Noname manuscript No.
(will be inserted by the editor)

Image-driven Unsupervised 3D Model Co-segmentation

Juncheng Liu[1,2] · **Paul L. Rosin**[2] · **Xianfang Sun**[2] · **Jianguo Xiao**[1] · **Zhouhui Lian**[1*]

the date of receipt and acceptance should be inserted later

Abstract Segmentation of 3D models is a fundamental task in computer graphics and vision. Geometric methods usually lead to non-semantic and fragmentary segmentations. Learning techniques require a large amount of labeled training data. In this paper we explore the feasibility of 3D model segmentation by taking advantage of the huge number of easy-to-obtain 2D realistic images available on the Internet. The regional color exhibited in images provides information that is valuable for segmentation. To transfer the segmentations, we first filter out inappropriate images with several criteria. The views of these images are estimated by our proposed texture-invariant view estimation Siamese Network. The training samples are generated by rendering-based synthesis without laborious labeling. Subsequently, we transfer and merge the segmentations produced by each individual image by applying registration and a graph-based aggregation strategy. The final result is obtained by combining all segmentations within the 3D model set. Our qualitative and quantitative experimental results on several model categories validate effectiveness of our proposed method.

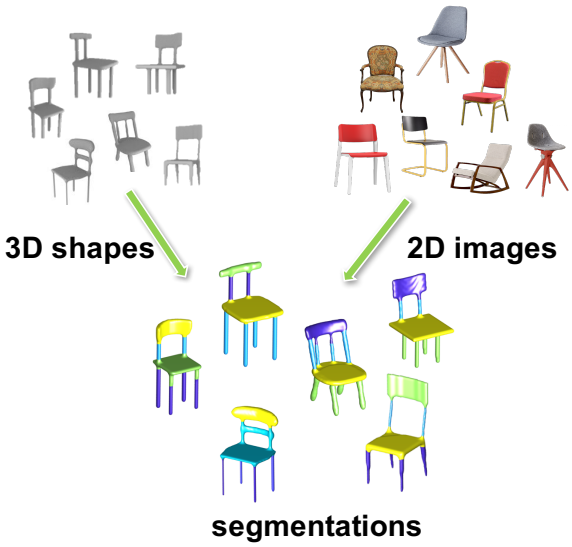


Fig. 1: The basic idea of our proposed method: segmenting 3D models by taking advantage of freely-available realistic 2D images online. Our system takes as inputs a set of 3D models, a search keyword for image retrieval, and segmentations are yielded accordingly.

1 Introduction

Segmentation of 3D models is a fundamental task in both the graphics and vision communities. Plausible segmentations facilitate not only the perception of 3D models but also subsequent processing. Despite its importance, current segmentation methods are far from satisfactory. The traditional methods leverage 3D mesh properties for segmentation. They often yield over segmentations and semantically meaningless parts. Recently, there are more and more methods proposed for semantic 3D model segmentations. However, the learning-based methods always require a large amount of hand-labeled training data, which is very time and effort consuming to obtain. Labeling faces of 3D meshes is much

Juncheng Liu
E-mail: liujuncheng@pku.edu.cn
Zhouhui Lian
E-mail: lianzhouhui@pku.edu.cn
* Corresponding author
[1] Institue of Computer Science and Technology, Peking University, Beijing, China
[2] School of Computer Science & Informatics, Cardiff University, UK

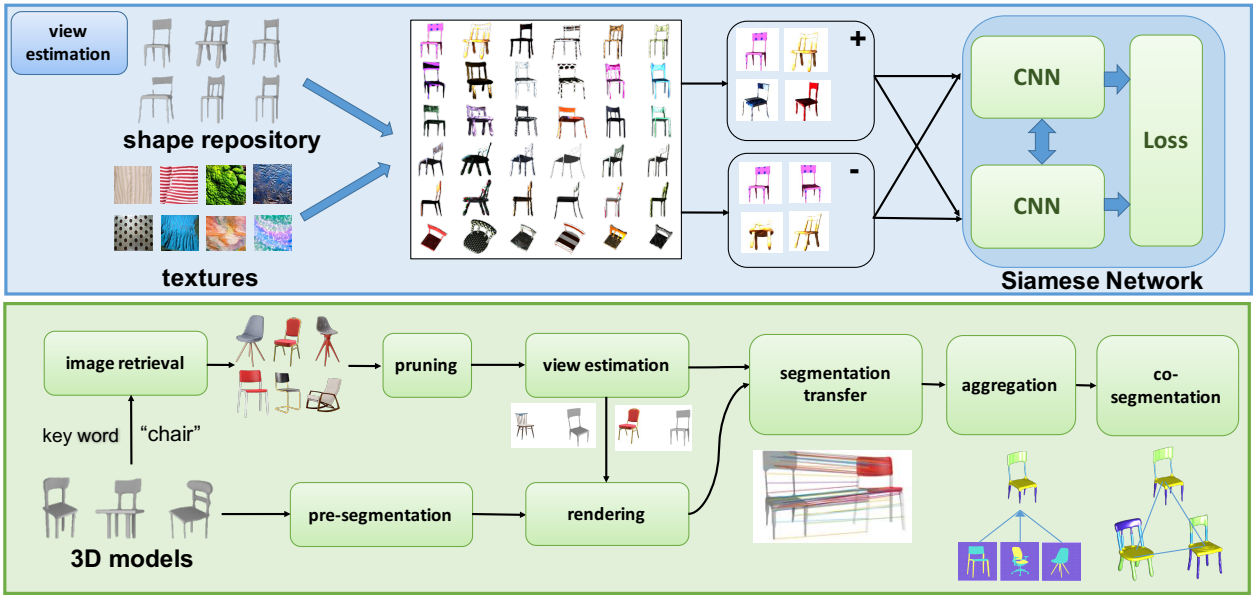


Fig. 2: The pipeline of our proposed method. The lower flowchart illustrates the main procedures for segmentation. The upper chart describes the training stage of view estimation network in detail.

more difficult than labeling the pixels of images. Besides, humans always hold different opinions upon segmentation, which makes it difficult to obtain a set of consistent training samples.

