Sealed Decomposition of a Triangular Mesh with Tetrahedral Mesh Segmentation

[1]Caiyun Yang, [2]Yutaka Ohtake and [3]Hiromasa Suzuki

[1]The University of Tokyo, yang@den.rcast.u-tokyo.ac.jp
[2]The University of Tokyo, yu-ohtake@den.rcast.u-tokyo.ac.jp
[3]The University of Tokyo, suzuki@den.rcast.u-tokyo.ac.jp

ABSTRACT

We propose a method for decomposing a closed 2-manifold triangular mesh (tri-mesh) into a set of sub-meshes. This approach has two novel features. First, the decomposed tri-meshes are "sealed". This type of subset tri-mesh generally has boundaries, but our method automatically fills these boundaries to close the subset mesh in a technique known as "sealed decomposition". Second, the mesh is decomposed along its concave features; the outcome tends to be a convex mesh decomposition, which is useful for a variety of applications. In addition, we smooth those parts of the set of decomposed sub-meshes not belonging to the closed 2-manifold tri-mesh using the umbrella algorithm.

Keywords: sealed decomposition, tetrahedral mesh segmentation, graph cut.
DOI: 10.3722/cadaps.2011.421-433

1 INTRODUCTION

A common technique for handling large, complex objects is to compute the division of the model into parts so as to enable a large number of applications, including ray tracing, rendering, collision detection and 3D mesh editing. In the past few decades, a vast amount of research work has been conducted to segment 3D surface mesh shapes into visually salient components [3], [7], [15-17], [20]. With this technique, the total mesh is cut into a set of sub-meshes, which therefore have boundaries even though the total mesh has none. Instead, decomposing a 3D geometric mesh into closed surface meshes has a large number of advantages when the geometric shape is understood, analyzed and edited. In addition, a variety of useful characteristics (such as the volume and the boundary surface area size of segmented components) can be ascertained much more simply when the corresponding geometric object consists of a set of sub closed surface meshes. We call this approach "sealed decomposition".

In addition, we aim to compute perceptually meaningful decomposition in which the sub-meshes take almost-convex shapes with the total mesh divided along its concave features. In order to locate concave features, our method requires segmentation of the total mesh as an input with segment boundaries along the concave features. We apply the approaches proposed in [8] and [24], though other methods can be used if their segmentation can recognize concave features.

To accomplish such sealed decomposition, we use tetrahedral meshes (tet-meshes) for computation. It is obvious that the boundary triangles of a tet-mesh form a closed triangular mesh (tri-mesh). Accordingly, we first generate a tet-mesh from the target mesh so that the boundary of the former is consistent with that of the later. Then, we decompose the tet-mesh along its concave features to generate a number of segmented tet-meshes. Finally, we extract the boundary of each decomposed sub-component, which forms a closed tri-mesh not only with surface triangles but also with the interior triangles of the tet-mesh. This decomposition is performed using the graph cut method and the given mesh segmentation is used to define the source and sink nodes.

The continuous improvement of tet-mesh generation techniques has made it possible to produce tet-mesh models in recent years. The interior triangles produced are usually rugged, so we smooth them using the umbrella algorithm.

The algorithms we propose here are completely automatic and easily implemented.

2 RELATED WORK

Graph cut Graph cut methods have been extensively applied to compute the segmentation of images into non-overlapping segments [1], [4], [18]. In recent years, their application in dealing with 3D geometric mesh shapes has also been introduced. Papers[10], [24] first compute the partition of 3D mesh models and use graph cut methods to smooth the boundaries shared by two adjacent mesh surfaces in post-processing. The boundary smoothing method based on graph cut in [24] is extended by [10] to improve the quality of the resulting smoothed boundaries. In [14], a hierarchical clustering method based on geodesic distance is used to segment a mesh shape into segments and generate a fuzzy strip between two neighboring parts. A graph cut method is applied to remove these mesh triangles in the fuzzy strip into the adjacent relevant parts and define an accurate boundary shared by the two neighboring components.

Mesh segmentation Depending on applications, two principal types of mesh segmentations are made, one is part-type segmentation targeting at partitioning a mesh into meaningful parts [6]; the other is patch-type segmentation which segment a mesh into patches [8], [24]. Many papers [26] have been presented on the partition of 3D mesh models into compact and non-overlapping portions for a wide range of applications such as geometry processing and object recognition. Papers [12], [23] present a method of convex decomposition. Such method generates a hierarchical volumetric representation of 3D shapes and recognizes feature volumes from boundary information. Then it generates a new object by subtracting an object from its convex hull recursively until all sub-objects are convex. Spectral clustering applies an intrinsic metric to build an infinity or Laplacian matrix [19], and the eigenvector of the matrix is used to realize the segmentation of input mesh models. [14] and [26] apply geodesic distance to compute the partition of 3D mesh models and negative curvature to obtain an accurate boundary between two adjacent parts. Defining an error metric L2 measuring the distance between the vertices and the corresponding fitting primitives, a hierarchical clustering algorithm is proposed to obtain useful segments from mesh shapes [2]. [17] computes a union of tight bounding volumes based on an e-tightness measure to divide a 3D model into segments using a variational approach. In recent years, the concept of convex hulls has been used to compute the partition of mesh shapes [3] and [16]. A bottom-up algorithm hierarchically clusters a mesh shape into meaningful components and generates a tree with the root representing the whole model and the leaves becoming non-overlapping tight parts [3]. The paper [6] suggests the best benchmarks of part-type segmentation which were manually made, thus our purpose is to achieve these segmentation automatically.

