EUROGRAPHICS 2010 / T. Akenine-Möller and M. Zwicker
(Guest Editors)

Volume 29 (2010), Number 2

Mesh Decomposition with Cross-Boundary Brushes

Youyi Zheng and Chiew-Lan Tai

The Hong Kong University of Science & Technology

Abstract

We present a new intuitive UI, which we call cross-boundary brushes*, for interactive mesh decomposition. The user roughly draws one or more strokes across a desired cut and our system automatically returns a best cut running through all the strokes. By the different natures of part components (i.e., semantic parts) and patch components (i.e., flatter surface patches) in general models, we design two corresponding brushes: part-brush and patch-brush. These two types of brushes share a common user interface, enabling easy switch between them. The part-brush executes a cut along an isoline of a harmonic field driven by the user-specified strokes. We show that the inherent smoothness of the harmonic field together with a carefully designed isoline selection scheme lead to segmentation results that are insensitive to noise, pose, tessellation and variation in user's strokes. Our patch-brush uses a novel facet-based surface metric that alleviates sensitivity to noise and fine details common in region-growing algorithms. Extensive experimental results demonstrate that our cutting tools can produce user-desired segmentations for a wide variety of models even with single strokes. We also show that our tools outperform the state-of-art interactive segmentation tools in terms of ease of use and segmentation quality.*

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

1. Introduction

Mesh decomposition is a classical problem in computer graphics. Geometry applications such as parameterization, morphing, shape retrieval, matching, modeling, often involve segmentation as an initial step. Automatic decomposition of a mesh into meaningful semantic components that match human intuition is a hard problem. The reasons are twofold. First, it is hard to define a measure that captures the semantic information of a given shape. Second, human perception on segmentation is subjective even towards the same object.

In recent years, interactive tools for mesh decomposition have become increasingly popular due to their ease of use. Previous interactive mesh decomposition systems provide interfaces that specify the cuts via either explicit (with along-cut strokes) or implicit ways (with in-segment strokes). These interactive segmentation systems share three common objectives: interactive response, easy-to-use and user-intention driven. However, whether the latter two objectives are met is arguable. Interfaces that rely on along-cut strokes [LLS*04, FKS*04, CGF09] provides good user control by allowing the user to specify a set of sparse points (either by sketching or clicking) along the cutting boundary. This however could be a laborious task since the user inputs have to lie on the cutting boundaries and it usually involves multiple rotations and sketching. The tools based on in-segment strokes [JLCW06, WPP*07, MJL08, BMB09]

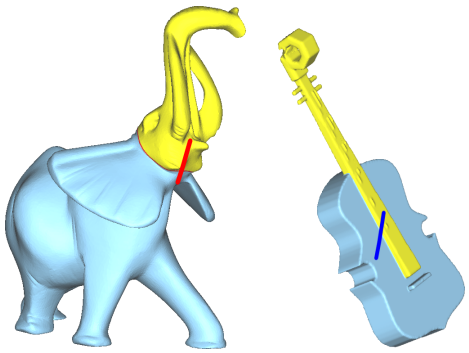

Figure 1: Our user interface: the user draws a single *cross-boundary* stroke to execute a cut in most cases. Left: a part-brush stroke (in red) is drawn to segment out a semantic part-component; Right: a patch-brush stroke (in blue) is drawn to segment out a flatter surface patch.

strive in terms of ease of use by allowing free strokes to specify the foreground/background regions. However, the user loses control over the cutting boundaries. In addition, their underlying region growing algorithms tend to get stuck at local minima, making the segmentation results unpredictable.

We present here an intuitive UI, called *cross-boundary brushes*, which aims at achieving both ease of use and good user-control. The user draws strokes roughly across a desired boundary. We minimize the user inputs by requiring only a single stroke to execute a cut in most cases. When necessary, additional strokes are sketched to refine a cut locally. We observed that real world objects contain two main types of components that merit cuts. The first type are semantic part-components with protruding shapes such as legs and arms (Figure 1 left). The second type are patch-components which are flatter such as the eyes, chests of character models and facets of CAD models (Figure 1 right). Their significant geometrical differences motivate the design of two different tools in our system, namely, *part-brush* and *patch-brush*. A common user interface is shared by these two brushes, enabling easy switch between them.

