Co-Segmentation of Textured 3D Shapes with Sparse Annotations

Mehmet Ersin Yumer
Carnegie Mellon University
meyumer@cmu.edu

Won Chun
Google
wonchun@google.com

Ameesh Makadia
Google
makadia@google.com

Abstract

We present a novel co-segmentation method for textured 3D shapes. Our algorithm takes a collection of textured shapes belonging to the same category and sparse annotations of foreground segments, and produces a joint dense segmentation of the shapes in the collection. We model the segments by a collectively trained Gaussian mixture model. The final model segmentation is formulated as an energy minimization across all models jointly, where intra-model edges control the smoothness and separation of model segments, and inter-model edges impart global consistency. We show promising results on two large real-world datasets, and also compare with previous shape-only 3D segmentation methods using publicly available datasets.

1. Introduction

3D shape segmentation is a core problem that facilitates high level shape processing [17]. Recent high level shape analysis works [13, 29, 30] either assume a low level segmentation is already present, or use manual segmentation to circumvent the limitations of current shape segmentation methods. Moreover, most of these methods require a compatible segmentation over a class of objects to facilitate intra-class shape operations.

In order to address the need for segmentation at scale, recent efforts have focused on fully supervised learning methods [14, 25] and *co-segmentation* approaches [10, 24, 28, 26] that leverage similarity of multiple objects to generate compatible segmentations. However, the limitations of these methods (such as requiring shape homogeneity within a category for unsupervised co-segmentation, requiring fully segmented models for learning segmentation, or working with only untextured models with clean manifold geometry and uniform discretization) make them still unsuitable for use in many practical applications.

What is missing is a segmentation process that can handle real-world datasets that exhibit large intra-category variation yet does not require the arduous process of collecting fully-segmented 3D models for training. To this end we introduce a new shape segmentation method tailored for real-word datasets of textured 3D models. Our process co-segments multiple models of the same category and requires only *sparse* annotations of foreground segments (see Figure 1(g)).

Our primary contributions are:

- We introduce a new problem in shape segmentation from sparse annotations, where shapes have not only geometry in $\mathbb{R}^3$ but also appearance in $\mathbb{R}^2$ in the form of texture images.
- We present a novel discretization for 3D shapes with textures, where we exploit well known superpixelization methods in $\mathbb{R}^2$ projected back into $\mathbb{R}^3$, which captures variability of appearance properties as well as geometric ones.
- We formulate a fully coupled shape co-segmentation process as an energy minimization task that incorporates both intra- and inter-model constraints.

Our promising results on challenging datasets indicating that with minimal sparse labeling we can consistently outperform unsupervised co-segmentation, and also come close in performance to fully supervised segmentation methods.

2. Related Work

Much of the existing 3D shape segmentation work has been inspired by advances in image segmentation. Image co-segmentation has been a long studied problem in computer vision (e.g. [19, 12, 27]). Interactive segmentation [18] and co-segmentation [15, 2] have also been proposed, where typically only foreground-background subtraction is desired and users indicate pixels on the foreground and background through scribbles. In our setting we are seeking to co-segment multiple foreground part segments per model, and our assumption is that we have only a single user-provided point per foreground segment (and no indication for the background).

As we highlighted in the previous section, automating 3D mesh segmentation has been an area of recent interest.

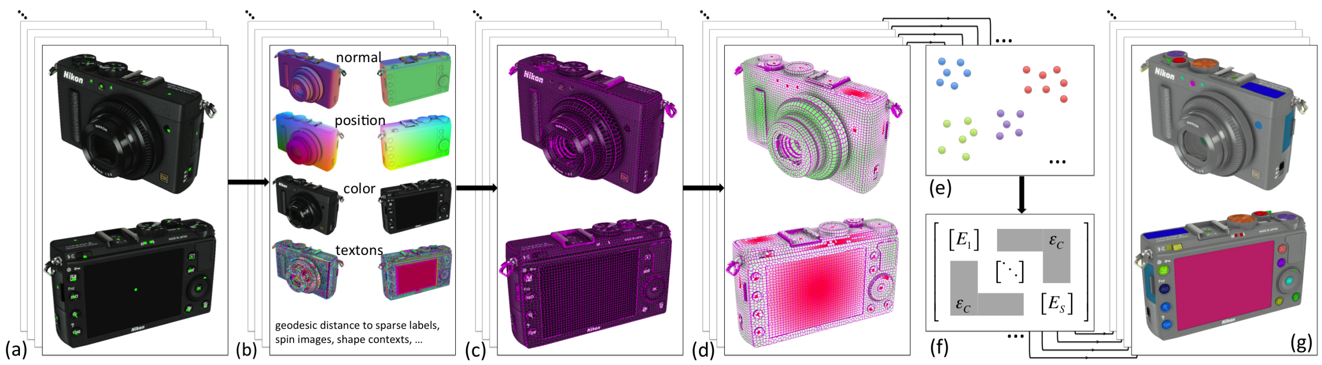

Figure 1. Given a set of textured models and sparse annotations (a), we first extract geometry and appearance descriptors (b). We create a discretization of the model leveraging its texture space and projecting back into 3D (c). We propagate the sparse label information to initialize constraints (d), followed by a GMM clustering (e). Finally, we construct a coupled energy minimization system (f) for a co-segmentation of the set (g).

Early works in mesh segmentation focus on partitioning a single mesh based on a single metric (See [21] for an extended survey).

