

Embedding 3D Geometric Features for Rigid Object Part Segmentation

Yafei Song¹, Xiaowu Chen¹, Jia Li^{1,2}*, Qinping Zhao¹

¹State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University ²International Research Institute for Multidisciplinary Science, Beihang University

Abstract

Object part segmentation is a challenging and fundamental problem in computer vision. Its difficulties may be caused by the varying viewpoints, poses, and topological structures, which can be attributed to an essential reason, i.e., a specific object is a 3D model rather than a 2D figure. Therefore, we conjecture that not only 2D appearance features but also 3D geometric features could be helpful. With this in mind, we propose a 2-stream FCN. One stream, named AppNet, is to extract 2D appearance features from the input image. The other stream, named GeoNet, is to extract 3D geometric features. However, the problem is that the input is just an image. To this end, we design a 2Dconvolution based CNN structure to extract 3D geometric features from 3D volume, which is named VolNet. Then a teacher-student strategy is adopted and VolNet teaches GeoNet how to extract 3D geometric features from an image. To perform this teaching process, we synthesize training data using 3D models. Each training sample consists of an image and its corresponding volume. A perspective voxelization algorithm is further proposed to align them. Experimental results verify our conjecture and the effectiveness of both the proposed 2-stream CNN and VolNet.

1. Introduction

Object part segmentation(OPSeg), which aims to label the right semantic part for each pixel of the objects in an image, is an important problem in computer vision. OPSeg can be regarded as a special case of semantic segmentation that focuses on part information. Specially, part information is useful for some fine-grained tasks, *e.g.*, image classification [29, 15], fine-grained action detection [23]. It is also necessary for some specific tasks of robots, *e.g.*, when a robot wants to open the hood to fix a car, the hood should be correctly segmented.

Object images are usually captured under different lightings, viewpoints and poses. Moreover, a class of ob-



Figure 1. Our basic idea is to exploit 3D geometric features during object part segmentation. While 2D appearance features have been widely used, 3D geometric features are rarely employed as the corresponding 3D shape is usually unavailable.

jects usually have various materials, textures, and topological structures. These diversities lead to the difficulties in OPSeg. Taken together, the difficulties can be attributed to an essential reason, *i.e.*, the objects are 3D models rather than 2D figures in real world. Therefore, besides 2D appearance features, 3D geometric features should also be useful. However, 3D information is usually unavailable. To this end, as shown in Fig. 1, our basic idea is to exploit 3D geometric features during OPSeg only with a single image.

In the existing literature, there are many semantic segmentation methods, but a few methods for OPSeg. Before the popularity of deep learning, previous methods [26, 19] usually 1) extract hand-crafted features for every pixel, 2) get the initial semantic probability distribution via classifier or other models, 3) construct MRF/CRF to optimize the final segmentation result. Entering deep learning era, step 1) and 2) are combined together. A milestone work based on deep learning may be FCN [20], which firstly performs semantic segmentation end-to-end. Chen et al. [3] further use atrous convolution to avoid the up-sampling layers in FCN and achieve better results. Moreover, DeepLab [3] also obtains good results on person-part segmentation. And several recent works of OPSeg [33, 31] are based on DeepLab. Besides FCN based methods, LSTM is also applied to exploit the context information for OPSeg [17, 18].

^{*}Corresponding Authors: Xiaowu Chen and Jia Li. E-Mail: {chen, jiali}@buaa.edu.cn

Most methods, *e.g.*, [20], [3], and [33], adapt and finetune models which have been pre-trained on large-scale dataset, *e.g.* ImageNet [5]. Since a pre-trained model can be regarded as a feature extractor, all these methods may use 2D image appearance features. On the other hand, as shown in [25, 10], depth or 3D geometric features are also essential for OPSeg. However, 3D information is usually unavailable, which makes it difficult to exploit 3D geometric features. Inspired by recent large-scale 3D model datasets [32, 2] and several works [27, 9] of recovering the 3D information from a single image, we believe that 3D information can be extracted from a 2D image to facilitate OPSeg.

With the basic idea in mind, we propose a 2-stream CNN to segment an object image based on FCN framework. One stream, named AppNet, is initialized by pre-trained ResNet-101 [11] model, which can extract 2D image appearance features. The other stream, named GeoNet, is to exploit 3D information. The GeoNet is trained via adopting a *teacher*student strategy [1]. Intuitively, it is easier to extract 3D geometric features from an image than directly recover its 3D structure. Therefore, we train a 2D-convolution based CNN model to perform OPSeg on volume, which is named as VolNet. The VolNet serves as a *teacher* model. GeoNet is an approximation of VolNet and extracts 3D geometric features from an image, which can be regarded as a student model. Though the input of VolNet is 3D volume, it only consists of 2D convolution layers. Theoretically, 3D convolution is a special case of 2D convolution. Experiments also show its effectiveness on 3D volume. To pre-train the student model GeoNet, we synthesise training data using ShapeNet [2]. Moreover, to align a rendered image and its corresponding volume, we further propose a perspective voxelization algorithm along with DCT pre-processing.

