Dot Scissor: A Single-Click Interface for Mesh Segmentation

Youyi Zheng, Chiew-Lan Tai, and Oscar Kin-Chung Au

Abstract—This paper presents a very easy-to-use interactive tool, which we call *dot scissor*, for mesh segmentation. The user's effort is reduced to placing only a single click where a cut is desired. Such a simple interface is made possible by a directional search strategy supported by a concavity-aware harmonic field and a robust voting scheme that selects the best isoline as the cut. With a concavity-aware weighting scheme, the harmonic fields gather dense isolines along concave regions which are natural boundaries of semantic components. The voting scheme relies on an isoline-face scoring mechanism that considers both shape geometry and user intent. We show by extensive experiments and quantitative analysis that our tool advances the state-of-the-art segmentation methods in both simplicity of use and segmentation quality.

Index Terms—Interactive mesh segmentation, dot scissor, concavity aware, harmonic fields, voting.

1 Introduction

Mesh segmentation has received extensive research attention in recent years [1], [2], [3], [4], [5]. Segmentation algorithms, either automatic or interactive, are designed for specific purposes such as recognition, modeling, and rigging. While humans can easily understand shape semantics, it is extremely difficult to design shape analysis tools that automatically establish such semantics. There exists no single automatic segmentation algorithm that can consistently produce human intended results, demanding interactive tools to incorporate user's intent. This paper focuses on the design of interactive segmentation tools, presenting a sufficiently easy-to-use tool for the user to cut out meaningful shape components.

Existing interactive approaches can be roughly classified into three groups: *on-boundary brushes* (e.g., the intelligent scissors [6], [7]), *in-segments brushes* (e.g., the foreground/background brushes [2], [4], [8], [9], [10]), and *cross-boundary brushes* [5]. On-boundary tools require the user to draw strokes or click along the desired cut boundary, which may require substantial user efforts. In-segment tools made the user interaction easier by allowing the user to draw free strokes indicating the background and foreground regions. However, free strokes provide looser constraints, thus weakening the user control on the cutting boundaries. The *cross-boundary* brushes provide a nice balance between the user control and the ease of use. Executing a cut only requires the user to roughly sketch a stroke across a desired cutting boundary. Though easy to use and providing good control over cutting boundaries, the cross boundary tools have notable drawbacks. First, it often requires multiple strokes for refining the cutting boundaries locally due to the underlying harmonic field being oblivious to the shape geometry. Second, when refining a cutting boundary, strokes need to be drawn in a consistent general direction, and the stroke length may affect the segmentation result since the desired cut is assumed to run roughly across the center of the strokes.

Most of the above-mentioned interactive segmentation tools often require multiple strokes (points) to execute a single cut. The in-segment tools [2], [4], [8], [9], [10] need both foreground and background strokes; the on-boundary approaches need multiple inputs along the cut boundary; the cross-boundary brushes [5] need multiple strokes to refine a cutting boundary. The multiple inputs serve the purpose of providing more information indicating how the cutting boundary should run across the surface. For example, the foreground and background strokes indicate that the cutting boundaries should lie in between them, while the cross-boundary strokes specify that the desired cutting boundary lies roughly perpendicular to the strokes.

In this paper, we ask the question of whether it is necessary to specify these additional information to execute a single cut. Naturally, the design of interactive segmentation tools should first ponder the following question: What is the simplest interface for a user to specify a desired segmentation cut? We argue that the simplest possible way is a single mouse click. We show in this work that it is possible to achieve quality segmentations using only single mouse clicks without any additional user input.

We call our interactive segmentation tool the *dot scissor*. The user only needs to place a dot circle near where a cut is desired and click the mouse (Fig. 1). A best cut that runs *through the circle* is automatically returned. We choose not to restrict to finding a best cut that runs *through the clicked point*, for the reason of flexibility. Such a design allows the user to place the circle more casually without paying great attention. To allow precision control over clicking flexibility when desired, we let the user reduce the circle size using the mouse wheel.

• *Y. Zheng and C.-L. Tai are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: {youyi, taicl}@cse.ust.hk.*
• *O.K.-C. Au is with the School of Creative Media, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong. E-mail: kincau@cityu.edu.hk.*

Manuscript received 2 Mar. 2011; revised 16 June 2011; accepted 29 July 2011; published online 4 Aug. 2011.
Recommended for acceptance by L. Kobbelt.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2011-03-0047.
Digital Object Identifier no. 10.1109/TVCG.2011.140.

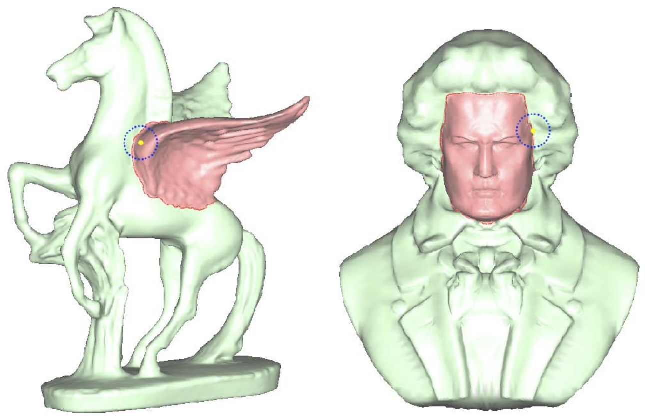

Fig. 1. Interface of our *dot scissor*. The user cuts out a component by moving a dot circle and clicking at a location where a cut boundary is desired. Our system automatically returns a best cut boundary that respects the local geometry features.

This simple interface is made possible by a directional *search-and-vote* strategy that first samples the search space to get a set of potentially good cutting boundaries lying near the clicked point and then finds a best cut from this search space. Like the cross-boundary brushes [5], our candidate cutting boundaries are isolines of harmonic fields; but in our case, no directional input information is provided to determine the directions of the isolines, demanding a search strategy. We generate the search space by defining a set of harmonic fields that propagate in different directions and sample the isolines that pass through the dot circle as candidate cuts. We adopt a variant of the concavity-aware harmonic field [11] which has strong differentiating power toward concave creases and seams, producing dense isolines at concave regions where natural boundaries lie. With the candidate cuts defined, we apply a robust face-based voting scheme that considers both shape geometry information and user intent to select the best cut.

