Received: 27 December 2017 | Revised: 27 November 2018 | Accepted: 29 November 2018

DOI: 10.1002/cav.1869

RESEARCH ARTICLE

WILEY

Building hierarchical structures for 3D scenes based on normalized cut

Xi Zhao | Zhenqiang Su | Xinyu Yang

School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China

Correspondence
Xinyu Yang, School of Electronic and Information Engineering, Xi'an Jiaotong University, 28 West Xianning Road, Xi'an 710049, China.
Email: xyyang@mail.xjtu.edu.cn

Funding information
China Postdoctoral Science Foundation, Grant/Award Number: 2015M582664; National Natural Science Foundation of China, Grant/Award Number: 61602366

Abstract

The growing number of 3D scene data available online brings in new challenges for scene retrieval, understanding, and synthesis. Traditional shape processing methods have difficulty to manage 3D scenes because such methods ignore the contextual information, that is, the spatial relationship between the objects or groups of objects, which plays a significant role in describing scenes. Therefore, a context-aware representation is needed to deal with such a problem. In this paper, we propose a method to build scene hierarchies based on contextual information. Given a 3D scene, we first use the interaction bisector surface to measure the affinity between different objects/elements of the scene and then apply the normalized cut method to build a hierarchical structure for the whole scene. The resulting hierarchical structure contains not only the relationship between the individual objects but also the relationship between object groups, which provides much richer information of the scene compared with a flat structure that only describes the contacts or affinity between the individual objects. We test our method using several public databases and show that the resulting structure is more consistent with the ground truth. We also show that our method can be used for point cloud segmentation and outperforms previous methods.

KEYWORDS

hierarchical structure, normalized cut, scene analysis, segmentation

1 | INTRODUCTION

Recently, an increasing number of 3D scene databases have been available online because of the popularity of depth sensors and user-friendly modeling software. The ability to reuse such databases is critical for many content-hungry applications such as virtual worlds and games. To understand and retrieve existing 3D scene data, or synthesize new scenes based on existing ones, an essential problem is how to encode the scene information with a structural representation, which is compact, organized, and easy to use.

Representing 3D scenes effectively is not a trivial task. The 3D scenes can be in a different format, such as depth data, point cloud data,[1] and artificial 3D models.[2] No matter what the format is, the representation should be able to encode not only the geometry of each object but also, more importantly, the contextual information of the scene. The spatial relationship between objects or object groups plays a significant role in describing the scene context. It describes how one object locates relative to another object, that is, A is on top of B, or A hooks on B, and how one object group locates relative to other groups, that is, group A is beside group B, or groups A and B are quite close while group C is a bit far away.

Existing methods use either a flat graph or a hierarchical/tree structure to represent 3D scenes. A flat graph usually consists of nodes representing single objects and edges representing relations between objects. It has been used for sketch-based scene modeling,[3] context-based object retrieval,[4] scene retrieval,[5] etc. However, the flat graph requires further processing such as segmentation and feature extractions because the flat graph treats each scene elements equally and is lacking of explicit representation of groups or local regions. Tree structures, which are usually built by an agglomerative or divisive method, can represent the subgroups of a scene more explicitly because the branches of the tree naturally represent different groups of objects. The existing methods for building a tree structure for 3D scenes can be divided into two types: learning-based methods, such as that in the work of Liu et al.,[6] and analytic methods, such as those in the works of Zhao et al.,[7] Hu et al.,[8] and Xu et al.[9] Among the analytic methods, we are more interested in the work of Zhao et al.,[7] which can deal with complex relations and is not restricted to small-scale scenes. However, although this method can build a tree structure that considers the spatial relationship for both object pairs and object groups, the speed of merge of scene elements is controlled by a merge parameter, which is hard to optimize for different datasets. With a too small merge parameter, this method may get unbalanced results because it does not consider the global structure of the scene when merging scene elements.

In this paper, we propose an approach for building tree structures for scenes with an improved merge strategy. Our method contains two main steps. After measuring the affinity between all pairs of objects in the scene based on the interaction bisector surface (IBS[7]), we merge objects iteratively based on a normalized cut (Ncut) criterion. In other words, for each merge, our method chooses the pair that leads to the minimum Ncut value. We present two ways to apply the Ncut method to different types of data.

There are two main contributions of our work. First, our approach presents an improved definition of IBS-based affinity measurement between elements in a scene. It can be used to compute more general and precise weights on IBS ridges. Second, by using the normalized cut criterion, our method can avoid the unbalance merge and guarantee a global optimal merge of scene elements in each iteration. Our method is general in the sense that the method for evaluating the affinity can be applied to different types of geometry elements, such as points of an object, object subparts, single object, or a group of objects. We show that the proposed method can be naturally used for segmentation of point clouds, as each level of the hierarchy is a segmentation of the scene. Furthermore, by measuring the approximation between scene elements, our method can convert a flat scene graph to a hierarchical structure so that it can be used as a supplement to other scene parsing systems.

2 | RELATED WORK

Our work is closely related to the research on building the structure of 3D objects. Structure analysis is the key feature of the state-of-art techniques of 3D object processing.[10] Analyzing the structure enables many smart applications such as shape deformation,[11] shape synthesis,[12–16] and reconstructions.[17] Among these works, many rely heavily on a hierarchical structure, which is generally believed to be more structure revealing. Wang et al.[18] build a hierarchical structure based on symmetry analysis. Jain et al.[19] blend two shapes by matching two hierarchical structures. In the work of van Kaick et al.,[20] the hierarchical structure is built for a set of shapes of the same class by co-analysis. Li et al.[16] produce the object structure first and generate the model part individually by the deep learning method.

Building structures for 3D scenes is very important for many applications such as context-based analysis and retrieval,[5,7–9,21] scene parsing,[6,22] and scene synthesis.[23] Flat graphs are a commonly used structure representation, where the nodes of which normally represent single objects and the edges represent the relationship between objects. Although the flat graph is commonly used, a hierarchical structure is proven to be more structure revealing and informative for scene parsing,[6] functional analysis,[8,20] and scene decoration.[24]

Measuring the relationship between scene elements is the first step in building scene hierarchy. While the relationship between elements in a scene can be both semantic and geometric, we focus on the geometric measurement here. Different from measuring the relationship between subparts of objects, which is mainly about contacts and symmetry, objects of a scene are apart from each other for most cases. To compare scenes, Paraboschi et al.[25] use the distance between the barycenters, height distance, and geodesic distance to encode the relationship of adjacent objects. Yu et al.[26] use not only distance information but also the orientation and ergonomic measures between furniture. Fisher et al.[27] use relative vectors to represent the spatial relationships. In the work of Liu et al.,[6] the minimum distance between points on different objects, the distance between bounding box centers, and the ratio between bounding box areas are used as the relative layout descriptor of objects. To quantitatively represent the spatial relationship, especially a complex relationship,

Zhao et al.[7] and Hu et al.[8] use the IBS and its features. In this paper, we also use IBS to measure the affinity between scene elements.