In this paper we explore the possibility of segmenting 3D models by taking advantage of realistic 2D images which are freely available online. The key observations are as follows: a large portion of real-world objects can be properly segmented according to their color and texture attributes, e.g. the chairs in Figure 1. 3D models tend to be produced without assigned textures, whereas 2D images always carry abundant depictions of color and textures. Since we have large availability of both 2D realistic images and 3D models, we can bridge the two types of representation and segment 3D models with the guidance of 2D images without supervision.

The main difficulties of realizing the aforementioned idea lie in the following aspects: (1) 3D models always exhibit spatially variant geometric details and are colorless (also know as white models) while realistic images contain abundant texture and material depictions, but are lacking explicit geometric information. These bring difficulties when establishing correspondences between these two types of data; (2) One image contains only one single-view depiction of an object while the 3D model has a full description of the mesh geometry; (3) Segmentations are always not consistent between different 2D images. Different images may favor different segmentations. The designed method should be able to integrate these individual segmentations in a plausible way.

We address the above problems by the following strategies which also summarize the main contributions of our paper:

- We propose a texture and geometry-invariant view estimation Siamese network that precisely recovers the camera pose of a realistic image.
- The training samples are generated and selected by an automatic rendering-based synthesis pipeline.
- Different segmentations are fused by a graph-based merging scheme, which allows a co-segmentation both image-model and model-model-wise.
- The method is unsupervised. Namely, no labeled data is required throughout the whole process.

2 Related Work

Geometry based segmentations. There exist many methods [1, 2] which segment 3D meshes according to their geometry properties such as shape diameter function, surface curvatures, normal distributions, etc.

These methods often cluster faces with similar geometric properties, and the level of segmentation can usually be controlled by parameters. However, these hand-crafted features are insufficient for exploring the semantic attributes of different parts.

Learning based segmentations. There is a vast amount of existing work on learning based segmentation of 2D images, among which the fully-convolutional

network (FCN) [3] is a most widely-used architecture. Based on it, several improvements have been made to achieve a higher accuracy [4].

Learning techniques can also be utilized in the segmentation of 3D meshes. Kalogerakis et al. [5] employ a Conditional Random Field for a consistent face labeling. Guo et al. and Shu et al. [6, 7] use hand-crafted classic local mesh descriptors along with Convolutional Neural Networks (CNNs) for more comprehensive high-level features. Recently, Kalogerakis et al. [8] proposed end-to-end projective convolutional networks that segment projected views instead, and integrate them using a Conditional Random Fields (CRF) layer.

As a semi-supervised algorithm, Wang et al. [9] segments a set of models by active learning and a small number of user interactions.

Compared to these existing methods, the main advantage of our method is that we do not need labeled training data which is time and energy consuming to obtain. We use segmentations contained in realistic images instead.

Image-guided segmentation. Several previous works have already explored 3D model segmentation with image guidance. [10] treats a 3D model as a collection of 2D projection silhouettes. Labeled 2D images are used as guidance and the segmentation is transfered to the 3D model by calculating bi-class Hausdorff distances. However, their method requires many labeled 2D images and only silhouettes are used which is insufficiently informative. The method described in [11] is the most similar to our work. It extracts small patches of rendered views of 3D models and realistic images. SIFT flow [12] is then used to calculate similarities. Patches are aligned by optimizing a free-form deformation function. Having the dense correspondence, the segmentation is achieved by aggregating all segments in each image.

Shape co-segmentation. There are also data-driven methods that segment a set of models all at once such as [13, 14]. These methods often deal with a set of models instead of one-by-one as do traditional methods, and usually higher accuracy is achieved by integrating segmentations produced by each individual model. The main difficulty lies in the correspondence establishment since there exist large diversity within each model set.

Our proposed method shares the similar idea with [15] yet we face different circumstances: their method aims to discover a view of natural images, which always includes a main object and a background, while as we will see in the following sections, our segmentation-oriented pipeline only leverages "clear-background" realistic images. In our observation, it is always easy to obtain these images with a transparent background by including this requirement in the search specification. Therefore we do not need to consider the background contained in images. Secondly, we randomly pick textures for model rendering while they leave the model untextured. We adopt this strategy to force our network to learn a common feature when two objects have different color and geometric appearance.

Fig. 3: Image segmentations by k-means. Top row: 5 original retrieved images after pruning. Middle row: the corresponding segmentation results (colomap indicating different segmented regions) by setting $k = 2$. Bottom row: the Lab color space (L component for lightness is discarded) of all pixels along with 2 cluster centroids marked as green stars.

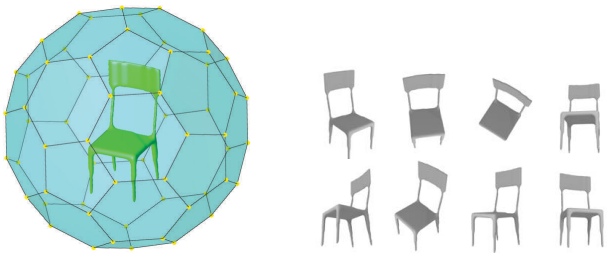
3 Method overview

The basic idea of our system is to leverage the large availability of natural images and use them for 3D model segmentation without supervision. Since 3D models tend to be generated without textures, using the appearance of its image counterparts can be more informative for segmentation. We expect the two kinds of data will complement each other and enable improved segmentation accuracy.