We demonstrate the power of our single-click dot scissor through extensive experiments. We compare it with the state-of-the-art interactive segmentation techniques and show that our dot scissor advances others in both simplicity of use and segmentation quality. In addition, we quantitatively evaluate our segmentation results for all the models in the Princeton Segmentation Benchmark (PSB) and show that our tool consistently locates high-quality cuts in agreement with the ground truth segmentations using only a single click for each cut.

2 Related Work

We briefly review previous mesh segmentation methods, focusing on interactive ones. Automatic mesh segmentation has a long history and there is a large body of work in the literature. The survey paper by Shamir [12] offers an excellent overview.

2.1 Interactive Segmentation Tools

Interactive mesh segmentation was first attempted by Wong et al. [13], Stalling and Hege [14]. In their methods, the user specifies points on the cutting boundary and the cut is completed by finding the shortest path connecting the points. This interface is simple and straightforward (used also in [15]), however the user needs to rotate the model multiple times to specify the boundary points. Later, the intelligent scissor tools were introduced to allow drawing of boundary strokes [6], [7], and a closed loop is automatically formed by a geometric snake or an invariant shortest path on the mesh surface. These tools produce cutting boundaries that closely follow the user strokes, however specifying the boundary strokes require careful user attention.

Arguably, the most well-known interactive mesh segmentation methods are those that use the foreground/background snapping tools [2], [4], [8], [9]. Initial seeds are specified by drawing free strokes on the mesh to specify the foreground/background regions, then either a graph cut [2], [4] or region growing [8], [9] algorithm is employed to partition the mesh into two regions. Although intuitive to use, these tools provide limited control over the cutting boundaries. They are also sensitive to noise.

The recently introduced *cross-boundary* brushes [5] provide the *part* brush and the *patch* brush to segment the part and patch types of components, respectively. They require drawing rough strokes *across* the imaginary desired cut boundary. A single stroke input provides both location and orientation information of the desired boundary. However, since the cut boundaries are smooth isolines of fields that are oblivious to geometric features (see Fig. 12), multiple strokes are often needed to refine the boundaries. Most recently, Fan et al. [10] proposed a tool called "painting" brush for mesh segmentation, which simplifies the user effort to drawing a single stroke within the component to be cut out while providing interactive feedback. In contrast, our proposed tool simplifies the user interface to a single click near the desired cutting boundary.

2.2 Harmonic Field

Harmonic field has played an important role in surface processing. It is often used for interpolation in applications such as shape approximation and editing [16], [17]. More recently, harmonic fields have also been exploited for mesh segmentation. In interactive segmentation, the *cross-boundary* brushes use isolines as the cutting boundaries. In automatic segmentation, the recent work by Au et al. [11] introduced a concavity-aware segmentation field to capture the shape concavities along the propagation paths of the harmonic fields. In this paper, we use a simpler variant of the concavity-aware field for our dot scissor (see Section 4.1).

2.3 Voting

Voting has been widely used in recent research work [18], [19], [20] due to its robustness against local errors or missing data. To the best of our knowledge, there has been only one previous mesh segmentation method that uses voting as a basic technique, i.e., the randomized cut [1]. Seven automatic segmentation methods are employed to find the cutting boundaries which then vote for the best one. In contrast, our method is based on selecting the best cut from a set of sampled candidate isolines of harmonic fields. It is computationally efficient and does not contain any randomized process.

3 System Overview

Clearly, the simplest possible interface for segmentation is requiring only a single mouse click that roughly specifies

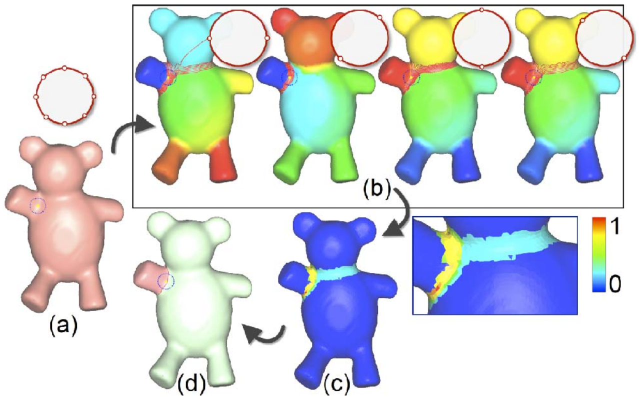

Fig. 2. Basic flow of our method. (a) The user clicks on a region where a cutting boundary is desired. Four pairs of points are sampled on the dotted circle and projected onto the surface to form constraints of the concavity-aware harmonic fields (b). All four sets of isolines extracted between pairs of constraints are candidates of cutting boundaries. The faces through which the isolines pass are assigned scores (c) and vote to elect the best isoline as the cutting boundary (d).

where a cut is desired. The cutting direction and the exact placement of the cut are automatically determined. In some situations, the user may prefer to have precise control over where the cutting boundary will pass through. Therefore, a practical interface should allow the user to choose between a precise or flexible placement of the mouse click. These considerations motivate the key ingredients of our dot scissor interface. We use a circle centered at the mouse point to indicate the region where a desired cut lies and we allow the user to vary the size of the circle to control the desired flexibility in placing the circle. To execute the cut, the user only needs to place the dot circle at a desired boundary region. Figs. 1 and 2 show the interface of the dot scissor.

An additional benefit of the variable circle size is that it reduces the need of zooming in and out during interaction. Users can use different circle sizes to segment components of different sizes. Fig. 3 shows an example of the user first using a large circle to segment out the head of the Neptune model and then uses a smaller circle to cut out a finger without zooming in. We will show that the user has considerable leeway in specifying the circle size (Fig. 8). In all our experiments, we default the circle size to be 10 percent of the window size and let the user change the size if necessary.

The placement of the dot circle indicates that the user's intent to have a cutting boundary near it. No cutting direction is specified. Hence, we need to solve the problem in which direction a good cutting boundary should run across and how to search for it. These questions motivate the design of our *search-and-vote* strategy. By using different point pairs on the dot circle as constraints, we define multiple harmonic fields propagating in different directions on the object surface. Specifically, we uniformly sample K pairs of opposite points on the dot circle and use them as boundary constraints to define K harmonic fields. From these fields, we sample K sets of isolines that intersect the dot circle to be the candidate cutting boundaries. Among them we then search for the best one as the final cut. We use $K = 4$ in our experiments, see detailed discussion in Section 5. To ensure that the sampled points on the circle are projected to unique surface points, we limit the circle size to be not smaller than 10 pixels.