Tetrahedral mesh generation The difficulty of generating a mesh represented by tetrahedrons means that the issue has a long history in both computer science and in engineering. Part of the difficulty relates to forcing a 3D tet-mesh model to conform to the sharp corners and edges of an object while maintaining quality. However, with the development of approaches for generating tet-meshes over the last decade, a 3D geometric mesh represented by a tet-mesh can be easily and quickly generated without sacrificing the quality of the corresponding surface mesh. As a result, a variety of software is now produced that uses advanced techniques to generate tet-meshes. One of these is TetGen [21], which can generate tet-meshes of any 3D polyhedral domain. This application

produces 3D Delaunay tet-meshes conforming to boundaries as well as exact constrained Delaunay meshes represented by tetrahedrons. TetGen can also generate high-quality tet-meshes suitable for numerical approaches, including the finite element and finite volume methods. It can easily be applied to compute a mesh model represented by tetrahedrons from the corresponding surface mesh. Accordingly, we use TetGen to generate a tet-mesh model from the corresponding surface mesh while keeping the boundary triangles fixed.

Surface mesh fairing A signal processing technique is presented to smooth arbitrary topology connectivity surface meshes by minimizing expensive energy functionals in [22]. Transforming the problem of fairing arbitrary polyhedral surfaces into discrete functions is based on two-dimensional discrete surface signals. As the Laplacian is discretely approximated, its eigenvectors are taken as the frequency of the polyhedral surface mesh. This frequency is tailored by repeatedly applying the linear operator, and a smoothed arbitrary surface mesh can be obtained. The method is linear in terms of time and space, and makes it quite easy to process meshes composed of a large number of triangles into visually smooth surfaces. Kobbelt [13] extends the technique to fair arbitrary surfaces by combing similar discrete approximations of Laplacian and mesh subdivision methods. This is applied for arbitrary connectivity models to edit multi-resolution areas [11]. These methods are quite easy to implement and are suitable for handling large meshes, which has led to their extensive use in the field. However, a number of issues remain for irregular arbitrary connectivity polyhedral meshes, including slow convergence for large meshes and inadequate control over total behavior. Accordingly, Desbrum et al. in [9] computed solutions for these problems to enhance the fairing of arbitrary connectivity polyhedral surface meshes. The method in [11] can be simply implemented and our fuzzy meshes are regular connectivity meshes, thus we apply the method in our research.

3 SEALED DECOMPOSITION OF A TRIANGULAR MESH

3.1 Algorithm Summary

The input for our method is a 2-manifold tri-mesh M without boundaries. We also need a segmentation of M, $R = \{R_i\}$, where R_i is the i-th region of the segmentation and satisfies the following properties: $R_i \subset M$, $\cup R_i = M$ and $R_i \cap R_j = \Phi\ (i \neq j)$. The region boundary b_{ij} is a set of triangle edges between the two adjacent regions R_i and R_j. As mentioned above, we use the segmentation methods proposed in [24] and [8]. A tet-mesh is denoted by $T = \{t_i\}$, where t_i is the i-th tetrahedron of T. We can define a closed tri-mesh $Tri(T)$, which is made of the boundary triangles of the tet-mesh T. We assume T can be generated with consistency to M such that $Tri(T) = M$. In our implementation, we use TetGen software [21].

First, we find a concave boundary and let it be b_{ij} between the two regions R_i and R_j. We decompose the tet-mesh T along this boundary using the graph cut algorithm. The graph's nodes are tetrahedrons of T, and arcs are formed between all adjacent tetrahedrons sharing a common surface in the two regions. For the graph cut, we assign tetrahedrons whose boundary triangles are associated with R_i as source nodes, and those with R_j as sink nodes. The costs of the arcs will be introduced later.

After the graph cut, T is decomposed into two sub-tet-meshes. Our algorithm works recursively in a top-down manner. That is, we continue applying the above graph cut decomposition for the two sub-tet-meshes decomposed as long as concave boundaries remain.

After finishing this tetrahedral decomposition, we have a set of decomposed tet-meshes T_i. These tend to take convex shapes as they are decomposed along concave boundaries. We then extract the boundary tri-meshes M_i of T_i, that is, $M_i = Tri(T_i)$. Some region of triangles of M_i is not on M but on a surface of section of tet-mesh T. Thus it is jagged and need to be smoothed.

In our algorithm, decomposition is performed along concave boundaries between regions of segmentation, meaning that it is not dependent on convex or smooth boundaries. Accordingly, decomposition does not involve extensive changes as all the concave features are captured in the segmentation.