Works on *co-analysis* exploit the latent information embedded in a larger shape set. Golovinskiy *et al.* [7] utilize rigid shape alignments and clustering of initial segmentations among a set of shapes to compute consistent segmentations. Fully supervised mesh segmentation and labeling algorithms that exploit a set of segmented training models [14, 25] require densely labeled training data. The acquisition of such data is labor intensive, and therefore cannot be scaled easily. Conversely, we require only very limited sparse labeling information.

Huang *et al.* [10] and Sidi *et al.* [24] introduced unsupervised consistent segmentation algorithms that first segment each model in a shape set independently before creating a joint segmentation. Wang *et al.* [28] extended the descriptor space clustering of [24] to improve co-analysis by iteratively introducing user provided constraints to the joint segmentation problem. They also utilize limited user input similar to ours, but the active learning framework requires user interaction during co-segmentation. In our framework, the sparse labels are can be collected in advance, which can easily be scaled.

3. Methodology

We begin with the assumption that all 3D shapes in the collection to be processed belong to a single category. A shape S in category $\mathcal{C}$ is represented by its surface in $\mathbb{R}^3$ as well as a texture image that is mapped onto its surface.

Let $G = \{V, E\}$ be a discretization of S, where nodes V each represent a relatively small part in S $(Area(x_i) \ll Area(S)|x_i \in V)$, and an edge $(x_i, x_j) \in E$ exists between nodes x_i and x_j if they are neighbors in S. Note that V need not be the faces of the polygonal mesh representation (*i.e.* triangles in most common cases). We use superpixels in 3D (Section 3.2).

Let $\mathcal{D} = \{l_1, l_2, \ldots, l_{|\mathcal{D}|}\}$ be a dictionary of foreground segment labels for shapes in $\mathcal{C}$. Given $\mathcal{D}$, we assume that each shape S in the collection has a sparse labeling $\mathcal{Y}_S = \{(x_i, \gamma_i)|i = 1, \ldots, n\}$, where $x_i \in V$, $\gamma_i \in \mathcal{D}_S$ and $\mathcal{D}_S \subset \mathcal{D}$. We further assume that for each separate segment in S, only a single (x_i, γ_i) pair is given $(n \ll |V|)$. However, we allow multiple segments of the same label $(n \geq |\mathcal{D}_S|)$, and all segments are separated by background regions.

Let $\mathcal{Z}_S = \{(x_i, z_i)|i = 1, \ldots, |V|\}$ be a dense labeling of shape S, where each entity $x_i \in V$ is labeled. We assume that there is an additional label l_b in the dense labeling which indicates the background (i.e. regions of the model that do not belong to any label in $\mathcal{D}$). Hence, $z_i \in \{\mathcal{D}_S \cup \{l_b\}\}$. Given $\mathcal{D}$ and sparsely labeled shapes $(S_i, \mathcal{Y}_{S_i})$ that belong to set $\mathcal{C}$, our task is to jointly determine a dense labeling $\mathcal{Z}_{S_i}$ for each shape $S_i \in \mathcal{C}$.

3.1. Descriptors

The surface of shape S is given by points $P \in \mathbb{R}^3$, and planar faces F (usually triangles) whose corners are defined by points in P. The appearance of a textured shape is specified by a 2D image $\mathcal{I}$. We also use these conventions for our geometry and appearance definitions. We compute geometry descriptors on the discrete mesh given by $\{V, F\}$, and the appearance descriptors in the image space given by $\mathcal{I}$. Prior to descriptor computation, we scale each mesh uniformly to match a unit bounding box in $\mathbb{R}^3$. Furthermore, we also down sample the textures $\mathcal{I}$ of each mesh uniformly in $\mathbb{R}^2$ to match the number of pixels of smallest texture.

3.1.1 Geometry Descriptors

Numerous geometry descriptors have been developed for shape analysis tasks (e.g. see [8]). At each point in P we extract common local features such as *Curvature* [6, 14],

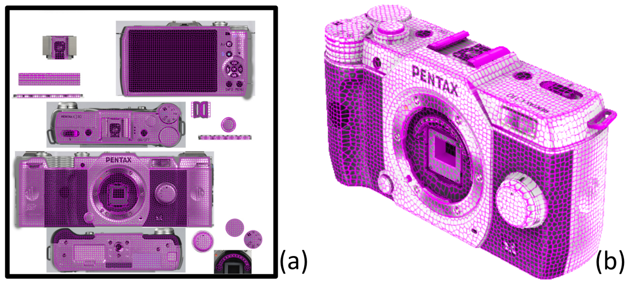
Figure 2. (a) Superpixelized charts in 2D (only the active pixel regions are superpixelized). (b) Projected superpixels in 3D, where chart boundaries are repaired.

Spin Images [11], *3D Shape Contexts* [3], *Neighborhood-PCA* [14], and *Average Geodesic Distance* [9] (for implementation details please see the supplemental material). Furthermore, we introduce the following novel descriptor.

Geodesic Distance to Landmark Labels. Let $\tilde{\mathcal{D}} \subset \mathcal{D}$ be the set of landmark labels present on all models in $\mathcal{C}$ (example of landmarks are *lens* and *shutter button* for the *cameras* category). For any point on a model we compute a Landmark Distance Descriptor (LDD) vector $\overrightarrow{x} \in \mathbb{R}^{|\tilde{\mathcal{D}}|}$ where the value in dimension i is the geodesic distance to the landmark point labeled with $l_i \in \tilde{\mathcal{D}}$. This descriptor captures the spatial context of a point by measuring the distances to landmark label locations.