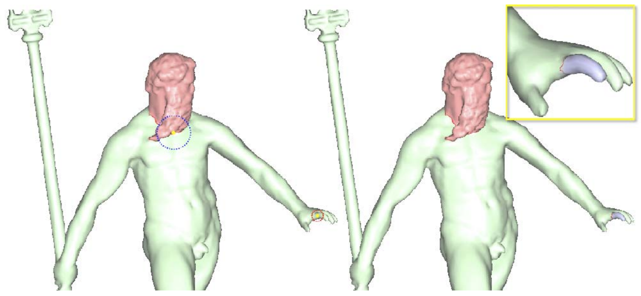

Fig. 3. Our resizable dot circle facilitates user interaction during segmentation. The user can change the circle size by mouse wheel to cut out components of different scales without zooming in and out on parts of model.

To find a best cut in the user-clicked region, we consider the collection of all the sampled isolines as raw data to vote for the best boundary cut. The basic idea is to determine which faces have high probability of lying on the best cutting boundary. We assign a face score to each face measuring its quality of being on a cutting boundary and use the face score to vote for the final best isoline. The face score is determined based on the quality of the isolines passing through the face as well as the concavity of the face. To evaluate the quality of an isoline, we combine three measures: concavity of the shape region on which the isoline lies, tightness of the isoline, and its proximity to the user clicked position. Each face will receive scores from all isolines that pass through it. Finally, to find the best isoline, each face votes by sending their scores to the isolines that pass through it and the isoline with the highest vote is returned as the best cut.

4 The Dot Scissor

We now introduce the details of our approach. Our framework consists of two main components, a directional search space sampling that constructs a set of candidate isolines as the potential cutting boundaries and a robust voting mechanism devoted to finding the best isoline as the final cutting boundary. In the following, we present the techniques used in each component.

4.1 Search Space Sampling

Our candidate cutting boundaries are isolines of harmonic fields. We adopt a variant of the concavity-aware segmentation field introduced by Au et al. [11]. By placing very small weights on edges that lie in concave regions, the weighting scheme is able to generate large field variation along concave regions. The isolines at concave regions then follow closely the concave creases and seams, forming good cutting boundaries for segmentation (in line with the Minima rule [21], [22]). To reduce sensitivity to surface noise and fine details, we remove the curvature term used in [11]. The weighting term is defined as

$$w_{ij} = \begin{cases} \frac{\|v_i - v_j\|}{\bar{e}}\Theta & \text{if either } v_i \text{ or } v_j \text{ is concave,} \\ 1 & \text{otherwise.} \end{cases} \quad (1)$$

Here, $\bar{e}$ is the average edge length and Θ is a small constant (0.01 in our experiments). Figs. 2 and 4 show examples of the segmentation fields. The constant field colors on the components demonstrate the strong differentiating power

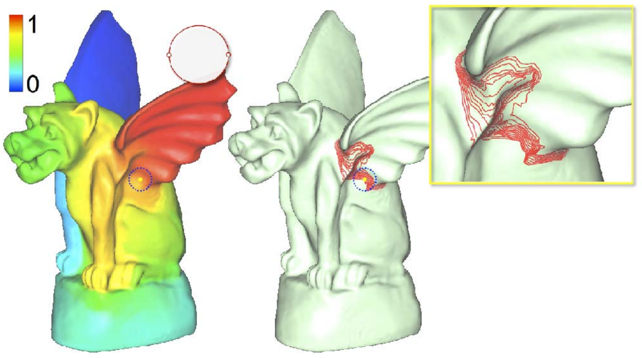

Fig. 4. The concavity-aware harmonic fields gather isolines toward local concave regions such as creases and seams. The field values are normalized to $[0, 1]$.

of the fields. In addition, Figs. 2 (last two bear examples of the top row) and 10 show that the generated fields are not sensitive to small changes in the placement and orientation of the constraints. This allows the use of only a small number of fields propagating in the major directions (four in our experiments) to sufficiently identify all desirable cutting boundaries.

To construct the candidate set of cutting boundaries, we compute a field corresponding to each pair of sampled points on the dot circle (Section 3) and extract candidate isolines from each field. For each field, we set its boundary constraints at two projected surface points (from points on the circle) with boundary values 0 and 1 (but in fact the order and magnitude does not affect the candidate isolines generation). We uniformly sample the isolines using isovalues between the two boundary values such that the sampled isolines lie between the projected point pairs, passing through the user-defined circle area. The collections of isolines sampled from all fields form the set of candidates from which the final cut will be selected.

Note that, for a given model, the computation of different harmonic fields differs only in the boundary constraints, which can be expressed as a penalty term in [23]. Hence, we employ the fast Cholesky factorization update/downdate scheme [24] for fast fields generation and achieve real-time response for interactive segmentation.

4.2 Face-Based Voting

For each harmonic field, we extract a set of isolines lying between its two constraints. All K sets of isolines form our search space. We design a robust face-based voting scheme to select the best cut from the candidate set.

We assign a *face score* s_i to each face to measure its likelihood of lying on a cutting boundary. The face score is determined by the isolines that pass through it and the face's concavity. For each candidate isoline, we analyze its quality of being a cutting boundary based on concavity of the local region on which it lies, the isoline's tightness and proximity to the user click (see details below) and assign an *isoline score* ψ to it. Each face accumulates the scores from all the isolines that pass through it, weighted by the in-face isoline segment length, as follows:

$$t_i = \frac{\sum_{j\in\Psi_i} \ell_{ij}\psi_j}{\sum_{j\in\Psi_i} \ell_{ij}}, \tag{2}$$

where ψ_j denotes the score of isoline j, ℓ_{ij} is the segment length of isoline j in face i, and Ψ_i is the set of isolines that pass through face i.

To determine the concavity of a face, we use the *summed field gradient magnitude* g as in [11], based on the observation that concave regions have larger field gradients. Specifically, to compute g_i for each face i, we simply sum up all the magnitudes of the gradients from all the K fields and linearly map them to $[0, 1]$. The face score s_i is then defined as

$$s_i = g_i * t_i, \tag{3}$$

which indicates the total votes the face i holds. Then, the faces vote for the final best cut. Each face sends its votes to each isoline that passes through it, again weighted by the in-face isoline segment length. Finally, the isoline that gets the most normalized vote (total votes divided by the isoline length) is returned as the final cut.

4.2.1 Isoline Scores

Designing robust measures for good cutting boundaries is crucial to the voting process. The score of an isoline takes into consideration both the shape geometry and the user intent. Specifically, we measure three quantities: concavity, tightness, and proximity to user click.

