# 3D Shape Segmentation with Projective Convolutional Networks

Evangelos Kalogerakis[1]    Melinos Averkiou[2]    Subhransu Maji[1]    Siddhartha Chaudhuri[3]

[1]University of Massachusetts Amherst    [2]University of Cyprus    [3]IIT Bombay

## Abstract

*This paper introduces a deep architecture for segmenting 3D objects into their labeled semantic parts. Our architecture combines image-based Fully Convolutional Networks (FCNs) and surface-based Conditional Random Fields (CRFs) to yield coherent segmentations of 3D shapes. The image-based FCNs are used for efficient view-based reasoning about 3D object parts. Through a special projection layer, FCN outputs are effectively aggregated across multiple views and scales, then are projected onto the 3D object surfaces. Finally, a surface-based CRF combines the projected outputs with geometric consistency cues to yield coherent segmentations. The whole architecture (multi-view FCNs and CRF) is trained end-to-end. Our approach significantly outperforms the existing state-of-the-art methods in the currently largest segmentation benchmark (ShapeNet). Finally, we demonstrate promising segmentation results on noisy 3D shapes acquired from consumer-grade depth cameras.*

## 1. Introduction

In recent years there has been an explosion of 3D shape data on the web. In addition to the increasing number of community-curated CAD models, depth sensors deployed on a wide range of platforms are able to acquire 3D geometric representations of objects in the form of polygon meshes or point clouds. Although there have been significant advances in analyzing color images, in particular through deep networks, existing semantic reasoning techniques for 3D geometric shape data mostly rely on heuristic processing stages and hand-tuned geometric descriptors.

Our work focuses on the task of segmenting 3D shapes into labeled semantic parts. Compositional part-based reasoning for 3D shapes has been shown to be effective for a large number of vision, robotics and virtual reality applications, such as cross-modal analysis of 3D shapes and color images [60, 24], skeletal tracking [42], objection detection in images [11, 30, 36], 3D object reconstruction from images and line drawings [54, 24, 21], interactive assembly-based 3D modeling [5, 4], generating 3D shapes from a small number of examples [25], style transfer between 3D objects [33], robot navigation and grasping [40, 8], to name a few.

The shape segmentation task, while fundamental, is challenging because of the variety and ambiguity of shape parts that must be assigned the same semantic label; because accurately detecting boundaries between parts can involve extremely subtle cues; because local and global features must be jointly examined; and because the analysis must be robust to noise and undersampling.

We propose a deep architecture for segmenting and labeling 3D shapes that simply and effectively addresses these challenges, and significantly outperforms prior methods. The key insights of our technique are to repurpose image-based deep networks for view-based reasoning, and aggregate their outputs onto the surface representation of the shape in a geometrically consistent manner. We make no geometric, topological or orientation assumptions about the shape, nor exploit any hand-tuned geometric descriptors.

Our view-based approach is motivated by the success of deep networks on image segmentation tasks. Using rendered shapes lets us initialize our network with layers that have been trained on large image datasets, allowing better generalization. Since images depict shapes of photographed objects (along with texture), we expect such pre-trained layers to already encode some information about parts and their relationships. Recent work on view-based 3D shape classification [47, 38] and RGB-D recognition [15, 46] have shown the benefits of transferring learned representations from color images to geometric and depth data.

A view-based approach to 3D shape segmentation must overcome several technical obstacles. First, views must be selected such that they together cover the shape surface as much as possible and minimize occlusions. Second, shape parts can be visible in more than one view, thus our method must effectively consolidate information across multiple views. Third, we must guarantee that the segmentation is complete and coherent. This means all the surface area, including any heavily occluded portions, should be labeled, and neighboring surface areas should likely have the same label unless separated by a strong boundary feature.

Our approach, shown in Figure 1, systematically addresses these difficulties using a single feed-forward network. Given a raw 3D polygon mesh as input, our method generates a set of images from multiple views that are automatically selected for optimal surface coverage. These images are fed into the network, which outputs confidence maps per part via image processing layers. The confidence maps are
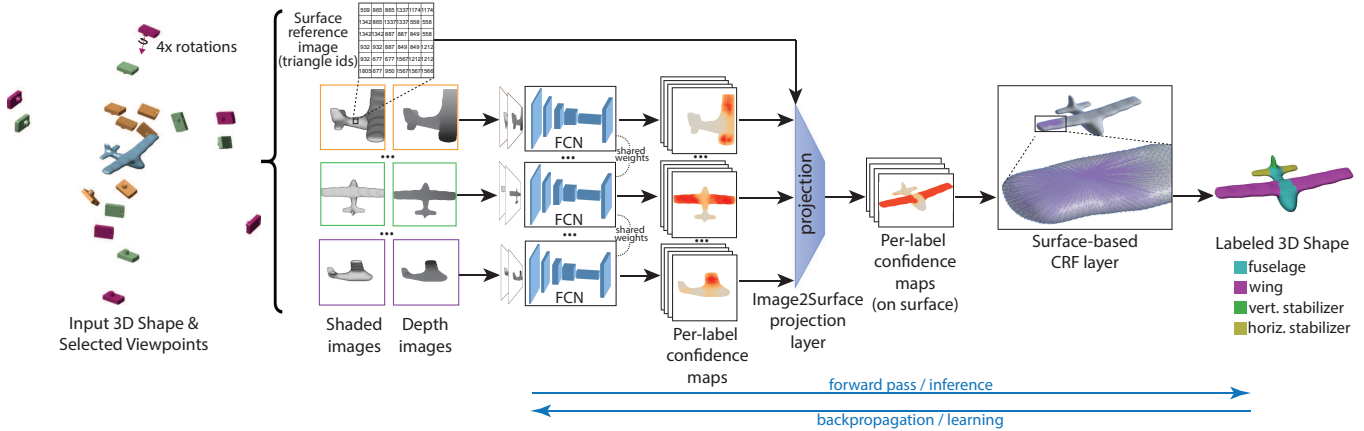
1

Figure 1. Pipeline and architecture of our method for 3D shape segmentation and labeling. Given an input shape, a set of viewpoints are computed at different scales such that the viewed shape surface is maximally covered (left). Shaded and depth images from these viewpoints are processed through our architecture (here we show images for three viewpoints, corresponding to 3 different scales). Our architecture employs image-based Fully Convolutional Network (FCN) modules with shared parameters to process the input images. The modules output image-based part label confidences per view. Here we show confidence maps for the wing label (the redder the color, the higher the confidence). The confidences are aggregated and projected on the shape surface through a special projection layer. Then they are further processed through a surface-based CRF that promotes consistent labeling of the entire surface (right).

fused and projected onto the shape surface representation through a projection layer. Finally, our architecture incorporates a surface-based Conditional Random Field (CRF) layer that promotes consistent labeling of the entire surface. The whole network, including the CRF, is trained in an end-to-end manner to achieve optimal performance.

