Exemplar-Based 3D Shape Segmentation in Point Clouds

Rongqi Qiu
University of Southern California
Los Angeles, USA
rqiu@usc.edu

Ulrich Neumann
University of Southern California
Los Angeles, USA
uneumann@usc.edu

Abstract

This paper addresses the problem of automatic 3D shape segmentation in point cloud representation. Of particular interest are segmentations of noisy real scans, which is a difficult problem in previous works. To guide segmentation of target shape, a small set of pre-segmented exemplar shapes in the same category is adopted. The main idea is to register the target shape with exemplar shapes in a piecewise rigid manner, so that pieces under the same rigid transformation are more likely to be in the same segment. To achieve this goal, an over-complete set of candidate transformations is generated in the first stage. Then, each transformation is treated as a label and an assignment is optimized over all points. The transformation labels, together with nearest-neighbor transferred segment labels, constitute final labels of target shapes. The method is not dependent on high-order features, and thus robust to noise as can be shown in the experiments on challenging datasets.

1. Introduction

Shape segmentation is an important step towards shape analysis, shape matching and scene understanding. In particular, segmentation of real-scan point clouds is very difficult due to severe data problems, such as occlusion, noise, outliers, *etc*. Most existing methods [2, 18] work on segmentation of CAD models or synthetic scans, but their performances drop rapidly in the case of real scans. One important reason is that many prior works rely on high-order local features (*e.g*., [14, 28]). These features are sensitive to noise and the robustness of these methods depends heavily on the quality and accuracy of input data.

To overcome this weakness, we present a new method which only relies on low-order information (*i.e*., geometric positions and normals). In addition, a set of pre-segmented synthetic shapes in the same category is used as examples to guide the task of segmenting real scans (Figure 1). These examples can be either manually or automatically segmented during some pre-process. This framework has the following advantages:

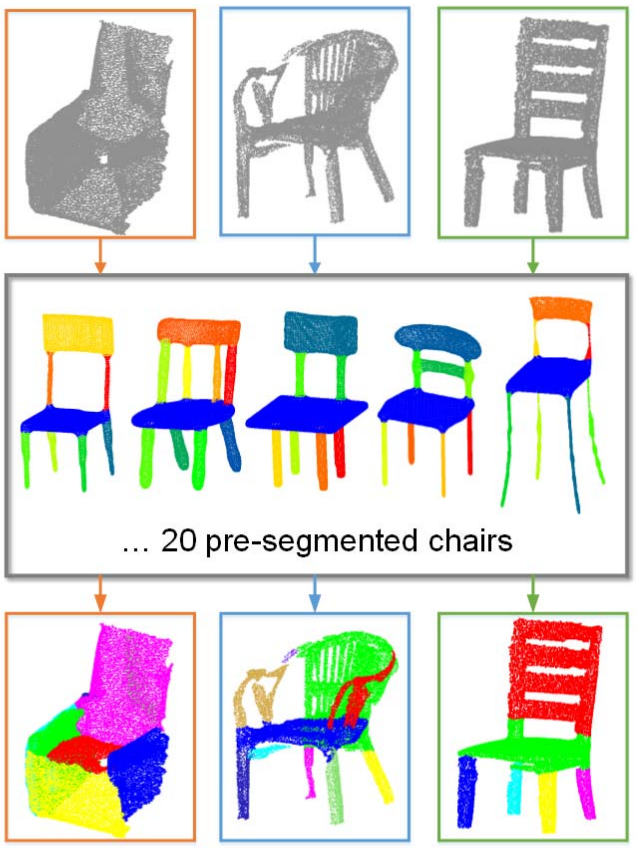

Figure 1: Noisy and incomplete real scans of chairs are segmented using 20 pre-segmented CAD models of chairs as examples. Note that each real scan is segmented individually.

1. The method is robust to noise in real scans because it does not rely on high-order features.

2. The method can segment shapes that are geometrically ambiguous by learning from similar exemplar shapes.

3. The method works with a relatively small set of exam-

978-1-5090-5407-7/16 $31.00 © 2016 IEEE
DOI 10.1109/3DV.2016.29

ples compared to typical machine learning approaches.

A key observation is that if two points of the target shape belong to the same segment, they can be aligned with an exemplar shape part under the same rigid transformation. This inspires us to register the target shape with exemplar shapes in a piece-wise rigid manner. In other words, we apply a set of rigid transformations on all target shape points to align its parts with exemplar shapes. Here it is assumed that each part of the target shape should appear at least once in exemplar shapes. This alignment provides an important clue in this segmentation task, as segment labels of the target shape can be transferred from exemplar shapes via simple nearest-neighbor matching.

This goal is achieved in three steps. First, starting with an initial segmentation of the target shape, we generate an over-complete set of candidate rigid transformations via part-wise registration. Second, we treat each transformation as a label and optimize an assignment over all points of the target shape. In particular, we seek a labeling minimizing a combination of (1) fitting error (data term), (2) boundary of different labels (smoothness term), and (3) number of appeared labels (label cost). Third, we apply post-processing to further smooth the results. The resulting segmentation can be fed into the processing pipeline again to iteratively improve the result.