4.2.2 Concavity

The concavity of an isoline measures the concaveness of the region in which it lies. To avoid sensitivities to local geometric features or noise, we employ a similar idea proposed in [5], which we call the *global concavity* ς here. Specifically, given the length distribution of a set of isolines $I_1, I_2, \ldots, I_t$ from a specific harmonic field, the global concavity of an isoline is computed as

$$\varsigma_i = \frac{\sum_k f(k)\triangle_{ik}}{\sum_k f(k)}, \tag{4}$$

where $f(k)$ is a Gaussian function and $\triangle_{ik} = 2r_i - r_{i-k} - r_{i+k}$ is the multiscale stepped concavity. For details, see [5]. Note that the value ς for each isoline is computed within the set of isolines from the same field.

4.2.3 Tightness

This term measures the tightness of the segmentation boundaries (isolines). This is motivated by the shortcut rule [25], which states that, other things being equal, human perception prefers to use the shortest path to parse silhouettes. We observe that although the isolines are constrained to pass through the dot circle, some of them may form looser loops diverging far from the desired boundary region. At branching regions where multiple components conjunct, some sampled isolines may pass through multiple boundaries (e.g., the isolines passing through both the underarm and the neck of the teddy model in Fig. 2b). Such lengthy isolines are not desirable cutting boundaries and should be rejected as candidate cuts. Thus, we introduce the *tightness* term in our isoline score under the assumption that good cuts should be tight loops. For each set of isolines, we want to filter out those that are much longer than the others. To avoid locality of measure, we construct a filter using all K sets of isolines. The tightness of an isoline is defined as the ratio of

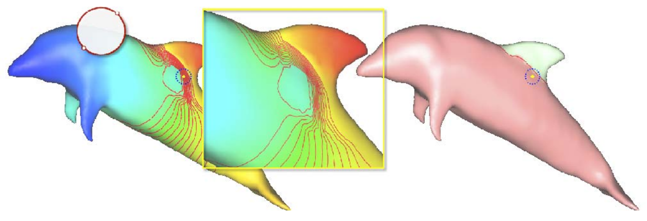

Fig. 5. Some harmonic fields may have small isolines lying on flat surface regions. We filter out such isolines by analyzing the flatness of the isolines (Section 4.2). The right image shows the final cut selected from another set of isolines not shown here.

the isoline length l_i and the minimum length l_b of all K sets of isolines, i.e.,

$$\tau_i = \frac{l_b}{l_i}. \quad (5)$$

For efficiency, we simply disregard any isoline whose length is larger than 1.5 times the minimum length among all the isolines and remove them from the voting process.

In some cases, we observe that the improper locations of constraints (projected to the surface) at flat regions lead to the field propagation getting stuck, forming swirls in the field and resulting in small isorings (Fig. 5). We consider these isolines as outliers and also remove them. To do this, we apply the simple strategy of collecting the normals of the faces through which an isoline passes and consider the isoline as lying on a flat region if more than 80 percent of the collected face normals lie on the same hemisphere. Isolines on flat regions are removed unless all the sampled isolines of a harmonic field lie on a flat region. Since this indicates that the current component to be cut out is a patch-like segment (e.g., a human face), we cancel out this filtering step.

4.2.4 *Proximity to User Click*

Though the user's click position is rough, we assume that the position conveys the intent that a cutting boundary shall lie close to it. Thus, we introduce a proximity measure. For an isoline i, we define its proximity ϑ_i to the clicked point c as the average distance of c to k sampled points $p_1, p_2, \ldots, p_k$ on the isoline. For efficiency, we use the euclidean distance here instead of the geodesic distance. We calculate this proximity measure within each set of isolines.

Finally, to calculate the score for each isoline, we linearly scale three terms to be in the same range of $[0, 1]$ and combine them. Specifically, the score for each isoline i is the product of the three terms

$$\psi_i = g(\varsigma_i) \times g(\tau_i) \times g(\vartheta_i). \quad (6)$$

Here, g is a monotonic function that controls the influence of the three terms. We set it to be $g(x) = 1/(1 + x^2)$ to encourage smaller values of all three terms. Note that the first term ς (global concavity) can be negative, therefore it was first scaled to $[-1, 1]$ and then linearly mapped to $[0, 1]$.

5 Robustness

The dot scissor has several desirable properties. In Fig. 8, we show that the segmentation results are not sensitive to the clicked position along a boundary. Similar cutting boundaries are also returned using different circle sizes as long as the dot circle covers the user-desired cutting boundary. Further, our robust directional searching and voting strategy enables the dot scissor to have very good performance even when the part to be segmented is rather casually oriented (Figs. 5 and 9). Such cases would be hard for existing techniques to specify or draw the regions of interest.

The concavity-aware harmonic fields are solved globally in a least squares sense and are stable in the presence of noise, therefore the located cut boundaries are also not sensitive to noise (Fig. 10). In Fig. 11, we show that our dot scissor is also largely insensitive to different resolutions and surface tessellations, thanks to the design of the harmonic field and our robust voting scheme.

5.1 Dot Scissor with Different Supporting Fields

The weighting scheme for the Laplacian matrix is crucial to the quality of the solved harmonic fields for our use in the interactive dot scissor. Beside the proposed concavity-aware weighting scheme, we have also used the cotangent weighting and the feature preserving weight of [26] and evaluate the segmentation results. Fig. 12 shows a comparison of the segmentation results using the different weighting schemes. It is interesting to note that the dot scissor produces reasonable quality segmentations with all these fields. But only the concavity-aware field produces boundaries that follow the geometric features well. Hence, we choose that as the underlying supporting field.

5.2 Parameter Analysis

An important parameter of our system is the number of sampled pairs on the dot circle, i.e., the parameter K, whose projected points are used as boundary constraints for the generation of the harmonic fields. It specifies how many directions to propagate the fields in search for the best cut. Fewer searching directions are desirable for computation efficiency. More searching directions would theoretically discover candidate isolines that would otherwise miss. However, we observe that, since the sampled isolines are mostly located along concave creases, they are insensitive to small change in the orientation dictated by the two constraints. This feature makes a small number of fields suffices in capturing the desired candidate isolines for subsequent robust voting process. In practice, we find that sampling in four major directions ($K = 4$) suffices to ensure that all desired cutting boundaries are covered by at least one of the computed fields (see Fig. 13 for an example).

Another parameter to evaluate is the number of isolines sampled from each field. In practice, we find sampling above 10 isolines would produce similar segmentation results since the isolines mostly gather in small concave regions. Hence, we sample a small number (15) of isolines per field for all examples in this paper. This suffices to render the method effective.