Our contributions mainly include three aspects: 1) We design a 2-stream FCN for object part segmentation. One stream, named AppNet, can extract 2D appearance features from an image. The other stream, named GeoNet, can extract 3D geometric features from an image. 2) A 2D convolution based CNN model, named VolNet, is proposed to extract 3D geometric features from 3D volume effectively, which also serves as a teacher model to teach GeoNet how to extract 3D geometric features. 3) To generate training data for VolNet and GeoNet, we synthesise an RGB image and its corresponding volume from a 3D model. And we further propose a perspective voxelization algorithm along with DCT pre-processing to align the image and volume.

2. Related Work

This section reviews some important works on semantic segmentation as well as object part segmentation. As the basic idea of this paper is mainly to exploit 3D information, 3D geometric features learning methods and applications also are reviewed briefly.

Semantic segmentation. Before the popularity of deep learning, previous methods [26, 19] usually 1) extract handcrafted features for every pixel, 2) get the initial semantic probability distribution via classifier or other models, 3) construct MRF/CRF to optimize the final segmentation result. Specifically, Shotton et al. [26] propose texton features along with bag-of-visual-word. These features are fed to a random-forest model, which can output the initial probability distribution of each semantic label. Besides the learning based methods, Liu et al. [19] propose the idea of label transfer, which extracts dense SIFT features for each pixel and directly transfers label information from labelled training image to an input image via feature matching. Entering deep learning era, step 1) and 2) are combined together. A milestone work based on deep learning may be FCN [20], which can perform semantic segmentation endto-end. Chen et al. [3] further use atrous convolution to replace the up-sampling in FCN and achieve better results. Another group of important works is R-CNN based [8, 10], which first extract plenty of segment proposals and then classify each proposal. These methods mainly exploit 2D appearance features, but also imply that 3D geometric features are useful, e.g. the depth [10].

Object part segmentation. Compared with semantic segmentation, there are fewer works on OPSeg. Some earlier methods usually has drawbacks, e.g., the method of Eslami and Williams [6] seems to be sensitive to viewpoint, and the method of Lu et al. [21] needs user to point landmarks. Besides semantic segmentation, DeepLab [3] also obtains good results on person-part segmentation. Several recent works of OPSeg [33, 31] enhance DeepLab from different aspects. Xia et al. [33] pay attention to the scale. And Wang et al. [31] combine object-level and part-level potentials. Recently, LSTM is also successfully applied to encode image context and perform OPSeg [17, 18]. There is also a series of works focusing on human/clothes parsing, e.g. [16, 36, 25]. Some methods also exploit 2D geometric features or 3D geometric information, e.g., 2D shape [30], the depth [25]. However, our work focus on exploiting 3D geometric features without inputting 3D information.

3D geometric features learning and application. Along with the construction of large-scale 3D shape dataset, *e.g.*, ModelNet [32] and ShapeNet [2], there are more and more works on 3D geometric features learning and applications. Some methods [27, 9] recover the 3D information from a single image, which inspired this work. And several other methods [32, 22, 34, 24] have been proposed for model classification. These methods mainly are based on 3D CNN with 3D volume as input, while our VolNet adopts 2D CNN. When using the same number of feature maps, 2D CNN takes less computation and storage than 3D CNN. The method of Qi *et al.* [24] also has similar idea, but our VolNet is more effective.



Figure 2. Overview of our method. We design a 2-stream CNN under FCN framework. The stream AppNet exploits appearance features and is initialized by ResNet-101 model. The stream GeoNet aims to exploit 3D geometric features, which is pre-trained on synthetic data.

3. The Approach

3.1. Overview

In order to simultaneously exploit the appearance and 3D geometric features, we design a 2-stream CNN under FCN framework. As illustrated in Fig. 2, one stream, named AppNet, is initialized by ResNet-101 [11] model. As ResNet-101 is trained on ImageNet to perform image classification task, it can be regarded as an extractor of 2D image appearance features. Moreover, to balance the storage and precision, each spatial dimension of the output is an eighth of the input. Following the DeepLab system [3], the pool5 layer is removed, the last two down-sampling strides are set as 1, and the convolution kernels of all res4 and res5 layers are transformed to atrous kernels to fit the input and output size. More details can be found in [3] and [11].