2. Related Work

2.1. Shape Segmentation

Segmentation of shapes is a fundamental problem which is traditionally studied on mesh models. According to Shamir [26], segmentation methods can be roughly categorized into two types, *i.e.*, surface-type and part-type. Surface-type segmentation partitions input meshes into patches under various criteria [6], while part-type segmentation decomposes input shapes into meaningful parts [2, 18]. Our method falls into the latter category. However, these methods cannot be directly applied to point clouds, as the vertex connectivity in meshes is necessary in these methods. Meanwhile, such information cannot be robustly recovered by data-driven reconstruction [3], especially for noisy real-world scans.

Among prior works of point cloud segmentation, many addressed the problem of segmenting objects from large scenes [20, 9, 11, 22], whereas our work focus on segmenting a single object into parts. Van Kaick *et al.* [14] proposed a point cloud segmentation method based on convexity analysis, but their method cannot handle shapes with semantic parts that possess concavities.

2.2. Segmentation of Shape Collections

Besides segmenting individual shapes, research efforts are also made on segmenting shape collections. Prior works explored consistent segmentation of shape collections using spatial relations (*e.g.*, [10]) and feature relations (*e.g.*, [28]). Recently, with even larger input shape collections, many methods applied machine learning techniques, including unsupervised (*e.g.*, [13, 29, 31]), semi-supervised (*e.g.*, [32]) and supervised (*e.g.*, [15, 12]) co-segmentation or joint segmentation. However, these methods usually need a very large input dataset (hundreds or thousands), whereas our method can work with a relatively small dataset (about 20 in our experiments).

2.3. Segmentation Transfer

Another way of shape segmentation is transferring from pre-segmented shapes to a new one. This can be achieved by computing dense mapping between shapes. A large volume of prior works focused on dense correspondences of shapes (*e.g.*, [16, 17, 33, 4]). However, in real-world cases, full correspondence of shapes may not exist due to different topologies. For example, it is impossible to find full correspondence between two ladders with different number of steps (*e.g.*, Figure 3(a)(b)). Instead, researchers sought part correspondences and analogies of shapes (*e.g.*, [27, 30]. For example, Alhashim *et al.* [1] computes part correspondence of two shapes using a deformation-driven approach, but their method requires manual pre-segmentatation. Ma *et al.* [19] finds shape analogies enabling style transfer. Our method, under a similar philosophy, finds shape analogies to transfer segmentation.

3. Overview

The goal is to decompose a target shape T (in point cloud representation) into semantic parts, using pre-segmented shapes S in the same category (also in point clouds) as examples. Here S is treated as a single shape - in the case of multiple shapes, we simply concatenate them into one shape with margins between adjacent ones.

A key observation is that if two points of the target shape belong to the same segment, they can be aligned with an exemplar shape part under the same rigid transformation. Thus by aligning the target shape with exemplar shapes in a piece-wise rigid manner, we can obtain important clues on segmenting the target shape from both transformation assignment and segment labels transferred from exemplar shapes.

A basic pipeline is shown in Figure 2. Starting with pre-segmented shape S and target shape T, the first step is generating an over-complete set of candidate rigid transformations M (Section 4). The candidate set M is generated by registering all pairs of parts from S and T. Here, an initial segmentation of T is needed, which can be obtained by an automatic algorithm (*e.g.*, [14, 24]) or from the last iteration of this pipeline. Next, we assign each point of the target shape a transformation from M in a globally opti-

