COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2011; **22**:335–342
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.422

SPECIAL ISSUE ARTICLE

iCutter: a direct cut-out tool for 3D shapes

Min Meng, Lubin Fan and Ligang Liu*

Department of Mathematics, Zhejiang University, Hangzhou 310027, China
State Key Laboratory of CAD & CG, Zhejiang University, Hangzhou, China

ABSTRACT

We present a novel sketch-based tool, called *iCutter* (short for *intelligent cutter*), for cutting out semantic parts of 3D shapes. When a user performs a cutting task, he only needs to draw a freehand stroke to roughly specify where cuts should be made without much attention. Then, iCutter intelligently returns the best cut that meets the user's intention and expectation. We develop a novel scheme for selecting the optimal isoline from a well-designed scalar field induced from the input stroke, which respects the part saliency as well as the input stroke. We demonstrate various examples to illustrate the flexibility and applicability of our iCutter tool. Copyright © 2011 John Wiley & Sons, Ltd.

KEYWORDS

mesh cutting; semantic parts; harmonic field; isolines

***Correspondence**

Ligang Liu, Department of Mathematics, Zhejiang University, Hangzhou 310027, China.
E-mail: ligangliu@zju.edu.cn

Supporting information may be found in the online version of this article.

1. INTRODUCTION

Decomposing 3D shapes into semantic parts is an important ingredient in computer graphics and has received increasing research attention during the last decade [1,2]. However, semantic parts can hardly be measured quantitatively as human perception on semantics is subjective. Therefore, interactive tools for mesh decomposition, which can aid the user to extract semantic parts at ease, have become increasingly popular in recent years [3–6].

Existing interactive interfaces for mesh decomposition can be classified into two categories: *boundary*-based interfaces [3,5], which require the user to specify points or strokes at the desired cutting boundary, and *region*-based interfaces [4,6], which allow the user to freely draw strokes on the foreground and background regions. The latter provides the user little control on the cut, thus generally requiring the user to put *extra* effort on specifying more hints to refine the cut [4,7].

Before he performs the cut, the user generally knows what he wants to cut out from a model and roughly knows where the cut should be made. In this paper, we present a novel interactive tool, called *iCutter* (short for *intelligent cutter*), for cutting out semantic *parts* of 3D shapes. The user roughly draws a stroke around a desired cut. From the user's perspective, the meaning of the input stroke is "I want to cut the object along the best cut around here". From the tool's perspective, it specifies a hint and constraint on where the cut must pass near the stroke. Then, our iCutter tool intelligently returns satisfied cuts, which respect the user's intension, as shown in Figure 1. This tool is so intelligent to meet the user's intention and expectation that the user roughly draws the strokes without paying much attention.

The contributions of our iCutter are as follows:

- iCutter provides an intuitive and easy-to-use tool for cutting out semantic parts from 3D shapes. The user only needs to draw rough strokes without paying much attention. A number of experimental results show that iCutter produces satisfactory cutting results and provides users a favorable experience on cutting out meshes in both ease of use and cutting quality, which provides a direct cut-out tool for 3D shapes.
- We have developed a novel and non-trivial technique for computing a cut from the input stroke. The core of the technique is to compute a scalar field that is obtained by averaging some harmonic fields defined by the estimated foreground and background pairs induced from the sampling points on the stroke.
- We have conducted two user studies to evaluate the performance of iCutter. The studies show that iCutter is not only better than the previous boundary-based user interfaces but also it is easier to use than the region-based user interfaces like the easy mesh cutting.

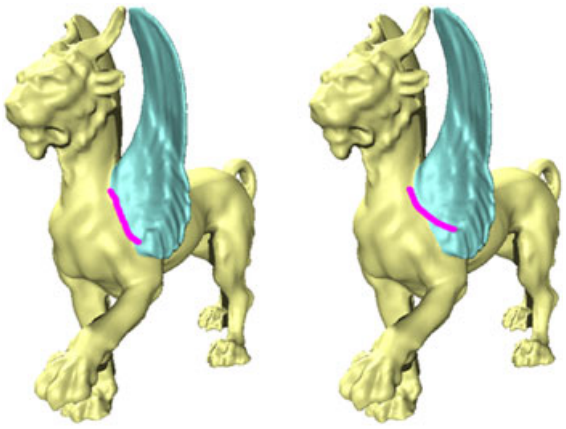

Figure 1. Two mesh cutting results produced by our iCutter tool. The user only needs to roughly draw a stroke (in purple) around the desired cutting boundary. iCutter obtains the similar cuts no matter how the user draws the strokes in different ways.