Our main contribution is the introduction of a deep architecture for compositional part-based reasoning on 3D shape representations without the use of hand-engineered geometry processing stages or hand-tuned descriptors. We demonstrate significant improvements over the state-of-the-art. For complex objects, such as aircraft, motor vehicles, and furniture, our method increases part labeling accuracy by a remarkable ∼8% over the state of the art on the currently largest 3D shape segmentation dataset.

## 2. Related work

Our work is related to learning methods for segmentation of images (including RGB-D data) and 3D shapes.

**Image-based segmentation.** There is a vast literature on segmenting images into objects and their parts. Most recent techniques are based on variants of random forest classifiers or convolutional networks. An example of the former is the remarkably fast and accurate human-pose estimator that uses depth data from Kinect sensors for labeling human parts [42]. Our work builds on the success of convolutional networks for material segmentation, scene labeling, and object part-labeling tasks. These approaches use image classification networks repurposed for dense image labeling, commonly a fully-convolutional network (FCN) [32], to obtain an initial labeling. Several strategies for improving these initial estimates have been proposed including techniques based on top-down region-based reasoning [10, 16], CRFs [6, 31], atrous convolutional layers [6, 57], decon-

volutional layers [35], recurrent networks [59], or a multi-scale analysis [34, 17]. Several works [29, 1, 2] have also focused on learning feature representations from RGB-D data (e.g. those captured using a Kinect sensor) for object-level recognition and detection in scenes. Recently, Gupta et al. [15] showed that image-based networks can be repurposed for extracting depth representations for object detection and segmentation. Recent works [14, 45, 18] have applied a similar strategy for indoor scene recognition tasks.

In contrast to the above methods, our work aims to segment geometric representations of 3D objects, in the form of polygon meshes, created through 3D modeling tools or reconstruction techniques. The 3D models of these objects often do not contain texture or color information. Segmenting these 3D objects into parts requires architectures that are capable of operating on their geometric representations.

**Learning 3D shape representations from images.** A few recent methods attempt to learn volumetric representations of shapes from images via convolutional networks that employ special layers to model shape projections onto images [55, 39]. Alternatively, mesh-based representations can also be learned from images by assuming a fixed number of mesh vertices [39]. In contrast to these works, our architecture discriminatively learns view-based shape representations along with a surface-based CRF such that the view projections match an input surface signal (part labels). Our 3D-2D projection mechanism is differentiable, parameter-free, and sparse, since it operates only on the shape surface rather than its volume. In contrast to the mesh representations of [39], we do not assume that meshes have a fixed number of vertices, which does not hold true for general 3D models. Our method is more related to methods that learn view-based shape representations [47, 38]. However, these methods only learn global representations

for shape classification and rely on fixed sets of views. Our method instead learns view-based shape representations for part-based reasoning through adaptively selected views. It also uses a CRF to resolve inconsistencies or missing surface information in the view representations.

**3D geometric shape segmentation.** The most common learning-based approach to shape segmentation is to assign part labels to geometric elements of the shape representation, such as polygons, points, or patches [53]. This is often done through various processing stages: first, hand-engineered geometric descriptors of these elements are extracted (e.g. surface curvature, shape diameter, local histograms of point or normal distributions, surface eigenfunctions, etc.); then, a clustering method or classifier infers part labels for elements based on their descriptors; and finally (optionally) a separate graph cuts step is employed to smooth out the surface labeling [26, 41, 43, 19, 58]. Recently, a convolutional network has been proposed as an alternative element classifier [13], yet it operates on hand-engineered geometric descriptors organized in a 2D matrix lacking spatially coherent structure for conventional convolution. Another variant is to use two-layer networks which transform the input by randomized kernels, in the form of so-called "Extreme Learning Machines" [52], but these offer no better performance than standard shallow classifiers.

Other approaches segment shapes by employing non-rigid alignment steps through deformable part templates [27, 20], or transfer labels through surface correspondences and functional maps between 3D shapes [48, 22, 50, 27, 23]. These correspondence and alignment methods rely on hand-engineered geometric descriptors and deformation steps. Wang et al. [51] segment 3D shapes by warping and matching binary images of their projected views with segmented 2D images through Hausdorff distances. However, the matching procedure is hand-tuned, while potentially useful surface information, such as depth and normals, is ignored.

In contrast to all the above approaches, we propose a view-based deep architecture for shape segmentation with four main advantages. First, our architecture adopts image processing layers learned on large-scale image datasets, which are orders of magnitude larger than existing 3D datasets. As we show in this work, the deep stack of several layers extracts feature representations that can be successfully adapted to the task of shape segmentation. We note that such transfer has also been observed recently for shape recognition [47, 38]. Second, our architecture produces shape segmentations without the use of hand-engineered geometric descriptors or processing stages that are prone to degeneracies in the shape representation (i.e. surface noise, sampling artifacts, irregular mesh tesselation, mesh degeneracies, and so on). Third, we employ adaptive viewpoint selection to effectively capture all surface parts for analysis. Finally, our architecture is trained end-to-end, including all image and surface processing stages. As a result of these contributions, our method achieves better performance than prior work on big and complex datasets by a large margin.

## 3. Method

Given an input 3D shape, the goal of our method is to segment it into labeled parts. We designed a projective convolutional network to this end. Our network architecture is visualized in Figure 1. It takes as input a set of images from multiple views optimized for maximal surface coverage; extracts part-based confidence maps through image processing layers (pre-trained on large image datasets); combines and projects these maps onto the surface through a projection layer, and finally incorporates a surface-based Conditional Random Field (CRF) that favors coherent labeling of the input surface. The whole network, including the CRF, is trained end-to-end. In the following sections, we discuss the input to our network, its layers, and the training procedure.