The other stream, named GeoNet, aims to exploit 3D geometric features. However, as the 3D information is usually unavailable, we adopt a teacher-student strategy. The teacher model extracts 3D geometric features from volume, while the student model learns to extract these features from an image. Following this strategy, we design a 2D convolution based model as the teacher model, named VolNet. Compared with 3D convolution based model, VolNet takes less computation and storage. The details of VolNet are presented in Sec. 3.3.

In order to teach GeoNet extracting 3D geometric features from an image, the outputs of GeoNet are regressed to the outputs of VolNet, and the details are in Sec. 3.4. As there is few data consisting of images and the corresponding 3D information, we synthesise training data using 3D models. For each model, an RGB image is rendered under random settings, including viewpoint, translation, and lighting. The corresponding volume is then generated according to the same viewpoint and translation. To align the image and volume, we further propose a perspective voxelization algorithm in Sec. 3.2.

3.2. Perspective Voxelization Algorithm

To generate training data for GeoNet and VolNet, the RGB images are generated via perspective rendering algorithm. As a result, if the volume is generated via traditional orthogonal voxelization algorithm, its correspondence with the rendered image cannot be accurately established. To align the image \mathcal{I} and volume \mathcal{V} , we propose a perspective voxelization algorithm (PVA). Before applying this algorithm, the 3D model is firstly rotated and translated to the same setting as rendering the RGB image. The basic rules of PVA are: 1) a voxel is set as 0 unless it is in the 3D model or interacts with the surface of the 3D model. 2) a volume vector \mathcal{V}_i is associated with an image pixel \mathcal{I}_i , and all voxels in \mathcal{V}_i are in the line determined by the original point \mathcal{O} and the pixel \mathcal{I}_i . 3) If two voxels have the same distance from the original point \mathcal{O} , they should be in the same volume plane.

Following these three rules, for an input 3D model, PVA outputs the volume \mathcal{V} . A 3D model consists of a vertex set **V** and a triangular face set **F**. Each triangular face has three vertexes $\langle u_0, u_1, u_2 \rangle$. The camera settings include the focal length f, the height resolution H and the width resolution W of the rendered image. And the depth resolution n_d also should be set up. The details are illustrated in Alg. 1.

PVA first initializes each voxel in the volume \mathcal{V} as 0. The size of the volume \mathcal{V} is $\langle \mathrm{H}, \mathrm{W}, n_d \rangle$. Then for each vertex $u \in \mathbf{V}$, PVA calculates its depth d_u from the original point \mathcal{O} and its coordinates $\langle x_u, y_u \rangle$ at image plane. By the way,

Algorithm 1 Perspective voxelization algorithm. Input: $\mathbf{V}, \mathbf{F}, f, \mathbb{H}, \mathbb{W}, n_d$ Output: V1: Initialize: $\mathcal{V} \leftarrow 0, d_{min} \leftarrow \infty, d_{max} \leftarrow 0$ 2: for all $u \in \mathbf{V}$ do $\begin{array}{l} d_u \leftarrow \sqrt{x_u^2 + y_u^2 + z_u^2} \\ x_u \leftarrow -f \times \frac{x_u}{z_u}, y_u \leftarrow -f \times \frac{y_u}{z_u} \\ d_{min} \leftarrow \min(d_{min}, d_u) \end{array}$ 3: 4: 5: $d_{max} \leftarrow \max(d_{max}, d_u)$ 6: 7: end for 8: while $\mathbf{F} \neq \emptyset$ do $\langle u_0, u_1, u_2 \rangle \leftarrow \mathsf{pop}(\mathbf{F})$ 9: for $j \leftarrow 0$ to 2 do 10:
$$\begin{split} & i \leftarrow \texttt{round} \langle -y_{u_j} + \frac{\texttt{H}}{2}, x_{u_j} + \frac{\texttt{W}}{2} \rangle \\ & k \leftarrow \texttt{round} (\frac{d_{u_j} - d_{min}}{d_{max} - d_{min}} \times n_d) \end{split}$$
11: 12: $\mathcal{V}_i^k \leftarrow 1$ 13: end for 14: for $j \leftarrow 0$ to 2 do 15: $k \leftarrow \operatorname{mod}(j+1,3), l \leftarrow \operatorname{mod}(j+2,3)$ 16:
$$\begin{split} & \text{if } (abs(x_{u_j} - x_{u_k}) > 1 \text{ or } abs(y_{u_j} - y_{u_k}) > 1 \\ & \text{ or } abs(x_{u_j} - x_{u_k}) > 1 \text{ or } abs(y_{u_j} - y_{u_k}) > 1 \\ & \text{ or } abs(n_d \times \frac{d_{u_j} - d_{u_k}}{d_{max} - d_{min}}) > 1) \text{ then } \\ & x_v \leftarrow \frac{x_{u_j} + x_{u_k}}{2}, y_v \leftarrow \frac{y_{u_j} + y_{u_k}}{2} \\ & z_v \leftarrow \frac{z_{u_j} + z_{u_k}}{2}, d_v \leftarrow \frac{d_{u_j} + d_{u_k}}{2} \\ & \text{ push}(\mathbf{V}, v) \end{split}$$
17: 18: 19: 2021: $\mathtt{push}(\mathbf{F}, \langle u_j, v, u_l \rangle)$ 22: $push(\mathbf{F}, \langle u_k, u_l, v \rangle)$ 23: break 24: end if 25: end for 26: 27: end while 28: $fill(\mathcal{V})$