3.1.2 Appearance Descriptors

We incorporate the appearance cues implicit in the texture of the 3D models by processing the original texture image $\mathcal{I}$. We use a color feature (pixels represented in $L^*a^*b^*$ color space) as well as textons (we adopt the 17 filter bank and texton clustering of [23]). Please see the supplemental material for details.

3.1.3 Descriptor Fusion

We first rasterize the geometry descriptors defined on the surface of S in $\mathbb{R}^3$ into $\mathbb{R}^2$ using the Geometry-Buffer [20]. However, instead of rendering into the image plane (which is the conventional use case of the G-buffer for viewpoint camera rendering), we rasterize into the texture space by using the texture coordinates associated with each vertex of the mesh. This procedure allows us to define a vector of concatenated (geometry and appearance) descriptors in $\mathbb{R}^{129+|\tilde{\mathcal{D}}|}$.

3.2. Superpixelization

Prior to segmentation, we generate a superpixelization of each shape independently, in their respective texture spaces. We follow SLIC superpixelization [1], however our distance measure d_{ij} also considers 3D proximity and orientation:

$$d_{ij} = d_{lab} + \frac{m_I}{\sqrt{A_s}} d_I + \frac{m_P}{d_{box}} d_P + m_\alpha * sin(\alpha) \quad (1)$$

where d_{lab}, d_I, and d_P are Euclidean distances between ith and jth pixels in $L^*a^*b^*$ color space, image space, and 3D model space respectively. A_s is the average superpixel size in pixels, d_{box} is the bounding box diameter of the 3D model, and α is the acute angle between the 3D normals assigned to pixels i and j. m_I, m_P, and m_α control the compactness of superpixels in image, 3D position, and 3D normal spaces respectively [1].

We compute a graph $G = \{V, E\}$, where nodes V are the superpixels, and edges E are the edges between neighboring superpixels in the texture space. Recall that we enable a trivial transfer of any function defined on the surface of a shape into its texture space (Section 3.1.3). Thus, for each pixel that belongs to a chart in the texture image there is a valid 3D position. Using the rasterized 3D position we project the superpixels back into 3D space and complete the edges between neighboring superpixels at chart boundaries and other 3D intersections (Figure 2). A superpixel's descriptor vector is computed using the average of the rasterized descriptors (Section 3.1.3) associated with its pixels.

3.3. Co-Segmentation

Recall that for each foreground segment we have just a single labeled point on the surface of the mesh, and furthermore there is no indication about the background (Figure 1(a)). To go from such input to full segmentations we introduce the novel co-segmentation process outlined below.

3.3.1 Semi-Supervised Gaussian Mixture Model

The shapes we are concerned with exhibit a considerable amount of detail both in the foreground segments as well as in the background. It is clear from the nature of the data (Figure 8) that some of the foreground segments, and most of the background, are multi modal. We therefore use Gaussian Mixture Model (GMM) clustering to model part segments as well as the background. We will estimate this GMM using all the objects in a category, and also we wish to exploit the sparse input labeling to guide the clustering (*i.e.* points in different segments should map to different clusters). To incorporate these sparse label inputs into the GMM model we follow closely the approach of [16].

Overloading our earlier notation, let $X = \{x_i\}, i = 1, \ldots, N$ be the observed descriptors from all the superpixels of all shapes in category $\mathcal{C}$. The traditional M-component GMM is:

$$P(x|\Theta) = \sum_{k=1}^{M} \pi_k P(x|\theta_k) \quad (2)$$

[1]In experiments: $m_I = 30$, $m_P = 300$, $m_\alpha = 10$, and $A_s = 250$.

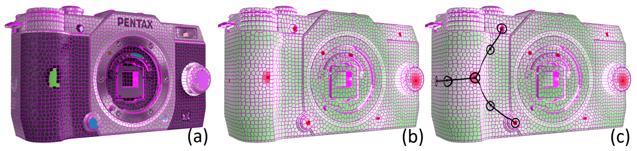

Figure 3. (a) Region growing initialization for sparse labels. (b) Corresponding confidence map. (c) An example constraint set generated by $n = 3$ nearest geodesic neighbors (for visual clarity, only one label's constraints are shown).

where $\Theta = (\pi_1, \dots, \pi_M, \theta_1, \dots, \theta_M)$ are the parameters of the GMM ($P(x|\theta_k) = \mathcal{N}(\sigma_k, \Sigma_k)$ is a normal distribution and π_i are the mixing coefficients).

If we let $Y = \{y_i | i = 1, \dots, N\}, y_i \in \{1, \dots, M\}$ be the latent variables (cluster assignments of the observed data in the GMM), then the complete data-likelihood for the GMM becomes

$$P(X, Y|\Theta) = P(X|Y, \Theta)P(Y|\Theta) \tag{3}$$

where $P(X|Y, \Theta) = \prod_{i=1}^{N} P(x_i|\theta_{y_i})$, and $P(Y|\Theta) = \prod_{i=1}^{N} \pi_{y_i}$.

Following [16] we can incorporate constraints into the model by introducing a weighting function $g(Y)$

$$\begin{aligned} P(Y|\Theta, G) &= \frac{1}{C}\left(\prod_{i=1}^{M} \pi_{y_i}\right) g(Y) \qquad (4) \\ g(Y) &= \prod_{i \neq j} e^{W_{ij}\delta(y_i, y_j)} \end{aligned}$$

Here $C = \sum_Y \left(\prod_j \pi_{y_j}\right) g(Y)$ is a normalization term. Prior clustering constraints can be incorporated through the weights $W_{ij} = W_{ji}$. $W_{ij} > 0$ indicates a soft *link* preference and $W_{ij} < 0$ indicates a soft *do-not-link* preference between x_i and x_j (details on how we select W are provided below).