2. RELATED WORK

There has been a number of work on mesh decomposition [1,2]. We only review relevant work on interactive mesh cutting and their user interfaces.

2.1. Boundary-Based Interactive Mesh Cutting

Boundary-based user interfaces require the user to input some hints that are on or close to the desired cuts. The work of [2,8] requires the user to specify a sequence of points, or draw a scree-space "lasso" [9], or paint a stroke or path [3,10] on/near the desired cutting boundaries. Then, the cuts are automatically obtained by connecting the points or closing the stroke via shortest paths. 3D geometric snake [11] is often applied to move the initial contour by considering the part salience until it settles to define the final cut position. The recent work of cross-boundary brushes [5] requires the user to draw strokes across a desired cut. However, the user has to pay much attention on the directions of strokes, and multiple strokes are often needed to obtain desired results.

Similar to the interfaces used in [3,10], our iCutter tool allows the user to draw rough strokes on the mesh to specify where cuts should be made. Instead of using the shortest paths as the cuts, we estimate some foreground and background points at both sides of the stroke and compute a scalar field on the basis of these points. The cut is then obtained by finding an optimal isoline from the scalar field [5].

2.2. Region-Based Interactive Mesh Cutting

Region-based user interfaces require the user to draw freehand strokes on the foreground and background regions. Starting from the initial seeds, various techniques, such as region growing [4], graph cut [12,13], random walks [6,14], and hierarchical aggregation [15], have been proposed to extract the foreground and background regions.

One drawback of these approaches is that the user has lack of control on refining the cutting boundaries. These approaches do not always guarantee that the exact cuts that the user wants are obtained, thus also allowing the user to make *extra* effort on interactively refining the cuts as a postprocessing. The easy mesh cutting [4] allows the user to draw freehand strokes to replace a segment of the cut. In the mesh snapping [6], the user is allowed to specify vertices or triangles where he wants the cut to pass. A recent work [7] allows both soft and hard constraint inputs to modify the cuts.

Our iCutter tool shows that the user can directly cut out the mesh through specifying rough strokes on the cuts instead of making it as a postprocess of those region-based user interfaces.

3. THE ICUTTER TOOL

To execute a cutting, the user draws a stroke on the mesh to specify where cuts should be made, as shown in Figure 1.

3.1. Adaptive Sampling

The input stroke in the image plane is denoted as $\mathcal{S}$. Thus, $\mathcal{S}$ roughly indicates the position and the direction of the cut. The basic idea is to sample some pairs of foreground and background points along $\mathcal{S}$, which provide initial seeds for computing the cut.

3.1.1. Stroke sampling.

First, we uniformly sample a sequence of n points (every $d^{ST} = 5$ pixels), denoted as $\mathcal{S} = \{S_1, S_2, \cdots, S_n\}$, on $\mathcal{S}$ (Figure 2 (left)). By projecting these sampling points

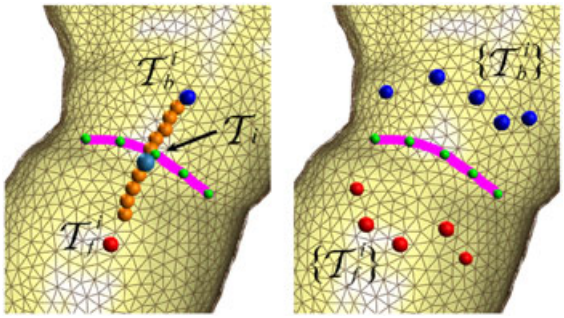

Figure 2. Illustration of adaptive sampling. Left: a sequence of points $\{T_1, T_2, \cdots, T_n\}$ (in green) are uniformly sampled on the stroke $\mathcal{T}(\mathcal{S})$ (in purple). For each T_i, we sample a set of points (in orange) along its normal direction and choose the one (in light blue) with minimum negative curvature as the feature point. Then, the foreground and background candidate points T_f^i (in red) and T_b^i (in blue) for T_i is computed. Right: all the foreground and background candidate points $\{T_f^i\}_{i=1}^n, \{T_b^i\}_{i=1}^n$ are shown.

