EUROGRAPHICS 2011 / M. Chen and O. Deussen
(Guest Editors)

Volume 30 (2011), Number 2

Paint Mesh Cutting

Lubin Fan[1,2], Ligang Liu[†1,2], Kun Liu[1]

[1]Department of Mathematics, Zhejiang University, China
[2]State Key Laboratory of CAD&CG, Zhejiang University, China

Abstract

We present a novel progressive painting-based mesh cut out tool, called Paint Mesh Cutting, for interactive mesh segmentation. Different from the previous user interfaces, the user only needs to draw a single stroke on the foreground region and then obtains the desired cutting part at an interactive rate. Moreover, the user progressively paints the region of interest using a brush and has the instant feedback on cutting results as he/she drags the mouse. This is achieved by efficient local graph-cut based optimizations based on the Gaussian mixture models (GMM) on the shape diameter function (SDF) metric of the shape. We demonstrate a number of various examples to illustrate the flexibility and applicability of our system and present a user study that supports the advantages of our user interface.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Computer Graphics—Geometry/Mesh Segmentation¡¦User Interface

1. Introduction

Segmentation of 3D shapes into semantic parts or patches is a key ingredient in many graphics applications. Development of mesh segmentation algorithms has received much attention and numerous algorithms have been developed over the last decade [AKM*06, Sha08].

However, it remains a challenge to develop automatic mesh segmentation algorithms due to the complicated human perception. Therefore, interactive tools for mesh segmentation have become very popular in recent years. There have emerged three main types of user interfaces for cutting meshes so far. The simplest interfaces rely on along-cut strokes, asking the user to directly specify a set of points along the cutting boundary [FKS*04, CGF09]. This could be tedious and time-consuming as it requires great care on the part of the user when specifying the boundary points. The more intuitive interfaces are based on foreground/background strokes, allowing the user to freely draw strokes to specify the foreground and background regions [JLCW06, ZWC*10]. It is sometimes unnatural for the user to have to specify the background parts that he does not want to cut out. Recently, a cross-boundary brush based interface was proposed [ZT10]. The user draws strokes across a desired cutting boundary. The user has to know where the cutting boundary is and has to carefully specify the strokes across the precise boundary. In addition, the cutting boundary is sensitive to the stroke directions which makes the cutting results unstable.

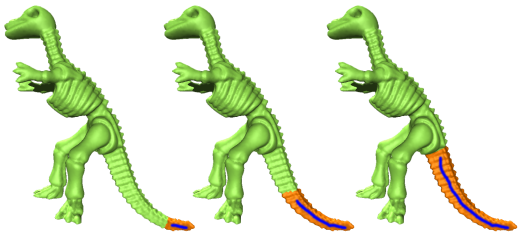

Figure 1: *Illustrations of Paint Mesh Cutting. The user paints the region of interest with a single brush (in blue) on a mesh surface and then obtains the cutting part (in orange). Our system can provide the user instant feedback of the cutting results during mouse dragging (from left to right). Note that we only show 3 snapshots of the continuously sampled point interaction here.*

† Corresponding: liangliu@zju.edu.cn

Our approach. In this work, we propose *Paint Mesh Cutting*, a progressive painting-based tool for mesh segmentation, which is inspired by the work of interactive image segmentation [LSS09]. However, extending the foreground brush user interface for mesh cutting is nontrivial. The challenges are twofold. First, what shape metric should we use? Second, how to gain an interactive rate for progressive drawing? Based on the Gaussian mixture models (GMM) on the shape diameter function (SDF) metric of the shape, we achieve it by graph-cut based optimizations which are performed locally and efficiently.

Our system is easy to use and efficient. It provides a novel and intuitive user interface where users cut out parts by directly painting the region of interest with a brush. Specifically, users only need to freely paint strokes on the foreground part. Unlike the previous interactive cutting tools, users do not need to paint over the background part or the boundary. The cutting results can be automatically expanded from users' paint brush and aligned with the part boundary. By inspecting the new foreground region, users can continuously paint the brush to expand the foreground while holding the left mouse button, until they are satisfied, as shown in Figure 1. Many experiments show that our system extracts semantic parts precisely and efficiently while requiring little skill or effort from the user.

To the best of our knowledge, this is the first time the foreground brush has been used to cut out mesh surfaces in such an easy manner.

2. Related work

Automatic mesh segmentation. We refer the interested readers to [Sha08] for a survey of state-of-the-art studies on mesh segmentation. Some representative work includes techniques based on graph cuts [KT03], hierarchical clustering [GG04], spectral clustering [LZ04], core extraction [KLT05], primitive fitting [AFS06], random walks [LHMR08], randomized cuts [GF08], and so on. Recently, benchmarks based on a ground-truth corpus of human segmented 3D models are presented for evaluating automatic mesh segmentation approaches in a quantitative way [CGF09, BVLM09].