The key to the design of our part-brush is using the isolines of a harmonic field as cutting boundaries. Harmonic field has been found useful in geometry applications such as deformation and quadrilaterization [ZRKS05, DBG*06]. By constraining the harmonic field with the user strokes, the isolines of the field all run across the user strokes, serving as good cutting boundary candidates. We design a robust selection scheme to locate the best isoline as the cutting boundary. This choice of design is motivated by three reasons. First, each isoline is a smooth connected loop, thus is well suited to serve directly as a cutting boundary. Second, the underlying harmonic field as well as the extracted isolines are inherently smooth, insensitive to noise, pose, tessellation, and variation in user's stroke. Third, the computation of the harmonic field is rather efficient, involving only a sparse linear system. With modern numerical tools, each new stroke merely invokes a fast updating of the factorization in the linear system, providing interactive feedback to the user.

The part-brush by itself already provides a powerful tool for segmenting general models with semantic part components. However, it cannot segment flatter patches that contain sharp boundaries such as the embossed characters on the bunny model (Figure 2) or facets of CAD models (Figure 13), since no isoline runs along the sharp boundaries, even when multiple strokes are drawn. Therefore, to complement the functionality of the part-brush, we introduce the patch-brush, which is based on a region-growing algorithm. The design rationale is that sharp edges form local extrema that are ideal for a greedy algorithm to stop at. We define a novel face-based metric applied on a filtered normal field for our region-growing algorithm. We show that such design alleviates the common problem of sensitivity to noise and fine surface details in greedy region-growing algorithms.

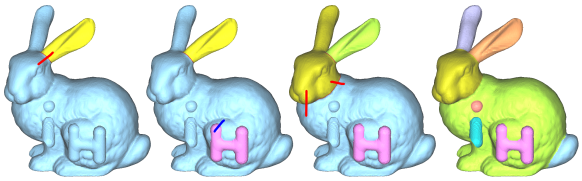

Figure 2: Usage of our system: the user first cuts out the bunny's ears using the part-brush (left), then cuts out the character 'H' using the patch-brush (middle left), then switch back to the part-brush to cut out the head with two strokes (middle right), the final segmentation result (right).

The main contribution of this work is an easy-to-use and effective interactive mesh segmentation system, including

- ◇ A new user interface requiring only a single stroke for most cuts;
- ◇ A novel segmentation algorithm based on cutting along an isoline of a harmonic field;
- ◇ A robust face-based surface metric defined on a filtered normal field for an improved region-growing algorithm.

2. Related Work

We classify the previous mesh segmentation methods into two classes: fully automatic methods and interactive methods.

Automatic mesh segmentation A variety of methods for automatically partitioning a mesh into parts have been proposed in the last decade. They are commonly classified into part-based and patch-based [Sha08]. We exclude patch-based methods here, since unlike our patches which are meaningful flatter regions that merit cuts, a patch in these patch-based techniques refers to a collection of connected faces used for further processing (i.e., parameterization, morphing etc.).

A majority of methods rely on clustering mesh vertices or faces using either top-down (hierarchical) [STK02, GWH01, GG04, KT03] or bottom-up (region growing) [MW99, Zuc02] approaches. Other methods use some primitive extraction [MPS*04] or fitting [AKM*06] techniques to find semantic components. Mesh segmentation methods based on structural information [LKA06, ATC*08, BDBP09] or shape analysis tools [LZ07, GF08] have also been proposed. We refer to [APP*07, Sha08] for recent surveys and [CGF09] for a comprehensive study.

