactive 3D labeling and learning at your fingertips. ACM Trans. Gr. (TOG) **34**(5), 154 (2015)
41. Vineet, V., et al.: Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 75–82. IEEE (2015)
42. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-CNN: octree-based convolutional neural networks for 3D shape analysis. ACM Trans. Gr. (TOG) **36**(4), 72 (2017)
43. Wu, Z., et al.: 3D shapeNets: a deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1912–1920 (2015)
44. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)