Overall, our system has a training stage and a main processing stage as illustrated in Figure 2. The main processing stage deal with the 3D model segmentation task while the training stage trains a Siamese network which is used to estimate camera poses in which objects are depicted in realistic images.

In the training stage, we train the so-called texture-invariant view estimation Siamese network which is designed to be robust to the large variations of geometry and topology, and more significantly, the textures exhibited in realistic images. We achieve this by automatically synthesizing a large number of training samples by rendering randomly textured aligned models.

Fig. 4: Image pruning. The first column is unsuitable because each image contains multiple instances. The second column is discarded due to the homogeneous appearance of each object. The third column shows the images .



(a) model and sampled view points (b) sample rendered views

Fig. 5: 60 sampled view points on a soccer ball surface, marked as yellow dots.

To segment a 3D model set, we first pre-segment each individual model into smaller patches. Meanwhile, we retrieve a 2D realistic image dataset according to the user-specified searching keywords. After pruning inappropriate images, we estimate the camera pose of each kept image. 3D models are subsequently rendered based on these estimated camera poses. Then we establish dense pixel-level correspondences between realistic images and rendered views of the 3D models to be segmented. The segmentations then can be transfered to the surfaces of 3D models. These different segmentations are aggregated by a graph-based majority voting strategy and smoothed by alpha-expansion multi-label graph cut [16]. Finally, by combining all the segmentations yielded by each model, we achieve consistent co-segmentation results.

Inputs. Our system takes as input a set of aligned 3D models of the same kind, a search keyword for image retrieval, and an estimated minimum number of parts n.

Outputs. Final results are segmented 3D models each having approximately n parts.

4 Image Pruning and 3D Model Pre-segmenation

To obtain 2D realistic images we search the Internet by providing a key word indicating the semantic category such as “chairs”, “guitars” etc. The first row of Figure 3 presents some examples retrieved by using “chairs” as the key word.

We also require the retrieved images to have a transparent background which is indicated in the alpha channel. Since there is such a large number of images available, we can assume there are is always a sufficient amount of images that satisfy this criterion.

However, not all of the retrieved images are appropriate for our application as shown in Figure 4. To avoid multiple instances being included in a single image, we first filter out images with more than one connected foreground region.

As the images are used to provide valuable information for segmentation, the abundance of color is expected to be neither too plain nor too complex. Images of homogeneous textures provide little useful information of segmentation while images with too many regions are likely to be over-segmented or merely complex texture patterns. The complexity of a 2D image is determined in Lab color space to eliminate the influence of illumination as shown in the middle and bottom rows of Figure 3.

Clustering centroids are then discovered by applying the k-means method in this space. The number of clusters is indicated manually. However, we have observed that most realistic images of artificial objects contain a very limited number of color regions. Therefore we simply use two cluster centroids for robustness. That is, the object in each image is split into two parts. However, this does not mean each model would finally be cut into two individual parts. We will discuss the aggregation of segmentations in detail in the following sections. Having the centroids in the color space, we calculate the variance between the centroids and the variance within each cluster. The variance is regarded as a measurement of color abundance. We remove all images which have too small between-cluster variance and too large within-cluster variance. The images with too small between-cluster variance usually contains homogeneous color and texture, which convey little information for segmentation. Examples of images with different between-cluster variance are shown in Figure 6.

This pruning will not discard all problematical images. However, for each 3D model we only select the M most visually similar images for segmentation ($M = 5$ in our implementation). The similarity can be calculated by any vision-based descriptors (we employ HoG [17]).

Fig. 6: Color complexity measured by between-cluster variance. The larger the number is, the more varied the color of the image is. Improper images are marked with red square.

As a prerequisite, we segment each 3D model into small patches using geometric properties such as curvature and normal consistency. This procedure does not have to be accurate, and the model can be over-segmented. We employ shape diameter function [18] in our implementation.

5 Texture-invariant View Estimation

5.1 Network Architecture

A crucial step of linking 3D models and 2D images is the view estimation of 2D images. Traditional methods use hand-crafted descriptors such as HoG [17] and SIFT [19] to retrieve the most similar view. However, they are not suitable for our circumstances due to the following reasons: most of 3D models often come without any texture or material assignments. The rendered views of them are hence colorless as shown in Figure 5b. On the other hand, the 2D images available on the Internet are mostly depictions of real objects with abundant color information. Additionally, the objects in images are not necessarily identical to the 3D models in terms of geometry and topology. Therefore, the features we used for view estimation must be invariant and suitable for both different textures and topologies.

To achieve the texture and topology-invariant features, we train a Siamese network [20] consisting of two parallel networks sharing the same parameters. The outputs of the last layers are then fed to a contrastive loss function, which calculates the similarity between the two inputs. We adopt the architectures of AlexNet [21] without the fully connected layers.

Fig. 7: Examples of textures and textured models. Each row shows the original model along with its 4 textured counterparts.

In the training stage, the network takes as input pairs of training images $(\mathbf{x}_1, \mathbf{x}_2)$ labeled as "genuine" $(y = 1)$ or "impostor" $(y = 0)$. The loss function is defined as:

$$L(\mathbf{x}_1, \mathbf{x}_2) = yD + (1 - y)\max(\sigma - D, 0), \quad (1)$$

where $D = \|f(\mathbf{x}_1) - f(\mathbf{x}_2)\|$ is the feature distance and $f(\cdot)$ is the output of the last layer of the network. The methodology of generating and choosing genuine and impostor pairs will be introduced in Section 5.2 and 5.3. This objective function favors a feature space where the genuine samples are projected close to each other while impostors pairs are should be placed far apart from each other.