**Input.** The input to our algorithm is a 3D shape represented as a polygon mesh. As a preprocessing step, the shape surface is sampled with uniformly distributed points (1024 in our implementation). Our algorithm first determines an overcomplete collection of viewpoints such that nearly every point of the surface is visible from at least $K$ viewpoints (in our implementation, $K = 3$). For each sampled surface point, we place viewpoints at different distances from it along its surface normal (distances are set to 0.5, 1.0 and 1.5 of the shape's bounding sphere radius). In this manner, the surface is depicted at different scales (Figure 1, left). We then determine a compact set of informative viewpoints that maximally cover the shape surface. For each viewpoint, the shape is rasterized under a perspective projection to a binary image, where we associate every "on" pixel with the sampled surface point closest to it. The coverage of the viewpoint is measured as the fraction of surface points visible from it, estimated by aggregating surface point references from the image. For each of the scales (camera distances), the viewpoint with largest coverage is inserted into a list. We then re-estimate coverages at this scale, omitting points already covered by the selected viewpoint, and the viewpoint with the next largest coverage is added to the list. The process is repeated until all surface points are covered at this scale. In our experiments, with man-made shapes and at our selected scales, approximately 20 viewpoints were enough to cover the vast majority of the surface area per scale.

After determining our viewpoint collection, we render the shape to shaded images and depth images. For each viewpoint, we place a camera pointing towards the surface point used to generate that viewpoint, and rotate its up-vector 4 times at 90 degree intervals (i.e, we use 4 in-plane rotations). For each of these 4 camera rotations, we render a shaded, greyscale $512 \times 512$ image using a typical computer graphics shader (Phong reflection model [37]) and a depth image, which are concatenated into a single two-channel image. These images are fed as input to the image processing module (FCN) of our network, described below. We found that both shaded and depth images are useful inputs. In early experiments, labeling accuracy dropped 2.5% using depth alone. This might be attributed to the more

"photo-realistic" appearance of shaded images, which better match the statistics of real images used to pretrain our architecture. We note that shaded images directly encode surface normals relative to view direction (shading is computed from the angle between normals and view direction).

In addition to the shaded and depth images, for each selected camera setting, we rasterize the shape into another image where each pixel stores the ID of the polygon whose projection is closest to the pixel center. These images, which we call "surface reference" images, are fed into the "projection layer" of our network (Figure 1).

**FCN module.** The two-channel images produced in the previous step are processed through identical image-based Fully-Connected Network (FCN) modules (Figure 1). Each FCN module outputs $L$ confidence maps of size $512 \times 512$ per each input image, where $L$ is the number of part labels. Specifically, in our implementation we employ the FCN architecture suggested in [57], which adopted the VGG-16 network [44] for dense prediction by removing its two last pooling and striding layers, and using dilated convolutions. We perform two additional modifications to this FCN architecture. First, since our input is a 2-channel image, we use 2-channel $3 \times 3$ filters instead of 3-channel (BGR) ones. We also adapted these filters to handle greyscale rather than color images during our training procedure. Second, we modified the output of the original FCN module. The original FCN outputs $L$ confidence maps of size $64 \times 64$. These are then converted into $L$ probability maps through a softmax operation. Instead, we upsample the confidence maps to size $512 \times 512$ through a transpose convolutional ("deconvolution") layer with learned parameters and stride 8. The confidences are later converted into probabilities through our CRF layer.

**Image2Surface projection layer.** The goal of this layer is to aggregate the confidence maps across multiple views, and project the result back onto the 3D surface. We note that both the locations and the number of optimal viewpoints can vary from shape to shape, and they are not ordered in any manner. Even if the optimal viewpoints were the same for different shapes, the views would still not necessarily be ordered, since we do not assume that shapes are oriented consistently. As a result, the projection layer should be invariant to the input image ordering. Given $M_s$ input images of an input shape $s$, the $L$ confidence maps extracted from the FCN module are stacked into a $M_s \times 512 \times 512 \times L$ image. The projection layer takes as input this 4D image. In addition, it takes as input the surface reference (polygon ID) images, also stacked into a 3D $M_s \times 512 \times 512$ image. The layer outputs a $F_s \times L$ array, where $F_s$ is the number of polygons of the shape $s$. The projection is done through a view-pooling operation. For each surface polygon $f$ and part category label $l$, we assign a confidence $P(f, l)$ equal to the maximum label confidence across all pixels and input images that map to that polygon according to the surface reference images. Mathematically, this projection operation is formulated as:

$$\tilde{C}(f, l) = \max_{\substack{\forall m, i, j: \\ I(m, i, j) = f}} C(m, i, j, l) \qquad (1)$$

where $C(m, i, j, l)$ is the confidence of label $l$ at pixel $(i, j)$ of image $m$; $I(m, i, j)$ stores the polygon ID at pixel $(i, j)$ of the corresponding reference image $m$; and $\tilde{C}(f, l)$ is the output confidence of label $l$ at polygon $f$. We note that the surface reference images omit polygon references at and near the shape silhouette, since an excessively large, nearly occluded, portion of the surface tends to be mapped onto the silhouette, thus the projection becomes unreliable there. Instead of using the max operator, an alternative aggregation strategy would be to use the average instead of the maximum, but we observed that this results in a slightly lower performance (about $1\%$ in our experiments).

**Surface CRF.** Some small surface areas may be highly occluded and hence unobserved by any of the selected viewpoints, or not included in any of the reference images. For any such polygons, the label confidences are set to zero. The rest of the surface should propagate label confidences to these polygons. In addition, due to upsampling in the FCN module, there might be bleeding across surface convexities or concavities that are likely to be segmentation boundaries.