Sketching mesh segmentation. Fully automatic segmentation is typically impossible as defining a semantic subpart for shapes still remains a challenging task. Sketch-based interfaces, which are simple and intuitive and help users easily express their intensions, have been successfully used in interactive mesh segmentation. Instead of specifying the boundary directly [FKS*04, CGF09], the user simply and quickly draws freehand sketches on the mesh to roughly mark out two types of regions (foreground and background) [JLCW06]. During the last few years, a series of two-region sketch-based methods, based on multiple techniques, such as region growing [JLCW06, WPP*07], graph cut [BMB09], random walks [LHMR08, ZWC*10], hierarchical aggregation [XFT09], etc., have been proposed for interactive mesh segmentation. A very recent work presents a system for allowing the user to draw strokes across a desired cutting boundary [ZT10]. We aim to develop a single foreground painting interface for mesh segmentation. The user only specifies strokes on the foreground region which is simpler and more intuitive than the previous approaches.

Surface metrics. A surface metric, which defines a scalar function over the mesh surface, plays an important role in mesh segmentation. It is a critical key to find the right metric which can capture the essence of the semantic components. The minimal rule, which induces part boundaries along negative curvature minima, is a classic metric to capture parts [HR84]. Other well-known surface metrics include geodesic distance based curvature map [GGGZ05], an isophotic metric [PSH*04], a feature-sensitive metric based on geodesic and isophotic metrics [LZH*07], diffusion distance [dGGV08], curvature tensor based anisotropic geodesic metric [SJC08] etc. Other metrics rely on volumetric information [DZM08]. *Shape diameter function* (SDF) [SSCO08] measures the diameter of the object's volume in the neighborhood of each point on the surface, and thus provides a mapping from volumetric information onto the surface boundary. And it is invariant to pose changes. The volumetric shape image (VSI) [LZSCO09] is also a part-aware shape metric, which captures relevant visibility information inside the shape's enclosed volume by combining the SDF distance with geodesic distance and normal variation. However, computation of VSI costs much more than computation of SDF. In this work we adopt the SDF metric to capture the surface essence of semantic parts.

3. Graph-cut based optimization

We extend the foreground painting user interface [LSS09] to mesh cutting. However, the extension is nontrivial. We adopt the similar mechanism with the work of [LSS09] and some implementation issues have to been carefully handled.

In our system, we use SDF as the shape metric [SSCO08]. We first compute the SDF values $M(\cdot)$ for all vertices after the mesh is loaded into the system, see Figure 2 (left) for the SDF values on the Armodilo model.

The heart of our segmentation technique is minimum graph-cut optimization [BJ01]. We consider the triangular mesh $\mathcal{M} = \{\mathcal{V}, \mathcal{E}\}$ as a graph with mesh vertices $\mathcal{V}$ as graph nodes and mesh edges $\mathcal{E}$ as the corresponding graph edges. Vertices are selected as the foreground *seed vertices* $\mathcal{S}^f$ by projecting each mouse point on the user's painting brush $\mathcal{P}$ onto the mesh surface and including all vertices within a certain distance (10 vertices) from that point. Our goal is to compute the foreground part $\mathcal{F}$ starting from the foreground seed vertices $\mathcal{S}^f$.

At the very beginning of the user interaction, the back-

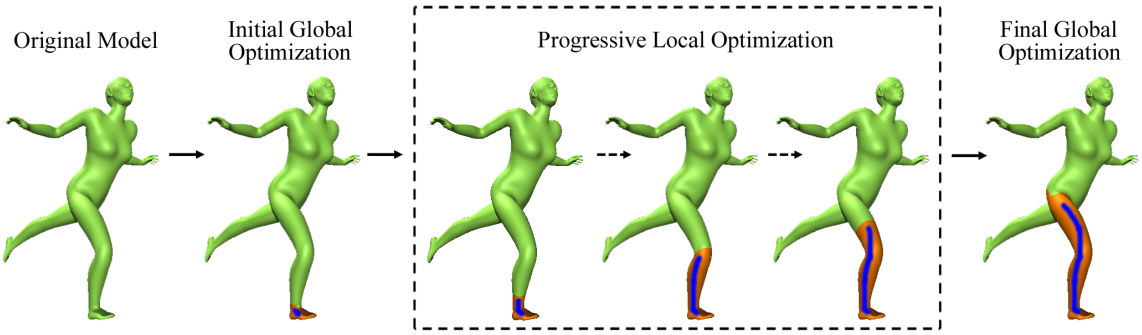

Figure 3: *Main steps of Paint Mesh Cutting. Starting from an input model, we progressively paint brush on the region of interest and get the desired cutting results.*