We use a soft-threshold σ as a limit when the distance increases.

5.2 Training Sample Synthesis

In this section we describe the strategy we use to generate the training samples fed into the aforementioned network.

We first prepare a 3D model repository including various aligned 3D models of the same category as well as a texture database. 60 view points are sampled as the vertices of a soccer ball as illustrated in Figure 5. For each view, each 3D model is automatically parametrized and assigned with 10 randomly picked textures from the describable texture dataset [22]. These view points and textures are then used to render each individual model, forming an $N \times 60 \times 10$ image set where N is the number of models. Figure 7 shows two textured chairs along with their shading images. In the training stage, two rendered views are granted a "genuine" label if they come from the same view point and an "impostor" otherwise.

The 3D repository should cover variations of a certain kind of objects and be aligned beforehand. We expect the variations of shapes and textures to reinforce the ability of generalizations of our network.

Fig. 8: Matching process. From left to right: the pre-segmented 3D model, the dense correspondence between rendered view of model and 2D image, the label-transfered samples on mesh, the labeled 3D patches by majority voting.

Please note that the models we use for training sample synthesis do not necessarily coincide with the models to be segmented as long as they share the same semantic label and are aligned in the same way. In our experiments however, we use the same shape set for both tasks.

5.3 Label Generation

As stated in the previous section, two rendered views with the same camera pose are regarded as a "genuine" pair and "impostor" otherwise. We add more restrictions for more precise and effective learning. Note that the network is expected to be able to extract discriminative view-sensitive features of both rendered images of 3D models and 2D realistic images regardless of the color attributes and the geometry layout. Therefore, a genuine pair should be identical in terms of camera pose and different in other aspects (texture and geometry). On the contrary, we should leave everything identical except for the camera pose for the impostors, that is, we select impostor pairs from rendered views of the same model and the same texture.

By using this strategy, the network is forced to learn features that are only sensitive to the view difference and invariant to the color and geometry.

5.4 View Estimation

After training, we calculate the features of all the training samples, forming a view-sample feature space. Note the view index of each training sample is already known. When estimating the camera pose of an image, we first extract its feature by feeding it to the trained network. The estimated view is the one voted by the nearest neighbors in the view-sample feature space (10 neighbors in our experiment).

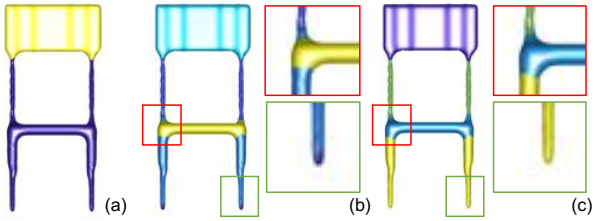


Fig. 9: Graph-based segmentation merging. (a) Segmentation with a smaller threshold where only 2 components are produced. (b) Segmentation with a larger threshold where 7 components are produced. (c) Graph-cut smoothed result. Note the small pieces are smoothed out.

6 Segmentation transfer and aggregation

After the view estimation we are now able to establish the dense correspondence between a 2D image and the rendered view of a model. Since we have the foreground mask for both the two types of images, we first uniformly sample a fixed number of points in the foreground regions denoted as sets $\mathcal{X}_1$ and $\mathcal{X}_2$. Then we employ the Coherent Point Drift (CPD) [23] method to register $\mathcal{X}_1$ and $\mathcal{X}_2$ as shown in Figure 8. Let ϕ denote the mapping function which maps $p \in \mathcal{X}_2$ to $p' \in \mathcal{X}_1$.

We keep face indices while rendering views. Therefore the sampled points on rendered views can be directly mapped to the 3D mesh. For simplicity, we assume labels are directly transfered to the 3D mesh from the segmented images. Naturally, the pre-segmented 3D patches partition $\mathcal{X}_2$ into smaller subsets P_i. Then P_i is assigned with the label which has the most occurrences as in Figure 8 in the following scheme:

$$l(P_i) = \arg\max_{\ell \in \mathcal{L}} \sum_{p \in P_i} \mathbb{1}\{\Phi(\phi(p)) = \ell\}, \tag{2}$$

where $\mathbb{1}(\cdot)$ is an indicator function and $\Phi(\cdot)$ represents the image segmentation which assigns a label to each position of an image from a label set $\mathcal{L}$. The above equation describes the procedure of majority voting within a 3D patch.

For each 2D image we repeat the above procedure, after which we expect to have M segmentations for the same model, where M is the number of 2D images we used for each model ($M = 5$ in our implementation). We have evaluated the face coverage rate of using different number of 2D images as shown in Figure 10. Since we have already estimated the view of each image in Section 5, we are able to compute the covered faces of each corresponding image (by visibility test). In Figure 10 we evaluate three types of face coverage

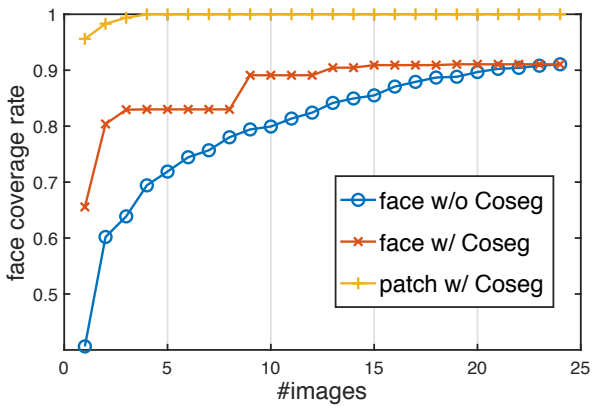


Fig. 10: Face coverage rate with number of used images.