We define a CRF operating on the surface representation to deal with the above issues. Specifically, each polygon $f$ is assigned a random variable $R_f$ representing its label. The CRF includes a unary factor for each such variable, which is set according to the confidences produced in the projection layer: $\phi_{\text{unary}}(R_f = l) = \exp(\tilde{C}(f, l))$. The CRF also encodes pairwise interactions between these variables based on surface proximity and curvature. For each pair of neighboring polygons $(f, f')$, we define a factor that favors the same label for polygons which share normals (e.g. on a flat surface), and different labels otherwise. Given the angle $\omega_{f, f'}$ between their normals ($\omega_{f, f'}$ is divided by $\pi$ to map it between $[0, 1]$), the factor is defined as follows:

$$\phi_{adj}(R_f = l, R_{f'} = l') = \begin{cases} \exp\left(-w_{\text{adj}} \cdot w_{l, l'} \cdot \omega_{f, f'}^2\right), & l = l' \\ \exp\left(-w_{\text{adj}} \cdot w_{l, l'} \cdot (1 - \omega_{f, f'}^2)\right), & l \neq l' \end{cases}$$

where $w_{\text{adj}}$ and $w_{l, l'}$ are learned factor- and label-dependent weights. We also define factors that favor similar labels for polygons $f$, $f'$ which are spatially close to each other according to the geodesic distance $d_{f, f'}$ between them. These factors are defined for pairs of polygons whose geodesic distance is less than 10% of the bounding sphere radius in our implementation. This makes our CRF relatively dense and more sensitive to long-range interactions between surface variables. We note that for small meshes or point clouds, all pairs could be considered instead. The geodesic distance-based factors are defined as follows:

$$\phi_{dist}(R_f = l, R_{f'} = l') = \begin{cases} \exp\left(-w_{\text{dist}} \cdot w_{l, l'} \cdot d_{f, f'}^2\right), & l = l' \\ \exp\left(-w_{\text{dist}} \cdot w_{l, l'} \cdot (1 - d_{f, f'}^2)\right), & l \neq l' \end{cases}$$

where the factor-dependent weight $w_{\text{dist}}$ and label-dependent weights $w_{l, l'}$ are learned parameters, and $d_{f, f'}$

Figure 2. Labeled segmentation results for alternative versions of our CRF (best viewed in color).

represents the geodesic distance between $f$ and $f'$. Distances are normalized to $[0, 1]$.

Based on the above factors, our CRF is defined over all surface random variables $\mathbf{R}_s = \{R_1, R_2, \ldots, R_{F_s}\}$ of the shape $s$ as follows:

$$P(\mathbf{R}_s) = \frac{1}{Z_s} \prod_f \phi_{\text{unary}}(R_f) \prod_{\text{adj } f, f'} \phi_{\text{adj}}(R_f, R_{f'}) \prod_{f, f'} \phi_{\text{dist}}(R_f, R_{f'})$$

$$(2)$$

where $Z_s$ is a normalization constant. Exact inference is intractable, thus we resort to mean-field inference to approximate the most likely joint assignment to all random variables as well as their marginal probabilities. Our mean-field approximation uses distributions over single variables as messages (i.e. the posterior is approximated in a fully factorized form – see Algorithm 11.7 of [28]). Figure 2 shows how segmentation results degrade for alternative versions of our CRF, and when the unary term is used alone.

**Training procedure.** The FCN module is initialized with filters pre-trained on image processing tasks [57]. Since the input to our network are rendered grayscale (colorless) images, we average the BGR channel weights of the pre-trained filters of the first convolutional layer, i.e. the $3 \times 3 \times 3$ filters are converted to color-insensitive $3 \times 3 \times 1$ filters. Then, we replicate the weights twice to yield $3 \times 3 \times 2$ filters that can accept our 2-channel input images. The CRF weights are initialized to 1.

Given an input training dataset $S$ of 3D shapes, we first generate their depth, shaded, and reference images using our rendering procedure. Then, our algorithm fine-tunes the FCN module filter parameters $\boldsymbol{\theta}$ and learns the CRF weights $w_{\text{adj}}, w_{\text{dist}}, \{w_{l,l'}\}$ to maximize their log-likelihood plus a small regularization term:

$$L = \frac{1}{|S|} \sum_{s \in S} \log P(\mathbf{R}_s = \mathbf{T}_s) + \lambda ||\boldsymbol{\theta}||^2 \qquad (3)$$

where $\mathbf{T}_s$ are ground-truth labels per surface variable for the training shape $s$, and $\lambda$ is a regularization parameter (weight decay) set to $10^{-3}$ in our experiments. To maximize the above objective, we must compute its gradient w.r.t. the FCN module outputs, as required for backpropagation:

$$\frac{\partial L}{\partial C(m, i, j, l)} = \begin{cases} 1 - P(R_f = l) & \text{if } l = T_f \text{ and } I(m, i, j) = f \\ P(R_f = l) & \text{if } l \neq T_f \text{ and } I(m, i, j) = f \\ 0 & \text{otherwise} \end{cases}$$

$$(4)$$

Computing the gradient requires estimation of the marginal probabilities $P(R_f)$. We use mean-field inference to estimate the marginals (same inference procedure is used for training and testing). We observed that after 20 iterations,

mean-field often converges (i.e. marginals change very little). We also need to compute the gradient of the objective function w.r.t. the CRF weights. Since our CRF has the form of a log-linear model, gradients can be easily derived.

Given the estimated gradients, we can train our network through backpropagation. Backpropagation can send error messages towards any FCN branch i.e., any input image (Figure 1). One strategy to train our network would be to set up as many FCN branches as the largest number of rendered images across all training models. However, the number of selected viewpoints varies per model, thus the number of rendered images per model also varies, ranging from a few tens to a few hundreds in our datasets. Maintaining hundreds of FCN branches would exceed the memory capacity of current GPUs. Instead, during training, our strategy is to pick a random subset of 24 images per model, i.e. we keep 24 FCN branches with shared parameters in the GPU memory. For each batch, a different random subset per model is selected (i.e. no fixed set of views used for training). We note that the order of rendered images does not matter – our view pooling is invariant to the input image ordering. Our training strategy is reminiscent of the DropConnect technique [49], which tends to reduce overfitting.

At test time all rendered images per model are used to make predictions. The forward pass does not require all the input images to be processed at once (i.e., not all FCN branches need to be set up). At test time, the image label confidences are sequentially projected onto the surface, which produces the same results as projecting all of them at once.

**Implementation.** Our network is implemented using C++ and Caffe [1]. Optimization is done through stochastic gradient descent with learning rate $10^{-3}$ and momentum $0.9$. We implemented a new Image2Surface layer in Caffe for projecting image-based confidences onto the shape surface. We also created a CRF layer that handles mean-field inference during the forward pass, and estimates the required gradients during backpropagation.

## 4. Evaluation

We now present experimental validations and analysis of our approach.