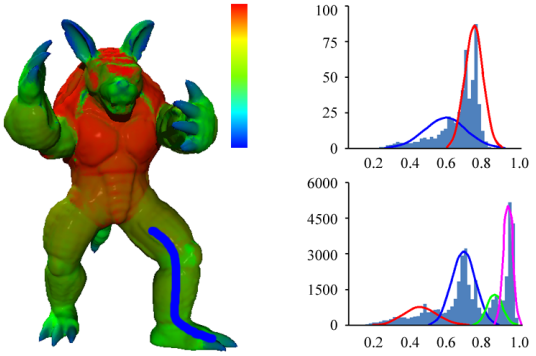

Figure 2: *Left: the SDF values of the Armodilo model (large in red and small in blue). The user draws a blue brush on the left leg region of the model. Upper right: the GMM model $p_f(\cdot)$ with $k_f = 2$ components on the foreground (the blue brush); Lower right: the GMM model $p_b(\cdot)$ with $k_b = 4$ components on the background (the other region).*

ground $\mathcal{B} = \mathcal{V}$ and the foreground $\mathcal{F} = \emptyset$. A background SDF model $p_b(\cdot)$ is initialized by randomly sampling a number (typically 1000) of the vertices from $\mathcal{V}$ by fitting a Gaussian Mixture Model (GMM) with k_b components using the greedy Expectation-Maximization (EM) algorithm [VL02]. Similarly, a foreground SDF model $p_f(\cdot)$ (a GMM with k_f components) is built based on $\mathcal{S}^f$ (see Figure 2(right)).

Then, with the two SDF models p_f and p_b, we apply a graph-cut based optimization to obtain the foreground $\mathcal{F}$. Specifically, we formulate the segmentation as a vertex-labeling problem in terms of energy minimization. The binary labels $L = \{l_v | v \in \mathcal{V}\}$ of the vertices $\mathcal{V}$ are obtained by minimizing the following energy function:

$$E(L) = \sum_{v \in \mathcal{V}} E_d(l_v) + \lambda \sum_{(v,u) \in \mathcal{E}} E_s(l_v, l_u), \qquad (1)$$

where the data term E_d depicts the penalty of assigning a label l_v to vertex v (1-foreground, 0-background), the smoothness term E_s describes the penalty for assigning different labels to two adjacent vertices v and u, and λ is the weight.

Data term. We define the data term as follows:

$$E_d(l_v) = \begin{cases} (1 - l_v) \cdot K, & \forall v \in \mathcal{S}^f \\ l_v \cdot L_v^f + (1 - l_v) \cdot L_v^b, & \texttt{otherwise} \end{cases} \qquad (2)$$

where K is a sufficiently large constant, $L_v^f = -\ln(p_f M(v) + \varepsilon)$ and $L_v^b = -\ln(p_b M(v) + \varepsilon)$, ε is a small threshold number to avoid zero value in the log function (we set $\varepsilon = 10^{-6}$), and $M(v)$ is the SDF value of vertex v.

Smoothness term. The smoothness term is defined as:

$$E_s(l_v, l_u) = -|l_v - l_u| \cdot \ln\left((1 - \beta) n(v,u) + \beta g(v,u)\right), \qquad (3)$$

where $n(v,u) = \frac{1 - \mathbf{n}_v \cdot \mathbf{n}_u}{2}$, $g(v,u) = \frac{e(v,u) - e_{\min}}{e_{\max} - e_{\min}}$, $\mathbf{n}_v$ and $\mathbf{n}_u$ are normals of adjacent vertices v and u respectively, $e(v,u)$ is the length of edge $(v,u) \in \mathcal{E}$, $e_{\max}$ and $e_{\min}$ are the maximum and minimum of edge lengths respectively, and β is a weight (we set $\beta = 0.05$). Like the VSI metric [LZSCO09], we combine the SDF metric with normal variation $n(v,u)$ and geodesic distance $g(v,u)$ in this smoothness term.

4. Paint mesh cutting system

In this section we present our novel framework of paint mesh cutting.

4.1. User interface

Our interface is simple and easy to use. To execute a cut out for a subpart, the user paints the part of interest (foreground)

with a brush while holding the left mouse button. Unlike the previous UIs which compute results after the mouse button is released, we trigger a segmentation optimization process once the user starts to drag the mouse into the background, see Figure 3 (second). Once the segmentation process is triggered, we apply a progressive segmentation algorithm, described below, to expand the foreground part. The expanded foreground region is computed in a very short time interval (usually under 0.1 seconds) and instantly displayed to the user.

By inspecting the new foreground region, the user can continuously drag the mouse to expand the foreground, until satisfied, as illustrated in Figure 3 (also see the accompany video). The user needs not paint over the entire area since the foreground can be properly expanded from the brush to the nearby part boundaries. The user can expand and refine the foreground by drawing more brush strokes.

4.2. Progressive expansion algorithm

Here, we introduce the progressive expansion algorithm which supports the progressive painting interface.

Initial global optimization. At the very beginning of the user interaction, we set $k_b = 4$ for building the background GMM model $p_b(\cdot)$. Once the user starts to draw a stroke (which covers 3-ring vertex neighbors), an initial global optimization is triggered. The foreground GMM model $p_f(\cdot)$ is built by fitting the seed vertices with $k_f = 2$ components. With the two GMM models, we apply the graph cut optimization as described in Equation 1 and obtain the initial foreground region $\mathcal{F}$, see Figure 3 (second).

Progressive local optimization. Given the existing foreground $\mathcal{F}$ and current brush $\mathcal{P}$, a local optimization is triggered to compute a new and expanded foreground $\mathcal{F}'$ in the background $\mathcal{B}$, as shown in Figure 4 (left). Once the new foreground $\mathcal{F}'$ is obtained, the existing foreground is updated as $\mathcal{F} = \mathcal{F} \cup \mathcal{F}'$ for the next user interaction.