Fig. 11: Samples of the view retrieval. For each tuple, left is the query image and right is the retrieved view. For simplicity, we only use the rendered views of one same model for each category.

rates: face-based without referring to other peer models (face w/o Coseg), face-based with referring to other peer models (face w/ Coseg) and patch-based with referring to other peer models (patch w/ Coseg) which is the method used in our implementation. From the figure it can be observed that by using individual face-based method only 70% of faces are covered when 5 images are used, the number increases to 80% when referring to other peer models. When using the more efficient patch-based method, almost all visible faces are covered by using only 5 images. Therefore in our implementation we only use 5 images for each model's segmentation, which is already sufficient to cover all visible faces.

We then use a graph-based method to aggregate segmentations. Specifically, each patch P_i is considered as a node on graph. Two patches P_i and P_j are connected by an edge with a weight $w_{i,j}$ which is set to the number of segmentations which assign the two patches the same label as follows:

$$w_{i,j} = \sum_{k=1}^{M} \mathbb{1}\{l_k(P_i) = l_k(P_j)\}, \tag{3}$$

where l_k represents the segmentation favored by image $k = 1, \ldots, M$. The edges with weights which are smaller than a given threshold δ are discarded from the graph:

$$w'_{i,j} = \max(0, w_{i,j} - \delta). \tag{4}$$

We increase the threshold from 1 to M until the desired number of connected components n is reached as illustrated by Figure 9. That makes the segmented model have a number of parts no less than the indicated parts number.

The graph-based aggregation does not require consistent labels across different segmentations. In fact it only checks whether a tuple $\{P_i, P_j\}$ belongs together or not. As a side-effect, it always causes over-segmentation. For a more smooth result, we filter out small isolated pieces by applying the alpha-expansion multi-label graph cut [16]. The smoothing effects can be observed in Figure 9c. After this we obtain an aggregated segmentation $\mathcal{S}_i$ for each model $i = 1, \ldots, N$.

7 Co-segmentation across models

After obtaining each individual segmentation, we are now able to propagate and integrate these segmentations across models. It is reasonable to believe a set of models bring more useful information hence a higher accuracy can be achieved and outliers can be eliminated. Since the models are all aligned, we register them together by again employing the CPD algorithm. Similar to Equation 2, we redefine the mapping function ϕ as a mapping between pairs of models:

$$l(P_i) = \arg\max_{\ell \in \mathcal{L}} \sum_{p \in P_i} \mathbb{1}\{\mathcal{S}(\phi(p)) = \ell\}, \tag{5}$$

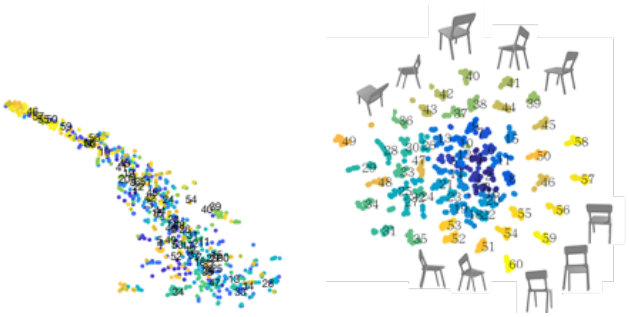
where $\mathcal{S}$ replacing the Φ assigns each face a label.

Similarly, the graph-based aggregation is again applied:

$$w_{i,j} = \sum_{k=1}^{N} \mathbb{1}\{l_k(P_i) = l_k(P_j)\}, \tag{6}$$

where l_k this time represents the segmentation favored by each individual model.

Subsequently, we repeat the graph-based aggregation steps as in Equation 4 until the final results are yielded. Multiple iterations can be run for the co-segmentation to converge to the optimum.



(a) feature space of pretrained VGG16 network

(b) feature space of our texture-variant Siamese network

Fig. 12: Feature space. Dimensionality reduced to 2 for visualization by t-SNE [24]. The colormap indicates each individual rendering view. View indices are marked at the mean position of all member points.

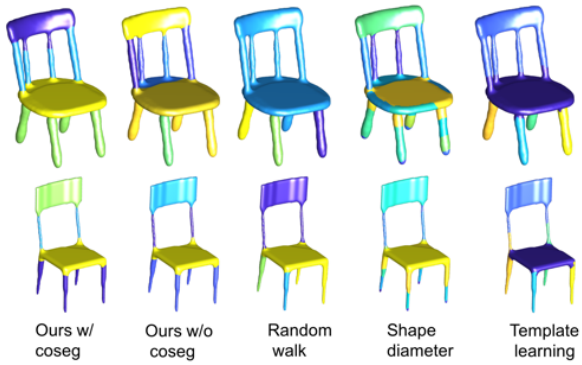


Fig. 13: Two samples of segmentation results yielded by different methods. From left to right: our result with co-segmentation procedure, our result without co-segmentation procedure, mesh random walk, shape diameter function and template learning. Note the four legs of the chair are segmented into one single part by our method while all other methods segment them into separate parts.

8 Experiments

8.1 Datasets and Environments

We evaluate our method on 8 categories (chairs, guitars, wine glass, lamps, cups,tables, glasses, pliers) of objects chosen from two public available datasets Shape COSEG [1] and Labeled-PSB [2] Dataset [5, 25]. These categories are more likely to have better segmentations with the guidance of images. Textures are chosen from the describable texture dataset [3], consisting of 5640 images, organized according to a list of 47 categories. There are 120 images for each category. Image sizes range between 300×300 and 640×640, and the images contain at least 90% of the surface representing the category attribute [22].