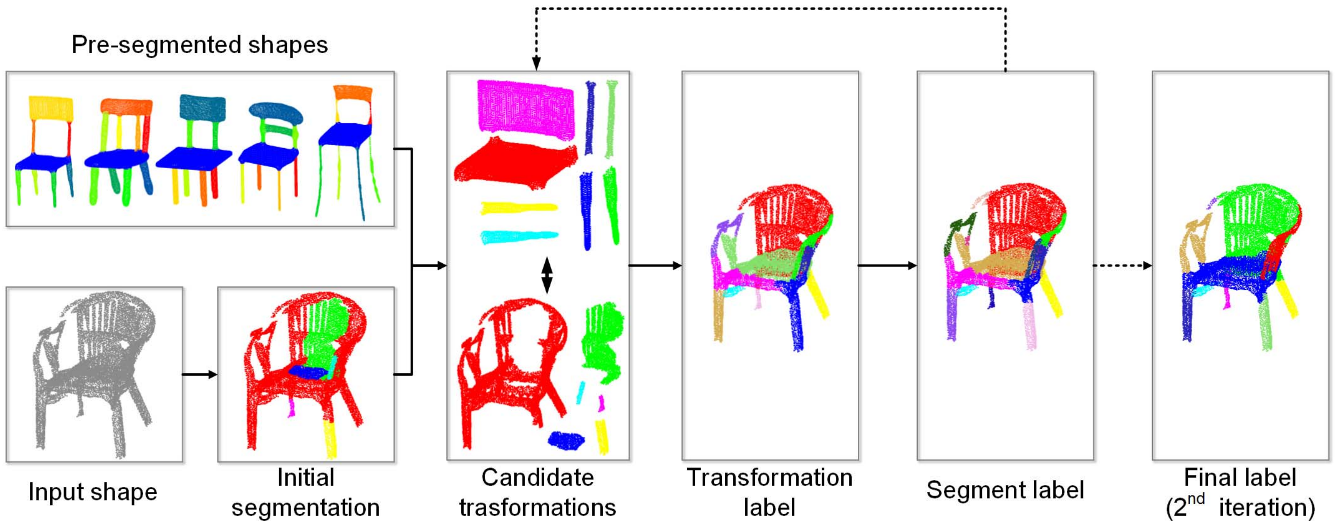

Figure 2: A pipeline of our method. Given pre-segmented exemplar shapes S and a target shape T with initial segmentation, our method first generates a set of candidate transformations M that potentially align parts of T to S. The set M is generated by part-wise registration of S and T. Then we assign each point of T a transformation in M in a globally optimized manner. After that, each point in T copies segment label from its nearest neighbor in exemplar shape S, which is combined with its transformation label to form its final label. The resulting segmentation can be fed into the pipeline again as initial segmentation to improve the result.

mized manner (Section 5) so that (1) the points are well aligned with exemplar shapes; (2) boundaries between different labels are small; and (3) the fewest labels are used in total. We formulate this as an energy optimization problem and solve it using graph cut algorithm [8]. Transformation label, combined with segment label transferred from exemplar shapes, constitute final label of target shapes. Finally, a post-process (Section 6) is applied to smooth the results, which can be fed into the pipeline again as initial segmentation to iteratively improve the results (Section 7).

4. Generating Candidate Transformations

The first step of our method is generating an over-complete set of candidate transformations M so that each point in T can pick one of them to align with S in a later stage.

A classic idea proposed by Mitra *et al.* [21] uses the following fact: a pair of points with their local frames uniquely defines a rigid transformation. Thus their method randomly picks a large number of point pairs from S and T, and computes a rigid transformation that aligns their positions and local frames (normals and principal directions). Mean-Shift clustering [7] in 6D transformation space yields a set of candidate transformations M.

However, we argue that this point-wise approach does not work on noisy real scans, because local high-order properties (*e.g.*, curvatures) are not reliable (Figure 3(c)). To improve it, we introduce a part-wise approach of generating candidate transformations. Instead of picking pairs of points, we utilize initial segments of the target shape T, and pick pairs of segments from S and T instead. Because we are targeting at an over-complete candidate set M, the initial segments may overlap but should cover all points in the shape T, increasing the probability that correct (or near correct) segments appear in M. In the first iteration, the initial segmentation can be computed by any automatic point-cloud segmentation algorithm (*e.g.*, [14, 24]); in later iterations, segmentation results from the last iteration can be used, as discussed in Section 7.

After an initial segmentation is generated, we enumerate all pairs of segments from S and T and compute rigid transformations to register each pair. Specifically, the centroids and three PCA axes of both segments are aligned as initial pose, and Iterative Closest Point (ICP) algorithm [25] is applied to refine it. Here a variant of ICP is applied that finds the closest point with compatible normal (with deviation less than $\frac{\pi}{4}$).

A comparison of point-wise and part-wise approach is shown in Figure 3. In this example, point-wise approach generates 1000 candidate transformations but still misses the correct match. The reason is that in real scans, local features (*e.g.*, curvatures) adopted in point-wise approach are extremely sensitive to noise, making the alignment based on these features very unstable. In contrast, part-wise transformations are much more robust due to usage of large patches and ICP refinement. As a result, it generates only about

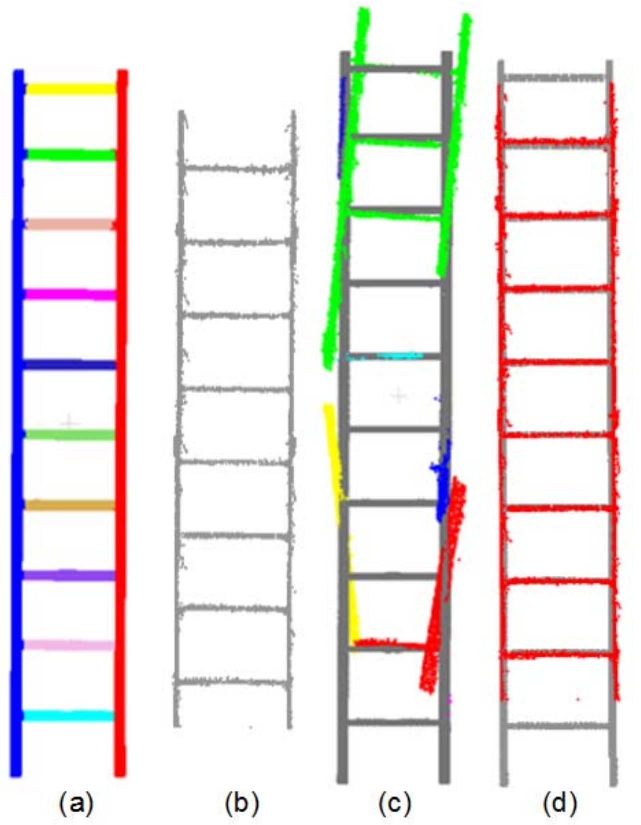

Figure 3: A comparison of point-wise and part-wise approaches of candidate transformations. The pre-segmented exemplar ladder (a) and target ladder (b) can be aligned using one rigid transformation. Point-wise approach generates 1000 candidate transformations (after clustering) but still misses the correct one. (c) shows an piece-wise registration using these 1000 transformations (optimized by a method described in Section 5), which is a not precise match. On the contrary, for part-wise approach, a set of about 20 candidate transformations is enough to contain the correct one (as shown in (d)), which is much more efficient.