We denote the intersection between the brush $\mathcal{P}$ and the background $\mathcal{B}$ as seed (red) vertices S^f ($S^f = \mathcal{P} \cap \mathcal{B}$). To obtain a stable estimation, we dilate the frontal foreground boundary *inwards* by a certain offset (typically 2 vertices) and set the (blue) vertices in the dilated region as local foreground vertices, as shown in Figure 4 (right). Using both seed vertices and local foreground vertices, we build a local foreground GMM model $p^f(\cdot)$ with $k_f = 1$ component. Using local foreground vertices makes the estimation more stable because the brush or seed vertex region may be very small.

Then, we update the background SDF model. In each subsequent user interaction, we replace the samples that were labeled as foreground in the previous interaction with the same number of vertices randomly sampled from the background. Estimating the background GMM using all the samples might cost much and does not meet the instant feedback requirement. Hence, we compute a region $\mathcal{R}$ by dilating the frontal foreground boundary *outwards* by a certain offset (typically 20 vertices). The background GMM model $p_b(\cdot)$ is built by fitting the vertices in $\mathcal{R} \cap \mathcal{B}$ with $k_b = 4$ components.

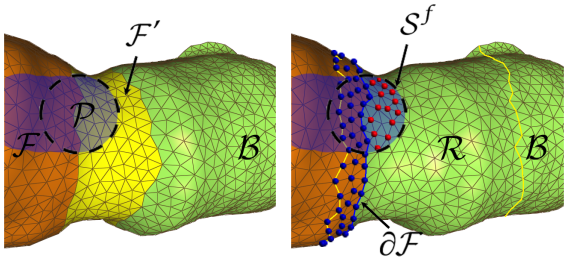

Figure 4: *Progressive foreground expansion. Left: Given the existing foreground $\mathcal{F}$ (in brown) and current brush $\mathcal{P}$ (in blue) which touches the background $\mathcal{B}$ (in green), a local optimization is triggered to compute a new and expanded foreground $\mathcal{F}'$ shown in yellow. Right: the seed vertices S^f (in red) is computed by the intersection between the brush $\mathcal{P}$ and the background $\mathcal{B}$; The vertices in the inward dilation of frontal foreground boundary is shown in blue. The region $\mathcal{R}$ is computed by outward dilating the frontal foreground boundary. The local graph-cut optimization is performed in the region between the two yellow boundaries.*

With the two GMM models, we apply the graph cut optimization in Equation 1 and obtain the expanded foreground region $\mathcal{F}'$ and update the foreground $\mathcal{F}$, see Figure 3 (middle).

Final global optimization. Once the user stops painting the brush by releasing the left mouse button, we apply the global graph cut optimization again, see Figure 3 (right). Before that, we update the foreground GMM $p^f(\cdot)$ and the background GMM $p^b(\cdot)$ by fitting the vertices in $\mathcal{F}$ and $\mathcal{B}$ respectively. In the final global optimization we set $k_f = 2$ and $k_b = 4$.

4.3. Implementation details

The reasons why progressive expansion can give users instant feedback on good segmentation results are twofold. First, SDF is a good shape metric which measure the essential characteristic (volumetric diameter) for shapes. Second, only a small portion of vertices participate in the progressive local optimizations, so it gives very quick updates while the user drags the brush. In the following we discuss some implementation issues and details.

Cutting boundary refinement. After we have the cutting result obtained by the final global optimization after the user

releases the mouse, we perform a boundary optimization on the segmentation boundary using snakes [JLCW06].

Background painting. Like the foreground/background sketching UI [JLCW06], we also allow the user to expand the background if necessary. This is done by painting strokes on background. In our system, the user uses the right mouse button to swap the roles of the foreground and background. Suppose $\mathcal{S}^b$ is the seed vertices on the background strokes, we modify the data term in Equation 2 as follows:

$$E_d(l_v) = \begin{cases} (1-l_v)\cdot K, & \forall v \in \mathcal{S}^f, \\ l_v \cdot K, & \forall v \in \mathcal{S}^b, \\ l_v \cdot L_v^f + (1-l_v)\cdot L_v^b, & \texttt{otherwise}. \end{cases} \quad (4)$$

Adaptive weights. The weight λ in Equation 1 plays an important role in the graph-cut optimization. Graph cut optimization with small value of λ tends to get the cutting part consisting of vertices with similar SDF values as the dominant data term tries to match the GMMs of SDF values. Graph cut optimization with large value of λ achieves good segmentation boundaries to match the minimal rule as the normal variation and geodesic distance are considered in the dominant smoothness term. We vary λ in different steps. In the initial global optimization step, we set small value of $\lambda = \lambda_{initial}$ to guarantee a good initial segmentation result guided by SDF values. Then we increase the value of $\lambda = \lambda_{local}$ to let the cutting boundary catch the concave creases during the progressive local optimization. Finally $\lambda = \lambda_{final}$ should be reduced to get a better segmentation result in the final global optimization phase. In our system, we set $\lambda_{initial} = 1, \lambda_{local} = 10$, and $\lambda_{final} = 2$.