A summary of the algorithm is given below:

Main

Begin

Input: M and R

Generate T s.t. $Tri(T) = M$ $\{T_i\}$ = Decompose (T, R)

For each T_i

$M_i = Tri(T_i)$

Smoothing (M_i)

end

end

Procedure Decompose (T, R)

begin

If concave boundaries exist then

Select the concave boundary b_{ij} with the minimum α_{ij} value

Set $Tet(R_i)$ to be the seeds of N_0 and also set $Tet(R_j)$ to be the seeds of N_1

Apply graph cut for T to separate it into $T0$ and $T1$

$R0$: set of regions in $Tri(T0)$ $R1$: set of regions in $Tri(T1)$

Return union of Decompose $(T0, R0)$ and Decompose $(T1, R1)$

else

Return $Tri(T)$

end

3.2 Concave boundary

The input surface mesh M is segmented by the methods outlined in [8] and [24] for our algorithm. Boundaries b_{ij} shared by the two adjacent regions R_i and R_j in the surface mesh M are then smoothed using our approach proposed in [25]. Along each boundary, a convexity measure α_{ij} is defined (Fig. **1**):

$$\alpha_{ij} = \frac{\sum_{m \in b_{ij}} \psi_m \delta_m}{\sum_{m \in b_{ij}} \delta_m} \tag{1}$$

where ψ_m is the dihedral angle between two triangles incident to edge m of b_{ij} , and δ_m is its length.

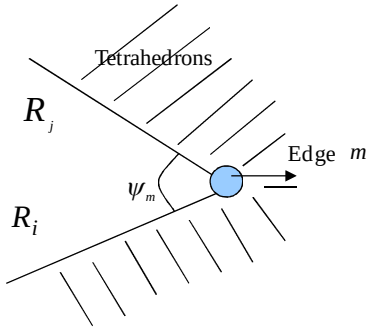

Fig. 1: Dihedral angle ψ_m definition along a concave boundary.

α_{ij} is the average dihedral angle along the boundary b_{ij} (see Fig. **1**). An angle is a concave angle if it is less than π. Thus, considering noise, when $\alpha_{ij} < 0.98\pi$, b_{ij} is defined as a concave boundary.

3.3 Tetrahedral mesh segmentation based on graph cut

3.3.1 Graph Cut

We apply the graph cut to separate a tet-mesh $T = \{t_i\}$ into two tet-meshes. The graph $G = (V, E)$ consists of all the tetrahedrons t_i of T as its nodes. We also add two more special nodes, n_0 and n_1, which are referred to as source and sink nodes, respectively. The undirected arcs E of G consist of the set of arcs E_1 between two adjacent tetrahedrons t_i and t_j, the set of arcs E_2 between every t_i and both n_0 and n_1, and the set of virtual arcs E_3 between two boundary tetrahedrons t_i and t_j, whose adjacent boundary triangles are in the same region of the surface segmentation R (see Fig. 2). This arc of E_3 is needed to avoid these connected two boundary tetrahedrons from being segmented into different parts.

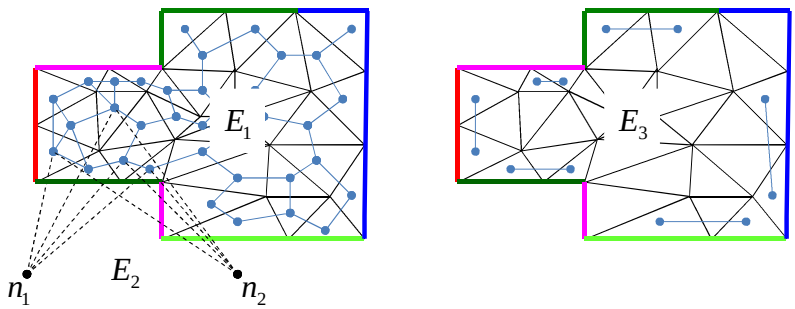

Fig. 2: E_1, E_2 and E_3 are shown in 2D where a tet mesh is represented as a triangular mesh. The colored edges represent the boundary triangles of the tet mesh, and one color edge shows one region. Only a part of the edges of E_2 are shown for brevity.

The arcs are associated with costs. The graph cut separates the set of nodes $V = \{t_i\} \cup \{n_0, n_1\}$ into two sets V_0 and V_1, s.t. $n_0 \in V_0$ and $n_1 \in V_1$ by cutting G along a cut (a set of arcs) so that its cost (i.e., the total cost of the arcs on the cut) is minimized.

We also define two sets of seed tetrahedrons N_0 and N_1 corresponding to $Tet(R_i)$ and $Tet(R_j)$. Tetrahedrons of N_0 are constrained to being members of V_0 by giving high costs to the arcs between n_0 and tetrahedrons of N_0 as hard constraints. In the same way, tetrahedrons of N_1 are constrained to being members of V_1 by giving high costs to the arcs between n_1 and tetrahedrons of N_1 as hard constraints. Tetrahedrons other than those in N_0 and N_1 (i.e., those in N_f) are called fuzzy.