Region growing initialization. Recall that we assume only one labeled point per foreground segment, and none for the background, is provided (Section 3). Since this information is not sufficient to fit our GMM model, we introduce a conservative breadth-first-search label propagation scheme. Specifically, for each foreground segment, we start from the initial labeled seed superpixel and continue merging adjacent neighboring superpixels if its descriptor vector is within a threshold γ relative to that of the seed superpixel (γ is 5% in our experiments). In each iteration we perform one merge step per segment before continuing, and we do not allow superpixels to belong to multiple segments. A sample result of such label propagation is given in Figure 3(a). All superpixels that are not claimed by region growing are considered to be part of the background at this initial stage.

Confidence Score. Each superpixel is given a confidence score $c \in [0, 1]$ to indicate the confidence that it belongs to the segment in which it was included in after the segment region growing step. This c is computed purely based on distance, so that the initial seed superpixel for the region is given score $c = 1$, and the superpixel in the segment furthest from the seed (at distance d_{max}) is given score $c = 0$. Any other superpixel in the segment at distance d from the seed is given score $c = \frac{d}{d_{\text{max}}}$. For any background region segment we consider the superpixel furthest from all non-background superpixels to be the region seed. See Figure 3(b) for a visualization.

GMM clustering constraints. For clustering we introduce soft *do-not-link* constraints between labeled regions and the background. For any two labeled region seed superpixels (say x_i, x_j), let x_k be the superpixel at the midpoint of the geodesic shortest path between x_i and x_j. We place a *do-not-link* constraint between (x_i, x_k) and (x_j, x_k). Such constraints help ensure separation between the segments during clustering. Specifically, we set $W_{ij} = K(-c_i - c_j)$ where c_i is the confidence score of x_i, and K is a constant that controls the contribution of W in the GMM. We will introduce these soft *do-not-link* constraints as described between each seed label and its geodesically n ($n = 3$ in our experiments) closest other seed labels (Figure 3(c)). We intentionally limit the number of constraints to a few fraction of the superpixels available in the shape since clustering time-complexity increases considerably with higher number of connected constraints.

With these constraints, we can fit the GMM model with EM [5] following the update rules of [16].

Cluster-to-Label Assignment. Overloading the notation we introduced in Section 3, Let $Z = \{z_i\}, i = 1, ..., N$ be label assignments for all superpixels of all models in the shape set, and $z_i \in \mathcal{L}$, where $\mathcal{L} = \{\mathcal{D} \cup \{l_b\}\}$ is the dictionary of labels including the background label ($\mathcal{D}$ given as in Section 3). Note that the fitted GMM model has more number of clusters than the number of labels ($M > |\mathcal{L}|$) since we assume a multi-component label representation. Furthermore, let this initial labeling generated by region growing be $Q = \{q_i\}, i = 1, ..., N$, where $q_i \in \mathcal{L}$. Hence, once the model fitting is completed, we can compute a label score for assigning cluster k to label l as follows:

$$\arg\max_{l \in \mathcal{L}} \sum_{i=1}^{N} s_i, \quad s_i = \begin{cases} e^{c_i} & \text{if } q_i = l \text{ and } y_i = k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Each cluster gets assigned to the label for which it scores the highest.

3.3.2 Co-segmentation as Energy Minimization

Once the we fit the constrained GMM model, we generate the final segmentation through a coupled energy minimization computed over all the models in the category. Given the superpixel graph of shape i as $G_i = V_i, E_i$, we construct a coupled graph $\mathcal{G}$ which encapsulates all individual shape graphs and additional edges across models:

$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}, \quad \mathcal{V} = \bigcup_{i=1}^{s} \{V_i\}, \quad \mathcal{E} = \left(\bigcup_{i=1}^{s} \{E_i\} \right) \cup \{\mathcal{E}_C\}, \tag{6}$$

here s is the number of models in the set, and $\mathcal{E}_C$ are the inter-model edges.

One of the novel contributions of our approach is the introduction of the inter-model edges $\mathcal{E}_C$ into the augmented graph using the structurally aware *LDD* descriptor introduced in Section 3.

We build the inter-model edges as follows:

$$\mathcal{E}_C = \bigcup_{a \in [1,n]} \; \bigcup_{b \in \{[1,n]-\{a\}\}} \; \bigcup_{x_i \in V_a} \; \bigcup_{x_j \in V_b} (x_i, x_j) \tag{7}$$
$$\forall (x_i, x_j) \quad \text{s.t.} \quad |\overrightarrow{x}_i - \overrightarrow{x}_j| \leq L_{ib} + \epsilon$$

where $\overrightarrow{x}_i$ is the LDD of superpixel x_i, L_{ib} is the minimum LDD distance between x_i to all superpixels in V_b, and ϵ is a small threshold (ϵ is the average superpixel size in 3D in our experiments).