We use the Blender Smart UV project to automatically parametrize 3D models for texturing. The views of 3D models are also rendered in Blender with the Cycles renderer and a default Lambert material. The proposed texture-invariant Siamese network is implemented in python and Tensorflow with GeForce GTX Titan X graphics card. The rest of our algorithm is implemented in MATLAB 2018a environments.

8.2 Analysis of Learned Features

We provide a collection of view retrieval results in Figure 11. To explain its effectiveness, we analyze the features learned by our proposed texture-invariant network in this section. The strategy we used in the training sample preparation aims to emphasize the view difference while eliminating the other factors such as textures and geometric variations. Therefore we expect the distribution of the learned feature space is able to map samples with the same rendered views close to each other. As illustrated in Figure 12b, the features learned by our network exhibit clustered distribution with each cluster representing the corresponding view.

Additionally, we observe concentric spirals formed by the feature distribution with similar views sharing a similar radius. This validates that our proposed network is capable of learning "view-sensitive" features by our task-specific design. For comparison, we also visualize the distributions of features produced by the-used CNN architecture VGG16 [26] in Figure 12a, in which we could not observe similar patterns as seen in Figure 12b. This is due to the fact that a recognition-oriented network such as [26] is sensitive to color and appearance while our strategy forces the learning to focus on the views.

8.3 Qualitative and Quantitative Results

In this section we present both qualitative and quantitative evaluation results of our segmentation method.

It is relatively difficult to evaluate our proposed method since it is unsupervised and neither a geometry-

[1] http://irc.cs.sdu.edu.cn/~yunhai/public_html/ssl/ssd.htm

[2] https://people.cs.umass.edu/~kalo/papers/LabelMeshes/index.html

[3] https://www.robots.ox.ac.uk/~vgg/data/dtd/index.html

	chairs			guitars			wine glass			lamps		
	RI	CD	HD	RI	CD	HD	RI	CD	HD	RI	CD	HD
Random Walk	0.36	0.38	0.24	0.26	0.55	0.23	0.3	0.55	0.19	0.1	0.31	0.078
Shape Diameter	0.15	0.48	0.18	0.13	0.22	0.068	0.25	0.67	0.098	0.11	0.26	0.084
Template Learning	0.12	**0.16**	0.15	0.12	0.19	0.052	**0.096**	**0.24**	**0.018**	0.076	0.18	0.042
Ours w/o coseg	0.1	0.24	0.089	0.15	0.4	0.046	0.22	0.49	0.08	0.085	**0.15**	0.049
Ours w/ coseg	**0.076**	0.22	**0.055**	**0.051**	**0.068**	**0.018**	0.17	0.44	0.037	**0.057**	0.16	**0.024**

Table 1: Segmentation accuracy of COSEG.

	cups			tables			glasses			pliers		
	RI	CD	HD	RI	CD	HD	RI	CD	HD	RI	CD	HD
Random Walk	0.43	1.0	0.22	0.25	0.34	0.16	0.29	0.79	0.21	0.48	0.35	0.42
Shape Diameter	0.61	1.1	0.34	0.42	0.45	0.27	0.33	0.4	0.29	0.3	0.5	0.24
Approximate Convexity	0.14	0.54	0.061	**0.092**	**0.11**	0.091	0.33	**0.37**	0.28	0.28	**0.15**	0.29
Ours w/o coseg	0.25	0.82	0.11	0.21	0.24	0.13	0.3	0.42	0.24	0.39	0.5	0.21
Ours w/ coseg	**0.11**	**0.37**	**0.054**	0.15	0.35	**0.066**	**0.17**	0.54	**0.13**	**0.29**	0.36	**0.17**

Table 2: Segmentation accuracy of Labeled-PSB.

Fig. 14: Failure cases. Categories such as animals and hands may not be suitable to use our method as textures within such objects are too homogeneous to provide useful segmentation hints.

based nor a pure semantic segmentation. We leverage the realistic 2D images to guide the segmentation of 3D models, which brings in semantics to some degree. However, since we do not import labels in our processing, the segmented parts are meaningless (without a semantic label attached) as opposed to most of the learning-based segmentation methods. Therefore we mainly use geometry-based evaluation criteria for our quantitative evaluations. For most learning based method, recognition rate is usually used to evaluate their segmentation accuracy. To compare with them, we first estimate the label correspondences and then calculate the face recognition accuracy. We also provide representative segmentation results picked from each category for qualitative judgments in Figure 15. The number of segmented parts might be slightly different due to the graph-based aggregation and graph-cut smoothing.

We compare our proposed method with both geometry and semantic learning-based methods. For geometry-based evaluation, we use Rand Index (RI), Cut Discrepancy (CD) and Hamming Distance (HD). Since the Shape COSEG dataset provides ground-truth segmentations, we evaluate the accuracy accordingly.

It is also worth noticing that there is not a uniform way to semantically segment a 3D object. For instance, a chair can be decomposed into 3 or 4 individual parts with different granularity.

Table 1 and 2 collect the evaluation results of four individual categories regarding the 3 criteria. Since our method is unsupervised therefore we only compare with methods requiring no labeling. For comparison, we use two representative geometric methods: mesh random walk [1] and shape diameter function [18] which we use for our pre-segmentation as well as a template learning co-segmentation method [27] (results available only for COSEG dataset) and an approximate convexity analysis method [28] (results available only for L-PSB dataset). We also provide the accuracy of the segmentation before cross-model co-segmentation (named as "ours w/o coseg") along with the final results (named as "ours w/ coseg"). Figure 13 also provides sample segmentation results produced by each method.