In this research, we used the following cost functions for the arcs of E_2:

$$c(n_0,u)=\begin{cases}\infty & u\in N_0\\ 0 & u\in N_1\\ 1-\dfrac{d(u,N_0)}{d(u,N_0)+d(u,N_1)} & u\in N_f\end{cases} \tag{2}$$

$$c(n_0,u)=\begin{cases}0 & u\in N_0\\ \infty & u\in N_1\\ 1-\dfrac{d(u,N_0)}{d(u,N_0)+d(u,N_1)} & u\in N_f\end{cases} \tag{3}$$

For those of E_1 and E_3 we used the following costs:

$$c(u,v)=\begin{cases}\lambda\dfrac{a(u,v)}{a(u,v)+\Lambda} & (u,v)\in E_1\\ \infty & (u,v)\in E_3\end{cases} \tag{4}$$

where $d(u,N_0)$ is the smallest distance of the shortest path between tetrahedron u and a tetrahedron of N_0. In the same way, $d(u,N_1)$ is the smallest distance of those of paths between u and a tetrahedron of N_1. The distance between the two nodes is defined as that between the centers of the two tetrahedrons. The shortest path is computed using Dijkstra's algorithm. $a(u,v)$ represents the area of the common triangle surface between two adjacent tetrahedrons. Λ denotes the average value of the total common triangle area sum in the tet-mesh. We set high costs to the virtual edges so that surface tetrahedrons with boundary triangles in the same region R_i will not be separated into different subcomponents.

The coefficient $\lambda \geq 0$ describes the relative importance of these two costs. We set λ to 100 for the examples shown in this paper. The minimum cut is obtained using the max-flow/min-cut algorithm proposed in [5].

3.3.2 Examples

An example of a graph cut is shown in Fig. 3 (see caption for details).

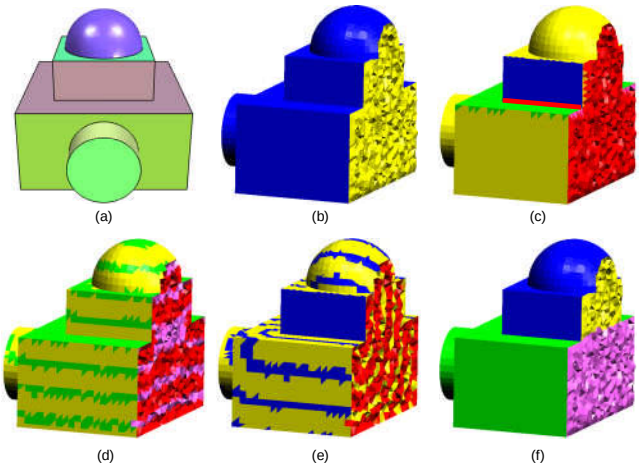

Fig. 3: (a) Segmentation result R for the surface mesh M. (b) Tet-mesh model T. (c) Two sets of seeds - N_0: blue tetrahedrons; N_1: green tetrahedrons. The thick red line shows the concave boundary between the green and blue tetrahedrons. (d) Eight green contours are sampled showing the shortest distance from seed set N_1. (e) Eight blue contours are sampled showing the shortest distance from seed set N_0. (f) Tet-mesh M segmented into two parts shown in blue and green.

In all, E_2 is represented by the smallest distance between two set of seeds and tetrahedron u, which means the tetrahedron u should be segmented into the part including the set of seeds, where there is smaller distance between tetrahedron u and them. E_1 is defined by the area between two adjacent tetrahedrons, the total area should be small when the minimum cut is achieved. Thus we can apply graph cut method to achieve right decomposition results.

3.4 Mesh Smooothing

As tet-mesh T_i is generated by cutting a tet-mesh, part of its boundary tri-mesh corresponding to the cut surface lacks smoothness.

To address this, we apply the discrete Laplacian as an umbrella operator [11] to this part of the tri-mesh (see Fig. 4).

$$U(v_i) = \begin{cases} v_i, & v_i \in O \\ \frac{1}{2}\sum_{j=0}^{1} v_j - v_i, & v_i \in I, v_j \in O \cup I \\ \frac{1}{n}\sum_{j=0}^{n-1} v_j - v_i, & v_i \in R, v_j \in O \cup I \cup R \end{cases} \tag{5}$$

where v_j represents the neighboring vertices of v_i, the vertices in $O \cup I \cup R$ consist of those in the common triangle mesh between two adjacent components, O is the set of vertices on the tet-mesh surface, I is the set of vertices of the boundary tri-mesh corresponding to the cut surface intersecting with another boundary mesh corresponding to another cut surface except for the vertices in O, and R is the set of vertices besides those in I and R. We keep the boundary vertices frozen for discrete boundary optimization.

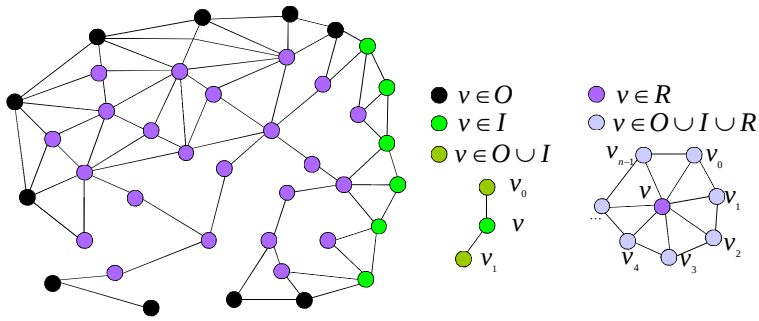

Fig. 4: In the top picture, the vertexes of the common surface shared by two adjacent components incorporate those shown in black, green and purple. The black vertexes belong to those on the surface of the tet-mesh. The green vertexes are those of the common surface intersecting with another common surface other than those of the black vertexes. Other vertexes are shown in purple.