We now define the total energy to be minimized in order to compute the joint segmentation for all models in the set:

$$\mathbf{E}(z) = \sum_{x_i \in \mathcal{V}} \mathbf{E}_{GMM}(x_i, z_i) + \lambda_S \sum_{(x_i,x_j) \in \mathcal{E}_S} \mathbf{E}_S(x_i, x_j, z_i, z_j)$$
$$+ \lambda_C \sum_{(x_i,x_j) \in \mathcal{E}_C} \mathbf{E}_C(x_i, x_j, z_i, z_j) \tag{8}$$

where z_i and z_j are the labels assigned to nodes x_i and x_j, and $\mathcal{E}_S = \{\mathcal{E} - \mathcal{E}_C\}$, and λ_S and λ_C[2] are regulating constants that control the influence of each smoothness in the total energy.

The GMM fit clustering energy is given by:

$$\mathbf{E}_{GMM}(x_i, z_i) = -log(p(z_i|x_i)) \tag{9}$$

where $p(z_i|x_i) = \sum_{y_j \in z_i} p(y_j|x_i)$. is the probability of assigning label z_i to superpixel x_i given by the constructed GMM.

Intra-model smoothing energy considers the continuity of label preferences for each model independently:

$$\mathbf{E}_S(x_i, x_j, z_i, z_j) = \begin{cases} 0 & \text{if } z_i = z_j \\ \infty & \text{if } z_i \neq z_j \wedge z_i \neq l_b \wedge z_j \neq l_b \\ -log\left(\frac{\beta_{ij}}{\pi}\right) & \text{otherwise} \end{cases} \tag{10}$$

where l_b is the background label, and β_{ij} is the dihedral angle between the superpixels x_i and x_j in 3D. Similar to previous 3D segmentation methods [22, 24, 14], our smoothing energy is favoring cuts through sharp 3D transitions on the surface of the mesh, however, we also incorporate our prior assumption (Section 3) of foreground segments not being adjacent to each other, but being separated by background.

[2] $\lambda_S = \lambda_C = 1.0$ after range normalization between energy terms for all experiments presented in this paper.

The inter-model smoothing energy is controlled by distances between LDD descriptors:

$$\mathbf{E}_C(x_i, x_j, z_i, z_j) = \begin{cases} 0 & \text{if } z_i = z_j \\ -log\left(\frac{|\overrightarrow{x}_i - \overrightarrow{x}_j|}{L_{max}}\right) & \text{otherwise} \end{cases} \tag{11}$$

where L_{max} is the maximum LDD distance between all superpixel pairs across models in the set.

Note that, while the intra-model smoothness ($\mathbf{E}_S$) improves the boundaries of segments by favoring homogeneous segment areas, inter-model edges ($\mathbf{E}_C$) favor similarly shaped and located segments across models. We solve the resulting energy minimization system with the multi-label approximate graph cut method [4]. In the final energy minimization system, the edge potentials are Potts potentials. We handle the infinite values by replacing them with practically large finite values (*i.e.* at least 6 orders of magnitude larger than any existing edge).

3.4. Generalization

Although our methodology is tailored for textured shapes, it is applicable to non-textured shapes with the following trivial changes: 1) Ignore appearance descriptors and 2) Skip superpixelization and assume that the mesh triangles are superpixels. Also, if the models to be segmented do not have any backgrounds, and we use *do-not-link* constraints ($W_{ij} = -w$, where w is a constant) between all sparse labels of each model independent of the other models. Lastly if the shape sets does not exhibit a shared part label seed in all the models, one can align the objects prior to computation, and select a vertex that is at the bottom of the objects to be utilized as the landmark label.

4. Datasets

Digital Cameras. This is a collection of 91 textured models of real digital cameras acquired from http://www.maybe3d.com. The sparse labels are provided by the modelers, and only encode the foreground segments of interest. The category consists of 37 foreground segment labels. Each label appears in at least 8 models, and on average only 19 out of the 37 labels appear on a model.

Video Recorders. Similarly to the cameras data, this is a collection of 47 textured models of real video recorders from http://www.maybe3d.com. There are 32 foreground segment labels. Each segment appears in at least 5 models, and on average only 21 out of the 32 labels appear on a single model. See the supplemental material for more details.

Note that both the video recorder and digital camera data shows large variation in the foreground segments (Figures 8 - 9, left column). There are very small segments such as buttons and connection ports alongside segments such as LCD screens that may cover up to 25% of the surface.

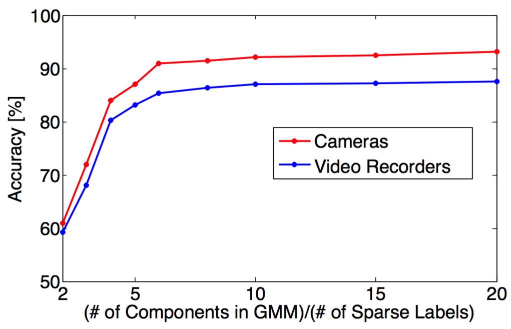

Figure 4. Accuracy of our method on two datasets vs. the number of components used in GMM normalized by the number of sparse labels in their dictionaries.

Table 1. Performance of different labeling schemes.

	Region Growing	w/o $\mathcal{E}_C$	w/ $\mathcal{E}_C$
Cameras	51.3%	82.3%	91.5%
V. Recorders	48.1%	77.2%	86.3%

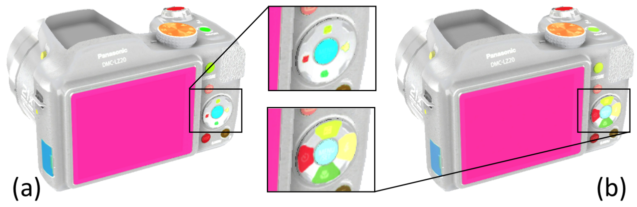

Figure 5. (a) Without using the inter-model energy. (*i.e*. Independent energy minimization). (b) With using inter-model energy.