the minimum and maximum depth d_{min}, d_{max} are recorded respectively. After that, a triangular face $\langle u_0, u_1, u_2 \rangle$ is popped out from the face set **F**. For each vertex u_i of the face, a voxel is set as 1, if it interacts with this vertex u_i . However, if a triangular face is large and can occupy nonadjacent voxels, there would be holes in the volume. To this end, such a triangular face will be divided into two small triangular faces via adding a vertex, *i.e.* the mid-point of the line, into the line which occupies two nonadjacent voxels. These two new triangular faces are subsequently added into the face set \mathbf{F} . The loop will stop until the face set \mathbf{F} is empty. At last, PVA fills the holes which are inside the volume to obtain a solid volume. Moreover, the function pop() is to pop an element from a set and delete it from the set, the function push() is to add an element into a set, the function round() is to get the nearest integer, the function abs() is to get the absolute value, the function mod() is to get the remainder after division.

3.3. 2D Convolution based VolNet to Extract 3D Geometric Features

To process a 3D shape using deep learning, previous methods, e.g., [32], [22], and [34] usually transform the 3D shape to volume and apply 3D convolution neural network. However, 3D convolution is actually a special case of 2D convolution. To concisely illustrate this, we assume that the input is a 3D volume \mathcal{V} and its dimension along with the depth direction is n_d . The output feature is denoted as \mathcal{F} , which has the same size with \mathcal{V} . The convolution kernel is denoted as \mathcal{K} . Each spatial dimension of \mathcal{K} is set as 3 without loss of generality. For 2D convolution layer, the size of \mathcal{K} is $3 \times 3 \times n_d$, and we have

$$\mathcal{F}_{i}^{j} = \sum_{k=1}^{n_{d}} \sum_{l \in \Omega(i)} \mathcal{K}_{l-i}^{k} \times \mathcal{V}_{l}^{k}, \qquad (1)$$

where the superscript indicates the index along the depth direction and the subscript indicates the spatial location, and $\Omega(i)$ is the adjacent domain of the location *i*. For 3D convolution, the size of \mathcal{K} is $3 \times 3 \times 3$, and we have

$$\mathcal{F}_{i}^{j} = \sum_{k=j-1}^{j+1} \sum_{l \in \Omega(i)} \mathcal{K}_{l-i}^{k-j} \times \mathcal{V}_{l}^{k}.$$
 (2)

From (1) and (2), the difference between 2D-CNN and 3D-CNN is that, a 3D convolution layer limits its receptive field and shares its weights in the third dimension, which is the depth direction in our case. While a 2D convolution layer does these only in the two dimensions of the spatial plane.

Receptive field and weights sharing can reduce the parameters and make the training easier. At the same time, we need to increase the amount of output features. For 2D-CNN, this is not a serious problem. However, for 3D-CNN, a lot of storage is needed. To balance these issues, we advise to process volume by applying 2D-CNN, which is named as VolNet. Moreover, to further reduce the storage of the data, we compress the volume by exploiting discrete cosine transformation (DCT). Specially, DCT is applied on each volume vector V_i as

$$\mathcal{F}_i = \mathsf{DCT}(\mathcal{V}_i). \tag{3}$$

This pre-processing also can be regarded as a convolution layer with constant parameters. As shown in Fig. 3, there is a sample output of DCT. Each DCT component can encode the global information across the depth from frequency aspect. For efficiency, we only reserve the lowest 16 frequency components.