DOI: 10.1002/cav

onto the mesh surface $\mathcal{M}$, we have a stroke on $\mathcal{M}$: $\mathcal{T} = \{T_1, T_2, \cdots, T_n\}$.

3.1.2. Feature points.

At each point S_i, we sample $m(=5)$ points (every d_i^{NR} pixels), $\{S_1^i, \cdots, S_m^i\}$, along its normal direction N_i. Similarly, we sample m points along $-N_i$ as $\{S_{-1}^i, \cdots, S_{-m}^i\}$. Denote $S_0^i = S_i$. Now we have $2m+1$ sample points around S_i as $\mathcal{S}_i = \{S_{-m}^i, \cdots, S_m^i\}$. Denote their projected points on $\mathcal{M}$ as $\mathcal{T}_i = \{T_{-m}^i, \cdots, T_m^i\}$. If T_i $(= T_0^i)$ is a concave vertex, we set $d_i^{NR} = 3$; otherwise $d_i^{NR} = 5$. We compute the principal curvatures of vertices $T_j^i (j = -m, \cdots, m)$ on $\mathcal{M}$ and find the vertex with the minimum negative curvature as $T_k^i (-m \le k \le m)$ (Figure 2 (left)).

3.1.3. Foreground/background candidate points.

In the image plane, we compute two points S_{f}^i and S_{b}^i with distance $d_i^{FB}(= (m+|k|)d_i^{NR})$ to S_k^i along N_i and $-N_i$, respectively. Then, we project S_{f}^i and S_{b}^i onto the mesh $\mathcal{M}$ and obtain T_{f}^i and T_{b}^i. The vertices T_{f}^i and T_{b}^i are regarded as foreground and background candidate points, respectively. Thus, we obtain a set of foreground vertices $T_{\mathrm{f}}^i (i = 1, 2, \cdots, n)$ and background vertices $T_{\mathrm{b}}^i (i = 1, 2, \cdots, n)$ on the mesh, as shown in Figure 2 (right).

3.2. Scalar Field

iCutter computes an isoline of a scalar field over the mesh as the cutting boundary. We compute the scalar field on the basis of some harmonic fields induced from the foreground and background candidate points.

A harmonic field is a scalar field which is the solution to the Laplace equation $\Delta u = 0$ subject to some Dirichlet boundary constraints. The discretization of the Laplace equation leads to a sparse linear system $Lu = b$, where the matrix L represents the discrete Laplace–Beltrami operator matrix [16]

$$L_{ij} = \begin{cases} -w_{ij}, & j \in N(i), \\ \sum\limits_{k \in N(i)} w_{ik}, & j = i, \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $N(i)$ is the one-ring neighbors of vertex i.

3.2.1. Naive harmonic field.

As in [5,13], we can regard all $T_{\mathrm{f}}^i, T_{\mathrm{b}}^i (i = 1, \cdots, n)$ as foreground and background points, respectively. A naive harmonic field F^0 can be obtained by solving the Laplace equation with the following Dirichlet boundary conditions:

$$\begin{cases} u_v = 1, v = T_{\mathrm{f}}^i, i = 1, \cdots, n, \\ u_v = 0, v = T_{\mathrm{b}}^i, i = 1, \cdots, n \end{cases} \tag{2}$$

We can see that the Dirichlet boundary conditions are regarded as hard constraints in the system. However, it is not guaranteed that $T_{\mathrm{f}}^i, T_{\mathrm{b}}^i (i = 1, \cdots, n)$ correctly located in the foreground/background regions. If there is one incorrect foreground and background pair, the harmonic field F^0 will be incorrect and thus result in incorrect segmentation (Figure 3 (left)).