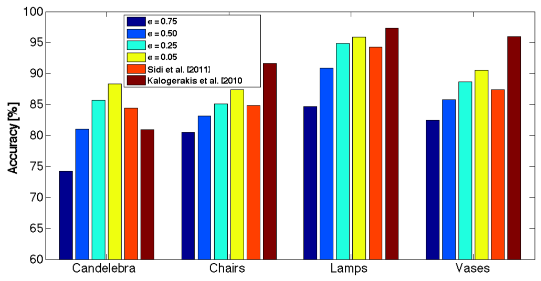

Figure 6. Comparison of our co-segmentation performance with Kalogerakis *et al*. [14] and Sidi *et al*. [24] with datasets in [24].

We have collected a full ground truth labeling of 4 cameras and 3 video recorders (Figures. 8 - 9, right column) performed by a 3D modeling artist to quantitatively evaluate a portion of our co-segmentation results.

For cameras and video recorders, the presence of texture maps present new challenges relative to existing datasets: while the texture brings in an additional cue which may translate to better segmentation, it also brings in a new dimension of variability across models in the data set.

Untextured datasets. For comparison with published results of existing segmentation methods, we present our results on synthetic untextured models of *Candelabras*, *Chairs*, *Lamps*, and *Vases*[3], which consist of 28, 20, 20, and 28 models, respectively.

4.1. Experiments and Analysis

To measure accuracy, we apply the metric of [24] to the superpixel graph, so that we define accuracy as the segmentation overlap of the predicted foreground pixels with the ground truth segmentation and labeling: $Acc = \frac{\sum_i a_i \delta(z_i = t_i \wedge z_i \neq l_b \wedge t_i \neq l_b)}{\sum_i a_i \delta(z_i \neq l_b \wedge t_i \neq l_b)}$, where a_i, z_i, t_i are the superpixel area, co-segmentation resulted label, and ground truth label respectively for superpixel i.

Cameras and Video Recorders. For each category we perform co-segmentation over the entire dataset, and evaluate the results on the subset of models for which we have full ground truth segmentation (Figures 8 - 9). Mean processing time for for both datasets less than 10 minutes on a workstation with an 8-core 2.0GHz processor bank.

An important parameter in our co-segmentation procedure is the number components used in the GMM. We empirically show, for both datasets, accuracy saturates when the number of components is roughly 8 times the size of the label dictionary (Figure 4). Figure 5 shows visually the benefit of incorporating the inter-model energy into our co-segmentation formulation.

Untextured 3D models. Since the untextured models do not contain sparse labeling input, we generate a sparse labeling by randomly selecting a single point from each foreground segment based on the distribution of point-to-boundary distances. We use a parameter $\alpha \in (0, 1]$ in our sampling (see the supplemental material for details). Essentially, for small α we bias the random point selection towards areas far from segment boundaries. We show results for $\alpha = \{0.05, 0.25, 0.50, 0.75\}$ and for each value we average accuracy over 10 trials of sampling.

In Figure 6, we compare our method with a supervised [14] and an unsupervised method [24]. As expected, our algorithm outperforms the unsupervised method, with the prior information (one vertex per segment), even with a con-

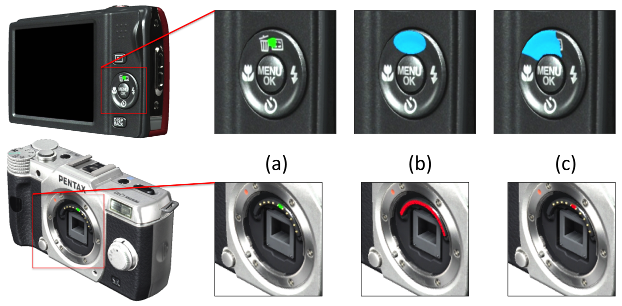

Figure 7. Failure: (a) sparse label, (b) ground truth, (c) our result.

[3]Shapes are available at: http://web.siat.ac.cn/ yunhai/ssl/ssd.htm

siderable variation in the position of the initially given label. However, when the given sparse annotation's centrality is very skewed (*e.g.* $\alpha = 0.75$), this misleads the algorithm and performance decreases.

Figure 7 shows two instances where our algorithm fails. In the top row, there is no geometric cue for the algorithm to find a fine boundary for the button, where as in the bottom row, a sparse pattern is missed by the algorithm.

5. Conclusion

We presented a novel co-segmentation method for textured 3D shapes. Our method leverages an assumption of sparse foreground labeling, and creates a co-segmentation of a set of objects that belong to the same category. Requiring only a sparse input labeling lends our algorithm to real-world segmentation tasks where data collection of fully-segmented training models may be impractical.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels. *École Polytechnique Fédéral de Lausssanne (EPFL), Tech. Rep*, 149300, 2010. 3

[2] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. icoseg: Interactive co-segmentation with intelligent scribble guidance. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 1

[3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, Apr. 2002. 3

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001. 5

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. of the Royal Statistical Society. Series B*, pages 1–38, 1977. 4

[6] C.-S. Dong and G.-Z. Wang. Curvatures estimation on triangular mesh. *Journal of Zhejiang Uni. Science*, 2005. 2