Exploiting recently progresses in deep learning, VolNet is constructed based on residual network [12] and atrous convolution [3]. Note that, high resolution volume is computationally expensive but only provide marginally information empirically. Therefore, each dimension of a volume



Figure 3. A sample of DCT components. From left-top to rightdown, low frequency components to high frequency components.

plane is a quarter of the image. As illustrated in Fig. 4 a), VolNet has eight full pre-activation units. The details of full pre-activation unit can be found in [12]. All convolution kernel size is 3×3 . In Res1a, the stride is set as 2 to fit the input and output size. In Res2a, Res2b, Res3a, Res3b, we use atrous convolution to obtain a large receptive field and maintain the output size as well. Res0a and Res0b both have 128 feature maps, while others have 256. Moreover, to fit the size of the feature maps, 1×1 convolution is also applied in the short path of Res0a and Res1a. The loss function used to train VolNet is cross-entropy.

3.4. Embedding 3D Geometric Features in a Layer-Wise Fashion

As soon as VolNet is trained, the parameters of it maintain unchanged. The outputs of GeoNet are then regressed to the outputs of VolNet. As illustrated in Fig. 4 b), GeoNet has a convolution layer along with batch normalization [13] and ReLU activation, a max pooling layer, and six residual building blocks. The convolution layer outputs 128 feature maps. And its kernel size is 7×7 , pad size is 3, stride size is 2. The kernel and stride sizes of max pooling are 3×3 and 2 respectively. The settings of each residual building block are the same as the corresponding block in VolNet.

As it is easier to train a shallow network than to train a deep network, the GeoNet is pre-trained in a layer-wise fashion. As shown in Fig. 4, at Step1, we train GeoNets Conv0 layer along with its following batch norm layer via minimizing the Loss1. At Step2, GeoNets Conv0 layer along with its following batch norm layer, Res1a block, and Res1b block are simultaneously trained via minimizing the Loss2. At this step, the Conv0 layer along with its following batch norm layer has been pre-trained at previous step. In the similar way, at Step3 and Step4, the Loss3 and Loss4 are minimized in turn. Benefitting from the layer-wise fashion, we avoid training the deep network GeoNet from scratch. Specially, we use mean square error as the loss function

$$\mathcal{L} = \sum_{i} \sum_{j} \frac{1}{2N} (\hat{\mathcal{F}}_{i}^{j} - \mathcal{F}_{i}^{j})^{2}, \qquad (4)$$

where $\hat{\mathcal{F}}_{i}^{j}$ is the output feature of GeoNet, and \mathcal{F}_{i}^{j} is the out-



Figure 4. The structures of VolNet and GeoNet, which use full pre-activation unit [12] as building block. F: the number of feature maps, S: stride, A: atrous rate, K: kernel size, P: pad. Moreover, GeoNet regresses to VolNet in a layer-wise fashion.

Table 1. Dataset statistic. We use three classes of data from PASCAL-Part to evaluate our method. For each class, training data for VolNet and GeoNet are synthesised using ShapeNet.

	Training	Training	Testing	Testing	Part
	model	sample	model	sample	number
PASCAL-Car	-	910	-	890	5
PASCAL-Aeroplane	-	368	-	373	5
PASCAL-Motorbike	-	305	-	305	5
ShapeNet-Car	1460	10220	364	2548	4
ShapeNet-Aeroplne	2152	8608	532	2152	4
ShapeNet-Motorbike	162	10368	40	2560	6

put feature of VolNet. And then, GeoNet is used to initialize one stream of our 2-stream model, which can be regarded as a 3D geometric features extractor.

4. Experiments

We use three classes of real-world image data from PASCAL-Part [7, 4] to evaluate our method. The three classes are Car, Aeroplane and Motorbike, respectively. For each class, we use ShapeNet [2] along with the part information [37] to synthesise training data for VolNet and GeoNet. The dataset statistic is summarized in Tab. 1. About the resolution, each input object image is cropped and resized to 224×224 , each volume plane is cropped and

Table 2. Results on PASCAL-Car, PASCA	L-Aeroplane and PA	ASCAL-Motorbike,	including IoU o	f each part, m	ean IoU and	mean pixel
accuracy. The proportion of each part is als	o presented.					