Speedup. We compute the SDF values when the user loads the mesh in our system. However, computation of SDF values on large meshes might be computational expensive. To reduce the waiting time for the user, we compute SDF values over the simplified mesh and then interpolate the SDF values over the original mesh using the Poisson equation [KGMS10]. Furthermore, we adopt a fast implementation for graph-cut optimizations in our system [BK04] and use the parallel and distributed graph-cut method to accelerate the algorithm [SK10].

4.4. Patch-based paint brushes

Similar to [ZT10], our system allows the user to switch between part-brush vs. patch-brush easily by pressing a button. Fortunately, our system provides a unified framework for both part-based and patch-based segmentations. We only need to change the weights λ in Equation 1 to adapt to the patch-based segmentation scenario. We set $\lambda_{initial} = 5, \lambda_{local} = 3$, and $\lambda_{final} = 5$ for patch-based brushes in our system. By using these weights, both initial and final optimization try to obtain the patches by clustering the vertices with similar SDF values, and the local optimization tries to catch the sharp features for cutting boundaries.

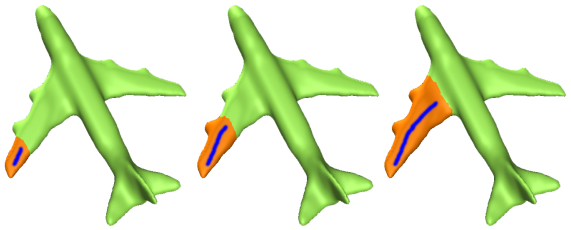

Figure 5: *The user progressively paints the brush to cut out a wing of the airplane.*

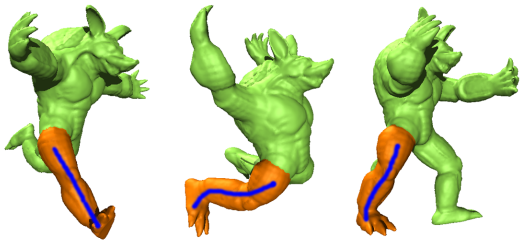

Figure 6: *The user can easily cut out the same parts from objects with different poses using the paint brush tool.*

5. Experimental results

We show some examples to illustrate the applicability and flexibility of our system. All the examples presented in this paper were made on a dual-core 3GHz machine with 4G memory.

Figure 5 shows one example of using the paint brush to cut a wing of the airplane in a progressive manner. Note that our approach obtains reasonable results even there are a couple of small bumps on the wing.

Thanks to the fact that the SDF values are invariant to pose changes, the user can easily cut out the same part from objects with different poses using the paint brush tool, see

Figure 7: *The cut out results are not that dependent on the specific brushes painted on the mesh surface. Thus users can obtain what they want to get by freely painting brushes on the surface.*

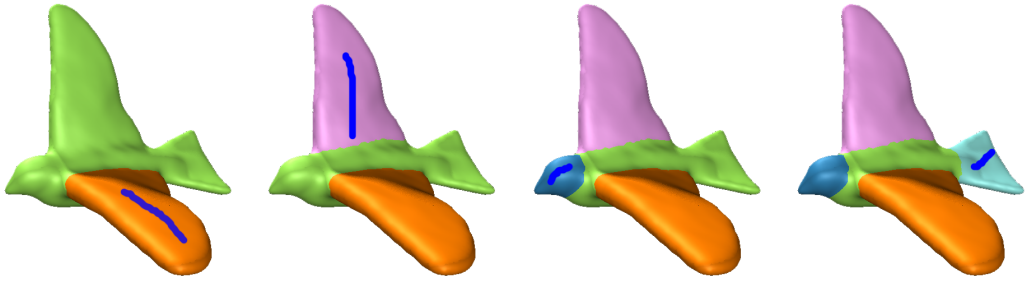

Figure 8: *A complete segmentation example of using our paint brush tool.*

Figure 11: *Segmentation results produced by our system.*

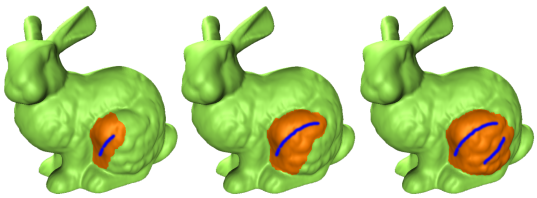

Figure 9: *An example of progressive patch-based cutting. Two foreground brushes are used.*

Figure 6. In Figure 7, users paint three different brushes on the left leg of the Armodillo model and obtain similar cutting results.

Figure 8 shows a complete example of segmentation using our tool. In each step, the user specifies one stroke on the foreground region and obtains the corresponding foreground part.

Figure 9 shows an example of progressive patch-brush cutting. The user applies a second brush to cut out the expected patch.

Our approach is insensitive to noise in the mesh, see Figure 10. A certain amount of noise was added to the original mesh, our system still obtains reasonable cutting results, either for part-type or for patch-type cuts. We tested our algorithm on the models in the Princeton benchmark database. Some of the results are shown in Figure 11.