For this linear system, we use iterative optimization approaches such as the Gauss-Seidel algorithm. The above smoothing method was run on a number of models, and corresponding efficient results were achieved (see Fig. 5 and Fig. 6).

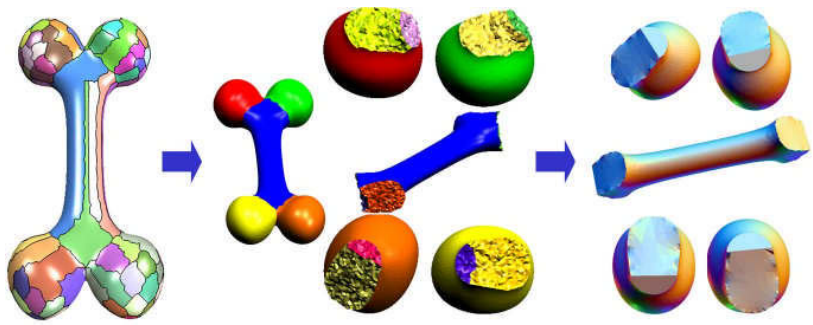

Fig. 5: Bone: The picture on the left presents the surface mesh segmentation results achieved using the methods proposed in [8] and [24]. The middle picture shows the components of the tet-mesh segmented. The common surfaces shared by two adjacent parts are smoothed in the picture on the right, and sealed sub-triangle meshes are achieved.

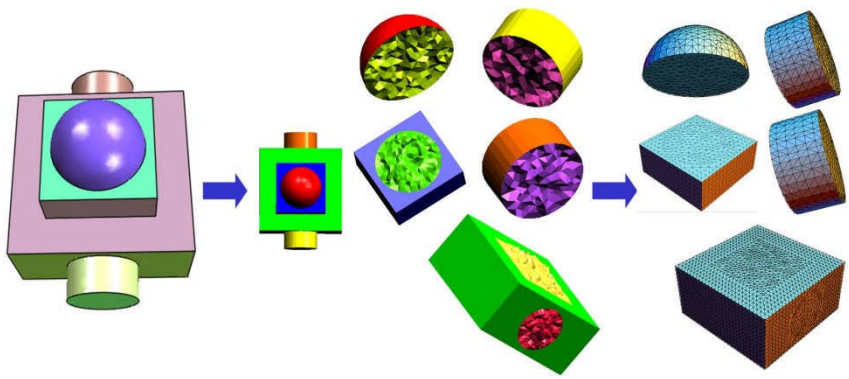

Fig. 6: Mechanical part: The picture on the left presents the surface mesh segmentation results achieved using the methods proposed in [24]. The middle picture shows the components of the tet-mesh segmented. The common surfaces shared by two adjacent parts are smoothed in the picture on the right, and sealed sub-triangle meshes are achieved.

4 RESULTS

In this section, a number of tested models are described to demonstrate the effectiveness of our method, and are compared with benchmark models segmented using the manual in [6]. These models are shown in Figures 7, 8, 9 and 10, where the top row shows the best benchmark results obtained using the manual in [6]. The middle row illustrates the results of patch type segmentation using the VSA based methods proposed in [8] and [24], and these are considered as the input files for our algorithms. The bottom row presents the results of part-type segmentation obtained using our method. It can be seen that the outcomes are similar to those of the best benchmark achieved using the manual in [6]. The segments are generated by the VSA-based method so as to have a sufficient number of segments. And our proposed method successfully reconfigure these segments by merging unnecessary ones and adjusting their boundaries. In addition, our segmented parts are all sealed. The difference between the input segmentations and our results should be noted. The input segmentations are not required to correctly capture the convex portions; rather, they need only include all the concave boundaries to generate good decompositions.

We ran all these models on a PC with a Core (TM) 3.00GHz CPU. Table 1 presents the running time and other statistics for model parts.

5 CONCLUSIONS AND FUTURE WORK

This paper outlines a method to decompose a closed 2-manifold tri-mesh into a set of sub-sealed tri-meshes. The algorithms used are fully automatic and efficient. There are some limitations in our method. Our results depend on the surface segmentation results shown as the middle row in Fig. 7, Fig. 8, Fig. 9 and Fig. 10, which means there should be segmented boundaries between regions on the concave parts of the 3D shape. But our results are not sensitive to them and they can be over-segmented (see the middle row in Fig. 8, Fig. 9 and Fig. 10,). And if there are no segmented boundaries in 3D shapes, there is no decomposition. Our computation time (see Tab. **1**) increases because we use tetrahedrons in our methods. But our results are achieved fully automatically and similar to the best benchmarks by manual (see Fig. 8, Fig. 9 and Fig. 10).