20 candidate transformations, including the correct one, enabling a accurate piece-wise registration in a later stage.

5. Optimizing Assignment of Transformations

With a set of candidate transformations M generated, the next step is to compute a piece-wise rigid registration of T to S. This is accomplished by treating each transformation as a label and assigning each point of T a transformation label from M. This labeling should be optimized in three aspects: (1) the fitting error should be small (data cost); (2) neighboring points tend to share the same label (smoothness cost); (3) the number of appeared labels should be small (label cost). To sum up, we minimize the following energy

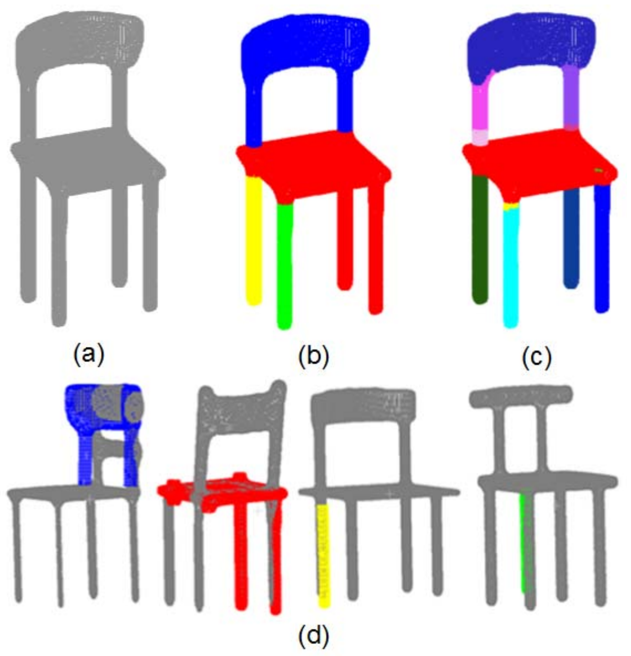

Figure 4: The target shape (a) is assigned four transformations on different regions (b). The four regions are aligned with exemplar shapes using their assigned transformations (d) and get segment labels transferred (c) using simple nearest-neighbor matching.

function:

$$E(L) = \lambda_d E_d(L) + \lambda_s E_s(L) + \lambda_l E_l(L),$$

where L is a transformation label assignment. We choose the factors $\lambda_d = 1$, $\lambda_s = 0.2$, $\lambda_l = 0.01$ in all our experiments.

Data cost E_d is defined as the sum of distances of all points in T to nearest neighbor in S. Here is the distance is defined as a linear combination of Euclidean distance and normal angular distance:

$$E_d(L) = \sum_{p \in T} \min_{q \in S} \{dist(p', q) + \alpha \langle N_p, N_q \rangle\},$$

where p' is obtained by transforming p with its associated transformation given by L; N_p and N_q are normals of p and q, respectively. We choose the balance factor $\alpha = 0.01$ empirically in all our experiments.

Smoothness cost E_s penalizes different labels of neighboring points, especially in flat areas:

$$E_s = \sum_{p,q \in T, (p,q) \in N} S(p, q),$$

where N denotes the relation of neighborhood, and $S(p, q)$ the individual smoothness penalty, which is defined as fol-

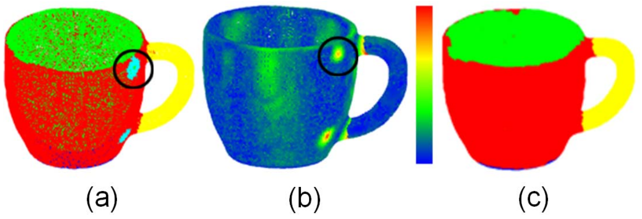

Figure 5: Labels computed in Section 5 may contain small holes with different labels from their surroundings (a). By plotting and examining the registration error map (b), we found that regions with high registration error tends to be mistakenly labeled. Thus we apply a post-processing step to smooth out these regions (c).

lows:

$$S(p,q) = \begin{cases} 0 & \text{if } L(p) = L(q), \\ 1 - \frac{\langle N_p, N_q \rangle}{\pi/2} & \text{if } L(p) \neq L(q). \end{cases}$$

Label cost E_l penalizes any appearance of labels:

$$E_l = \sum_{l \in L} App(l),$$

where $App(l)$ is an appearance indicator function:

$$App(l) = \begin{cases} 1 & \text{if } l \text{ appear in } L, \\ 0 & \text{if } l \text{ not appear in } L. \end{cases}$$

We minimize the energy using a graph-cut algorithm [8]. After the optimal assignment L^* is calculated, each point in T is transformed using its associated transformation given by L^*, and the segment label gets transferred from its nearest neighbor in S. The final label of T is the combination of transformation label and transferred segment label, *i.e.*, two points in T have the same final label if and only if they got the same segment label transferred under the same transformation.