It is worthwhile pointing out that we choose SDF as the

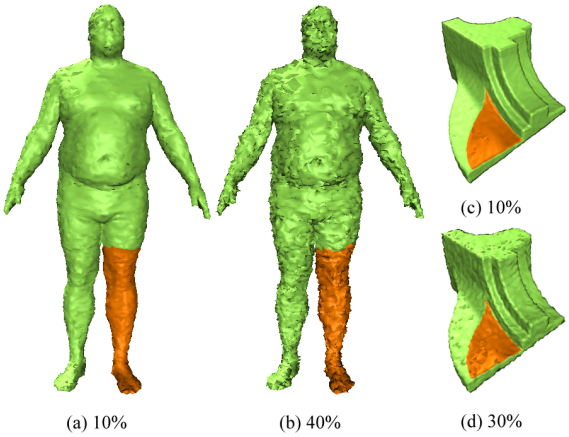

Figure 10: *Cutting results using our system for meshes with different amount of noise. The numbers below the figures show the amount of noise added on the original meshes.*

Figure 12: *Given the same specified foreground stroke, the cutting results obtained by different shape metrics on the surface (from left to right: Gauss curvature, mean curvature, SDF) are quite different. The metric of SDF gives the most meaningful part cutting result.*

surface metric in our system because it is a pose-invariant part-aware shape metric which measures the diameter of the object's volume in the neighborhood of each point on the surface. Thus our method based on SDF provides nice meaningful cutting results while other metrics such as Gaussian curvature and mean curvature cannot always give the correct results, see Figure 12. The reason is that the curvature based metrics are not part-aware. Points on much different parts might have the same curvature metric. Thus the GMM built based on the metric cannot reveal the local geometric property of the cutting part. On the other hand, the curvature based metric might work well for cutting the meaningful patches from meshes. Particularly, Gaussian curvature metric works much better than mean curvature metric. However, both curvature based metric are quite sensitive to the mesh noise and thus cannot give robust cutting patch results for noisy meshes. Our SDF metric based method can give stable cutting results as shown in Figure 10.

Table 1 lists the running time of the mesh segmentation results shown in this paper. As we can see, our system achieves a fast performance in all these examples.

Model	# Vertex	T_1 (ms)	T_2 (ms)	T_3 (ms)
Dino (Fig. 1)	28,150	53	10	178
Woman (Fig. 3)	5,691	8	6	27
Airplane (Fig. 5)	6,797	12	5	24
Armodilo (Fig. 6)	25,193	36	10	120
Bunny (Fig. 9)	34,835	54	11	248

Table 1: *Running time (in milliseconds) for different examples shown in the paper. T_1, T_2, T_3 denote the computation time of the three steps in our algorithm, i.e., the initial global optimization, each local optimization, and the final global optimization, respectively.*

5.1. User study

We conducted a usability study to compare the different user interfaces for mesh segmentation, including foreground/background brushes (FBB for short) [JLCW06], cross boundary brushes (CBB for short) [ZT10], and our foreground brushes (FB for short). See Figure 13 for the different user interfaces. We only compare the part-based segmentation results for three tools. We used two criteria, i.e., efficiency and accuracy, to evaluate these tools.

Assignment. We invited 16 individuals to participate in our experiment, of which 12 participants have experience in 3D interaction and 4 participants are not familiar with 3D interaction. A user guide and sufficient time were given to each participant, to familiarize himself/herself with the systems that would be used to do the experiment.

We collected models in 16 categories from the Princeton benchmark model database [CGF09], discarding the other 3 categories which are not suitable for part-based segmentations. We chose one model (with $8k - 15k$ vertices) from each category and thus our corpus contains 16 models. Each model is associated with an image describing the segmentation requirement. For the purpose of acquiring segmentations from participants for each model in the corpus, we divided the ground-truth randomly into 16 sets, ensuring that each set contains 6 models from different categories. Then each participant was assigned to segment the models of one set, each model with three segmentations using three interactive algorithms respectively.

Each participant was assigned to fill out a short questionnaire if he/she had finished the assignment. Users are requested to rate the following questions on a scale of 1-3: how easy the users specify the segmentation, how fast they carry out the expected segmentations, how accurate they consider their final segmentations, where 1 denotes lowest score, 3 denotes highest score. Finally, they were asked to rate the in-

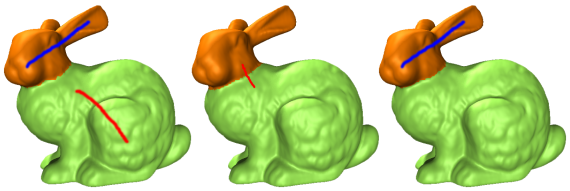

Figure 13: *Different brush-based user interfaces for mesh segmentation. Left: foreground/background brushes (FBB) [JLCW06]; Middle: cross boundary brushes (CBB) [ZT10]; Right: our foreground brushes (FB).*

teractive algorithms again on a scale of 1-3, indicating how good they perceived these algorithms to be.

Analysis. Based on the above studies, we made some analysis and comparisons among the three interactive segmentation tools.