From Table 1 and 2 we can observe significant improvements compared with traditional methods. This is due to the use of 2D images contributes semantic information which facilitates the segmentation. The legs of chairs, for instance, are supposed to belong to the same segment since they have the same semantic label. However, without the guidance of images, traditional geometry-based methods are always incapable of accomplishing this. However, this advantage is not evident in the first two metrics we employed. It can be captured by the Hamming distance as it calculates the region correspondence which links our produced labels and the ground-truth semantic labels. That also ex-

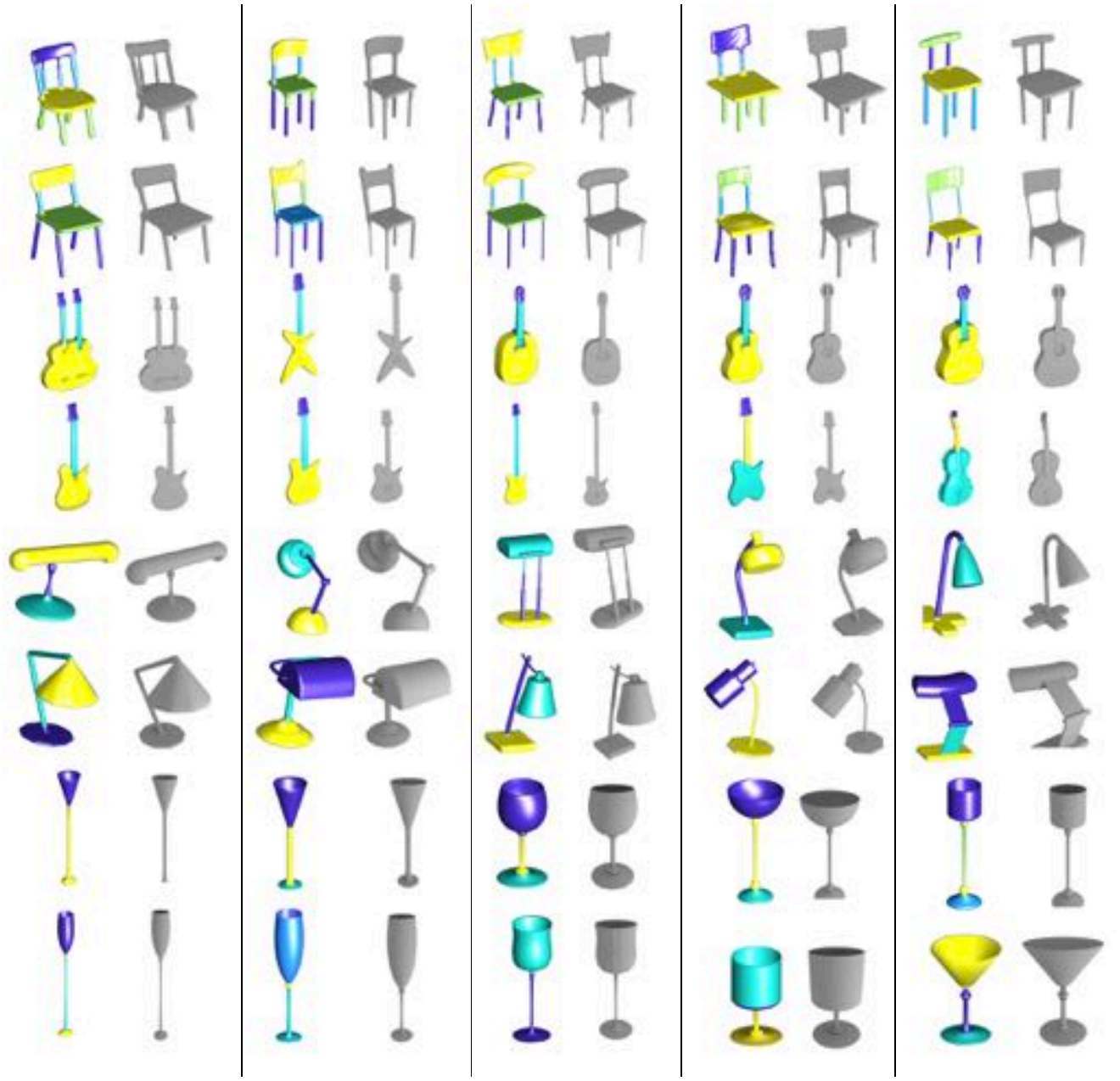


Fig. 15: Examples of segmentation results. Instances are chosen from "chairs","guitars","wine glass" and "lamps" within the Shape COSEG dataset. For each tuple, left is the segmented and right the original model. Note that the colormaps of these models are independent, that is, the color does not necessarily indicate the correspondence.

plained why the Hamming distance error of our method is much lower compared with the other two metrics.

Our method outperforms other methods in most cases. Especially the Hamming distance shows the significantly improved semantic accuracy by leveraging image guidance. In some non-rigid categories such as glasses and pliers, where diverse poses are presented, the accuracy might drop due to the inaccuracy of mesh correspondence. It can be observed that the accuracy improves significantly by applying the co-segmentations across models. The merging of each individual segmentation converges to the highest plausibility and the outliers are eliminated.