**Datasets.** We evaluated our method on manually-labeled segmentations available from the ShapeNetCore [56], Labeled-PSB (L-PSB) [7, 26], and COSEG datasets [50]. The dataset from ShapeNetCore currently contains 17,773 "expert-verified" segmentations of 3D models across 16 categories. The 3D models of this dataset are gathered

---

[1] Our source code, results and datasets are available on the project page: http://people.cs.umass.edu/kalo/papers/shapepfcn/
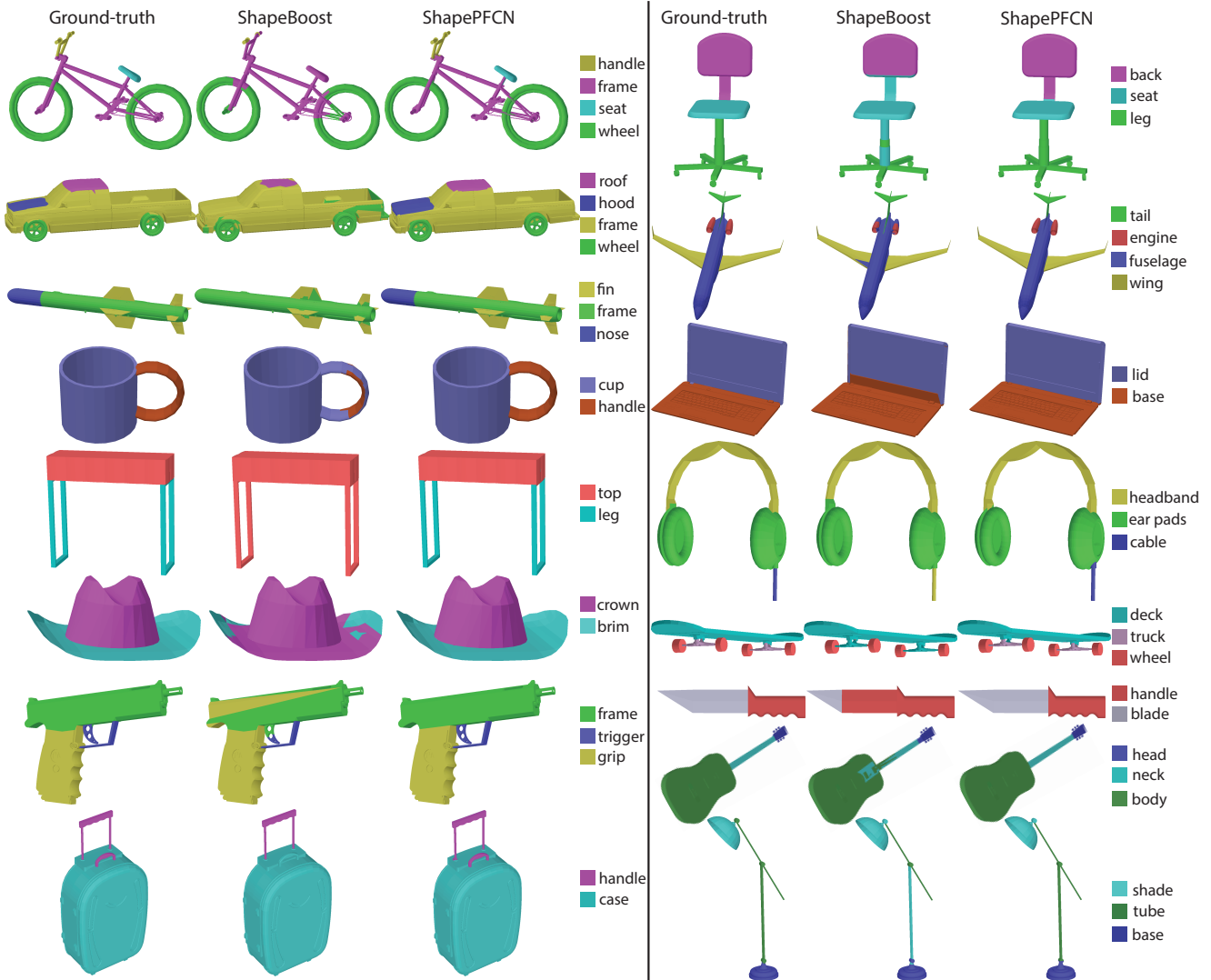
Figure 3. Ground-truth (human) labeled segmentations of ShapeNet shapes, along with segmentations produced by ShapeBoost [26] and our method (ShapePFCN) for test shapes originating from the ShapeNetCore dataset (best viewed in color).

"in the wild". They originate from the Trimble 3D Warehouse and Yobi3D repositories, and in general are typical representatives of objects created using 3D modeling tools for diverse applications. In contrast, the PSB and COSEG datasets are much smaller. PSB contains 380 segmented 3D models in 19 categories (20 models per category), while COSEG contains 190 segmented models in 8 categories, plus 900 synthetic variations of those in 3 categories. All models in the PSB have been carefully re-meshed and reconstructed such that their mesh representation is watertight with clean topology [50], facilitating use in geometry processing applications. Most shapes in COSEG are similarly preprocessed. As the authors of the PSB benchmark note [50], many 3D models cannot be re-meshed or re-constructed due to mesh degeneracies, hence they were not included in their datasets. From this aspect, our analysis is primarily focused on the dataset from ShapeNetCore, since it is by far the largest of the three datasets; contains

diverse, general-purpose 3D models; and was gathered "in the wild". Nevertheless, for completeness, we include comparisons with prior methods on all datasets (ShapeNetCore, L-PSB, COSEG).

**Prior methods.** We include comparisons with: (i) "ShapeBoost", the method described in [26], which employs graph cuts with a cascade of JointBoost classifiers for the unary term and GentleBoost for the pairwise term along with other geometric cues, and has state-of-the-art performance on the L-PSB; (ii) the recent method by Guo et al. [13] which reports comparable performance on the L-PSB dataset with ShapeBoost. This method employs graph cuts with a CNN on per-face geometric descriptors (also used in ShapeBoost), plus geometric cues for the pairwise term.