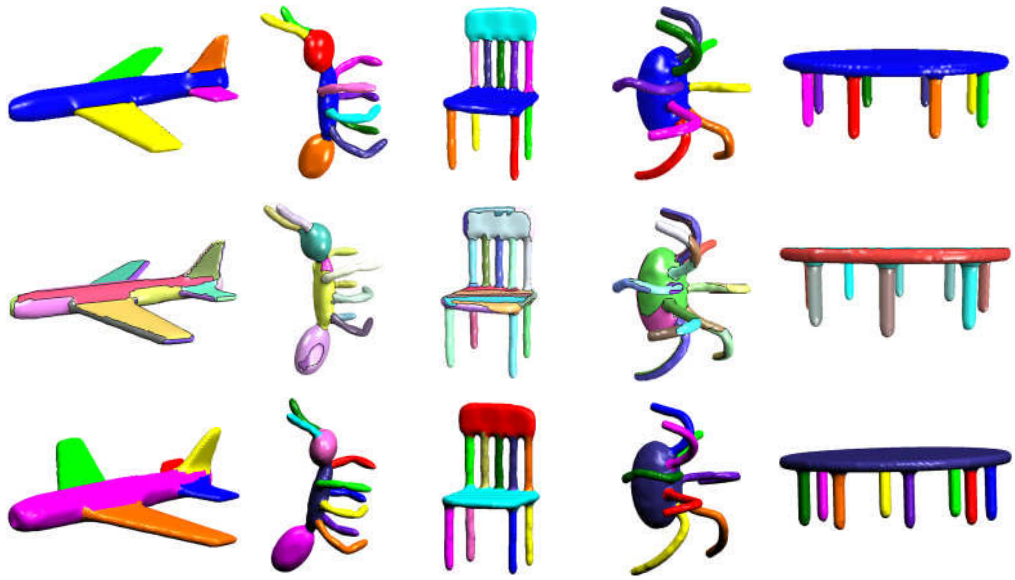

Fig. 7: The top row shows the benchmarks obtained using the manual in [6]. The middle row presents the surface mesh segmentation results achieved using the methods proposed in [8] and [24]. The bottom row describes the results of the technique proposed in our paper.

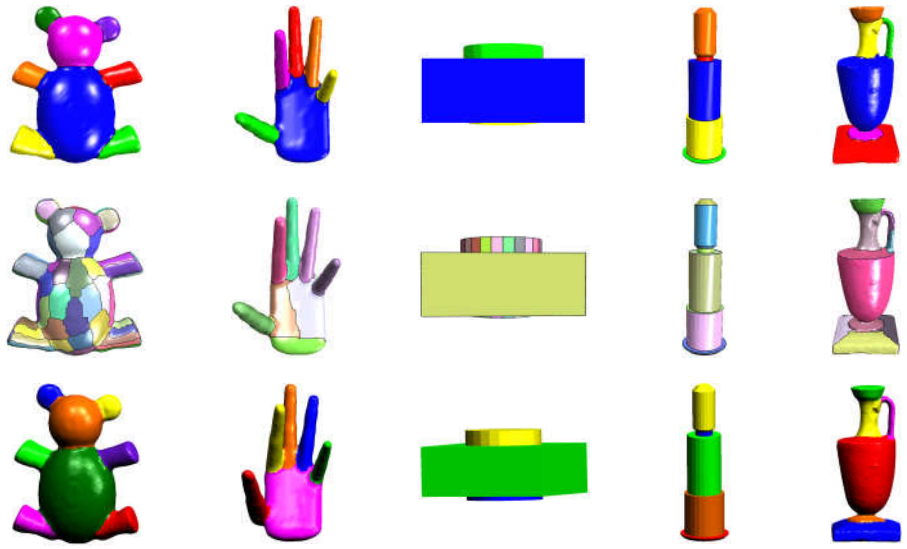

Fig. 8: The top row shows the benchmarks obtained using the manual in [6]. The middle row presents the surface mesh segmentation results achieved using the methods proposed in [8] and [24]. The bottom row describes the results of the technique proposed in our paper.