3.2.2. Our scalar field.

Instead, we compute a harmonic field F_i for each pair of points T_{f}^i and T_{b}^i by solving the Laplace equation with the following Dirichlet boundary conditions:

$$\begin{cases} u_v = 1, v = T_{\mathrm{f}}^i, \\ u_v = 0, v = T_{\mathrm{b}}^i \end{cases} \tag{3}$$

for $i = 1, \cdots, n$. Then, we compute a weighted averaged scalar field F as

$$F = \frac{1}{\sum_{i=1}^n \mu_i} \sum_{i=1}^n \mu_i F_i \tag{4}$$

where $\mu_i = \frac{1}{d_i + 0.1}$ and $d_i = \left| S_0^i - S_k^i \right|$.

The well-designed scheme of computing the scalar field in (4) has a few advantages. First, the individual harmonic field F_i has different reliability measured by μ_i. The closer the sampling point is to the local feature, the more reliable the foreground/background pair. Second, most of the foreground/background pairs $T_{\mathrm{f}}^i, T_{\mathrm{b}}^i$ are correct to respect the input stroke. Third, there is only a small number of incorrect pairs $T_{\mathrm{f}}^i, T_{\mathrm{b}}^i$ which has little effect on the weighted

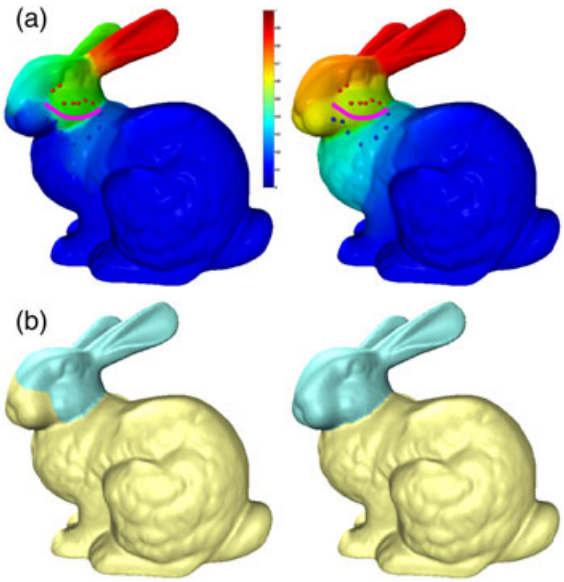

Figure 3. Comparison between the naive harmonic field (left) and our scalar field (right). (a) The harmonic fields; (b) The corresponding cutting results. There are a few pairs of sampling points incorrectly indicating the foreground and background along the input stroke.

averaged scalar field. Figure 3 (right) shows our scalar field and the correct segmentation result even if there are a few incorrect foreground/background pairs.

3.2.3. Geometry aware harmonic field.

In the discrete Laplace equation, we modify the weighting scheme on the basis of normal variation [13] to make the scalar values change abruptly along concave regions as $w_{ij} = \gamma\left(1 + \frac{\alpha_{ij}}{\text{avg}(\alpha_{ij})}\right)^{-1}$ where α_{ij} is the angle between the normals n_i and n_j of vertices v_i and v_j, respectively, $\text{avg}(\alpha_{ij})$ is the average of all α_{ij}, and $\gamma = 0.1$ if either v_i or v_j is concave; otherwise, $\gamma = 1$.

The improved weighting scheme generates better harmonic field, which reflects the concave regions, than the one used in [5], as shown in Figure 4.

3.3. Computation of Cutting Boundary

Similar to [5], we compute a set of isolines and choose the best one as the cutting boundary. Specifically, we compute N (we set $N = 15$) isolines $\{I_1, \cdots, I_N\}$ (with isoline I_i having the iso-value $i/(N+1)$) on the generated scalar field, as shown in Figure 5. We develop a novel scheme to select the best isoline on the basis of two factors: centerness and curvature concavity.

3.3.1. Centerness.

The best cut is expected to be close to the middle of the isolines. We use the same centerness measurement $\Phi_i = e^{-\frac{(i-t)^2}{2t^2}}, t = N/2$ as in [5].