5.3 Validation

We perform a rigorous experiment to further validate the robustness of our dot scissor. Instead of user-clicked points, we execute dot scissor using points randomly chosen from boundary loops of human segmentations. The aim is to

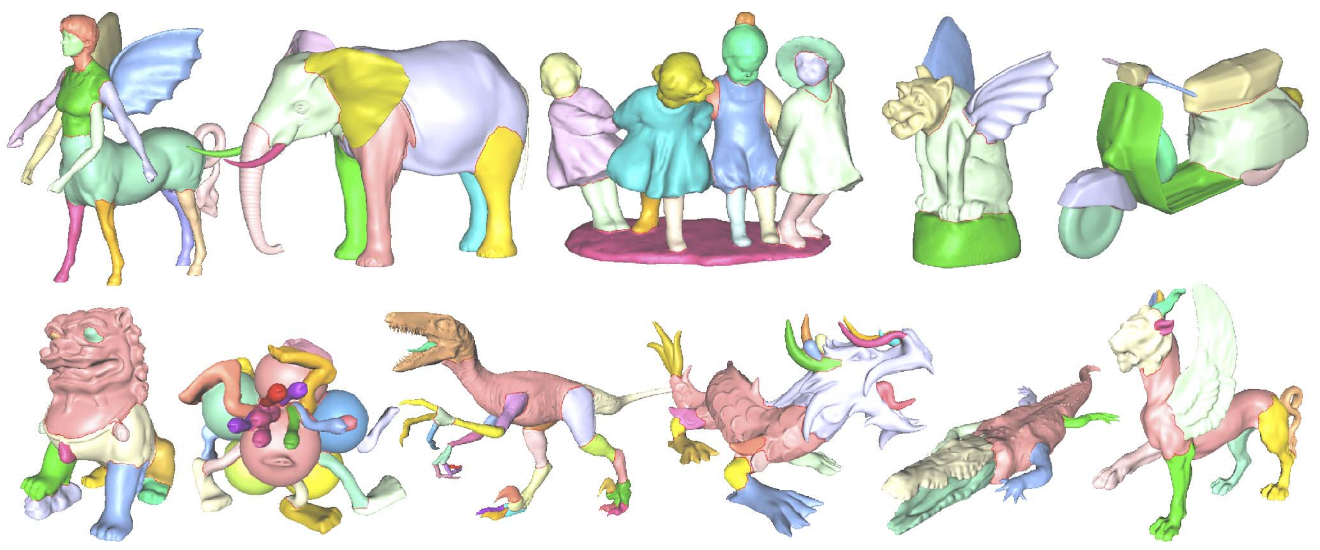
Fig. 6. Segmentation results of complex models using our dot scissor. All results were obtained using single clicks without any boundary refinement.

validate that dot scissor is able to faithfully reproduce the human segmentations. First, we collect all the human segmentation data from PSB which include different segmentations executed by different people for each of the 380 models. We then *randomly* select one human segmentation for each model and *randomly* pick a point on each of the boundary loops and apply our dot scissor using the point. To define the constraints for the fields, we use a local geodesic disk with the radius fixed at three times the average edge length and uniformly sample $\mathrm{K}(=4)$ pairs of points on the disk boundary. We apply this randomized process to all the 380 models, and quantitatively analyze the resulting segmentations using the protocals of Chen et al. [15].

Fig. 14 shows some segmentation visual examples and Fig. 15 shows the quantitative results. The experiment shows that our dot scissor performs well even with all points randomly selected and the constraints sampled with a fixed radius. For all the evaluations, segmentations using randomly selected points give slightly larger errors than segmentations using user-clicked points, except for the *cut discrepancy* (CD) error. This is because the computation of the CD error is based on proximity to the ground-truth human segmentations, and our automated segmentations try to reproduce the ground truth. In practice, we observe that errors are mainly introduced by complicated features such as hair and patch-like segments such as eyes. Nevertheless, our dot scissor is generally robust and capable of producing good segmentation results for most general models.

6 Results and Discussion

This section presents experimental results of using dot scissor to segment a large variety of models, including the entire data set of PSB [15]. Fig. 6 shows the segmentation results of some complex models. All the cuts were segmented using only single clicks. It can be observed that the cuts respect the geometric features. Each model takes an average of less than a minute to segment. More complicated models that need more cuts (e.g., the raptor and the bozbezbozzel) take under 2 minutes. Note that the dot scissor is able to segment flat patches such as the faces in the centaur, the dancing children models (Fig. 6), and the Beethoven model (Fig. 1).

6.1 Benchmark Evaluation

We use dot scissor to segment all the 19 categories of 380 models in the PSB data set. Each cut is executed with a single click, not following any specific instructions (e.g., how many parts are expected for a model), i.e., we just segment based on our perception. Fig. 7 shows samples of the segmentation

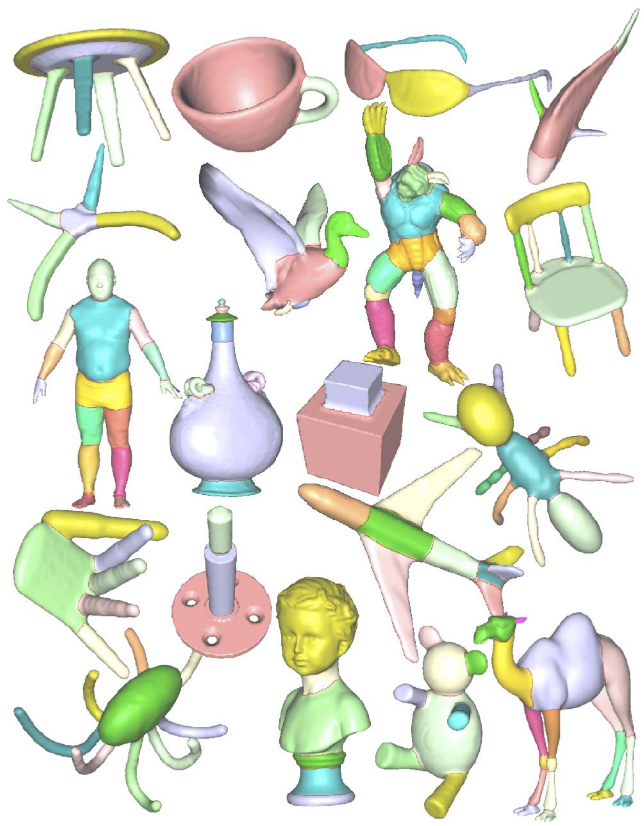
Fig. 7. Segmentation results showing a representative object from each of the 19 categories in the PSB data set.