[7] A. Golovinskiy and T. Funkhouser. Consistent segmentation of 3d models. *Computers & Graphics*, 33(3), 2009. 2

[8] P. Heider, A. Pierre-Pierre, R. Li, and C. Grimm. Local shape descriptors, a survey and evaluation. In *Eurographics Conference on 3D Object Retrieval*, pages 49–56, 2011. 2

[9] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Computer Graphics and Interactive Techniques*, pages 203–212. ACM, 2001. 3

[10] Q. Huang, V. Koltun, and L. Guibas. Joint shape segmentation with linear programming. In *ACM Transactions on Graphics*, volume 30, 2011. 1, 2

[11] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(5):433–449, 1999. 3

[12] A. Joulin, F. Bach, and J. Ponce. Discriminative clustering for image co-segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 1

[13] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun. A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics*, 31(4):55, 2012. 1

[14] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3D Mesh Segmentation and Labeling. *ACM Transactions on Graphics*, 29(3), 2010. 1, 2, 3, 5, 6

[15] A. Kowdle, D. Batra, W.-C. Chen, and T. Chen. imodel: interactive co-segmentation for object of interest 3d modeling. In *ECCV*, pages 211–224, 2012. 1

[16] Z. Lu and T. K. Leen. Penalized probabilistic clustering. *Neural Computation*, 19(6):1528–1567, 2007. 3, 4

[17] N. J. Mitra, M. Wand, H. Zhang, D. Cohen-Or, and M. Bokeloh. Structure-aware shape processing. In *Eurographics 2013-State of the Art Reports*, pages 175–197. The Eurographics Association, 2013. 1

[18] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH*, pages 309–314, 2004. 1

[19] C. Rother, T. Minka, A. Blake, and V. Kolmogorov. Cosegmentation of image pairs by histogram matching - incorporating a global constraint into mrfs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006. 1

[20] T. Saito and T. Takahashi. Comprehensible rendering of 3-d shapes. In *ACM SIGGRAPH*, volume 24, 1990. 3

[21] A. Shamir. A survey on mesh segmentation techniques. In *Computer graphics forum*, volume 27, pages 1539–1556. Wiley Online Library, 2008. 2

[22] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang. Contextual part analogies in 3d objects. *Int. J. of Computer Vision*, 89(3):309–326, 2010. 5

[23] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Int. J. of Computer Vision*, 81(1), 2009. 3

[24] 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. In *ACM Transactions on Graphics*, volume 30, 2011. 1, 2, 5, 6

[25] O. Van Kaick, A. Tagliasacchi, O. Sidi, H. Zhang, D. Cohen-Or, L. Wolf, and G. Hamarneh. Prior knowledge for part correspondence. In *Computer Graphics Forum*, volume 30, pages 553–562. Wiley Online Library, 2011. 1, 2

[26] O. van Kaick, K. Xu, H. Zhang, Y. Wang, S. Sun, A. Shamir, and D. Cohen-Or. Co-hierarchical analysis of shape structures. *ACM Transactions on Graphics*, 32(4), 2013. 1

[27] S. Vicente, C. Rother, and V. Kolmogorov. Object cosegmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2217–2224, 2011. 1

[28] Y. Wang, S. Asafi, O. van Kaick, H. Zhang, D. Cohen-Or, and B. Chen. Active co-analysis of a set of shapes. *ACM Transactions on Graphics*, 31(6), 2012. 1, 2

[29] K. Xu, H. Zhang, D. Cohen-Or, and B. Chen. Fit and diverse: Set evolution for inspiring 3d shape galleries. *ACM Transactions on Graphics*, 31(4):57, 2012. 1

[30] Y. Zheng, D. Cohen-Or, and N. J. Mitra. Smart variations: Functional substructures for part compatibility. In *Computer Graphics Forum*, volume 32, pages 195–204, 2013. 1