Similar to image segmentation, given the set of lowest level scene elements, and the relationship measurement between them, a hierarchical tree structure can be built by agglomerative or divisive algorithms. Zhao et al.[7] use agglomerative hierarchical clustering to merge from the fine details of the scene elements. For each iteration, elements of the scenes are merged if the affinity measurements between the elements are smaller than a threshold. Although this method is straightforward and fast to compute, it tends to merge all elements too fast. The threshold has to be adjusted to a proper value to control the merge speed. To solve this problem, a global optimization method, such as minimum cut (Mincut[28]) and Ncut[29] criteria can be used during the merge. To avoid imbalanced results, we choose to use the Ncut criterion in our method, which considers both the intergroup and the intragroup relationships. Although the Ncut method suffers from high computation cost and is sensitive to noise for the image segmentation problem when building scene hierarchy, we avoid such problems by an exhaustive search because the number of elements of a 3D scene is usually relatively small comparing to an image.

3 | OUR METHOD

Our goal is to build a hierarchical structure of 3D scenes. To achieve this goal, we first represent the input 3D scene as a graph $G = (V, E)$, where V is a set of scene elements and E is a set of edges that connect the elements. We assign a *weight* to each edge, which represents the affinity between two elements. Given a scene with n elements, we define a *weight matrix* with the size of $n \times n$, where each item is the edge weight. If there is no edge between two elements, then the weight is set to zero. With the weight matrix, we then cut the graph iteratively into binary partitions to build the hierarchy. More specifically, we apply the Ncut criterion to the weight matrix to find the optimal way to cut the graph. There are two fundamental problems to solve during this process. First is how to measure the affinity between objects or groups of objects. Second is how to apply the Ncut criterion to build the hierarchy. Next, we provide the details of our method for these two steps.

3.1 | Compute affinity between scene elements

We use IBS[7] for computing the affinity measure between scene elements and use them to build the weight matrix. IBS is a set of points that are equidistant to at least two elements of the scene. It is designed to compute the similarity of scenes based on the spatial relationships between the objects composing scenes.[7] The IBS of 3D data is a mesh composed of polygons (named *ridges*). Each ridge is equidistant to at least two samples of the scene data. There are two main advantages when using IBS to measure the affinity between objects. On the one hand, IBS can represent both simple and complex relations, whereas the traditional methods based on bounding boxes or contacts have difficulty to deal with complex relations. On the other hand, the geometry and topology of IBS are robust to small defects of scene models, such as small holes, bumpy surfaces, and triangle soups.

We use two features of IBS proposed in the work of Zhao et al.,[7] that is, the distance and the direction angle, to compute the weight matrix defined in the previous section. The features are defined as follows (see Figure 1): A ridge T on the IBS is equidistant to sample points p_1 and p_2 on objects1 and object2, respectively. Let us define a vector $\vec{v}$, which connects a point t on T to p_1, and define $\vec{n}$ as the normal vector of T. Then, distance d is the length of $\vec{v}$, and α is the angle between $\vec{n}$ and $\vec{v}$. The distance d and direction angle α define the relative location between t (from IBS) to points p_1 and p_2 (from objects). These two features are independent to each other. The IBS point t with large α may have very large or small d and vice versa.

The weight of the current ridge T, which is the affinity between p_1 and p_2, is computed by the surface integral of weights $w(t)$ over the ridge area:

$$W_T = \int w(t)ds \tag{1}$$

$$w(t) = w_{\mathbf{distance}}(t)^n \times w_{\mathbf{angle}}(t), \tag{2}$$

where $w_{\mathbf{angle}}$ and $w_{\mathbf{distance}}$ are defined as follows:

$$w_{\mathbf{angle}}(t) = 1 - \frac{\alpha}{\pi/2} \tag{3}$$

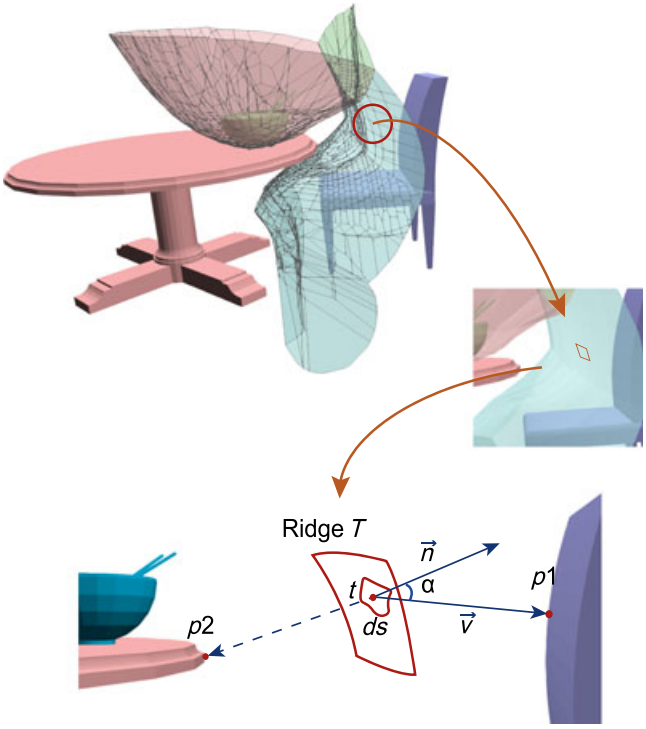

FIGURE 1 The distance and the direction angle of ridge T

$$w_{\mathbf{distance}}(t) = 1 - \frac{d}{D}. \tag{4}$$

$D = d_{\mathbf{diag}}/2$, where $d_{\mathbf{diag}}$ is the length of the diagonal of the bounding box of the whole scene. D is the upper bound of d. As $w_{\mathbf{angle}}$ and $w_{\mathbf{distance}}$ are in different range, n is used for adjusting the influence between the two parts. n is empirically set to 55 based on the ground truth hierarchies provided in the Princeton Database.

We approximate the surface integral over the ridge T in Equation (1):

$$W'_T = \sum w(t')\Delta s, \tag{5}$$

where t' is a random point on patch s on ridge T. When $\max(\Delta s)$ is small enough, W'_T is a good approximation of W_T. Then, the weight between object i and object j is computed by the sum of the weights of the ridges that belong to IBS(i,j):

$$W(i,j) = \sum_{T \in \mathrm{IBS}(i,j)} W'_T, \tag{6}$$

where IBS(i,j) is the IBS between object i and object j. Note that not all pairs of objects produce IBS between them. If IBS(i,j) does not exist, $W(i,j)$ is set to 0.

By computing the affinity value between all pairs of scene elements, we build the weight matrix based on which the graph partition algorithm can be applied. In Figure 2, we show the weights computed from a simple example scene and visualize the weights of the ridges. From the figure, we can see that $W(i,j)$ tends to be large when the objects contact with each other.