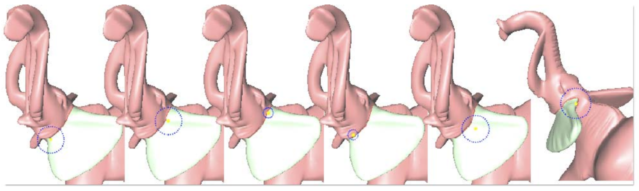
Fig. 8. Our dot scissor is not sensitive to the clicked position along a boundary. Different circle sizes also return similar boundaries as long as the circle covers the desired boundary.

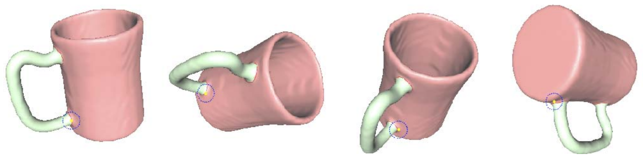
Fig. 9. The dot scissor does not require the input model to be oriented properly.

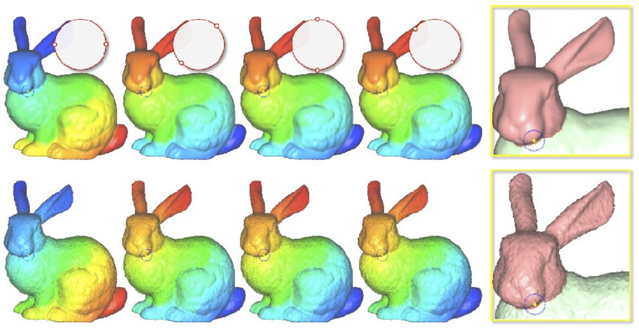
Fig. 10. Our concave-aware harmonic field is stable in the presence of surface noise. The top row shows the original model and the bottom row shows the same model with 20 percent mean edge length Gaussian noise. The four fields in each row correspond to four pairs of constraints. Observe the similar final cuts for both models (last column).

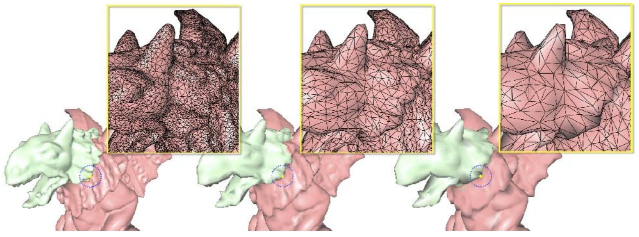
Fig. 11. The dot scissor is insensitive to different surface tessellations. (a)-(d) are Gargoyle models in different surface resolutions and tessellations. The same clicked position leads to a consistent cut.

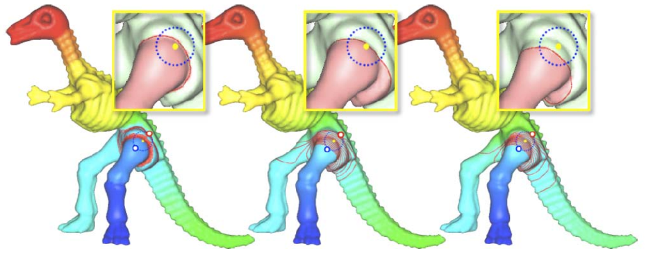
Fig. 12. Comparison of harmonic fields with different weighting schemes. (left to right) Our concave-aware field, field with cotangent weighting and field of [26]. Zoom-in images show the cutting results using these fields.

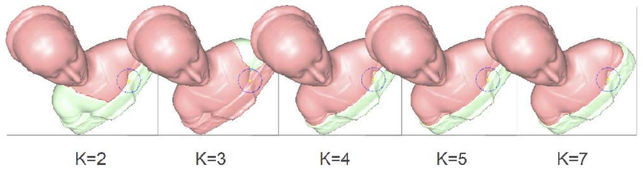

Fig. 13. The effect of different K to the final segmentation results. We found $K >= 4$ suffices to locate desired cutting boundaries in all our experiments.

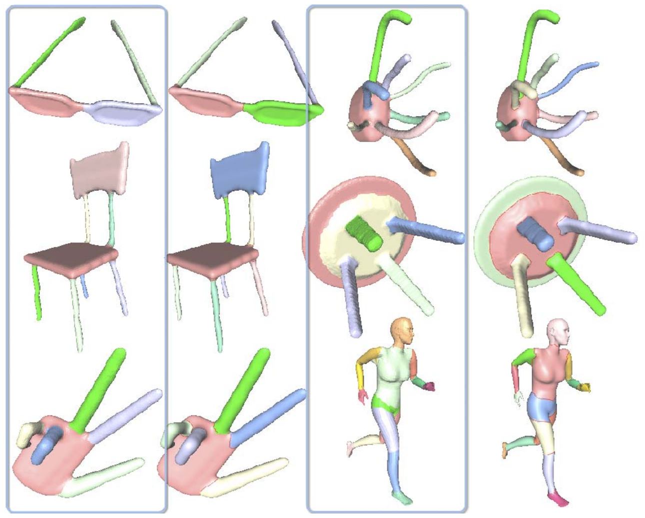
Fig. 14. We validate the robustness of dot scissor by showing that it is able to faithfully reproduce human segmentations when executed using randomly selected points on the boundary loops of human segmentations. Models enclosed by rectangles are human segmentations and the rest are segmentations produced by automatically executing dot scissor (see Section 5 for details).

results, where a representative model is selected from each category. The quantitative evaluation results in Fig. 15 show that our dot scissor gives lower or similar errors than the human segmentations in PSB. This is because the evaluation for human segmentations is carried out in a one-to-majority manner, indicating that our segmentation qualities are more consistent with the majority.

It is noteworthy that, while the cutting boundaries in our case are isolines that could pass through faces, the evaluation protocols of [15] requires the boundaries to be a subset of mesh edges (i.e., a face-based segmentation). Therefore, before evaluation, we adjust each cutting boundary by assigning a foreground/background value to each face according to ratio of the subareas divided by the cut. This adjustment approach is simple, but could produce segmentation results with zigzag boundaries that led to a relatively higher *cut discrepancy* error.