3.3.2. Concaveness.

We also expect the best cut to run across the concave shape areas. We uniformly sample a sequence of points

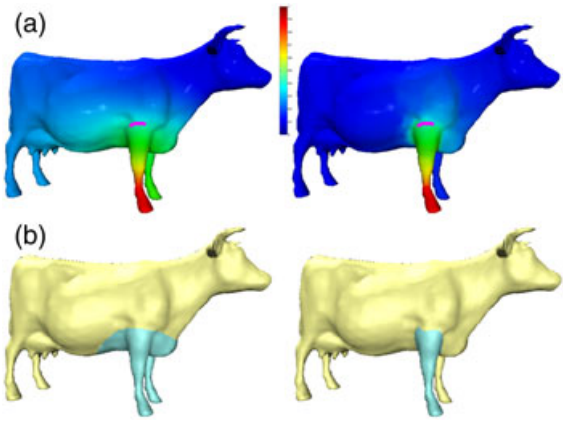

Figure 4. Comparison between the scalar fields using the weights in [5] (left) and our new weights (right). (a) The harmonic fields; (b) The corresponding cutting results.

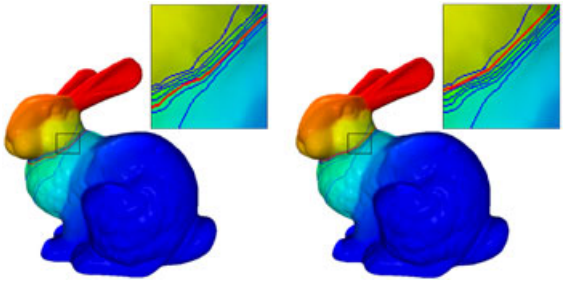

Figure 5. Comparison of isoline selection between the method of [5] (left) and our method (right). The isoline candidates are shown in blue. The best isolines are shown in red.

P_j^i on each isoline I_i and compute their minimum negative curvatures c_j^i. Then, the averaged minimum negative curvature of I_i is computed as $\gamma_i = \frac{1}{n}\sum_j c_j^i$. Denote $\gamma_{\min} = \min_{1 \le i \le N} \gamma_i$ and $\overline{\gamma} = \frac{1}{N}\sum_{i=1}^{N} \gamma_i$. We define a curvature concavity to measure how much the isoline respects the minimal rule as

$$\Gamma_i = e^{-k_\gamma \frac{(\gamma_i - \gamma_{\min})^2}{2(\overline{\gamma} - \gamma_{\min})^2}} \tag{5}$$

where the weight k_γ is set to 5 in our system.

Metric. Combining the above two factors, the metric for measuring each isoline is defined as

$$\Omega_i = \Phi_i \Gamma_i, i = 1, 2, \cdots, N \tag{6}$$

The isoline with the largest metric is selected as the best cut. The isoline is smooth, but might not follow all salient features of the mesh. As a postprocess, we can adopt some geometric snake approaches to refine the cutting boundary [4,6].

The concaveness measurement proposed in [5] only considers the length concaveness of the isolines whereas our scheme considers minimum negative curvatures and works better, as shown in Figure 5.

4. EXPERIMENTAL RESULTS

We have tested our iCutter tool to cut out a number of models and will illustrate the applicability and flexibility of the tool in this section (also see the accompanying video). All the examples presented in this paper were made on a dual-core 3-GHz machine with 4-G memory.

Thanks to the adaptive sampling scheme, the scalar field is not sensitive to the input strokes (their directions and lengths). Actually, the user does not have to pay much attention to how precisely the stroke lies to the exact cut. In Figure 1, two users adopt iCutter to cut out the left wing from the feline model by drawing two different strokes, respectively. The tool returns much similar cutting results.

DOI: 10.1002/cav

The harmonic field is insensitive to noise on the model (Figure 6 (middle)) and is invariant to poses of the model (Figure 6 (right)). iCutter can cut out various semantic parts with complex boundary features, as shown in Figure 7.

iCutter can also cut out the local semantic parts from the models. As shown in Figure 8, the user successfully cuts out the ear and nose from the Plank head model by specifying one stroke, respectively.

Generally, the user only needs to draw one stroke to cut out the semantic parts from models. In some complicated cases (e.g., high genus models), one stroke is not enough to obtain the desired cut. iCutter allows the user to draw other strokes to refine the cut in an interactive manner. This can be easily performed by adding more foreground and background candidate points in computing the scalar field. As shown in Figure 9, the user draws multiple strokes to refine the cutting results successively.

The cross-boundary brushes [5] only needs to compute one harmonic field. Our algorithm needs to compute a few harmonic fields to obtain the scalar field. As in previous work, we use the fast Cholesky factorization updating technique [17] for dynamic updating of the harmonic fields. Specifically, we have to compute the first harmonic field as in [5], then we only need to update the harmonic field dynamically with an extra computation cost no more than 10%.