3.2 | Use Ncut to build the hierarchy

To avoid the problem of the method,[7] which tends to merge scene elements too fast, we merge two parts of the scene at each time. An example of a too fast merge is shown in Figure 3. In this example scene hierarchy, which is produced by the method of Zhao et al.,[7] all the objects in the scene are merged into one group within one step. Instead, the result of our method will be a binary structure, which has been used in many methods.[6,8]

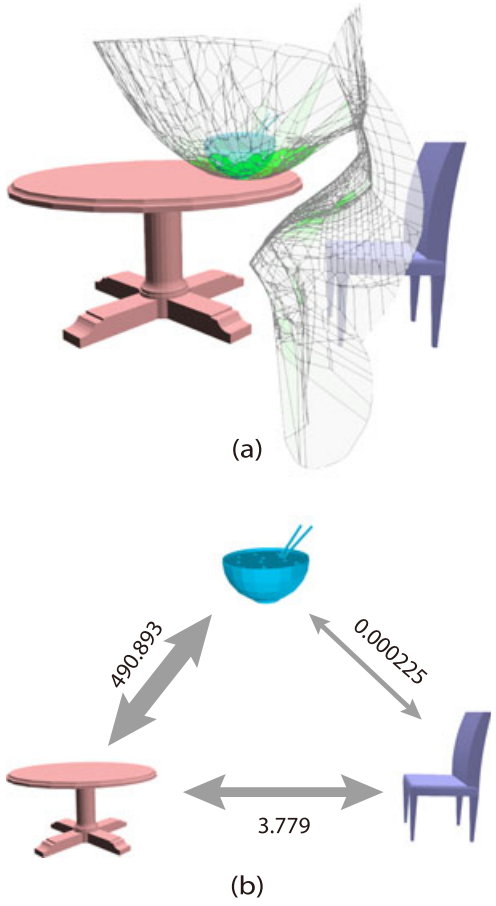

FIGURE 2 (a) Visualization of weights on the interaction bisector surface (the transparent mesh). Dark green represents larger values, and light green represents smaller values. (b) The graph for the example scene. The values are the actual weights of the edges

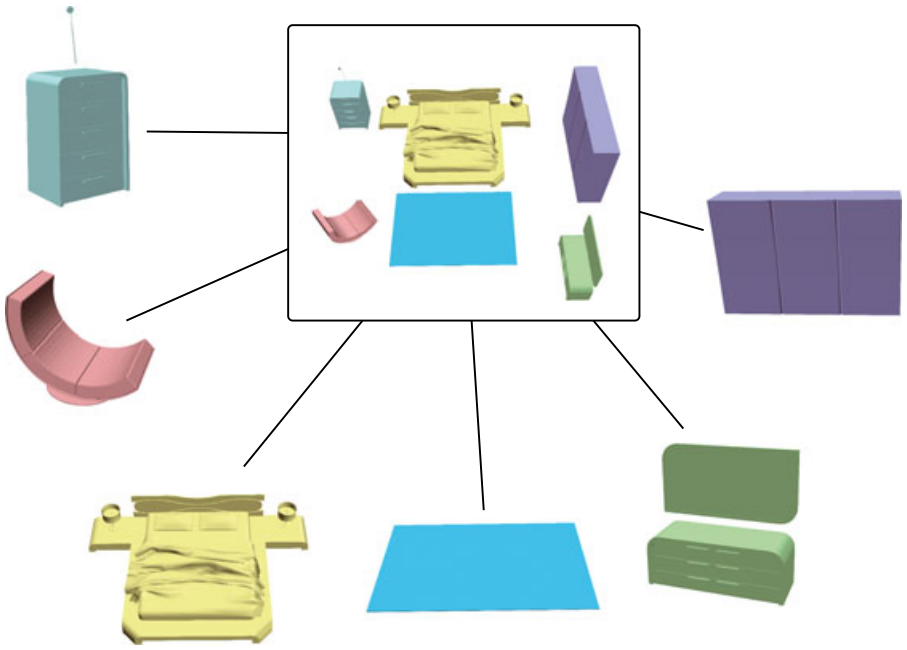

FIGURE 3 All the objects in this scene will be merged to one group within one step by the method in the work of Zhao et al.[7]

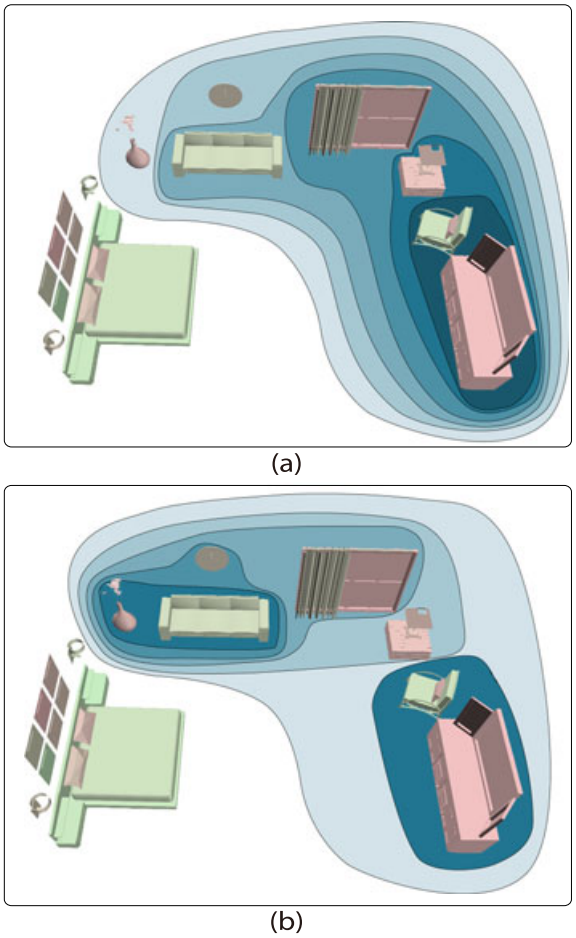

FIGURE 4 The merge order by (a) minimum cut and (b) Ncut criteria for an example scene. The dark to pale blue circles represent the groups created during the merge. The darker circles represent the earlier merge

The simplest way to adapt the method in the work of Zhao et al.[7] to binary merge is applying a constraint to merge two parts each time. According to the merge criterion, the pair of objects that share the largest weights is merged. This strategy is the same as the Mincut criterion. We tested this method, but we found that it tends to produce unbalanced results, as shown in Figure 4a. The greenish and reddish colored shapes are the scene objects, and the dark blue to pale blue circles represent the groups created during the merge. Darker circles represent earlier merges. We can see that, in Figure 4a, the vase, clock, and sofa on the left corner have never been merged into a group, although they are semantically and geometrically a good group of objects. To avoid unbalanced results, we choose to use the Ncut criterion.[29] Two types of methods can be used for applying the Ncut criterion.