Figure 4 shows an example of optimizing assignment of transformations. Out of a set of candidate transformations M, our algorithm chooses an optimal labeling L^* which consists of four transformations on four different parts. These parts are well aligned with corresponding parts of exemplar shapes, and get segment label transferred from them.

6. Post-Processing

In this section, a post-processing step is applied to refine segmentation. The goal of this step is smoothing out small segments, and aligning segment boundaries to shape features. For example, Figure 5(a) shows an example of segmentation before refinement, where the cup body contains small "holes", which is caused by mis-alignment of the handles: that region gets labels transferred from the handles of another cup in exemplar set. Thus a post-process is adopted to smooth out small segments that are mistakenly labeled, while preserving the correct ones. In addition to region sizes, we observe that registration error can be a good indication to distinguish these two cases. In other words, small regions with large registration error are likely mistakenly labeled, and thus should be smoothed out. For example, as Figure 5(b) shows, the wrong labels of the cup align well with regions with large registration error.

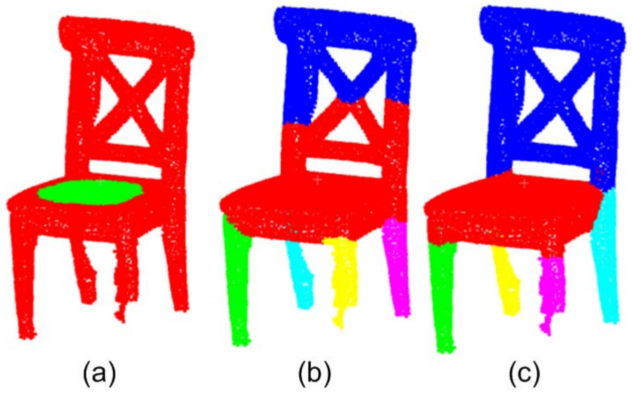

Figure 6: Illustration of multiple iterations. (a) Initial segments; (b) Result of first iteration; (c) Result of second iteration.

We formulate this problem as an optimization problem and again use a graph-cut algorithm [8] to solve it. The goal is to refine an existing labeling L over all points so that:

1. A data penalty occurs when a point changes label. In particular, the penalty is large in a high registration-error region.
2. A smoothness penalty occurs when two neighboring points have different labels.

More formally, the energy is defined as:

$$E'(L') = \lambda'_d \sum_{p \in T} E'_d(p, L') + \lambda'_s \sum_{p,q \in T, (p,q) \in N} S'(p,q,L').$$

We choose $\lambda'_d = 0.3$, $\lambda'_s = 1$ in all our experiments. Data penalty $E'_d(p, L')$ is defined as:

$$E'_d(p, L') = \begin{cases} 0 & \text{if } L'(p) = L(p), \\ \frac{dist(p, L'(p))}{err(p)} & \text{if } L'(p) \neq L(p). \end{cases}$$

where $dist(p, L'(p))$ is the distance of p to the nearest boundary of its new label $L'(p)$. Intuitively, if p needs to travel a long distance to its new label $L'(p)$, then p is inside a large patch and thus changing its label should be given large cost. $err(p)$ is the registration error of p using its old

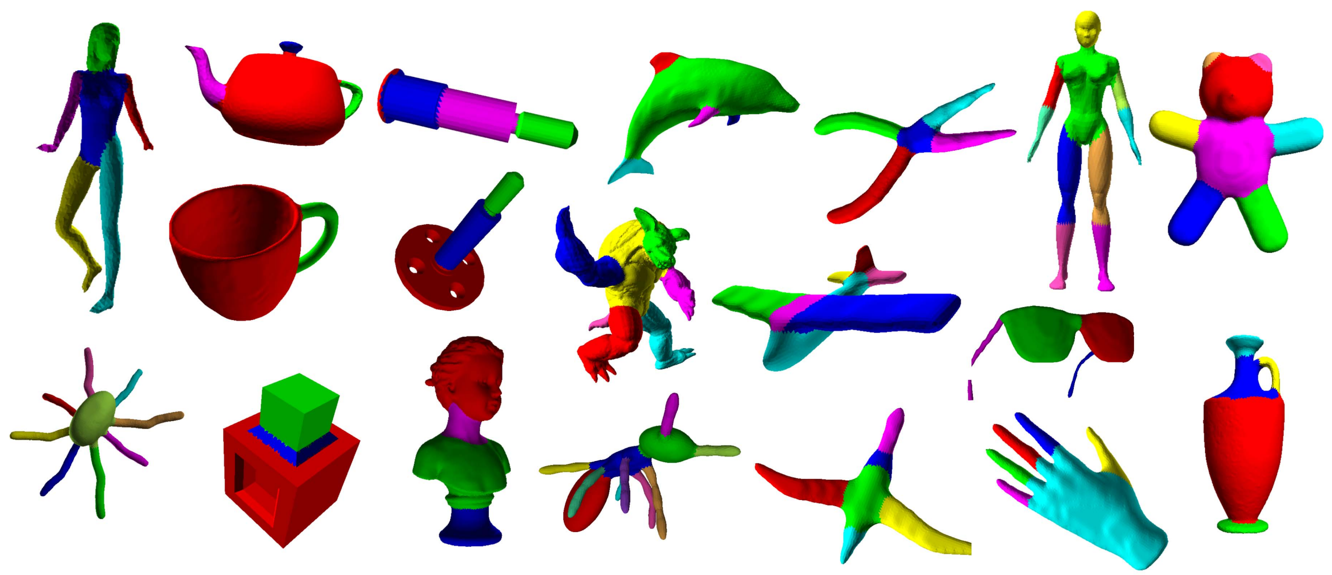

Figure 7: A gallery of our resulting segmentations of 3D mesh segmentation benchmark [5]. Each shape is segmented using the other 19 shapes in the same category.

transformation label $L(p)$, and a large $err(p)$ indicates a mis-aligned region and welcomes change of label.