Computing geometric descriptors on ShapeNetCore shapes is challenging since they are often non-manifold "polygon soups" (meshes with arbitrarily disconnected sets of poly-

| | #train/test shapes | #part labels | ShapeBoost | Guo et al. | ShapePFCN |
|---|---|---|---|---|---|
| Airplane | 250 / 250 | 4 | 85.8 | 87.4 | **90.3** |
| Bag | 38 / 38 | 2 | 93.1 | 91.0 | **94.6** |
| Cap | 27 / 28 | 2 | 85.9 | 85.7 | **94.5** |
| Car | 250 / 250 | 4 | 79.5 | 80.1 | **86.7** |
| Chair | 250 / 250 | 4 | 70.1 | 66.8 | **82.9** |
| Earphone | 34 / 35 | 3 | 81.4 | 79.8 | **84.9** |
| Guitar | 250 / 250 | 3 | 89.0 | 89.9 | **91.8** |
| Knife | 196 / 196 | 2 | 81.2 | 77.1 | **82.8** |
| Lamp | 250 / 250 | 4 | 71.7 | 71.6 | **78.0** |
| Laptop | 222 / 223 | 2 | 86.1 | 82.7 | **95.3** |
| Motorbike | 101 / 101 | 6 | 77.2 | 80.1 | **87.0** |
| Mug | 92 / 92 | 2 | 94.9 | 95.1 | **96.0** |
| Pistol | 137 / 138 | 3 | 88.2 | 84.1 | **91.5** |
| Rocket | 33 / 33 | 3 | 79.2 | 76.9 | **81.6** |
| Skateboard | 76 / 76 | 3 | 91.0 | 89.6 | **91.9** |
| Table | 250 / 250 | 3 | 74.5 | 77.8 | **84.8** |

Table 1. Dataset statistics and labeling accuracy per category for test shapes in ShapeNetCore.

| | ShapeBoost | Guo et al. | ShapePFCN |
|---|---|---|---|
| Category Avg. | 83.0 | 82.2 | **88.4** |
| Category Avg. (>3 labels) | 76.9 | 77.2 | **85.0** |
| Dataset Avg. | 81.2 | 80.6 | **87.5** |
| Dataset Avg. (>3 labels) | 76.8 | 76.8 | **84.7** |

Table 2. Aggregate labeling accuracy on ShapeNetCore.

gons) with inconsistently oriented surface normals. Working with the original publicly available ShapeBoost implementation, we tried to make the computation of geometric descriptors and graph cuts as robust as possible. We preprocessed the meshes to correctly orient polygons (front-facing w.r.t. external viewpoints), repair connectivity (connect geometrically adjacent but topologically disconnected polygons, weld coincident vertices), and refine any excessively coarse mesh by planar subdivision of faces until it has >3,000 polygons. We also computed the geometric descriptors on point-sampled representations of the shape so that they are relatively invariant to tessellation artifacts. We note that neither the prior methods nor our method make any assumptions about shape orientation. No method explicitly used any pre-existing mesh subpart information manually entered by 3D modelers. Finally, we note that in the absence of a publicly available implementation, we used our own implementation of Guo et al's architecture.

**Dataset splits.** Since a standard training/test split is not publicly available for the segmented ShapeNetCore dataset, we introduced one (full list in the supplementary material). We randomly split each category into two halves, 50% for training and the rest for testing. The number of 3D shapes varies significantly per category in ShapeNetCore, ranging from 66 for Rockets to 5266 for Tables. The computation of geometric descriptors used in prior methods is expensive, taking up to an hour for a large mesh (e.g, 50K polygons). To keep things tractable, we used 250 randomly selected shapes for training, and 250 randomly selected shapes for testing for categories with more than 500 shapes. Our dataset statistics are listed in Table 1. For the much smaller

PSB and COSEG datasets, we used 12 shapes per category for training, and the rest for testing. Each of the methods below, including ours, is trained and tested separately on each shape category, following the standard practice in prior 3D mesh segmentation literature. All methods used the same splits per category. Our evaluation protocol differs from the one used by Guo et al. [13], where different methods are evaluated on randomly selected but different splits of the same category, which may cause inaccurate comparisons.

**Results.** The performance of all methods at test time is reported in Table 1 for the ShapeNetCore dataset. The labeling accuracy for a given shape is measured as the percentage of surface points labeled correctly according to the ground-truth point labeling provided by Yi et al. [56]. When considering a simple average of the per-category accuracies, our method performs $5.4\%$ better than the best-performing prior work [26] (Table 2, category average). Note, however, that several categories have disproportionately few models. A possibly more objective aggregate measure would be to weight each category by the number of test shapes. In this case, our method improves upon the state-of-the-art by $6.3\%$ (Table 2, dataset average). Most importantly, our method has significantly higher performance in categories with complex objects, such as motor vehicles, aircraft, and furniture, where the labeling task is also more challenging. For categories with more than 3 part labels, where part labeling is not just binary or ternary, our method improves upon prior work by $7.8\%$ in the unweighted estimate (Table 2, category average, >3 labels), or by $7.9\%$ when weighting by category size (Table 2, dataset average, >3 labels). This clearly indicates that our method can handle difficult shape labeling tasks in classes with complex objects significantly better than prior approaches. We include labeling results for all test shapes in the supplementary material. Figure 3 demonstrates human-labeled (ground-truth) segmentations, along with results from the best performing prior method (ShapeBoost) and our method (ShapePFCN) for various test shapes. We found that ShapeBoost, which relies on geometric descriptors, often fails for shapes with complex structure and topology (e.g. bikes, chairs), shapes with fine local features that can distort local descriptors (e.g. gun trigger, baggage handles), and shapes with coarse geometry (e.g., hats).

We also evaluated labeling accuracy in the PSB and COSEG datasets. We did not exclude any shape categories from our evaluation. Often, geometric methods are only applicable to certain types of "well-formed" input (e.g. manifolds, upright-oriented shapes, isometric articulations of a template, etc), and hence are not tested on unsuitable categories [53]. Our method, by contrast, is broadly applicable. We obtain an improvement over state-of-the-art methods for these datasets (92.6% for our method, 90.6% for ShapeBoost [26], 86.3% for Guo et al. [13] averaged over both datasets, see supplementary material for accuracy per category). We note that both PSB and COSEG contain a small number of shapes with limited variability, which even a shallow classifier may handle with high accuracy.
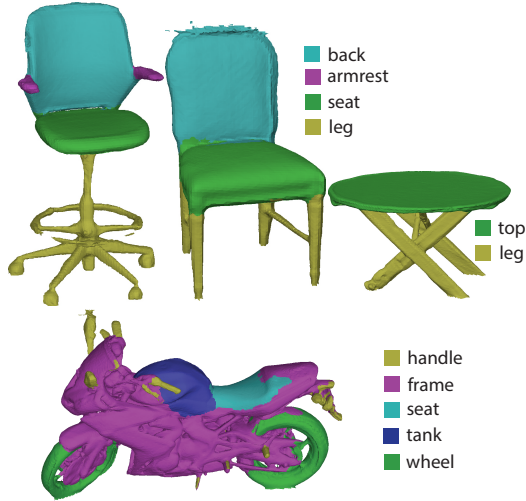
Figure 4. Labeled segmentations produced by our method on noisy objects reconstructed from RGBD sensor data.