When the number of elements is small (less than 100 according to the work of Shi et al.[29]), the "greedy pruning" method can be used. This method iteratively merges two subgroups of the scene at a time until the whole scene becomes one group. At each merge step, we compute an Ncut criterion for all pairs of subgroups and merge two subgroups if their merge leads to the minimum Ncut value. The Ncut criterion when merging elements i and j of the scene is defined as[29]

$$\mathrm{Ncut}_k(i,j) = \frac{\mathrm{cut}(g', V - g')}{\mathrm{assoc}(g', V)} + \sum_{m \neq i,j} \frac{\mathrm{cut}(g_m, V - g_m)}{\mathrm{assoc}(g_m, V)}, \tag{7}$$

where i and j are the index of the current pair of scene elements/groups, and g' is the new group after merging group i and group j: $g' = g_i \cup g_j$. By doing this, the groups in the scene gradually merged into one. This process provides us with a hierarchical structure. In our experiment, we use this greedy strategy for the data from the Stanford Scene Database and the Princeton Scene Database.

When the scene contains a large number of elements, the exhaustive search of group pairs that can produce minimum Ncut can be very slow. We use the approximation method based on the eigenvalue system proposed in the work of Shi et al.[29] to compute the optimized Ncut. After computing the Laplacian matrix of the scene graph based on the weight matrix computed in the last section, we compute the eigenvalues and eigenvectors of the Laplacian matrix and, then, use the eigenvector with the second smallest eigenvalue to bipartition the graph. We partition the graph recursively until reaching the individual scene elements. When applying our method for point cloud segmentation, we use this method because the number of scene elements is around 300.

For both conditions, we find that it is reasonable to merge scene elements that contact with each other earlier as the relationship between them is quite strong. Therefore, before applying the iterative Ncut algorithm, we detect all the contacts between objects and merge those that share contact relations. More specifically, we identify contacts between objects by computing the minimum distance between points on them. If the minimum distance is smaller than a threshold, the two objects are considered as contact with each other.

The pseudocode of the whole algorithm is shown as Algorithm 1. An example result of our method can be seen in Figure 4b. We can see that, by using the Ncut criterion, the merge is more reasonable and balanced because the furniture in the left corner of the scene forms a group before merging with the group on the right side.

Algorithm 1. Hierarchy construction

Data: a scene S with n objects

Result: a hierarchy H

The first level of grouping $G^0 = \{g_1^0, g_2^0, \dots g_n^0\}$;

The element of G is represented as g_i ;

Initialize the current level $G = G^0$;

Initialize $H = G$;

compute contact matrix $\mathbf{M2}_{n\times n}$:

$$\mathbf{M2}_{i,j} \leftarrow \begin{cases} 1, & \text{if } g_i \text{ and } g_j \text{ contact with each other} \\ 0, & \text{otherwise} \end{cases}$$

while $\exists \mathbf{M2}_{i,j} = 1$, $\mathbf{object}_i \in g_{n1}$ *and* $\mathbf{object}_j \in g_{n2}$ $(n1 \neq n2)$ **do**

  merge g_{n1} and g_{n2}, and update G ;

end

$H \leftarrow H \cup G$;

while size$(G) > 1$ **do**

  $m =$ size(G) ;

  compute matrix $\mathbf{M1}_{m*m}$: $\mathbf{M1}_{i,j} \leftarrow$ NCUT(i, j);

  find NCUT$(i_{\min}, j_{\min}) = min\{\mathbf{M1}\}$;

  merge $g_{i_{min}}$ and $g_{j_{min}}$, and update G ;

  $H \leftarrow H \cup G$;

end

Return H

There are two differences between the proposed method and the method in the work of Zhao et al.[7] First, our method uses the Ncut criterion, which considers not only the intergroup relation but also the intragroup relation, whereas Zhao et al.[7] use the Mincut criterion and the proximity within the groups is ignored. The Mincut criterion tends to leave out single small groups and leads to imbalanced grouping, as shown in Figure 4a. The other difference is that our method merges one pair of objects each step, whereas the method in the work of Zhao et al.[7] merges multiple subgroups. In their method, the speed of merge is controlled by a threshold that cannot be optimized for all type of data. A slightly larger threshold tends to merge too many objects at once, whereas a smaller threshold will make the merge stop in the middle. Instead, our method has no problem like that and guarantees that each step of merge is optimized.

4 | EXPERIMENTS AND RESULTS

4.1 | Results

We test our method on the Stanford Scene Database[27] and the Princeton Bedroom Database.[6] The Stanford Scene Database contains 133 indoor scene examples including kitchen, bedroom, living room, etc. The Princeton Bedroom Database contains 76 bedroom examples. For the data from these databases, we apply the "greedy pruning" method to compute the optimal Ncut. Some of the results are shown in Figures 5 and 6.