Smoothness penalty $E'_s(p,q,L')$ is defined in a similar manner as in Section 5:

$$E'_s(p,q,L') = \begin{cases} 0 & \text{if } L'(p) = L'(q), \\ 1 - \frac{\langle N_p, N_q \rangle}{\pi/2} & \text{if } L'(p) \neq L'(q). \end{cases}$$

After refinement, the wrong labels in Figure 5(c) are successfully smoothed out while the correct labels are preserved.

7. Multiple Iterations

After the above steps, we have computed a segmentation of target shape T from rough initial segments. Because we use example set S as guidance, the resulting segmentation (*e.g.*, Figure 6(b)) is almost always better than the initial segments (*e.g.*, Figure 6(a)). This inspires us to feed resulting segmentation to the pipeline again to further improve it (*e.g.*, Figure 6(c)). Theoretically, we can iterate the process until convergence. However, to keep the method efficient, we fix the number of iterations to be 2 in our experiments.

8. Experimental Results

The method is first tested on synthetic datasets (*i.e.*, both target shapes and example shapes are point clouds subsampled from 3D CAD models) with comparisons to other non-exemplar-based segmentation algorithms. Next, experiments are performed on real scans using CAD models as exemplar shapes.

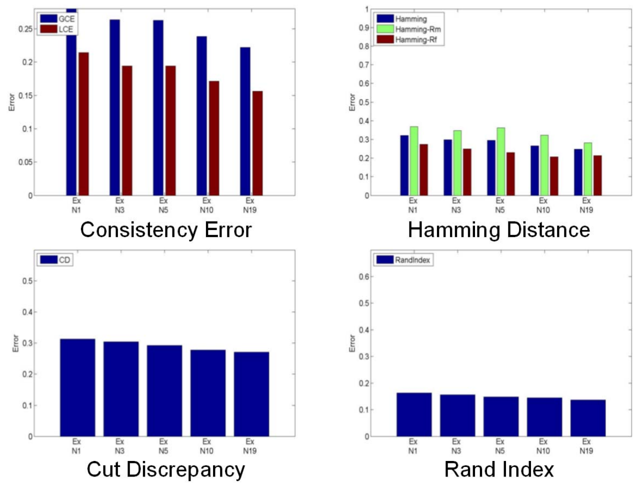

Figure 9: Performance as a function of number of examples. Five columns are results using 1, 3, 5, 10, 19 examples, respectively. Our method benefits from additional examples and its performance improves quickly as number of examples grows.

8.1. Synthetic Datasets

To demonstrate the capability of our method on handling large varieties of shapes, we test our method on 3D mesh segmentation benchmark [5]. It contains 19 categories of shapes, with 20 shapes in each category. To maximize the usage of this dataset, we set up our experiment so that each target shape is segmented using the other 19 shapes in the same category as examples. Each shape is