PASCAL-Car								
Method	Bkg	Body	Plate	Light	Wheel	Window	Mean	Pixel acc.
FCN [20]	75.11	56.72	0.18	1.23	37.26	35.78	34.38	78.04
DeepLab [3]	84.98	70.91	35.24	30.05	60.00	55.25	56.08	86.73
Ours(random initialize GeoNet)	85.39	69.76	39.58	33.29	57.63	56.39	57.01	86.83
Ours	87.30	73.88	45.35	41.74	63.30	59.99	61.93	88.54
Proportion	56.31	33.20	0.38	0.60	3.42	6.08	-	-
		PAS	CAL-Aero	plane				
Method	Bkg	Body	Engine	Wing	Tail	Wheel	Mean	Pixel acc.
FCN [20]	82.17	42.23	0.67	10.18	28.10	7.51	28.48	78.74
DeepLab [3]	89.09	59.56	19.15	27.85	48.53	26.40	45.10	85.83
Ours(random initialize GeoNet)	89.54	58.99	24.23	33.85	49.42	25.94	47.00	86.08
Ours	89.96	60.32	26.74	33.93	52.96	39.08	50.50	86.70
Proportion	69.61	17.73	1.81	4.91	5.44	0.48	-	-
		PAS	CAL-Moto	orbike				
Method	Bkg	Body	Wheel	Handle	Light	Saddle	Mean	Pixel acc.
FCN [20]	71.59	47.32	39.84	0.00	0.00	0.00	26.46	74.09
DeepLab [3]	79.80	62.32	64.66	0.00	15.64	0.00	37.07	83.17
Ours(random initialize GeoNet)	79.81	63.58	65.29	0.00	13.93	0.00	37.10	83.51
Ours	80.79	64.35	66.95	0.00	19.89	0.00	38.66	84.36
Proportion	56.16	28.92	13.89	0.33	0.59	0.10	-	-

resized to 56×56 , and the ground truth is resized to 28×28 . The segmentation result can be easily up-sampled using bilinear interpolation to obtain the result with the same size as the input.

Real-world data. For each class of the PASCAL-Part, we merge some part classes to form a new part class, *i.e.*, we reserve five part classes for each class. The reserved part classes can be found in Tab. 2.

Synthetic data. To synthesise the training data using ShapeNet, we follow the process in [28]. Specifically, for each class, the distribution of viewpoint and translation is estimated respectively using PASCAL-3D dataset [35]. A set of viewpoints and translations is then random generated for each model according to this distribution. To render an RGB image, the set-up of lightings is random generated. Each RGB image is rendered in Blender. Moreover, a background is random selected from PASCAL training dataset and added to each rendered image. The same viewpoint and translation are used to generate volume and segmentation ground truth. The focal length and image size are constant during the rendering process. For each class, 80% of all the models are random selected as training model, which are used to generate training data. And the rest 20% models are used to generate testing data.

4.1. Results on Real-world Dataset

We first compare our method with DeepLab [3] and FCN [20] on PASCAL-Part-Car, PASCAL-Part-Aeroplane,

and PASCAL-Part-Motorbike. For each class, the stream AppNet of our method is initialized by ResNet-101, the stream GeoNet is taught by the corresponding VolNet, which means it is pre-trained on the synthetic data and regress to VolNet. Actually, DeepLab method is the same as to only use the stream AppNet.

We use stochastic gradient descent (SGD) to fine-tune the model. The initial learning rate is set as 0.001, the momentum is set as 0.9, the weight decay is set as 0.0005, the batch size is set as 2. The learning rate is decreased using the policy "poly" (with power as 0.9). All models are obtained after 20 epochs.

We present the IoU of each part, the mean IoU and the mean pixel accuracy in Tab. 2, respectively. To verify the effectiveness of our motivation and basic idea, we also random initialize the GeoNet, then train AppNet and GeoNet to perform object part segmentation. As we can see, our method achieve the best performance on all these three datasets, which demonstrates that the 3D geometric features are useful for OPSeg and compatible with 2D appearance features.

From the results, we also find a phenomenon, the proportion distribution of the parts is very uneven. And the lower proportion of the part, the lower accuracy of the result, especially for the handle and saddle of the Motorbike. This may be a good problem for future work since sometimes these small parts may be the right concerns of some tasks. We also show some visual results in Fig. 5, which



Figure 5. Some visual results of FCN [20], DeepLab [3], and our method on PASCAL-Car, PASCAL-Aeroplane, PASCAL-Motorbike.

also show that our method can obtain visual better results compared with previous methods.

All neural network models are trained and tested using the deep learning frame work Caffe [14]. Our computer has an Intel Core i7-4790 CPU and a NVIDIA GTX 1080 GPU. It takes about 40 ms to segment an test image using GPU.

4.2. Results on Synthetic Dataset

To analysis our method, we also present the results on synthetic dataset in Tab. 3. Specially, VolNet(w/o DCT) presents the results which are obtained by VolNet on original volume data. VolNet presents the results which are obtained by VolNet on volume(after DCT). VolNet + AppNet means to simultaneously use VolNet on volume(after DCT) and AppNet on image, which gives an upper-bound of the proposed method.

We also use stochastic gradient descent (SGD) to train the model on synthetic dataset. Most of the superparameters are the same as on read-world dataset. The only difference is that we set the initial learning rate as 0.01 if a model is trained from scratch, *e.g.*, VolNet.