| | fixed views | disjoint training | unary term | without pretrain. | full method |
|---|---|---|---|---|---|
| Category Avg. | 87.2 | 87.0 | 83.5 | 86.3 | **88.4** |
| Category Avg. (>3 labels) | 83.2 | 82.8 | 78.8 | 82.5 | **85.0** |
| Dataset Avg. | 86.2 | 85.9 | 82.1 | 85.7 | **87.5** |
| Dataset Avg. (>3 labels) | 82.9 | 82.4 | 78.7 | 82.3 | **84.7** |

Table 3. Labeling accuracy on ShapeNetCore for degraded variants of our method.

**Analysis.** We also evaluated our method against alternative degraded variations of it, to identify major sources of performance gains. Table 3 reports test labeling accuracy on ShapeNetCore for the following cases: (i) instead of selecting viewpoints that maximize surface coverage at different scales, we select fixed viewpoints placed on the vertices of a dodecahedron as suggested in [47] for shape classification (see "fixed views" column), (ii) we train the FCN module and CRF separately ("disjoint training" column), (iii) we do not use the CRF, i.e. we rely only on the unary term ("unary term" column), (iv) we train the FCN module from scratch instead of starting its training from the pre-trained VGG. We note that the CRF in particular, whether trained jointly or separately, is responsible for a major performance improvement (compare "unary term" to other columns). Pretraining also offers a noticeable gain. Viewpoint adaptation and joint training contribute smaller but still useful gains.

**Generalization to RGB-D sensor data.** Even if our architecture is trained on complete, noise-free, manually-modeled 3D shapes, it can still generalize to noisy, potentially incomplete objects acquired from RGB-D sensors. Figure 4 presents segmentation results for various objects originating from Choi et al.'s dataset [9]. The dataset contains polygon meshes reconstructed from raw RGB-D sensor data. We trained our architecture on the ShapeNetCore chair, tables and motorbike categories (separately), then applied it to the reconstructed objects. We note that the scans included the ground that we removed heuristically through plane fitting. There was also background clutter that we re-

moved through connected component analysis (i.e. we kept the dominant object in the scene). In contrast to our method, prior works heavily rely on hand-coded geometric descriptors that are highly distorted by noisy, incomplete geometry and fail to produce meaningful results (see supplementary material for results on these objects).

## 5. Conclusion

We presented a deep architecture designed to segment and label 3D shape parts. The key idea of our approach is to combine image-based fully convolutional networks for view-based reasoning, with a surface-based projection layer that aggregates FCN outputs across multiple views and a surface-based CRF to favor coherent shape segmentations. Our method significantly outperforms prior work on 3D shape segmentation and labeling.

There are several exciting avenues for future extensions. Currently our method uses a simple pairwise term based on surface distances and angles between surface normals. As a result, the segmentations can become noisy and not aligned with strong underlying mesh boundaries (Figure 4, see motorbike). Extracting robust boundaries through a learned module would be beneficial to our method. Our method currently deals with single-level, non-hierarchical segmentations. Further segmenting objects into fine-grained parts (e.g. segmenting motorbikes into sub-frame components) in a hierarchical manner would be useful in several vision and graphics applications.

Another possibility for future work is to investigate different types of input to our network. The input images we used represent surface depth and normals relative to view direction. Another possibility is to consider the HHA encoding [14] or even raw position data. However, these encodings assume a consistent gravity direction or alignment for input 3D shapes. Although in a few repositories (e.g. Trimble Warehouse) the majority of 3D models have consistent upright orientation, this does not hold for all 3D models, and especially for other online repositories whose shapes are oriented along different, random axes. There have been efforts to develop methods for consistent orientation or alignment of 3D shapes [12, 14, 3], yet existing methods require human supervision, or do not work well for various shape classes, such as outdoor objects or organic shapes.

Finally, our method is currently trained in a fully supervised manner. Extending our architecture to the semi-supervised or unsupervised setting to benefit from larger amounts of data is another exciting future direction.

# References

[1] M. Blum, J. T. Springenberg, J. Wülfing, and M. Riedmiller. A learned feature descriptor for object recognition in RGB-D data. In *Proc. ICRA*, pages 1298–1303. IEEE, 2012. 2

[2] L. Bo, X. Ren, and D. Fox. Unsupervised feature learning for RGB-D based object recognition. In *Experimental Robotics*, pages 387–402. Springer, 2013. 2

[3] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q.-X. 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 abs/1512.03012, 2015. 8

[4] S. Chaudhuri, E. Kalogerakis, S. Giguere, , and T. Funkhouser. AttribIt: Content creation with semantic attributes. *ACM UIST*, 2013. 1

[5] S. Chaudhuri, E. Kalogerakis, L. Guibas, and V. Koltun. Probabilistic reasoning for assembly-based 3D modeling. *Trans. Graph.*, 30(4), 2011. 1

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *Proc. ICLR*, 2015. 2

[7] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3D mesh segmentation. *Trans. Graph.*, 28(3), 2009. 5

[8] H.-P. Chiu, H. Liu, L. Kaelbling, and T. Lozano-Pérez. Class-specific grasping of 3D objects from a single 2D image. In *IROS*, 2010. 1

[9] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun. A large dataset of object scans. *arXiv:1602.02481*, 2016. 8

[10] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *Proc. CVPR*, 2015. 2

[11] S. Fidler, S. Dickinson, and R. Urtasun. 3D object detection and viewpoint estimation with a deformable 3D cuboid model. In *Proc. NIPS*, 2012. 1

[12] H. Fu, D. Cohen-Or, G. Dror, and A. Sheffer. Upright orientation of man-made objects. *ACM Trans. Graph.*, 27(3), 2008. 8