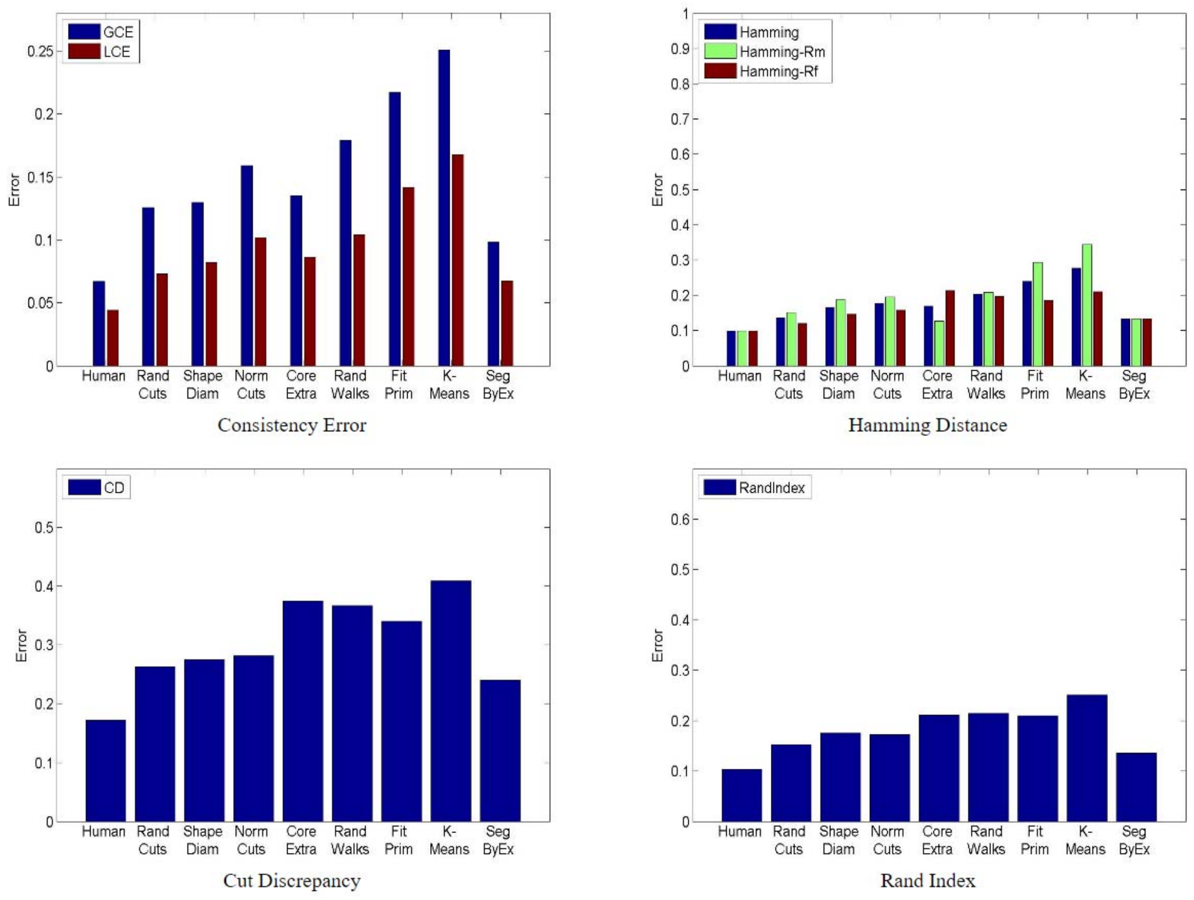

Figure 8: Statistical Results of 3D Segmentation Benchmark. Four metrics (*i.e.*, consistency error, hamming distance, cut discrepancy and rand index) are adopted to compare different methods. By making full use of pre-segmented exemplar shapes, our method outperforms all automatic non-exemplar-based methods. For details of comparing metrics and methods, we refer the readers to [5].

converted from triangle mesh to point cloud before fed into our algorithm, and converted back afterwards. In the case of synthetic datasets, initial segmentation is computed using Shape Diameter Function [28]. The method is implemented in C++ and run on a consumer-level desktop (Intel Core i7-3610QM 2.30GHz CPU with 6GB RAM). In our experiments, segmentation of each shape takes about 5 minutes in average. A gallery of resulting segmentations is shown in Figure 7. Quantitative analysis is shown in Figure 8, where four metrics (*i.e.*, consistency error, hamming distance, cut discrepancy and rand index) are used to compare segmentations in different aspects. We refer readers to [5] for details of these metrics and compared methods. By making full use of pre-segmented exemplar shapes, our method outperforms all non-exemplar-based methods.

In order to study the effect of the number of training examples, we repeat the experiment with 1, 3, 5, 10 training examples and compare their performances to that using full 19 training examples (Figure 9). Results show that our method still works with only a small training set but benefits from additional examples used.

8.2. Real-Scan Datasets

To demonstrate robustness of our method, we also experiment on real-scan datasets. We use a dataset from Qi *et al*. [23] as target shapes. These point clouds are registered from RGB-D sequences taken from a PrimeSense sensor, where the objects are extracted from the background. Specifically, two categories (*i.e.*, chairs and tables) from this dataset are used as our target shapes, and corresponding 20 chairs and 20 tables from 3D mesh segmentation benchmark [5] are used as examples. As Figure 10(a) shows, non-exemplar-based methods such as Van Kaick *et al*. [14] are dramatically affected by severe data problems in these real scans. Our method, on the other hand, remains robust due to usage of pre-segmented exemplar shapes and independence of high-order features (Figure 10(b)).

(a)

(b)

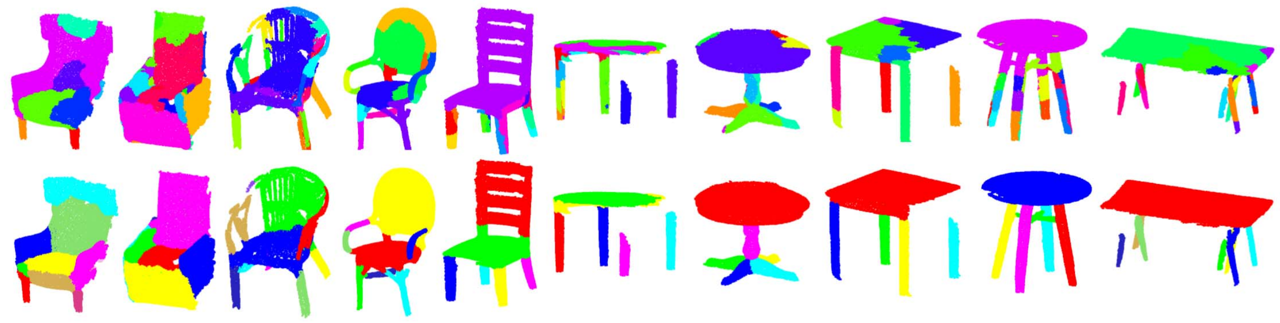

Figure 10: Segmentation results of real scans (dataset from Qi *et al*. [23]): (a) results of Van Kaick *et al*. [14]; (b) our results using pre-segmented shapes from 3D mesh segmentation benchmark [5] as examples. The scans are taken from a PrimeSense sensor, where the objects are extracted from the background.