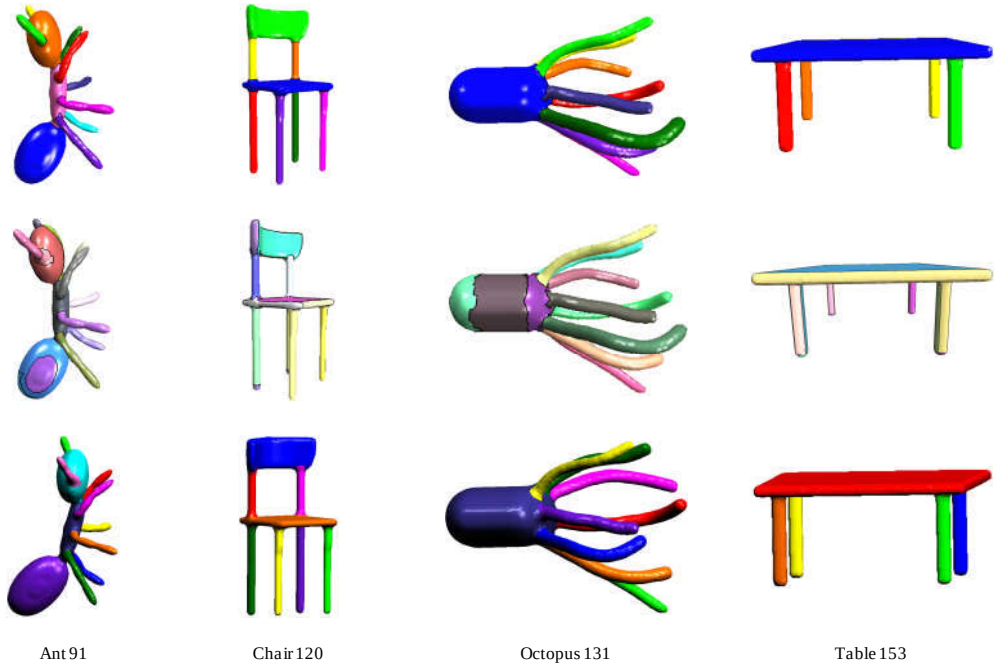

Fig. 9: The top row shows the benchmarks obtained using the manual in [6]. The middle row presents the surface mesh segmentation results achieved using the methods proposed in [8] and [24]. The bottom row describes the results of the technique proposed in our paper.

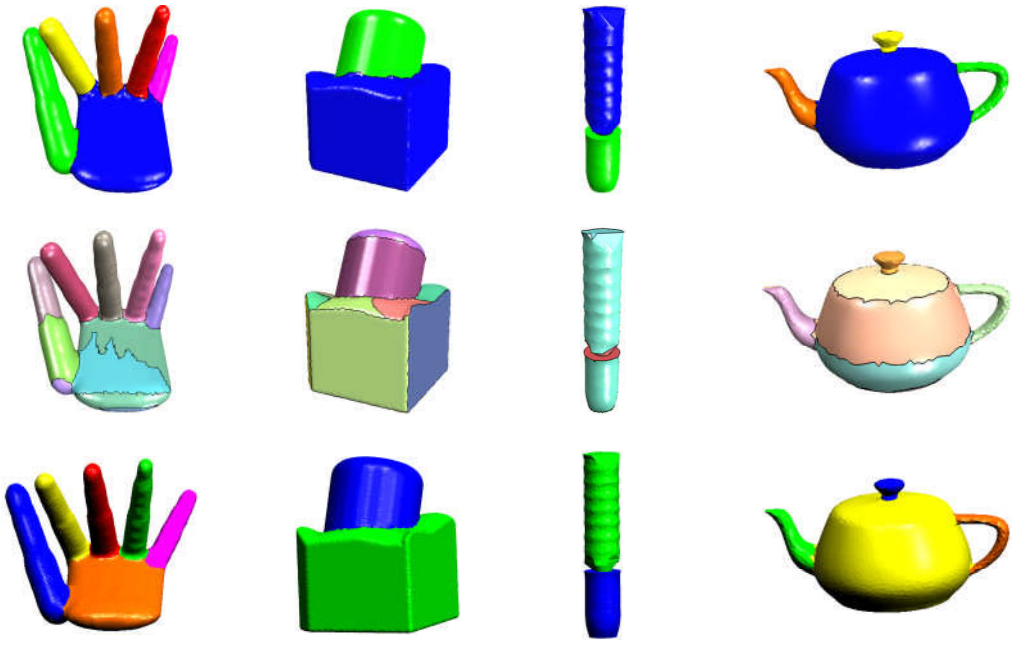

Fig. 10: The top row shows the benchmarks obtained using the manual in [6]. The middle row presents the surface mesh segmentation results achieved using the methods proposed in [8] and [24]. The bottom row describes the results of the technique proposed in our paper.

Our future work includes decomposition of a mesh with hollow regions where we need to decompose along convex boundaries between concave boundaries. This can be done by evaluating the convexity of each segment R_i of an input mesh.

Mesh	No. of tets	No. of proxies	Tet-mesh seg. (sec.)	Surface faring (sec.)	Total time (sec.)
Ant 91	43,490	17	123.875	0.25	124.125
Chair 120	39,883	18	77.719	0.312	78.031
Octopus 131	37,520	12	84.469	0.266	84.729
Table 153	44,645	26	72.61	0.14	72.75
Hand 189	55,994	10	98.28	0.891	99.171
Machine 338	83,093	15	84.75	2.844	87.594
Bearing 360	8,409	7	2.25	0.001	2.251
Vase 367	16,267	7	12.438	0.031	12.469

Tab. 1: Timing statistics.

ACKNOWLEDGMENT

The authors would like to thank Mr. Dongming Yan for providing the surface mesh segmentation software used here. Thanks also go to Mr. Xiaobai Chen for providing the basic meshes and the benchmarks adopted (http://segeval.cs.princeton.edu/). Author Caiyun Yang was supported through the “Global Center of Excellence for Mechanical Systems Innovation”, Global COE Program under the auspices of Japan’s Ministry of Education, Culture, Sports, Science and Technology.