Figure 14 shows the time statistics for segmentation tasks using three tools. We have the following observations. According to the interaction time, FB needs less time than FBB for users to specify the strokes and CBB needs the least time. According to the algorithm computation time, FBB is the fastest as it uses a region growing scheme. FB is faster than CBB as CBB has to solve a large Poisson equation. Although FB needs to solve a series of graph-cut optimizations, it is still computationally efficient.

We used a region-based measure to compute the consistency degree between the cutting result produced by the algorithms and the ground truth. The measure is based on binary jaccard index to compare the object accuracies of interactive image segmentations and can be defined as [MO10]

$$BJI(S_1, S_2) = \frac{||S_1^O \cap S_2^O||}{||S_1^O \cup S_2^O||}, \quad (5)$$

where S_1^O and S_2^O are the extracted regions of interactive segmentation methods S_1 and S_2 respectively. We set S_2^O as the results in the benchmark. The higher the value of $BJI(S_1, S_2)$ is, the more accurate the segmentation S_1^O is. Figure 15 shows the averaged *BJI* values of three algorithms. We can see that FB achieves the highest *BJI* value which means that FB generally obtains more accurate segmentation results than the other two algorithms.

According to the answers indicated by the participants on a scale of 1-3 on rating the algorithms to the questionnaire, FB received the highest rank, followed by CBB, and then FBB. That means, users prefer to using our novel interface FB to do the segmentation generally. Therefore, Paint Mesh Cutting gives the best experience for users to segment meshes interactively.

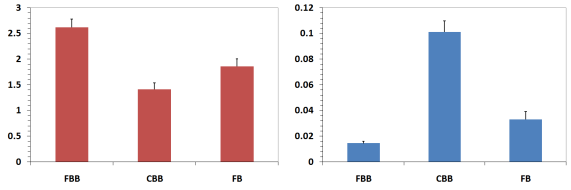

Figure 14: *Comparison of efficiency for three tools. Left: averaged time (show as vertical bars) and variance (show as vertical lines on top of the bars) of user interactions; Right: averaged time and variance of performing the segmentation algorithms.*

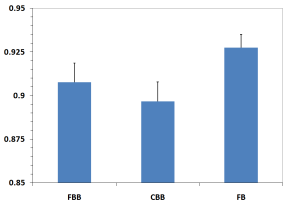

Figure 15: *Comparison of accuracy for three tools: averaged BJI value (show as vertical bars) and variance (show as vertical lines on top of the bars).*

5.2. Limitations

Paint Mesh Cutting suffers a few drawbacks due to the nature of the graph-cut optimization. One drawback is that this tool is difficult to cut out the partial part for smooth surfaces. For example, a small stroke on a smooth cylinder surface might result in obtaining the whole cylinder surface instead of a short segment of the surface using our tool. This is because the whole surface has the same SDF value at each vertex. To cut out a short segment of the cylinder surface, a background brush is also needed, just like in Easy Mesh Cutting [JLCW06]. The other drawback is that the user might have to specify many strokes to cut out some semantic parts from highly-detailed regions. This is because the optimization-guided contour tends to snap to sharp features and concave creases. However, these drawbacks can be alleviated to some extent by combining with other interactive segmentation tools.

6. Conclusion

We present a novel tool for interactive mesh segmentation, which allows users to only paint strokes on the region of interest. Our system obtains the cutting results instantly while users paint the brushes by holding the left mouse button. By inspecting the foreground region, users can continuously drag the mouse to expand the region of interest, until they are satisfied. Our paint based cutting system provides users a favorable experience on cutting mesh surfaces without consid-

ering the region of non-interests, which embodies the motif "what you paint is what you get (WYPIWYG)".

Acknowledgement. We thank anonymous reviewers for valuable feedback. We are thankful to Jie Xu for video narration. This work is supported by the National Natural Science Foundation of China (61070071) and the 973 National Key Basic Research Foundation of China (No. 2009CB320801).

References

[AFS06] ATTENE M., FALCIDIENO B., SPAGNUOLO M.: Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer 22*, 3 (2006), 181–193. 2

[AKM*06] ATTENE M., KATZ S., MORTARA M., PATANE G., SPAGNUOLO M., TAL A.: Mesh segmentation - a comparative study. In *Proc. Shape Modelling International* (2006), pp. 14–25. 1

[BJ01] BOYKOV Y. Y., JOLLY M.-P.: Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Proc. Internation Conference on Computer Vision (ICCV)* (2001), pp. 105–112. 2

[BK04] BOYKOV Y., KOLMOGOROV V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 9 (2004), 1124–1137. 5

[BMB09] BROWN S., MORSE B., BARRETT W.: Interactive part selection for mesh and point models using hierarchical graph-cut partitioning. In *Proc. Graphics Interface* (2009). 2

[BVLM09] BENHABILES H., VANDEBORRE J.-P., LAVOUÉ G., MOHAMEDDAOUDI: A framework for the objective evaluation of segmentation algorithms using a ground-truth of human segmented 3D-models. In *Proc. Shape Modeling International* (June 26-28 2009). short paper. 2

[CGF09] CHEN X., GOLOVINSKIY A., FUNKHOUSER T.: A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (Aug. 2009). 1, 2, 7