For a mesh with less than $10k$ vertices, iCutter can return the cutting results in an interactive rate. To evaluate the performance of iCutter, we test iCutter on meshes with $20k - 50k$ vertices in our experiments. Table I lists the running time in our experiments shown in the paper. For

Figure 6. iCutter is robust to users' inputs, noise on the object (middle), and object poses (right).

Figure 7. iCutter can cut out the different parts with complex features on the Neptune model.

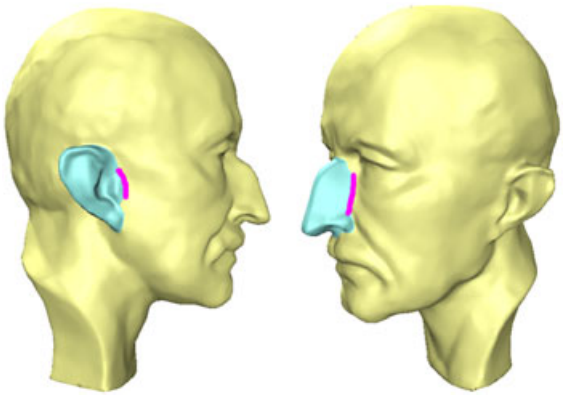

Figure 8. iCutter can cut out the local parts (ear, nose) on the Plank model.

Figure 9. The user draws multiple strokes (from left to right) to successively refine the cutting results.

Table I. Running time (in milliseconds) for the examples shown in the paper.

Model	# Vertex	RT_1	RT_2	RT_3
Feline (Figure 1)	49 864	952	921	49
Bunny (Figure 3(a))	34 839	842	858	47
Cow (Figure 4(b))	6938	172	141	3
Armadillo (Figure 6)	25 193	749	484	32
Plank (Figure 6)	25 445	609	546	32
Neptune (Figure 9)	28 052	687	561	31

RT_1, RT_2, and RT_3 denote the computation time of sampling, scalar field, and isoline selection, respectively.

large meshes, we can simplify the meshes by decimation as a preprocess in our system.

4.1. User Study

To further evaluate the performance of iCutter, we have conducted two user studies.

Ground-truth corpus. We have selected 70 models from the Princeton segmentation database [2] as the ground-truth corpus. Each model is associated with several images describing the required cutting part, which were shown to the subjects in the user studies. We divided the models randomly into 14 sets with five models in each set. Each subject was assigned to segment the models of one set.

First user study. The first study compared the performance of three boundary-based sketching cutting tools, that is, the mesh scissor [3], the cross-boundary brush [5], and our iCutter. Each subject was assigned to cut out the required parts out of three models in the set by using the different algorithms without knowing the order of the algorithms. Then, the subject was assigned to fill out a short questionnaire. Specifically, he was asked to select the best one of the three algorithms in six aspects: ease of use, accuracy, efficiency, stability, user intention, and user intuition.

Second user study. The second study compared the performance of our iCutter tool (boundary-based interfaces) with the easy mesh cutting [4] (region-based interfaces). Each subject was assigned to cut out the required parts out of the other two models in the set using iCutter and easy mesh cutting, respectively. Then, the subject was asked to vote for the tool (interface) he likes better. Also, he was asked to select the tool that obtained the better segmentation results.

Analysis. A total of 24 subjects participated in the studies. The subjects are men and women between the ages of 20 and 35 years. In [2], two criteria, the cut discrepancy (*CD*) and the rand index (*RI*), were used to measure the similarity between a segmentation and a ground truth. In the first user study, we adopt the normalized cut discrepancy $NCD = 1 - CD$ and *RI* to measure the accuracy of the obtained cutting results by the subjects with respect to the ground-truth segmentations. Figure 10 shows the evaluation of averaged *NCD* and *RI* and the variance of *RI* computed across all the models for each algorithm. The figures show that iCutter obtains better results and is more stable than the mesh scissor and the cross-boundary brushes. The result of the questionnaire is shown in Figure 11, which shows that iCutter performs better than the other two tools in most of the aspects except in efficiency.