6.2 Comparison with Other Interactive Tools

We compare the dot scissor with the state-of-the-art interactive segmentation tools. Fig. 16 shows an example. The cutting boundaries using the in-segments tools of [2], [8], and [9] were obtained by drawing the same foreground/background strokes. We observe that greedy algorithms like the methods in [8] and [9] have a relatively larger sensitivity to the stroke locations. To gain better control over the cutting boundary, the user needs to either specify more strokes or carefully draw strokes that extend to the appropriate regions. The graph-cut technique in [2] is sensitive to the areas of the source and sink regions, often requiring large regions as seeds. The methods in [4] and [5] obtain very good boundaries, but require multiple strokes, thus involve rotating the model. Also note that for the cross-boundary method [5], careful attention is needed to ensure multiple

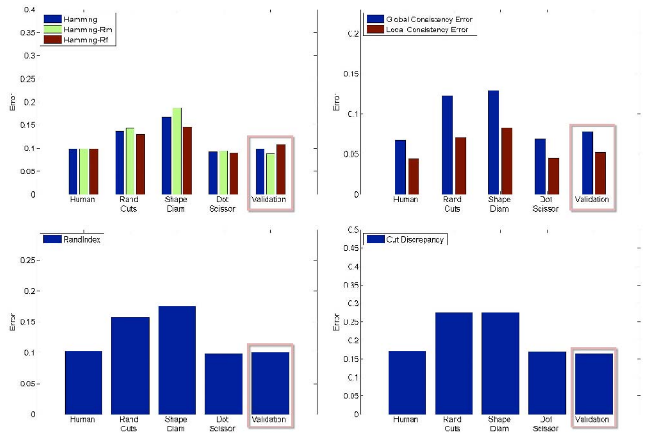

Fig. 15. Evaluation of segmentation results using the protocols of [15]. The test data include all the 380 models in the PSB data set. Our results give lower or similar errors than human segmentations. The validation results used random points from human segmented boundaries to execute dot scissor (see details in Section 5). Comparing with the state-of-the-art automatic methods, our interactive tool gives the expected better results.

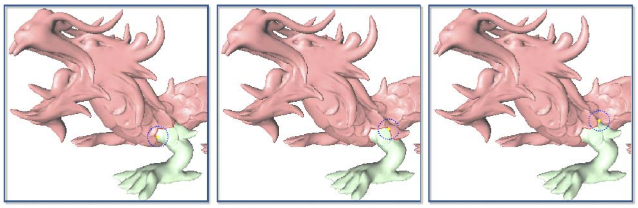

Fig. 17. When multiple concave seams coexist, placing the dot circle at different places may lead to different boundaries.

strokes are drawn in a consistent direction, which is not intuitive especially when the model is rotated. Our dot scissor produces high-quality segmentation by using only a single click (Fig. 16 bottom right), and the resulting boundary follows the geometry better than those of other methods.

6.3 Limitations

Our dot scissor produces high-quality cutting boundaries in most cases. However, in cases where no obvious concave region exists at the clicked point (e.g., cutting out the smooth head of the dolphin in Fig. 5) or when multiple concave seams coexist (e.g., the flake of the Asia dragon model in Fig. 17), the concavity-aware harmonic field may fail to distinguish among different concave seams. As a complemental tool for our dot scissor, we could include a state-of-the-art boundary refinement tool (e.g., [27]) to refine the segmentation boundaries. Another limitation is that, since the design focus of the dot scissor is in cutting out meaningful shape components, our tool is not suitable to segment flat piecewise smooth patches such as faces of CAD models. Like [5], our system is also built on harmonic fields, hence it cannot handle nonmanifold models or point cloud. This might be an interesting future direction to explore.

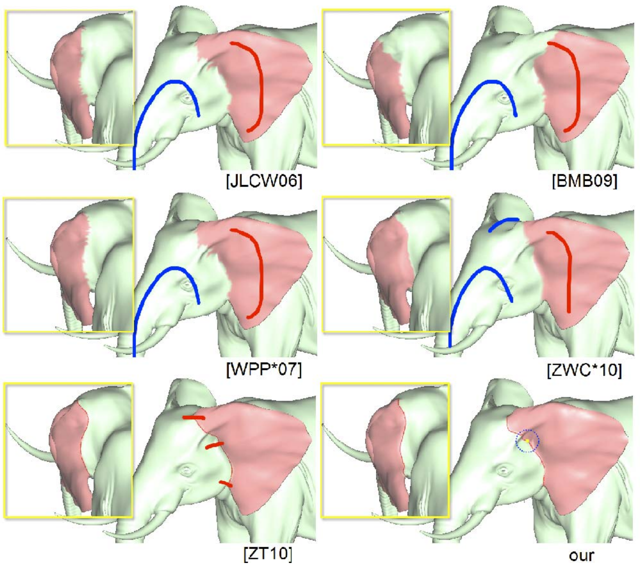

Fig. 16. Our dot scissor outperforms the state-of-art interactive segmentation tools in both simplicity of use and segmentation quality. Note the jaggy boundaries of the method in [2], [8], and [9]. It requires three strokes using the snapping tool of [4] and four strokes using the cross-boundary tool [5] to locate the desired boundary whereas ours is obtained with a single click without boundary refinement.

7 Conclusion

We have presented a very easy-to-use interactive tool for mesh segmentation. We reduce the user effort of cutting out a component to only requiring a single mouse click roughly at where a desired cutting boundary lie. The segmentation tool is built upon a concavity-aware harmonic field that is able to effectively capture the local concave shape features and a robust voting scheme that selects the best cutting boundary from candidate isolines sampled from the fields. The voting scheme increases the robustness of our tool and enables its flexibility and convenience of use. In addition, extensive experimental results and quantitative analysis show that our tool outperforms other interactive segmentation tools in both ease of use and segmentation quality.

There is a lack of qualitative analysis and user studies for comparing different kinds of interactive segmentation tools. We believe such user studies are essential in providing guidance in selecting the right tool for different applications. This remains as future work for our study in interactive segmentation tools.

Acknowledgments

We thank the anonymous reviewers for their valuable comments, Pengfei Xu for his help in the benchmark evaluation, Thomas Li for the video narration and Jackson Yuen for inspiring the single click interface during a discussion. This work was supported in part by the Hong Kong Research Grant Council (Project No. GRF619908, GRF619611) and grants from the City University of Hong Kong (Project No. SRG-Fd 7002642).