REFERENCE

[1] Agarwala, A.; Dontcheva, M.; Agarwala, M.; Drucker, S.; Colburn, A.; Curless, B.; Salesin, D.; Cohen, M.: Interactive digital photomontage, ACM Transactions on Graphics, 23(3), 2004. doi:10.1145/1015706.1015718

[2] Attene, M.; Falcidieno, B.; Spagnuolo, M.; Hierarchical mesh segmentation based on fitting primitives, The Visual Computer, 22(3), 2006, 181–193. doi:10.1007/s00371-006-0375-x

[3] Attene, M.; Mortara, M.; Spagnuolo, M.; Falcidieno, B.: Hierarchical convex approaximation of 3d shapes for fast region selection, Computer Graphics Forum, 27(5), 2008, 1323 – 1332. doi:10.1111/j.1467-8659.2008.01271.x

[4] Boykov, Y.; Jolly, M.: Interactive graph cuts for optimal boundary region segmentation of objects in n-d images, Computer Vision, 1, 2001, 105–112. doi:10.1109/ICCV.2001.937505

[5] Boykov, Y.; Kolmogorov, V.: Energy Minimization Methods in Computer Vision and Pattern Recognition, volume 2134, Springer Berlin / Heidelberg, 359–374.

[6] Chen, X.; Golovinskiy, A.; Funkhouser, T.: A benchmark for 3d mesh segmentation, in: Proc. of the ACM SIGGRAPH 2009, 28(3). doi:10.1145/1531326.1531379

[7] Cohen, E.; Singh, M.: Geometric determinants of shape segmentation: Tests using segment identification, Vision Research 47, 2007, 2825–2840. doi:10.1016/j.visres.2007.06.021

[8] Cohen-Steiner, D.; Alliez, P.: Desbrun M.: Variational shape approximation, ACM Transactions on Graphics 23(3), 2004, 905–914. doi:10.1145/1015706.1015817

[9] Desbrum, M.; Meyer, M.; Schroder, P.; Barr, H.: Implicit fairing of irregular meshes using diffusion and curvature flow, in: Proc. of the 26th annual conference on Computer graphics and interactive techniques 1999, 317–324. doi:10.1145/311535.311576

[10] Julius, D.; Kraevoy, V.; Sheffer, A.: D-charts: Quasi developable mesh segmentation, Computer Graphics Forum 24(3), 2005, 981–990. doi: 10.1111/j.1467-8659.2005.00883.x

[11] Kobbelt, L.; Campagna, S.; Vorsatz, J.; Seidel, H.: Interactive multi-resolution modeling on arbitrary meshes, in: Proc. of the ACM SIGGRAPH, 1998. doi:10.1145/280814.280831

[12] Kim, Y.: Convex decomposition and solid geometric modeling, Ph.D. thesis, Starnford University, USA, 1990.

[13] Kobbelt, L.: Discrete fairing, in: Proc. of the Seventh IMA Conference on the Mathematics of Surfaces 1997, 101–131.

[14] Katz, S.; Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and cuts, ACM Transactions on Graphics 22(3), 2003, 954–961. doi:10.1145/882262.882369

[15] Kraevoy, V.; Julius, D.; Sheffer, A.: Shuffler: Modeling with interchangeable parts, The Visual Computer, 2007.

[16] Lien, J.-M.; Amato, N.: Approximate convex decomposition of polyhedra, in: Proc. of the ACM Symposium on Solid and Physical Modeling (SPM) 2007. doi:10.1145/1186415.1186418

[17] Lu, L.; Choi, Y.-K.; Wang, W.; Kim, M.-S.: Variational 3d segmentation for bounding volume computation, Computer Graphics Forum, 26(3), 2007, 329–338. doi:10.1111/j.1467-8659.2007.01055.x

[18] Li, Y.; Sun, J.; Tang, C.; Shum, H.: Lazy snapping, ACM Transaction on Graphics 23(3), 2004, 303–308. doi:10.1145/1015706.1015719

[19] Liu, R.; Zhang, H.: Segmentation of 3d meshes through spectral clustering, in: Proc. of Pacific Graphics 2004, 298–305. doi:10.1109/PCCGA.2004.1348360

[20] Liu, R.; Zhang, H.; Shamir, A.; Cohen-or, D.: A part-aware surface metric for shape analysis, Computer Graphics Forum 28(2), 2009. doi:10.1111/j.1467-8659.2009.01379.x

[21] Si, H.: http://tetgen.berlios.de/, 2007.

[22] Taubin, G.: A signal processing approach to fair surface design, in: Proc. of the ACM SIGGRAPH 1995, 351–358. doi:10.1145/218380.218473

[23] Waco, D.; Kim, Y.: Geometric reasoning for matching feature using convex decomposition, Computer-Aided Design 26(6), 1994, 477–489. doi:10.1016/0010-4485(94)90069-8

[24] Yan D.; Liu, Y.; Wang, W.: Quadric surface extraction by variational shape approximation, in: Proc. of Geometric Modeling and Processing (GMP) 2006, volume 4077, 73–86. doi:10.1007/11802914_6

[25] Yang, C.; Suzuki, H.; Ohtake, Y.; Michikawa, T.: Boundary smoothing for mesh segmentation, in: Proc. of CAD/GRAPHICS 2009, 241–248. doi:10.1109/CADCG.2009.5246896

[26] Zhang, H.; Liu, R.: Mesh segmentation via recursive and visually salient spectral cuts, in: Proc. of Vision, Modeling and Visualization 2005, 429–436.