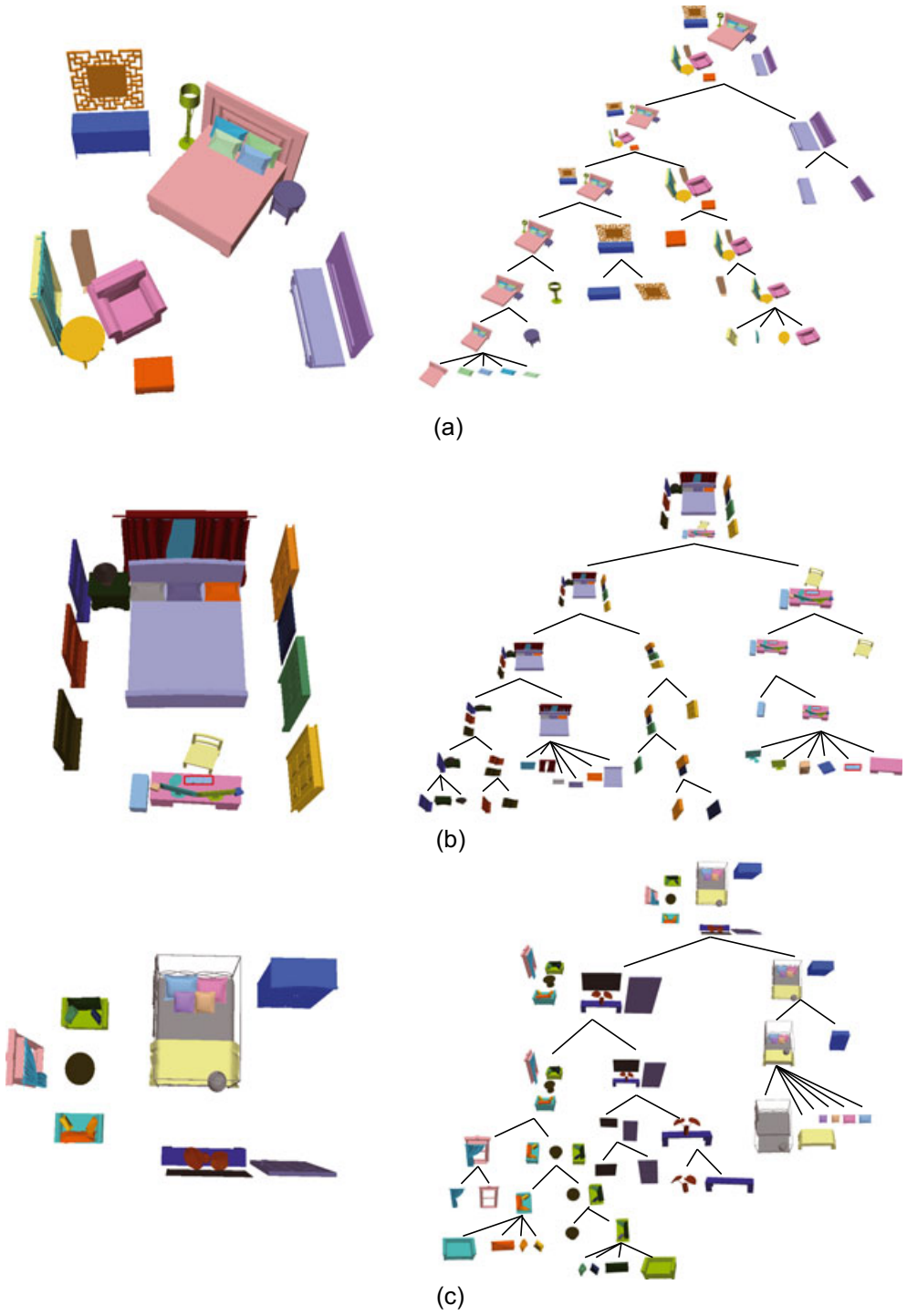

FIGURE 5 The original scenes (left) and the resulting hierarchies (right) for (a) scene 4,110, (b) scene 4,139, and (c) scene 4,919 from the Princeton Bedroom Database

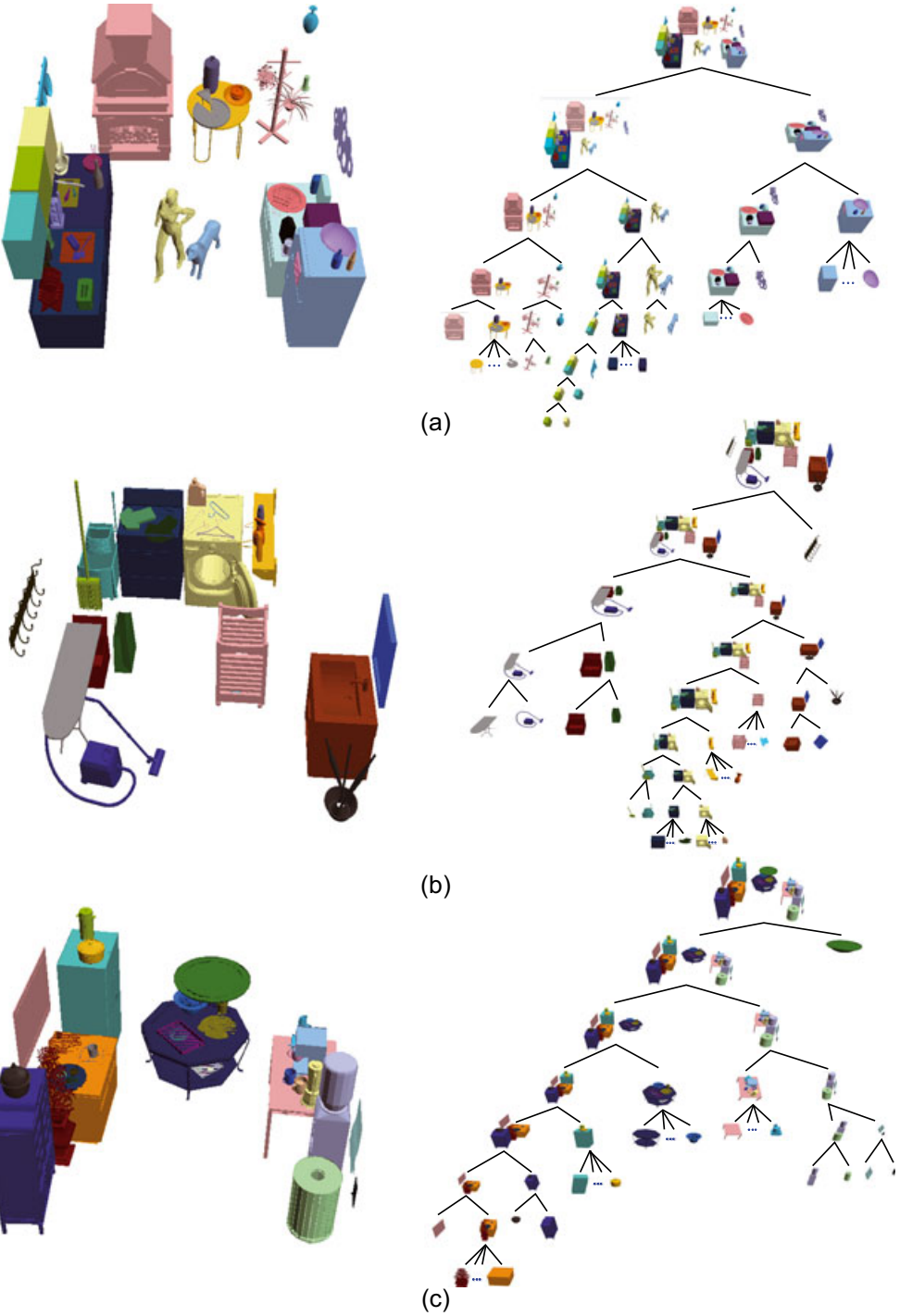

FIGURE 6 The original scenes (left) and the resulting hierarchies (right) for (a) scene 00000, (b) scene 00100, and (c) scene 00087 from the Stanford Scene Database

4.2 | Comparison and evaluation

To evaluate the hierarchy results, we use the Princeton Bedroom Database, which provides a ground truth made by users. We compare the similarity between the results and the ground truth by the method in the work of MacDonald et al.[30] The merge orders of the scene element pairs in two hierarchies are considered, and a final value γ is computed to describe the similarity. $\gamma = 1$ means the merge order is the same, whereas $\gamma = -1$ means totally different.

We compare the proposed method with following alternative methods.

- DIS. Just compute the integral of w_{distance} to build the weight matrix.

TABLE 1 The γ value of different methods

	MINCUT	NCUT
NEW MATRIX	0.5738	**0.6095**
OLD MATRIX	0.5468	0.5741
DIS	0.4232	0.4546
ANGLE	0.3741	0.5542

Note. The bold number corresponds to our method. $\gamma = 1$ means the resulting hierarchy has the same topology as the ground truth, whereas $\gamma = -1$ means totally different from the ground truth. Mincut = minimum cut; Ncut = normalized cut.

- ANGLE. Just compute the integral of w_{angle} to build the weight matrix.
- OLD-MATRIX. Use the method in the work of Zhao et al.[7] to build the matrix.
- MINCUT. Use the Mincut criterion each time the pair of elements with maximum affinity are merged.