Recently, Shapira et al. [SSCO08] use the Shape Diameter Function(SDF) to segment mesh surfaces. Like ours, they define a certain surface field and cut the surface along isovalues. However, their method is slow as computing the SDF function requires local ray-tracing. Besides, while the harmonic field supports the user's dynamic control through drawing new strokes, the SDF field is generally static and not suited for interactive segmentation framework.

Automatic methods implicitly require the users to adjust

parameters, such as the number of components in the segmentation or the termination condition. Yet, the segmentation results often do not agree with what the user has in mind. Besides, most automatic methods require a post-processing step to refine segmentation boundaries, using algorithms such as graph-cut [KT03, KT05], geometric snake [LLS*04] and active contour [KT09].

Interactive Mesh Segmentation There have been few methods proposed for interactive mesh segmentation. Earlier work of [FKS*04] and [LLS*04] use intelligent scissoring tools to cut the mesh. To execute a cut, the user needs to sketch along the cut boundary to specify a sparse set of points. This task could be quite tedious when the cutting boundary is complicated such as the head of Neptune (Figure 14). In addition, such tools may fail to achieve desirable boundaries for regions with noise or fine geometric details.

Recently, several interactive mesh cutting tools were proposed for fast and easy mesh decomposition [JLCW06, WPP*07, MJL08, BMB09]. The user draws strokes to specify foreground/background regions for each cut. These approaches have similar interfaces and employ either region-growing [JLCW06, WPP*07] or graph-cut [MJL08, BMB09] algorithms. Our patch-brush complementary tool shares similar properties with [JLCW06, WPP*07] in the underlying algorithm, but it uses a more robust face-based metric and is applied to the dual domain (i.e., mesh facets). Like our part-brush, the method in [MJL08] uses a harmonic field for segmentation, but it takes the field directly as input for graph-cut whereas our method considers isolines as cutting boundaries, avoiding common problems of greedy algorithms. In contrast with these previous interactive works, our novel cross-boundary brushes allow direct control over the cutting boundaries, thus simplifies the interactive segmentation task.

3. System Tools

Our interface is simple and easy to use. To execute a cut, the user draws a stroke using a simple mouse click-drag-release operation, and our system returns a cut automatically. The system allows the user to switch between part-brush vs. patch-brush, single-stroke vs. multiple-stroke easily by pressing a button. Figure 2 illustrates the use of the two brushes to segment different components of the bunny model. We now introduce the underlying algorithms of the two brushes provided by our system.

We introduce the formulation of our part brush for multiple strokes here, even though in most cases a single stroke is sufficient to segment a component. Denote $U = \{p_1, p_2, ..., p_c\}$ and $V = \{q_1, q_2, ..., q_c\}$ as the corresponding sets of start and end mesh vertex indices of the user strokes, here c is the number of strokes. A harmonic field is computed by solving the following Poisson equation:

$$\Delta\Phi = 0 \tag{1}$$

with boundary constraints $\Phi(x) = 1, x \in U$ and $\Phi(x) = 0, x \in V$ (or the other way round), where Δ is the Laplace operator with cotangent weighting [MDSB02].

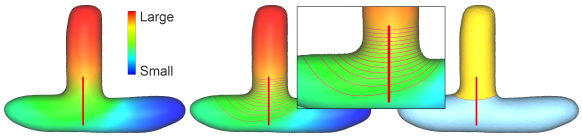

Figure 3: Overview of part-brush interface. (Left) user-drawn stroke and generated harmonic field; (middle) extracted isolines running across the stroke; (right) best cut isoline and segmentation result.