[dGGV08] DE GOES F., GOLDENSTEIN S., VELHO L.: A hierarchical segmentation of articulated bodies. 1349–1356. 2

[DZM08] DYER R., ZHANG H., MÖLLER T.: Surface sampling and the intrinsic voronoi diagram. 1393–1402. 2

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. *ACM Transactions on Graphics (Proc. SIGGRAPH) 23*, 3 (2004), 652–663. 1, 2

[GF08] GOLOVINSKIY A., FUNKHOUSER T.: Randomized cuts for 3D mesh analysis. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA) 27*, 5 (2008), Article No.: 145. 2

[GG04] GELFAND N., GUIBAS L. J.: Shape segmentation using local slippage analysis. In *Proc. Symposium on Geometry Processing* (2004), pp. 214–223. 2

[GGGZ05] GATZKE T., GRIMM C., GARLAND M., ZELINKA S.: Curvature maps for local shape comparison. In *Proc. Shape Modeling International* (2005), pp. 246–255. 2

[HR84] HOFFMAN D. D., RICHARDS W. A.: Parts of recognition. *Cognition 18* (1984), 65–96. 2

[JLCW06] JI Z., LIU L., CHEN Z., WANG G.: Easy mesh cutting. *Computer Graphic Forum (Proceedings of Eurographics) 25*, 3 (2006), 283–291. 1, 2, 5, 7, 8

[KGMS10] KOVACIC M., GUGGERI F., MARRAS S., SCATENI R.: Fast approximation of the shape diameter function. In *Proc. Workshop on Computer Graphics, Computer Vision and Mathematics (GraVisMa)* (2010). 5

[KLT05] KATZ S., LEIFMAN G., TAL A.: Mesh segmentation using feature point and core extraction. *The Visual Computer (Proc. Pacific Graphics) 21*, 8-10 (2005), 649–658. 2

[KT03] KATZ S., TAL A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics (Proc. SIGGRAPH) 22*, 3 (2003), 954–961. 2

[LHMR08] LAI Y.-K., HU S.-M., MARTIN R. R., ROSIN P. L.: Fast mesh segmentation using random walks. In *Proc. ACM Symposium on Solid and Physical Modeling* (2008), pp. 183–191. 2

[LSS09] LIU J., SUN J., SHUM H.: Paint selection. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (2009), Article No. 69. 2

[LZ04] LIU R., ZHANG H.: Segmentation of 3D meshes through spectral clustering. In *Proc. Pacific Graphics* (2004), pp. 298–305. 2

[LZH*07] LAI Y. K., ZHOU Q. Y., HU S., WALLNER J., POTTMANN H.: Robust feature classification and editing. *IEEE Transactions on Visualization and Computer Graphics 13*, 1 (2007), 34–45. 2

[LZSCO09] LIU R., ZHANG H., SHAMIR A., COHEN-OR D.: A part-aware surface metric for shape analysis. *Computer Graphics Forum (Proc. Eurographics) 28*, 2 (2009), 397–406. 2, 3

[MO10] MCGUINNESS K., O'CONNOR N.: A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition 43*, 2 (2010), 434–444. 8

[PSH*04] POTTMANN H., STEINER T., HOFER M., HAIDER C., HANBURY A.: The isophotic metric and its application to feature sensitive morphology on surfaces. In *Proc. European Conference on Computer Vision* (2004), pp. 560–572. 2

[Sha08] SHAMIR A.: A survey on mesh segmentation techniques. *Computer Graphics Forum 27*, 6 (2008), 1539–1556. 1, 2

[SJC08] SEONG J. K., JEONG W. K., COHEN E.: Anisotropic geodesic distance computation for parametric surfaces. In *Proc. Shape Modeling International* (2008), pp. 179–186. 2

[SK10] STRANDMARK P., KAHL F.: Parallel and distributed graph cuts by dual decomposition. In *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)* (2010). 5

[SSCO08] SHAPIRA L., SHAMIR A., COHEN-OR D.: Consistent mesh partitioning and skeletonization using the shape diameter function. *The Visual Computer 24*, 4 (2008), 249–259. 2

[VL02] VLASSIS N., LIKAS A.: A greedy em algorithm for gaussian mixture learning. *Neural Processing Letters 15*, 1 (2002), 77–87. 3

[WPP*07] WU H.-Y., PAN C., PAN J., YANG Q., MA S.: A sketch-based interactive framework for real-time mesh segmentation. In *Proc. Computer Graphics International* (2007). 2

[XFT09] XIAO C., FU H., TAI C.-L.: Hierarchical aggregation for efficient shape extraction. *The Visual Computer 25*, 3 (2009), 267–278. 2

[ZT10] ZHENG Y., TAI C.-L.: Mesh decomposition with cross-boundary brushes. *Computer Graphics Forum (Proc. Eurographics) 29*, 2 (2010), 527–535. 1, 2, 5, 7, 8

[ZWC*10] ZHANG J., WU C., CAI J., ZHENG J., CHENG TAI X.: Mesh snapping: Robust interactive mesh cutting using fast geodesic curvature flow. *Computer Graphics Forum (Proc. Eurographics) 29*, 2 (2010), 517–526. 1, 2