The first three alternatives are different ways of computing the weight matrix, and the last one is an alternative to the Ncut criterion when building the hierarchy.

In Table 1, we show the average γ value of all different combinations of the above methods applied to the Princeton Bedroom Database. From the results, we can see that the proposed method (NEW MATRIX + NCUT) gets the best results. We can also see that the γ values in the "NCUT" column are higher than those of the "MINCUT" column, and the γ values in the "NEW MATRIX" row are higher than those in the "OLD MATRIX" row. This means that both "NEW MATRIX" and the "NCUT" of the proposed method improved the final results. The γ values when using only w_{distance} or w_{angle} are lower than that when using our method, which implies that it is important to consider both the distance and direction angle when computing the weight matrix.

We now analyze and break down the complexity of the computational costs when applying our approach. It is composed of two main parts: computing the affinity matrix and applying the Ncut criterion to build the hierarchy. The main operation of the first part is computing the IBS. As we used the Quickhull algorithm[31] to calculate IBS, the worst case of the time complexity is $O(m^2)$, where m is the number of samples used for computing IBS. The first part takes 1.2 s on average in our experiments. In the second part of our method, if we use the exhaustive search to compute the best cut, the time complexity is $O(n^2k)$. If we use the approximation method,[29] the time complexity is just $O(nk)$ as the weight matrix is quite sparse. n is the dimension of the weight matrix, and k is the number of times that Ncut the method is applied for building the hierarchy. In our experiments, it takes around 0.01 s on average for the second part. Our experiments are done on a computer with i7 CPU, 3.4 GHz, and 16 GB of RAM.

4.3 | Generalization of our method

Here, we describe how our method can be applied to the point cloud segmentation. Given a raw point cloud scene data, we first oversegment it by the region-growing method[32] and, then, consider each segment as single elements. We use the method described in the previous section to build a weight matrix, where the items of which are the weights that describe the affinity between pairs of segments. We then apply the Ncut method to recursively divide the segments into two parts until reaching the stop condition. Because the number of segments here is much larger than the number of objects in a common 3D scene, we use the approximation method based on an eigenvalue system proposed in the work of Shi et al.[29] to compute the optimized partition. To stop the iteration, we check if the degrees of smoothness of the values in the second smallest eigenvector is lower than a threshold. The smoothness is computed by the ratio between the minimum and maximum bin values in the eigenvector histogram.

We test the above method on the SceneNN Database[33] and RGB-D Scenes Dataset v.2.[34] We segment each scene to 382 parts on average with the smoothness threshold $\pi/18$ and curvature threshold 1.2 for oversegmentation. The eigenvector smoothness threshold for stopping the iteration is set to 0.08 in our experiment.

We compute the Rand index to evaluate the segmentation results. Our method is compared with the following alternative methods.

- FULL-DIS: The method that considers the Euclidean distance between all pairs of 3D points in the input data. This method is a simple extension of the method of Shi et al.[29] from 2D to 3D.

TABLE 2 The Rand index of different methods

	OURS	FULL-DIS	DELAUNAY	REGION-GROW
SceneNN Database	0.9021	0.877	0.777	0.8028
RGB-D Scene Dataset v.2	0.7984	0.5015	0.4204	0.5302

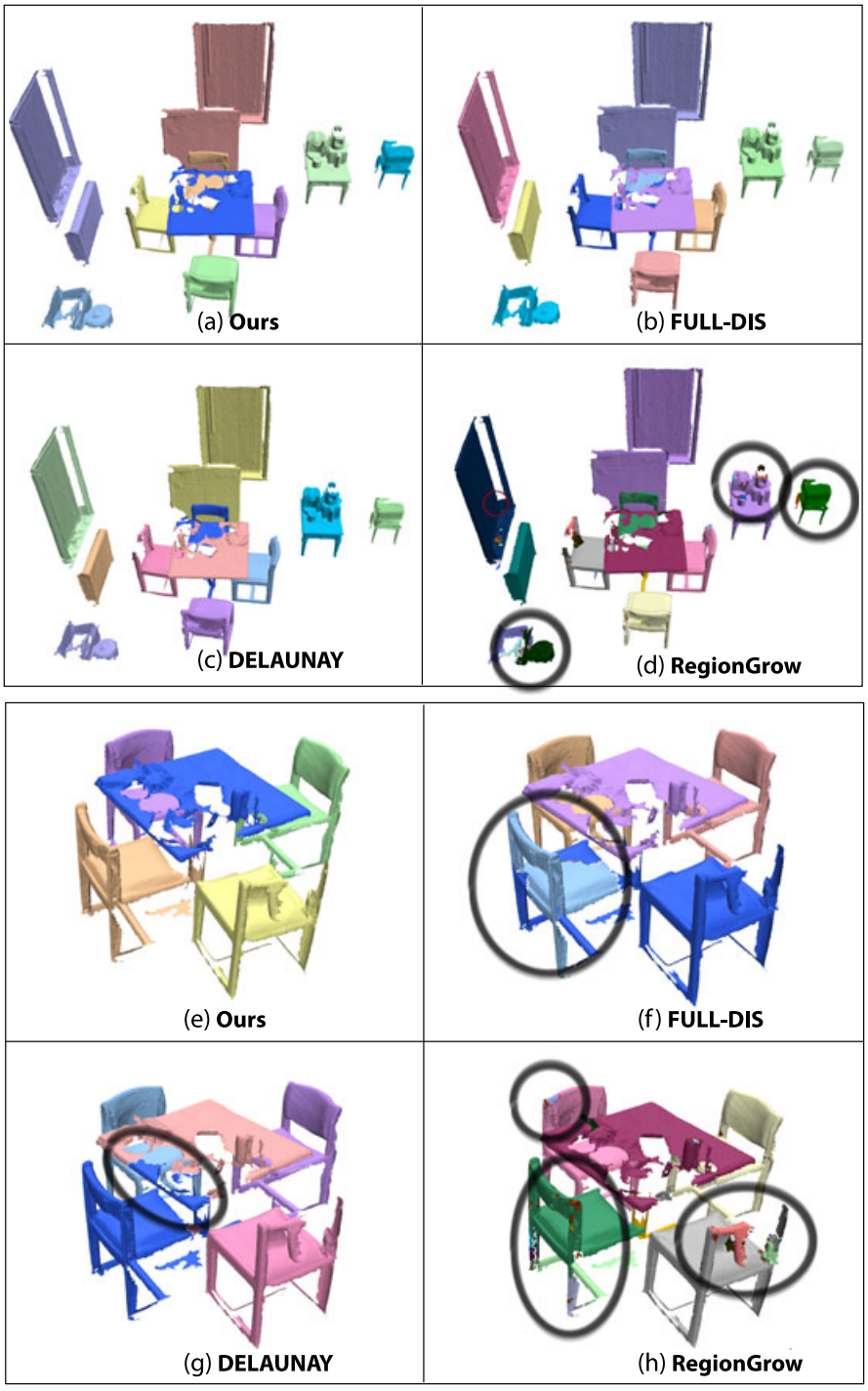

FIGURE 7 Segmentation result of scene 223 in the SceneNN Database by four methods. (a)–(d) show the segmentation results of the whole scene. (e)–(h) show the zoomed-out center area of the same scene. The wrong segmentation parts are highlighted by dark circles

- DELAUNAY: The Delaunay triangulation-based method. It is an alternative to the FULL-DIS method. Instead of using all pairs of distance, here, only the Delaunay edges of the point cloud are considered. The weight matrix by this method is much sparser compared with FULL-DIS.
- REGION-GROW: Directly use the region-growing method[32] to get the final segmentation result. Region-growing is a classic method that merges the neighborhood points to the seeds when the angle between their normal vectors is below a threshold. The smoothness threshold is set to $\pi/6$, and the curvature threshold is 2.2 in the experiments.