[13] K. Guo, D. Zou, and X. Chen. 3D mesh labeling via deep convolutional neural networks. *Trans. Graph.*, 35(1):3:1–3:12, 2015. 3, 6, 7

[14] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *Proc. CVPR*, 2013. 2, 8

[15] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *Proc. ECCV*, 2014. 1, 2

[16] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *Proc. ECCV*, 2014. 2

[17] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proc. CVPR*, 2015. 2

[18] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers. Fusenet: Incorporating depth into semantic segmentation via fusion-based CNN architecture. In *Proc. ACCV*, 2016. 2

[19] R. Hu, L. Fan, and L. Liu. Co-segmentation of 3D shapes via subspace clustering. *Comp. Graph. For.*, 31(5):1703–1713, 2012. 3

[20] H. Huang, E. Kalogerakis, and B. Marlin. Analysis and synthesis of 3D shape families via deep-learned generative models of surfaces. *Computer Graphics Forum*, 34(5), 2015. 3

[21] H. Huang, E. Kalogerakis, E. Yumer, and R. Měch. Shape synthesis from sketches via procedural models and convolutional networks. *TVCG*, 22(10):1, 2016. 1

[22] Q. Huang, V. Koltun, and L. Guibas. Joint shape segmentation with linear programming. *Trans. Graph.*, 30(6):125:1–125:12, 2011. 3

[23] Q. Huang, F. Wang, and L. Guibas. Functional map networks for analyzing and exploring large shape collections. *Trans. Graph.*, 33(4):36:1–36:11, 2014. 3

[24] Q. Huang, H. Wang, and V. Koltun. Single-view reconstruction via joint analysis of image and shape collections. *Trans. Graph.*, 34(4), 2015. 1

[25] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun. A probabilistic model for component-based shape synthesis. *Trans. Graph.*, 31(4):55, 2012. 1

[26] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3D mesh segmentation and labeling. *Trans. Graph.*, 29(4):102:1–102:12, 2010. 3, 5, 6, 7

[27] V. G. Kim, W. Li, N. J. Mitra, S. Chaudhuri, S. DiVerdi, and T. Funkhouser. Learning part-based templates from large collections of 3D shapes. *Trans. Graph.*, 32(4):70:1–70:12, 2013. 3

[28] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009. 5

[29] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *Proc. ICRA*, pages 1817–1824. IEEE, 2011. 2

[30] J. J. Lim, A. Khosla, and A. Torralba. FPM: Fine pose parts-based model with 3D cad models. In *Proc. ECCV*, 2014. 1

[31] G. Lin, C. Shen, I. Reid, et al. Efficient piecewise training of deep structured models for semantic segmentation. *arXiv preprint arXiv:1504.01013*, 2015. 2

[32] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, pages 3431–3440, 2015. 2

[33] Z. Lun, E. Kalogerakis, R. Wang, and A. Sheffer. Functionality preserving shape style transfer. *Trans. Graph.*, 2016. 1

[34] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *Proc. CVPR*, 2015. 2

[35] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proc. ICCV*, 2015. 2

[36] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Multi-view and 3D deformable part models. *PAMI*, 37(11):14, 2015. 1

[37] B. T. Phong. Illumination for computer generated pictures. *Comm. ACM*, 18(6), 1975. 3

[38] 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 *Proc. CVPR*, 2016. 1, 2, 3

[39] D. J. Rezende, S. M. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3D structure from images. In *NIPS*, 2016. 2

[40] R. B. Rusu. *Semantic 3D Object Maps for Everyday Robot Manipulation*. Springer, 2013. 1

[41] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang. Contextual part analogies in 3D objects. *IJCV*, 89(2-3):309–326, 2010. 3

[42] J. Shotton, A. Fitzgibbon, A. Blake, A. Kipman, M. Finocchio, R. Moore, and T. Sharp. Real-time human pose recognition in parts from a single depth image. In *Proc. CVPR*, 2011. 1, 2

[43] O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *Trans. Graph.*, 30(6):126:1–126:10, 2011. 3

[44] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR*, 2015. 4

[45] S. Song, S. P. Lichtenberg, and J. Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proc. CVPR*, 2015. 2

[46] S. Song and J. Xiao. Deep sliding shapes for amodal 3D object detection in RGB-D images. *arXiv preprint arXiv:1511.02300*, 2015. 1

[47] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *Proc. ICCV*, 2015. 1, 2, 3, 8

[48] O. van Kaick, A. Tagliasacchi, O. Sidi, H. Zhang, D. Cohen-Or, L. Wolf, and G. Hamarneh. Prior knowledge for part correspondence. *Comp. Graph. For.*, 30(2):553–562, 2011. 3

[49] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using DropConnect. In *Proc. ICML*, 2013. 5

[50] Y. Wang, S. Asafi, O. van Kaick, H. Zhang, D. Cohen-Or, and B. Chen. Active co-analysis of a set of shapes. *Trans. Graph.*, 31(6):165:1–165:10, 2012. 3, 5, 6

[51] Y. Wang, M. Gong, T. Wang, D. Cohen-Or, H. Zhang, and B. Chen. Projective analysis for 3D shape segmentation. *Trans. Graph.*, 32(6):192:1–192:12, 2013. 3

[52] Z. Xie, K. Xu, W. Shan, L. Liu, Y. Xiong, and H. Huang. Projective feature learning for 3D shapes with multi-view depth images. *Comp. Graph. For.*, 34(7):1–11, 2015. 3

[53] K. Xu, V. G. Kim, Q. Huang, and E. Kalogerakis. Data-driven shape analysis and processing. *Comp. Graph. For.*, 2016. 3, 7

[54] T. Xue, J. Liu, and X. Tang. Example-based 3D object reconstruction from line drawings. In *Proc. CVPR*, 2012. 1

[55] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3D object reconstruction without 3D supervision. In *NIPS*, 2016. 2

[56] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. A scalable active framework for region annotation in 3D shape collections. *Trans. Graph.*, in press, 2016. 5, 7

[57] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *Proc. ICLR*, 2016. 2, 4, 5

[58] M. E. Yumer, W. Chun, and A. Makadia. Co-segmentation of textured 3D shapes with sparse annotations. In *Proc. CVPR*, 2014. 3

[59] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *Proc. ICCV*, 2015. 2

[60] T. Zhou, P. Krähenbühl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3D-guided cycle consistency. In *Proc. CVPR*, 2016. 1