To compare with learning-based methods, we first estimate the label correspondence between the labels produced by our method and the ground-truth labels in a manner similar to bidirectional Hamming Distance:

$$\varphi(i) = \underset{j}{\operatorname{argmax}} \|S_1^i \cap S_2^j\|. \tag{7}$$

The recognition rate is calculated as the portion of the correctly labeled faces as shown in Table 3. It is worth noticing that our method is not learning-based hence no labeled training data is required compared to other methods in Table 3.

	ours	1	2	3	4
chairs	92.5	-	96.1	97.7	-
guitars	97.9	-	98.0	-	-
goblets	95.3	-	97.2	-	-
lamps	96.5	-	93.0	98.2	-
cups	94.6	96.3	93.8	98.3	96.0
tables	93.4	99.0	99.5	99.3	91.3
glasses	87.1	94.4	96.6	96.8	92.7
pliers	83.2	92.2	95.5	96.0	82.7

Table 3: Recognition rates of our method and other 4 learning-based methods for each category (1: Kalogerakis et al. [5], 2: Kalogerakis et al. [8], 3: Xie et al. [29], 4: Xu et al. [30]). The recognition rates of some methods for COSEG dataset are missing or partially missing since the methods were not tested over those categories.

9 Conclusion and Limitations

In this paper, we proposed a 3D model segmentation algorithm with the guidance of realistic images. The key observation is that the images of some certain kinds of real-world objects bring useful information of segmentations and hence can be used for 3D mesh collection segmentation.

We showed a complete pipeline integrating 3D model pre-segmentation, images pruning, view estimation, correspondence establishments and model co-segmentation. Additionally, we proposed a view estimation Siamese network along with a training sample synthesis and selection strategy.

We demonstrated that leveraging 2D images can significantly outperform existing unsupervised methods on the task of 3D model segmentation. Experiments have been conducted in four chosen categories of Shape COSEG datasets and qualitative and quantitative results both support our conclusion.

Limitations. The main drawback is that our proposed method could only be applied to certain kind of objects whose 2D images are expected to convey useful information of segmentation. When dealing with other objects such as animals and hands as shown in Figure 14, our method may fail as their fur can be homogeneous, from which useful segmentation hints are unlikely to be extracted. Our method is also incapable of dealing with objects with extremely large diversity within 3D models as the co-segmentation may fail due to the inaccuracy of correspondence. Lastly, since we do not use labels, the produced labels do not carry semantic meaning.

Acknowledgments

This work was supported by National Natural Science Foundation of China (Grant No.: 61672043 and 61672056), National Key Research and Development Program of China (2017YFB1002601) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). The support provided by China Scholarship Council (CSC) during a visit of Juncheng Liu to Cardiff University is acknowledged.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Yu-Kun Lai, Shi-Min Hu, Ralph R Martin, and Paul L Rosin. Fast mesh segmentation using random walks. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 183–191. ACM, 2008.
2. Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. *Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering*, volume 30. ACM, 2011.
3. Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
4. Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. Deep filter banks for texture recognition and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3828–3836, 2015.
5. Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3d mesh segmentation and labeling. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 102:1–102:12, New York, NY, USA, 2010. ACM.
6. Kan Guo, Dongqing Zou, and Xiaowu Chen. 3d mesh labeling via deep convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 35(1):3, 2015.

7. Zhenyu Shu, Chengwu Qi, Shiqing Xin, Chao Hu, Li Wang, Yu Zhang, and Ligang Liu. Unsupervised 3d shape segmentation and co-segmentation via deep learning. *Computer Aided Geometric Design*, 43:39–52, 2016.
8. Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3d shape segmentation with projective convolutional networks. In *Proc. CVPR*, volume 1, page 8, 2017.
9. Yunhai Wang, Shmulik Asafi, Oliver Van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Active co-analysis of a set of shapes. *ACM Transactions on Graphics (TOG)*, 31(6):165, 2012.
10. Yunhai Wang, Minglun Gong, Tianhua Wang, Daniel Cohen-Or, Hao Zhang, and Baoquan Chen. Projective analysis for 3d shape segmentation. *ACM Transactions on Graphics (TOG)*, 32(6):192, 2013.
11. Qixing Huang, Hai Wang, and Vladlen Koltun. Single-view reconstruction via joint analysis of image and shape collections. *ACM Transactions on Graphics (TOG)*, 34(4):87, 2015.
12. Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):978–994, 2011.
13. Qixing Huang, Vladlen Koltun, and Leonidas Guibas. Joint shape segmentation with linear programming. In *ACM transactions on graphics (TOG)*, volume 30, page 125. ACM, 2011.
14. Noa Fish, Oliver van Kaick, Amit Bermano, and Daniel Cohen-Or. Structure-oriented networks of shape collections. *ACM Transactions on Graphics (TOG)*, 35(6):171, 2016.
15. Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015.
16. Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, November 2001.
17. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.
18. Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249, 2008.
19. David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.
20. Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.
21. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.
22. M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.
23. Myronenko Andriy and Song Xubo. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 32(12):2262–75, 2010.
24. Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
25. Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.
26. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
27. Vladimir G Kim, Wilmot Li, Niloy J Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. Learning part-based templates from large collections of 3d shapes. *ACM Transactions on Graphics (TOG)*, 32(4):70, 2013.
28. Oliver Van Kaick, Noa Fish, Yanir Kleiman, Shmuel Asafi, and Daniel Cohen-Or. Shape segmentation by approximate convexity analysis. *Acm Transactions on Graphics*, 34(1):1–11, 2014.
29. Zhige Xie, Kai Xu, Ligang Liu, and Yueshan Xiong. 3d shape segmentation and labeling via extreme learning machine. In *Computer graphics forum*, volume 33, pages 85–95. Wiley Online Library, 2014.
30. Weiwei Xu, Zhouxu Shi, Mingliang Xu, Kun Zhou, Jingdong Wang, Bin Zhou, Jinrong Wang, and Zhenming Yuan. Transductive 3d shape segmentation using sparse reconstruction. In *Computer Graphics Forum*, volume 33, pages 107–115. Wiley Online Library, 2014.