Table 2 lists the Rand index of all methods applied to the SceneNN Database and the RGB-D Scenes Dataset v.2. It shows that our method outperforms the alternatives. To visually compare the results, we show example results in Figure 7. From (a)–(d), we can see that, except the region-grow method, all other three methods segment most individual objects well. However, if we take a close look at the desk-chair area in the middle from a different angle (e)–(h), we can see that the

FIGURE 8 Segmentation results for scenes from the SceneNN Database. Different colors represent different segments

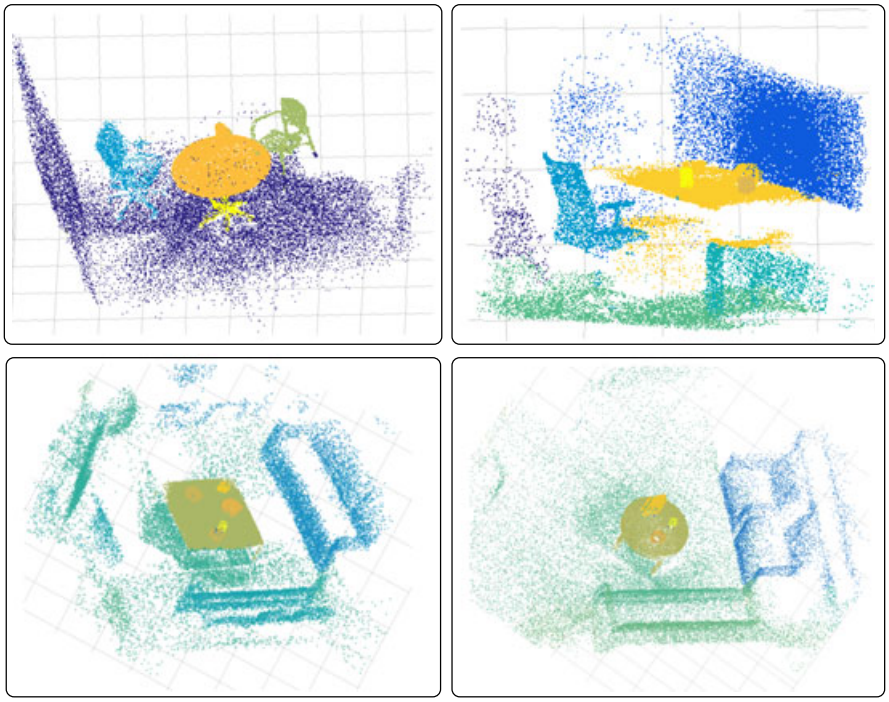

FIGURE 9 Segmentation results for scenes from the RGB-D Scenes Dataset. Different colors represent different segments

FULL-DIS method cuts the chair on the left side wrongly, the DELAUNAY method cuts one corner of the desk wrongly, and our method segments the four chairs cleanly. The region-grow method still has difficulty to avoid noise segmentation in the high-curvature area. More segmentation results of our method can be found in Figures 8 and 9.

5 | DISCUSSION AND CONCLUSION

In this paper, we introduce an improved method for building a binary hierarchical structure for 3D scenes. Our method is based on contextual information computed from the 3D geometry of the scene. It encodes the affinity between scene elements and uses the Ncut method to build the hierarchy iteratively. We demonstrate that the hierarchical structure computed by our method outperforms the previous handcrafted feature-based methods. We also show that our method can be applied to point cloud segmentation.

There are some limitations of our work. First, we do the bipartition of the scene data, which may lead to unnatural results for symmetric examples, such as a bed with two same nightstands on both sides. Second, our method is not good at dealing with incomplete point cloud data, which is quite common when the data is captured by depth sensors.

As a future work, we will first develop a method that considers the symmetry of the scenes when constructing the hierarchy. This can cope with the limitation of our current system mentioned above, where symmetric objects are joining the group one by one. Second, we can use other types of information in addition to geometric information, such as the semantic labels or user's interactive input[35] when dealing with noisy and incomplete data. Another interesting direction to explore is to develop a data-driven approach for constructing a scene hierarchy. The IBS of the ground truth scene, which is more abstract than the original scene, can be used as a low-level input feature for training. How to train a model that maps the geometry and semantics information to the hierarchy would be a problem that is worth investigating in future work.

ACKNOWLEDGEMENTS

This study was funded by the China Postdoctoral Science Foundation (2015M582664) and the National Natural Science Foundation of China (61602366).

ORCID

Xi Zhao https://orcid.org/0000-0002-3993-9870
Xinyu Yang https://orcid.org/0000-0001-5117-4914