References

[1] A. Golovinskiy and T. Funkhouser, “Randomized Cuts for 3D Mesh Analysis,” *ACM Trans. Graphics*, vol. 27, no. 5, pp. 1-12, 2008.
[2] S. Brown, B. Morse, and W. Barrett, “Interactive Part Selection for Mesh and Point Models Using Hierarchical Graph-Cut Partitioning,” *Proc. Graphics Interface (GI ’09)*, pp. 23-30, 2009.
[3] E. Kalogerakis, A. Hertzmann, and K. Singh, “Learning 3D Mesh Segmentation and Labeling,” *ACM Trans. Graphics*, vol. 29, no. 3, 2010.

[4] J. Zhang, C. Wu, J. Cai, J. Zheng, and X. cheng Tai, "Mesh Snapping: Robust Interactivemesh Cutting Using Fast Geodesic Curvature Flow," *Computer Graphics Forum (Eurographics),* vol. 29, no. 2, pp. 517-526, 2010.
[5] Y. Zheng and C.-L. Tai, "Mech Decomposition with Cross-Boundary Brushes," *Computer Graphics Forum (Eurographics),* vol. 29, no. 2, pp. 527-535, 2010.
[6] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin, "Modeling by Example," *ACM Trans. Graphics,* vol. 23, pp. 652-663, 2004.
[7] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel, "Intelligent Mesh Scissoring Using 3D Snakes," *Proc. 12th Pacific Conf. Computer Graphics and Applications (PG '04),* pp. 279-287, 2004.
[8] Z. Ji, L. Liu, Z. Chen, and G. Wang, "Easy Mesh Cutting," *Computer Graphics Forum (EuroGraphics),* vol. 25, no. 3, pp. 283-291, 2006.
[9] H.-Y. Wu, C. Pan, J. Pan, Q. Yang, and S. Ma, "A Sketch-Based Interactive Framework for Real-Time Mesh Segmentation," *Proc. Computer Graphics Int'l (CGI '07),* 2007.
[10] L. Fan, L. Liu, and K. Liu, "Paint Mesh Cutting," *Computer Graphics Forum (Eurographics),* vol. 30, no. 2, pp. 603-611, 2011.
[11] O.K.-C. Au, Y. Zheng, M. Chen, P. Xu, and C.-L. Tai, "Mesh Segmentation with Concavity-Sensitive Fields," *IEEE Trans. Visualization and Computer Graphics,* vol. 18, no. 7, pp. 1125-1134, July 2012.
[12] A. Shamir, "A Survey on Mesh Segmentation Techniques," *Computer Graphics Forum,* vol. 27, no. 6, pp. 1539-1556, 2008.
[13] K.C.-H. Wong, T.Y.-H. Siu, W. Tommy, Y.-H. Siu, P.-A. Heng, and H. Sun, "Interactive Volume Cutting," *Proc. Graphics Interface,* pp. 99-106, 1998.
[14] D. Stalling and H. christian Hege, "Fast and Intuitive Generation of Geometric Shape Transitions," *The Visual Computer,* vol. 16, pp. 241-253, 2000.
[15] X. Chen, A. Golovinskiy, and T. Funkhouser, "A Benchmark for 3D Mesh Segmentation," *ACM Trans. Graphics,* vol. 28, no. 3, pp. 1-12, 2009.
[16] O. Sorkine, D. Cohen-Or, D. Irony, and S. Toledo, "Geometry-Aware Bases for Shape Approximation," *IEEE Trans. Visualization and Computer Graphics,* vol. 11, no. 2, pp. 171-180, Mar./Apr. 2005.
[17] R. Zayer, C. Rössl, Z. Karni, and H.-P. Seidel, "Harmonic Guidance for Surface Deformation," *Computer Graphics Forum (EUROGRAPHICS),* vol. 24, no. 3, pp. 601-609, 2005.
[18] D. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition,* vol. 13, no. 2, pp. 111-122, 1981.
[19] N.J. Mitra, L. Guibas, and M. Pauly, "Partial and Approximate Symmetry Detection for 3D Geometry," *ACM Trans. Graphics,* vol. 25, no. 3, pp. 560-568, 2006.
[20] W. Chang and M. Zwicker, "Automatic Registration for Articulated Shapes," *Computer Graphics Forum (SGP '08),* vol. 27, no. 5, pp. 1459-1468, 2008.
[21] D. Hoffman, W. Richards, A. Pentl, J. Rubin, and J. Scheuhammer, "Parts of Recognition," *Cognition,* vol. 18, pp. 65-96, 1984.
[22] D.D. Hoffman and M. Singh, "Salience of Visual Parts," *Cognition,* vol. 63, no. 1, pp. 29-78, 1997.
[23] K. Xu, H. Zhang, D. Cohen-Or, and Y. Xiong, "Dynamic Harmonic Fields for Surface Processing," *Computers and Graphics Shape Modeling Int'l (SMI '09),* vol. 33, pp. 391-398, 2009.
[24] T. Davis, *User Guide for CHOLMOD: A Sparse Cholesky Factorization and Modification Package,* 2008.
[25] M. Singh, G.D. Seyranian, and D.D. Hoffman, "Parsing Silhouettes: The Short-Cut Rule," *Perception and Psychophysics,* vol. 61, no. 4, pp. 636-660, 1999.
[26] M. Meng, Z. Ji, and L. Liu, "Sketching Mesh Segmentation Based on Feature Preserving Harmonic Field," *J. Computer-Aided Design and Computer Graphics (in Chinese),* vol. 20, pp. 1146-1152, 2008.
[27] L. Kaplansky and A. Tal, "Mesh Segmentation Refinement," *Computer Graphics Forum,* vol. 28, no. 1, pp. 1995-2003, 2009.

Youyi Zheng received the BSc and MSc degrees in mathematics both from the Zhejiang University. Currently, he is working toward the PhD degree at the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests include digital geometric processing, computer graphics, and interactive techniques.

Chiew-Lan Tai received the BSc degree in mathematics from the University of Malaya, MSc degree in computer and information sciences from the National University of Singapore, and the DSc degree in information science from the University of Tokyo. Currently, she is working as an associate professor of computer science at the Hong Kong University of Science and Technology. Her research interests include geometry modeling and processing and computer graphics.

Oscar Kin-Chung Au received the PhD, MPhil, and BEng degrees, all from the Department of Computer Science and Engineering at Hong Kong University of Science and Technology. Currently, he is working as an assistant professor at the School of Creative Media, the City University of Hong Kong. His research interests include geometric modeling and processing, computer graphics, and interactive techniques.