9. Conclusion

This paper presents an exemplar-based automatic 3D shape segmentation method. It makes full use of pre-segmented example shapes to guide segmentation of target shape. In contrast to typical machine learning methods, our method only needs a relatively small training set (about 20 in our experiments). Our method is capable of segmenting noisy real scans because it only relies on low-order information (*i.e*., geometric locations and normals) as shown in the experiments on challenging datasets.

References

[1] I. Alhashim, K. Xu, Y. Zhuang, J. Cao, P. Simari, and H. Zhang. Deformation-driven topology-varying 3d shape correspondence. *ACM Transactions on Graphics (TOG)*, 34(6):236, 2015.

[2] O. K.-C. Au, Y. Zheng, M. Chen, P. Xu, and C.-L. Tai. Mesh segmentation with concavity-aware fields. *IEEE Transactions on Visualization and Computer Graphics*, 18(7):1125–1134, 2012.

[3] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[4] W. Chang and M. Zwicker. Automatic registration for articulated shapes. In *Computer Graphics Forum*, volume 27, pages 1459–1468. Wiley Online Library, 2008.

[5] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3d mesh segmentation. In *ACM Transactions on Graphics (TOG)*, volume 28, page 73. ACM, 2009.

[6] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Transactions on Graphics (TOG)*, 23(3):905–914, 2004.

[7] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[8] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov. Fast approximate energy minimization with label costs. *International journal of computer vision*, 96(1):1–27, 2012.

[9] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel. On the segmentation of 3d lidar point clouds. In *International Conference on Robotics and Automation (ICRA)*, pages 2798–2805. IEEE, 2011.

[10] A. Golovinskiy and T. Funkhouser. Consistent segmentation of 3d models. *Computers & Graphics*, 33(3):262–269, 2009.

[11] A. Golovinskiy and T. Funkhouser. Min-cut based segmentation of point clouds. In *IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 39–46. IEEE, 2009.

[12] K. Guo, D. Zou, and X. Chen. 3d mesh labeling via deep convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 35(1):3, 2015.

[13] Q. Huang, V. Koltun, and L. Guibas. Joint shape segmentation with linear programming. In *ACM Transactions on Graphics (TOG)*, volume 30, page 125. ACM, 2011.

[14] O. V. Kaick, N. Fish, Y. Kleiman, S. Asafi, and D. Cohen-OR. Shape segmentation by approximate convexity analysis. *ACM Transactions on Graphics (TOG)*, 34(1):4, 2014.

[15] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3d mesh segmentation and labeling. *ACM Transactions on Graphics (TOG)*, 29(4):102, 2010.

[16] V. G. Kim, Y. Lipman, and T. Funkhouser. Blended intrinsic maps. In *ACM Transactions on Graphics (TOG)*, volume 30, page 79. ACM, 2011.

[17] T. Liu, V. G. Kim, and T. Funkhouser. Finding surface correspondences using symmetry axis curves. In *Computer Graphics Forum*, volume 31, pages 1607–1616. Wiley Online Library, 2012.

[18] J. Lv, X. Chen, J. Huang, and H. Bao. Semi-supervised mesh segmentation and labeling. In *Computer Graphics Forum*, volume 31, pages 2241–2248. Wiley Online Library, 2012.

[19] C. Ma, H. Huang, A. Sheffer, E. Kalogerakis, and R. Wang. Analogy-driven 3D style transfer. *Computer Graphics Forum*, 33(2):175–184, 2014.

[20] O. Mattausch, D. Panozzo, C. Mura, O. Sorkine-Hornung, and R. Pajarola. Object detection and classification from large-scale cluttered indoor scans. In *Computer Graphics Forum*, volume 33, pages 11–21. Wiley Online Library, 2014.

[21] N. J. Mitra, L. J. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (TOG)*, 25(3):560–568, 2006.

[22] A. Nguyen and B. Le. 3d point cloud segmentation: a survey. In *IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 225–230. IEEE, 2013.

[23] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016.

[24] R. Qiu and U. Neumann. Ipdc: Iterative part-based dense correspondence between point clouds. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016.

[25] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings. Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. IEEE, 2001.

[26] A. Shamir. A formulation of boundary mesh segmentation. In *3D Data Processing, Visualization and Transmission*, volume 4, pages 82–89, 2004.

[27] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang. Contextual part analogies in 3d objects. *International Journal of Computer Vision*, 89(2-3):309–326, 2010.

[28] L. Shapira, A. Shamir, and D. Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249–259, 2008.

[29] 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 (TOG)*, volume 30. ACM, 2011.

[30] 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.

[31] 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 (TOG)*, 32(4):69, 2013.

[32] 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 (TOG)*, 31(6):165, 2012.

[33] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li. Dense human body correspondences using convolutional networks. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016.