From the experimental results, we can see that 3D geometric features usually are not as strong as 2D appearance features. However, 3D geometric features are compatible with 2D appearance features. The best performance is achieved when simultaneously exploit these features. Sometimes, neither 3D geometric features nor 2D appearance features can achieve good performance. But, the union of them can significantly reduce the ambiguities, e.g., for the Aeroplane in Tab. 3. The results also demonstrate that our method can obviously improve the object part segmentation performance.

4.3. Performance of VolNet

To verify the observation that 2D CNN is effective to process 3D volume data, we adapt VolNet to assign the right class label to an 3D model on Dataset ModelNet-40 [32]. ModelNet-40 consists of 40 classes of 3D mesh models, where 9843 models for training and 2486 models for testing. Each 3D mesh model is voxelized to a 30×30 volume as in [24]. We still use the full pre-activation unit as building-block. Our model has four residual blocks followed by three full-connection layers. And each buildingblock is followed by a 1×1 convolution layer to downsampling the feature map.

As shown in Table 4, we compare the performance with several 3D-Conv based methods VoxNet [22], E2E [32], 3D-NIN [24], SubvolumeSup [24], and one 2D-Conv based method AniProbing [24]. The referenced results are reported in [24]. Note that, we don't augment the data, while all the results of VoxNet, E2E, SubvolumeSup, AniProbing are obtained on augmented the data via random generating

Table 3. Results on ShapeNet-Car, ShapeNet-Aeroplane and ShapeNet-Motorbike, including IoU of each part, mean IoU and mean pixel accuracy. The proportion of each part is also presented.

			Sh	apeNet-C	ar				=
Method		Bkg	Wheel	Body	Roof	Hood	Mean	Pixel acc.	_
FCN [20)]	90.98	17.81	0.00	39.60	69.29	43.54	87.83	_
DeepLa	b [3]	93.74	43.57	47.61	59.29	77.72	64.39	91.85	
VolNet()	w/o DCT)	93.31	39.76	49.45	54.25	76.53	62.66	91.30	
VolNet		94.41	42.99	47.01	57.50	77.42	63.86	91.89	
VolNet -	+ AppNet	94.52	46.37	49.64	63.52	80.53	66.92	92.88	
GeoNet	+ AppNet	94.26	44.93	48.93	62.58	79.92	66.13	92.61	
Proporti	on	65.07	1.80	2.76	4.70	25.67	-	-	_
			Shape	Net-Aero	plane				=
Method		Bkg	Body	Engine	Wing	Tail	Mean	Pixel acc.	_
FCN [20)]	93.14	44.69	0.05	4.70	2.14	28.95	90.14	_
DeepLa	b [<mark>3</mark>]	93.84	51.40	9.31	26.01	10.71	38.26	91.26	
VolNet(w/o DCT)	93.17	40.92	12.24	15.16	10.67	34.43	89.14	-
VolNet		94.78	43.90	13.86	19.09	10.22	36.37	90.23	
VolNet -	+ AppNet	95.82	57.30	22.59	40.69	24.97	48.28	92.74	
GeoNet	+ AppNet	94.89	53.88	18.32	35.53	22.35	44.99	91.86	
Proporti	on	86.23	7.05	3.01	2.08	1.62	-	-	_
			Shape	Net-Aero	plane				
Method	Bkg	Gas tank	Seat	Wheel	Handle	Light	Body	Mean	Pixel acc
FCN [20]	87.47	8.62	0.58	40.46	3.09	0.01	56.60	28.12	83.48
DeepLab [3]	91.79	47.92	31.79	58.80	24.97	43.42	68.49	52.46	89.10
VolNet(w/o DCT)	89.00	46.01	29.70	51.22	19.70	0.00	65.12	42.96	87.13
VolNet	89.52	50.09	31.03	55.29	24.72	0.00	64.53	44.99	87.59
VolNet + AppNet	92.59	54.44	39.57	63.86	33.56	52.87	71.52	58.35	90.38
GeoNet + AppNet	92.42	53.05	38.58	63.18	34.00	53.34	70.77	57.91	90.11
Proportion	65.45	1.87	1.09	9.15	1.03	0.43	20.96	-	-

Table 4. 3D model classification results. Note that our method only use the initial data, while all other methods use augmented data.

Method	Instance accuracy
VoxNet [22]	83.8
E2E [32]	83.0
3D-NIN [24]	86.1
SubvolumeSup [24]	87.2
AniProbing [24]	85.9
Ours VolNet	86.8

azimuth and elevation rotation angles. And our method only exploits single orientation information of the input model.

We can see that though we don't use augmented data, our result is still comparable with the state-of-the-art 3D-Conv based method SubvolumeSup, which illustrates that VolNet is effective to process 3D volume data. At the same time, AniProbing also exploits the similar idea with VolNet. Its drawback is that some information may be lost as it first probes the 3D volume to one 2D feature map.