In the second user study, 84.6% of the subjects preferred to use our iCutter whereas 15.4% of the subjects chose the easy mesh cutting. This is because even if the easy mesh cutting can obtain the approximated segmentation

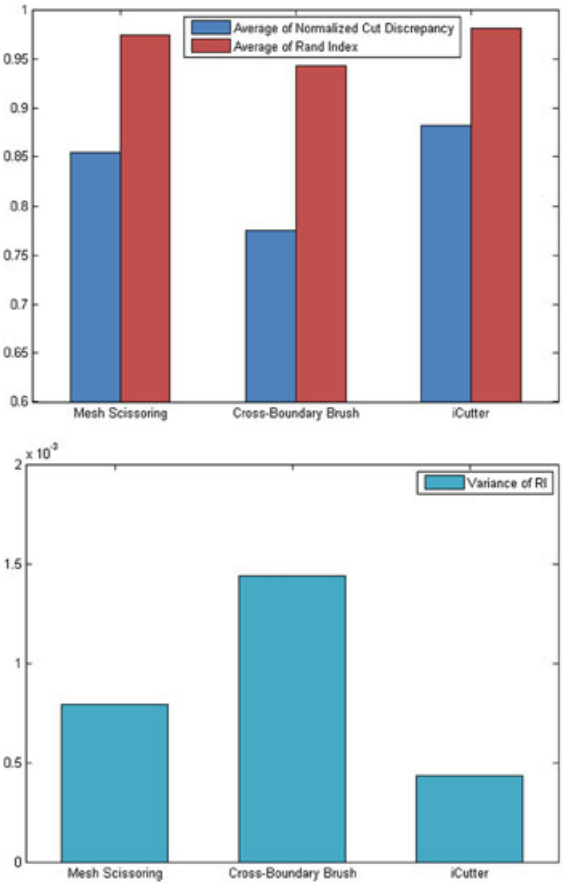

Figure 10. Evaluation of averaged normalized cut discrepancy and the rand index (left) and the variances of the rand index (right) in the first user study.

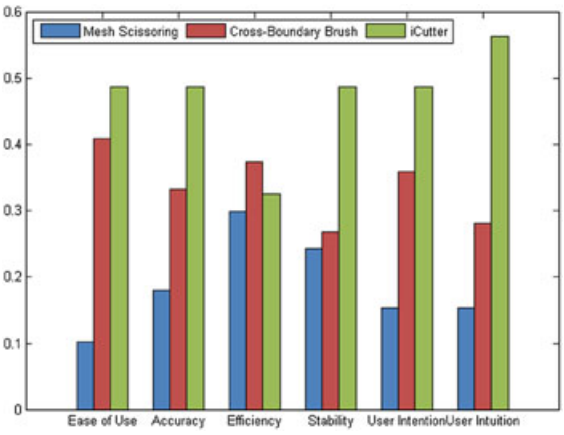

Figure 11. The result of the questionnaire in the first user study.

results, it is always required to refine the boundaries interactively. Almost half of the subjects selected iCutter that obtained the better segmentation results whereas the other half selected the easy mesh cutting, which means that both tools obtain much similar segmentation results.

DOI: 10.1002/cav

4.2. Limitations

Our iCutter tool suffers some limitations due to the nature of harmonic field. One drawback is that it is difficult to cut out parts (e.g., a short segment) from smooth surfaces (e.g., a cylinder). The other drawback is that this tool is not suitable for cutting out the patch-type components. The user might have to specify many strokes to cut out a patch [5]. However, both drawbacks can be alleviated to some extent by combining with other interactive segmentation tools.

5. CONCLUSION

We present an easy-to-use tool for interactive mesh cutting. The user roughly draws a stroke near the cut and the tool returns the desired cut result meeting his intention. A novel scheme based on isoline selection from a well-designed scalar field, which is computed by averaging a set of geometry aware harmonic fields induced from the input stroke, is proposed to compute the cut. A large number of experimental results have shown that iCutter provides users a favorable experience on cutting mesh surfaces, which embodies the motif "what you draw is what you get (WYDIWYG)". As a future work, we would like to perform an intensive user study on evaluating the various interactive cutting tools.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (61070071), the 973 National Key Basic Research Foundation of China (2009CB320801), and the Fundamental Research Funds for the Central Universities (2010QNA3039).