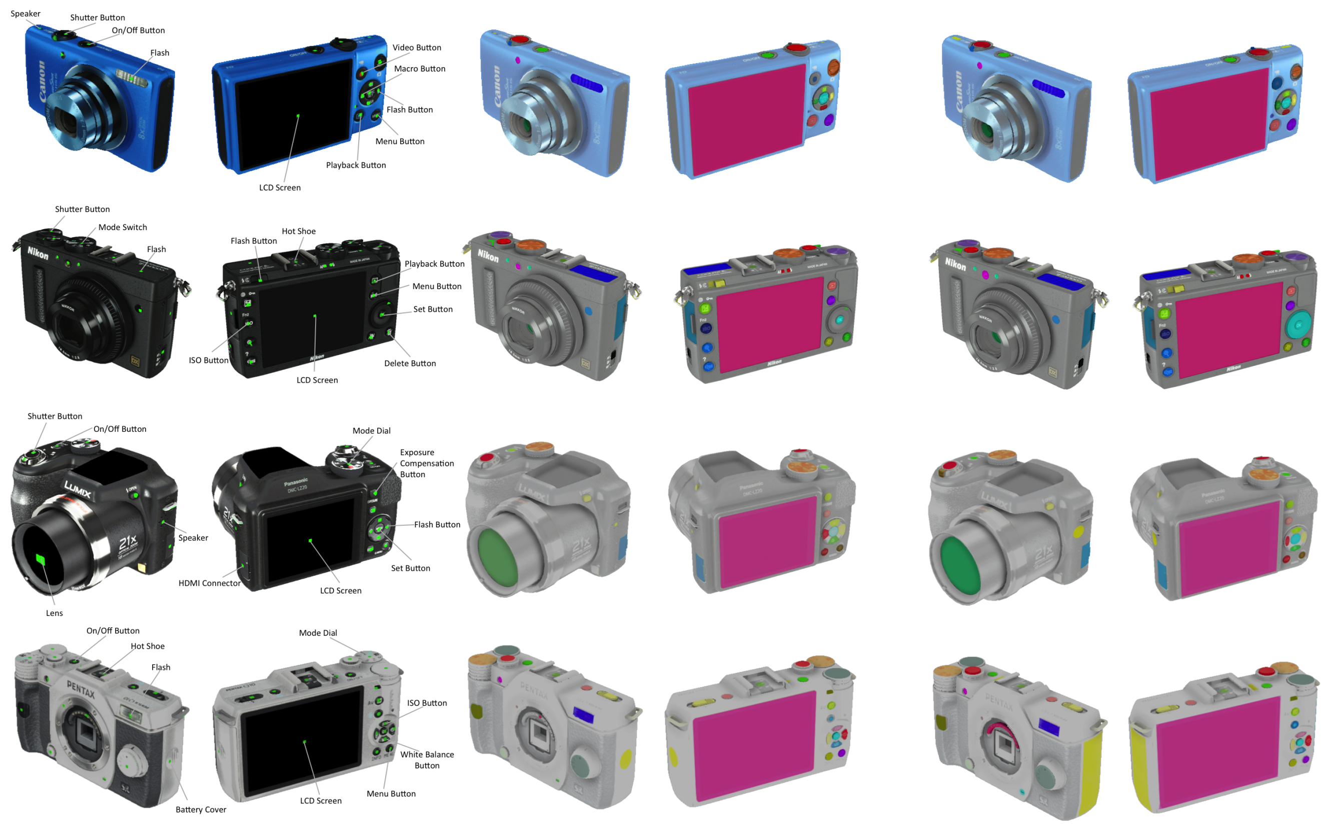

Figure 8. Left: Models with sparse labels (sparse labels are given as bright green, not all labels are annotated for visual clarity). Middle: Our co-segmentation result (generated using all models in the cameras dataset). Right: Ground-truth generated by a 3D artist. *Color overlays are the foreground segments (same colors in different shapes represent same labels), white overlay is the background.*

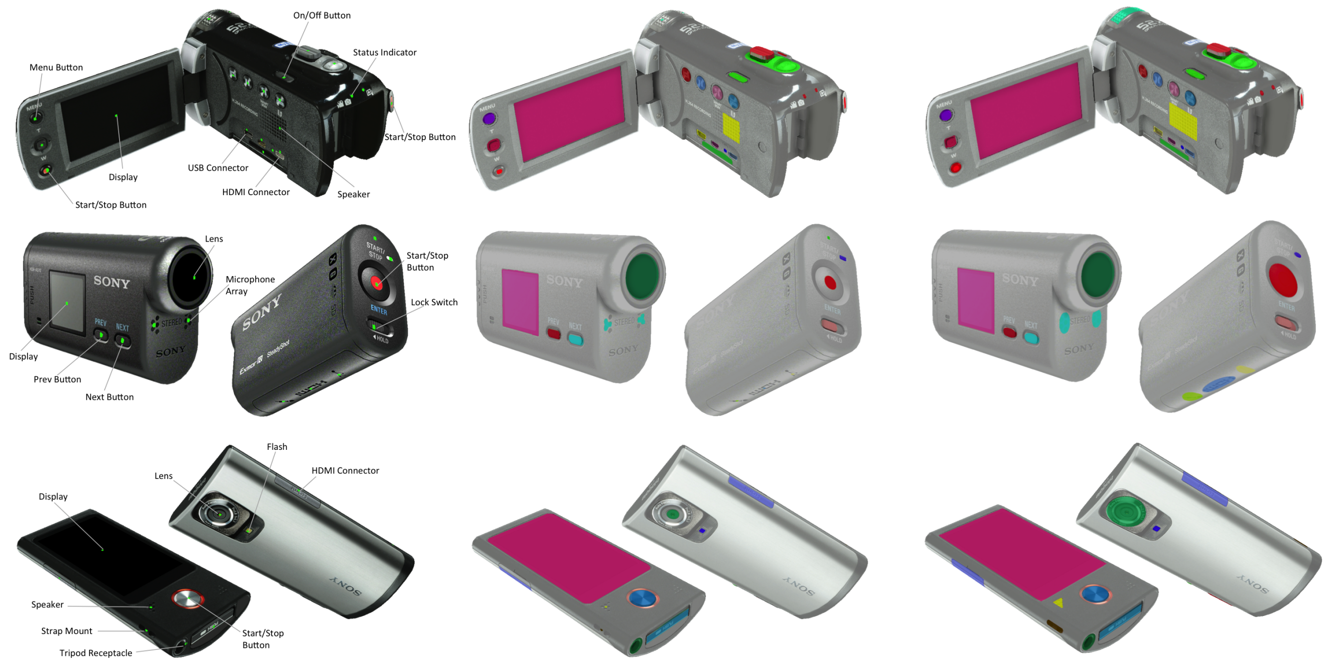

Figure 9. Left: Models with sparse labels (sparse labels are given as bright green, not all labels are annotated for visual clarity). Middle: Our co-segmentation result (generated using all models in the video recorders dataset). Right: Ground-truth generated by a 3D artist. *Color overlays are the foreground segments (same colors in different shapes represent same labels), white overlay is the background.*