5. Conclusion

This paper exploits 3D geometric features to perform object part segmentation. To this end, we design a 2-stream CNN based on FCN framework. One stream is to extract 2D appearance features from the input image. While the other stream, named GeoNet, is to extract 3D geometric features. To pre-train the GeoNet, we propose a perspective voxelization algorithm to generate training data using 3D models. We also present a 2D convolution based network to effectively extract 3D geometric features. Experimental results show that 3D geometric features are compatible with 2D appearance features for object part segmentation.

The major limitation of this work may be that the geometric features now are class-specific and limited to several rigid object classes. To overcome this limitation, a more larger and various 3D model dataset would be useful. Besides this, the experimental results also show that the proportion distribution is usually very uneven, which may be also an interesting problem for future work.

Acknowledgement. This work was partially supported by grants from National Natural Science Foundation of China (61532003, 61672072, 61325011 & 61421003).

References

- L. J. Ba and R. Caruana. Do deep nets really need to be deep? In NIPS, 2014. 2
- [2] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *arXiv*, 2016. 2, 5
- [3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv, 2016. 1, 2, 3, 4, 6, 7, 8
- [4] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. L. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *CVPR*, 2014. 5
- [5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 2
- [6] S. Eslami and C. Williams. A generative model for partsbased object segmentation. In *NIPS*. 2012. 2
- M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascalnetwork.org/challenges/VOC/voc2010/workshop/index.html.
- [8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2
- [9] E. Grant, P. Kohli, and M. van Gerven. Deep disentangled representations for volumetric reconstruction. In ECCV Workshops, 2016. 2
- [10] S. Gupta, R. B. Girshick, P. A. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In ECCV, 2014. 2
- [11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3
- [12] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In ECCV, 2016. 4, 5
- [13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv*, 2015. 5
- [14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv*, 2014. 7
- [15] J. Krause, H. Jin, J. Yang, and F. F. Li. Fine-grained recognition without part annotations. In CVPR, 2015. 1
- [16] X. Liang, S. Liu, X. Shen, J. Yang, L. Liu, J. Dong, L. Lin, and S. Yan. Deep human parsing with active template regression. *TPAMI*, 37(12):2402–2414, 2015. 2
- [17] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan. Semantic object parsing with graph lstm. In *ECCV*, 2016. 1, 2
- [18] X. Liang, X. Shen, D. Xiang, J. Feng, L. Lin, and S. Yan. Semantic object parsing with local-global long short-term memory. In *CVPR*, 2016. 1, 2
- [19] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In *CVPR*, 2009. 1, 2

- [20] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2, 6, 7, 8
- [21] W. Lu, X. Lian, and A. Yuille. Parsing semantic parts of cars using graphical models and segment appearance consistency. In *BMVC*, 2014. 2
- [22] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*), pages 922–928, 2015. 2, 4, 7, 8
- [23] B. Ni, V. R. Paramathayalan, and P. Moulin. Multiple granularity analysis for fine-grained action detection. In *CVPR*, 2014. 1
- [24] C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In CVPR, 2016. 2, 7, 8
- [25] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013. 2
- [26] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 81(1):2–23, 2009. 1, 2
- [27] H. Su, Q. Huang, N. J. Mitra, Y. Li, and L. Guibas. Estimating image depth using shape collections. In ACM SIG-GRAPH, 2014. 2
- [28] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *ICCV*, 2015. 6
- [29] J. Sun and J. Ponce. Learning discriminative part detectors for image classification and cosegmentation. In *ICCV*, 2013.
- [30] J. Wang and A. L. Yuille. Semantic part segmentation using compositional model combining shape and appearance. In *CVPR*, 2015. 2
- [31] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille. Joint object and part segmentation using deep learned potentials. In *ICCV*, 2015. 1, 2
- [32] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 2, 4, 7, 8
- [33] F. Xia, P. Wang, L.-C. Chen, and A. L. Yuille. Zoom better to see clearer: Human and object parsing with hierarchical auto-zoom net. In *ECCV*, 2016. 1, 2
- [34] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-driven 3d voxel patterns for object category recognition. In *CVPR*, 2015. 2, 4
- [35] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014. 6
- [36] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg. Parsing clothing in fashion photographs. In CVPR, 2012. 2
- [37] L. Yi, V. G. Kim, D. Ceylan, I.-c. S. Mengyan, H. Su, C. Lu, Q. Huang, A. Sheffer, L. Guibas, and U. T. Austin. A scalable active framework for region annotation in 3d shape collections. In ACM SIGGRAPH Asia, 2016. 5