REFERENCES

1. Shamir A. A survey on mesh segmentation techniques. *Computer Graphics Forum* 2008; **27**(6): 1539–1556.
2. Chen X, Golovinskiy A, Funkhouser T. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 2009; **28**(3): 1–12.
3. Lee Y, Lee S, Shamir A, Cohen-Or D, Seidel H-P. Mesh scissoring with minima rule and part salience. *Computer Aided Geometric Design* 2006; **22**(5): 444–465.
4. Ji Z, Liu L, Chen Z, Wang G. Easy mesh cutting. *Computer Graphic Forum (Proc. Eurographics)* 2006; **25**(3): 283–291.
5. Zheng Y, Tai C-L. Mesh decomposition with cross-boundary brushes. *Computer Graphics Forum (Proc. Eurographics)* 2010; **29**(2): 527–535.
6. Zhang J, Wu C, Cai J, Zheng J, Tai X-c. Mesh snapping: robust interactive mesh cutting using fast geodesic curvature flow. *Computer Graphics Forum (Proc. Eurographics)* 2010; **29**(2): 517–526.
7. Zhang J, Zheng J, Cai J. Interactive mesh cutting using constrained random walks. *IEEE Transactions on Visualization and Computer Graphics* 2010; **16**: 1–11.
8. Gregory AD, State A, Lin MC, Manocha D, Livingston MA. Interactive surface decomposition for polyhedral morphing. *The Visual Computer* 1999; **15**(9): 453– 470.
9. Funkhouser T, Kazhdan M, Shilane P, *et al.* Modeling by example. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 2004; **23**(3): 652–663.
10. Sharf A, Blumenkrants M, Shamir A, Cohen-Or D. Snappaste: an interactive technique for easy mesh composition. *The Visual Computer* 2006; **22**(9): 835–844.
11. Lee Y, Lee S. Geometric snakes for triangular meshes. *Computer Graphic Forum (Proc. Eurographics)* 2002; **21**(3): 229–238.
12. Brown S, Morse B, Barrett W. Interactive part selection for mesh and point models using hierarchical graph-cut partitioning, In *Proc. graphics interface*, 2009; 23–30.
13. Meng M, Ji Z, Liu L. Sketching mesh segmentation based on feature preserving harmonic field. *Journal of Computer-Aided Design and Computer Graphics (in Chinese) (Also see its technical report in English.)* 2008; **20**(9): 1146–1152.
14. Lai Y-K, Hu S-M, Martin RR, Rosin PL. Fast mesh segmentation using random walks, In *Proc. ACM Symposium on Solid and Physical Modeling*, 2008; 183–191.
15. Xiao C, Fu H, Tai C-L. Hierarchical aggregation for efficient shape extraction. *The Visual Computer* 2009; **25**(3): 267–278.
16. Zayer R, Rössl C, Karni Z, Seidel H-P. Harmonic guidance for surface deformation. *Computer Graphics Forum (Proc. Eurographics)* 2005; **24**(3): 601–609.
17. Xu K, Zhang H, Cohen-Or D, Xiong Y. Dynamic harmonic fields for surface processing. *Computers & Graphics (Proc. Shape Modeling International)* 2009; **33**(3): 391–398.

AUTHORS' BIOGRAPHIES

Min Meng was born in 1985 and is a PhD of the Department of Mathematics, Zhejiang University, China. Her main research interests include computer graphics, digital geometry processing, shape analysis.

Lubin Fan was born in 1986 and is a PhD candidate of the Department of Mathematics, Zhejiang University, China. His main research interests include computer graphics, digital geometry processing, shape analysis.

Ligang Liu was born in March 1975 and received a PhD degree in mathematics from Zhejiang University, China in 2001. He worked at Microsoft Research Asia during 2001-2004. Now he is a professor at Department of Mathematics, Zhejiang University. He paid an academic visit at Harvard University duing May 2009 and Jan. 2011. His research interests include geometric modeling, computer graphics, and image processing. His website is: http://www.math.zju.edu.cn/ligangliu

Comp. Anim. Virtual Worlds 2011; **22**:335–342 © 2011 John Wiley & Sons, Ltd.
DOI: 10.1002/cav