REFERENCES

1. Firman M. RGBD datasets: past, present and future. Paper presented at: CVPR Workshop on Large Scale 3D Data: Acquisition, Modelling and Analysis; 2016 June 26–July 1; Las Vegas, NV. Piscataway, NJ: IEEE; 2016.
2. Trimble Navigation Ltd. 3D Warehouse. 2006.
3. Xu K, Chen K, Fu H, Sun W-L, Shi-Min H. Sketch2scene: sketch-based co-retrieval and co-placement of 3D models. ACM Trans Graph. 2013;32(4). Article No. 123.
4. Fisher M, Hanrahan P. Context-based search for 3D models. ACM Trans Graph. 2010;29(6). Article No. 182.
5. Fisher M, Savva M, Hanrahan P. Characterizing structural relationships in scenes using graph kernels. ACM Trans Graph. 2011;30(4). Article No. 34.
6. Liu T, Chaudhuri S, Kim VG, Huang Q, Mitra NJ, Funkhouser T. Creating consistent scene graphs using a probabilistic grammar. ACM Trans Graph. 2014;33(6):211:1–211:12.
7. Zhao X, Wang H, Komura T. Indexing 3D scenes using the interaction bisector surface. ACM Trans Graph. 2014;33(6):22:1–22:14.
8. Hu R, Zhu C, van Kaick O, Liu L, Shamir A, Zhang H. Interaction context (ICON): towards a geometric functionality descriptor. ACM Trans Graph. 2015:34.
9. Xu K, Ma R, Zhang H, et al. Organizing heterogeneous scene collections through contextual focal points. ACM Trans Graph. 2014;33(4). Article No. 35.
10. Mitra N, Wand M, Zhang H, Cohen-Or D, Kim V, Huang Q-X. Structure-aware shape processing. Paper presented at: SA '13 SIGGRAPH Asia 2013 Courses; 2013 Nov 19–22; Hong Kong. New York, NY: ACM; 2013.
11. Gal R, Sorkine O, Mitra NJ, Cohen-Or D. iWIRES: an analyze-and-edit approach to shape manipulation. Paper presented at: SIGGRAPH '09 ACM SIGGRAPH 2009 Papers; 2009 Aug 3–7; New Orleans, LA. New York, NY: ACM; 2009.
12. Kraevoy V, Julius D, Sheffer A. Shuffler: modeling with interchangeable parts. Vis Comput J. 2007.
13. Xu K, Zhang H, Cohen-Or D, Chen B. Fit and diverse: set evolution for inspiring 3D shape galleries. ACM Trans Graph. 2012;31(4):57.
14. Kalogerakis E, Chaudhuri S, Koller D, Koltun V. A probabilistic model for component-based shape synthesis. ACM Trans Graph. 2012;31(4):55.
15. Zheng Y, Cohen-Or D, Mitra NJ. Smart variations: functional substructures for part compatibility. Comput Graph Forum. 2013;32:195–204.
16. Li J, Xu K, Chaudhuri S, Yumer E, Zhang H, Leonidas G. GRASS: generative recursive autoencoders for shape structures. ACM Trans Graph. 2017;36(4):1–12. arXiv: 1705.02090.
17. Shen C-H, Hongbo F, Chen K, Hu S-M. Structure recovery by part assembly. ACM Trans Graph. 2012;31(6):180:1–180:11.
18. Wang Y, Xu K, Li J, et al. Symmetry hierarchy of man-made objects. Comput Graph Forum. 2011;30:287–296. Wiley Online Library.
19. Jain A, Thormählen T, Ritschel T, Seidel H-P. Exploring shape variations by 3D-model decomposition and part-based recombination. Comput Graph Forum. 2012;31:631–640.
20. van Kaick O, Kai X, Zhang H, et al. Co-hierarchical analysis of shape structures. ACM Trans Graph. 2013;32(4):69:1–69:10.
21. Hueting M, Monszpart A, Mellado N. MCGraph: multi-criterion representation for scene understanding. Paper presented at: SIGGRAPH Asia 2014 Indoor Scene Understanding Where Graphics Meets Vision; 2014 Dec 3–6; Shenzhen, China. New York, NY: ACM; 2014.
22. Liang Y, Fei X, Zhang S-H, Lai Y-K, Taijiang M. Knowledge graph construction with structure and parameter learning for indoor scene design. Comput Vis Media. 2018;4(2):123–137.
23. Yeh Y-T, Yang L, Watson M, Goodman ND, Hanrahan P. Synthesizing open worlds with constraints using locally annealed reversible jump MCMC. ACM Trans Graph. 2012;31(4). Article No. 56.
24. Chen K, Kun X, Yizhou Y, Wang T-Y, Shi-Min H. Magic decorator: automatic material suggestion for indoor digital scenes. ACM Trans Graph. 2015;34(6). Article No. 232.
25. Paraboschi L, Biasotti S, Falcidieno B. 3D scene comparison using topological graphs. Paper presented at: Eurographics Italian Chapter Conference; 2007; Trento, Italy.
26. Yu L-F, Yeung S-K, Tang C-K, Terzopoulos D, Chan TF, Osher SJ. Make it home: automatic optimization of furniture arrangement. ACM Trans Graph. 2011;30(4). Article No. 86.
27. Fisher M, Ritchie D, Savva M, Funkhouser T, Hanrahan P. Example-based synthesis of 3D object arrangements. ACM Trans Graph. 2012;31(6):135:1–135:11.
28. Golovinskiy A, Funkhouser T. Min-cut based segmentation of point clouds. Paper presented at: 2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops); 2009 Sep 27–Oct 4; Kyoto, Japan. Piscataway, NJ: IEEE; 2009.
29. Shi J, Malik J. Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell. 2000;22(8):888–905.
30. MacDonald D, Lang J, McAllister M. Evaluation of colour image segmentation hierarchies. Paper presented at: The 3rd Canadian Conference on Computer and Robot Vision (CRV'06); 2006 Jun 7–9; Quebec, Canada. Piscataway, NJ: IEEE; 2006.
31. Bradford Barber C, Dobkin DP, Huhdanpaa H. The Quickhull algorithm for convex hulls. ACM Trans Math Softw. 1996;22(4):469–483.

32. Rabbani T, Heuvel FVD, Vosselmann G. Segmentation of point clouds using smoothness constraint. Int Arch Photogramm Remote Sens Spatial Inf Sci. 2006;36(5):248–253.
33. Hua B-S, Pham Q-H, Nguyen DT, Tran M-K, Yu L-F, Yeung S-K. SceneNN: a scene meshes dataset with annotations. Paper presented at: 2016 Fourth International Conference on 3D Vision (3DV); 2016 Oct 25–28; Stanford, CA. Piscataway, NJ: IEEE; 2016.
34. Lai K, Bo L, Fox D. Unsupervised feature learning for 3D scene labeling. Paper presented at: IEEE International Conference on Robotics and Automation; 2014 May 31–Jun 7; Hong Kong. Piscataway, NJ: IEEE; 2014.
35. Yang S, Jie X, Chen K, Hongbo F. View suggestion for interactive segmentation of indoor scenes. Comput Vis Media. 2017;3(2):131–146.

AUTHOR BIOGRAPHIES

Xi Zhao received the BSc degree from University of Shanghai of Science and Technology in 2007, MSc degree from Southeast University in 2010 and PhD degree from Edinburgh University in 2014. Currently she is a lecturer in the Department of Computer Science and Technology, Xi'an Jiaotong University. Her main research interests include shape analysis, geometry processing, character animation and 3D vision.

Zhenqiang Su received the BSc degree in the Department of Electric Engineering from Henan University of Technology, China, in 2014. He is currently working toward the Msc (Eng) degree in the Department of software, Xi'an Jiaotong University. His research interests include image processing and 3D vision.

Xinyu Yang received the BSc, MSc, and PhD degrees from Xian Jiaotong University, in 1995, 1997, and 2001, respectively. Currently, he is a professor in the Department of Computer Science and Technology, Xi'an Jiaotong University. His main research interests lie in the areas of image processing, data mining, and network security. He received the title of New Century Excellent Talent by the Chinese Ministry of Education in 2009.

How to cite this article: Zhao X, Su Z, Yang X. Building hierarchical structures for 3D scenes based on normalized cut. *Comput Anim Virtual Worlds*. 2018;e1869. https://doi.org/10.1002/cav.1869