3.1. Part-brush

The Poisson equation can be solved in the least-squares sense through the matrix form $\mathbf{A}\Phi = b$, with

$$\mathbf{A} = \begin{bmatrix} \mathbf{L} \\ \mathbf{WP} \end{bmatrix} \text{ and } b = \begin{bmatrix} \mathbf{0} \\ \mathbf{WB} \end{bmatrix}, \tag{2}$$

where $\mathbf{L}$ is the cotangent Laplacian matrix, $\mathbf{W} = diag\{w, ..., w\}$ is a $2c \times 2c$ positional weighting matrix ($w = 1000$ in our experiments). $\mathbf{P}$ is the positional matrix of size $2c \times n$, with n denoting the number of mesh vertices:

$$\mathbf{P}_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \{(0,p_1),(1,p_2),...,(2c-1,q_c)\} \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

and $\mathbf{B}$ is a $2c \times 1$ matrix $[1,1,...,0,0]^T$. Solving the above system is equivalent to solving the following normal equation $\Phi = (\mathbf{A}^\mathbf{T}\mathbf{A})^{-1}\mathbf{A}^\mathbf{T}b$.

The generated harmonic field is a smooth field (Figure 3). Each mesh vertex i is associated with a harmonic value $\Phi(i)$. We uniformly distribute a set of isolines $ISO = \{I_1, I_2, ..., I_N\}$ ($N = 15$ in our experiments) on the generated harmonic field, with the ith isoline I_i having the isovalue $i/(N+1)$. The resulting isolines generally run across the user's stroke (Figure 3).

It is noteworthy that for meshes containing tunnels, there might be multiple isolines corresponding to a single isovalue. To constrain the desired cut to lie across the user strokes, we simply discard the isolines that do not run through any user stroke. A key characteristic of the isolines is that each isoline is a connected loop dividing a mesh into two-subparts. Besides, the isolines extracted from such harmonic field are smooth, requiring no post-processing. This property holds even for noisy models, as we show in Section 3.1.3. Note that the isolines are transparent to the user. We design a multi-scale isoline-selection scheme to automatically select the best isoline as the cutting boundary based on two factors: user's intention and local shape information.

◇ **User intention** Our system assumes that the user's strokes convey certain reasonable information. When the user draws a stroke across a region that merits a cut, it signals that the best cut should runs across somewhere in the middle of the stroke, but not near its two ends. We therefore

introduce a term called *centerness* that measures the location of each isoline along the stroke: $C_i = e^{-\frac{(i-t)^2}{2t^2}}$ with $t = N/2$. When multiple strokes are drawn, we simply use the first C_i.

◇ **Local shape information** The minima rule suggests cuts on concave regions [HRP*84,HS97]. It is true that from the human perception, a protruding branch (part component) always introduces concave boundaries, this motivates us to design a metric that captures the local shape's concaveness, as in our case the concaveness of isolines. Specifically, for each isoline, we first compute its local radius r_i as its length l divide by 2π. The radius distribution of the isoline set represents the local volume variation of the underlying object (radius serves as local thickness). Then, we compute a metric $\triangle_i = 2r_i - r_{i-1} - r_{i+1}$ for each isoline I_i. Small negative $\triangle_i$ values mean large concave regions. A similar formulation is defined in the segmentation method of [ATC*08]. However, this fact may be violated when the isolines lie too close to each other since $\triangle_i$ tends to zero. In that case, it would be necessary to use a larger step by considering $\triangle_{ik} = 2r_i - r_{i-k} - r_{i+k}$ for some $k > 1$. Therefore, instead of considering only adjacent isolines, we use a multi-scale metric $\triangle_i = \sum_k \triangle_{ik}$. To penalize isolines of large distances from I_i, we convolve $\triangle_i$ with a Gaussian function $f(k) = e^{-\frac{(k-1)^2}{2\sigma^2}}$ ($\sigma = 2$ in our experiments). The final $\triangle_i$ is defined as

$$\triangle_i = \frac{\sum_k f(k)\triangle_{ik}}{\sum_k f(k)}. \tag{4}$$

Combining the two terms, the final metric for each isoline is $M_i = C_i\triangle_i$. The isoline with the smallest M_i is identified as the best cut.

3.1.1. Dynamic Updating of Harmonic Field

Each time the user draws stroke(s) to execute a new cut, our system updates a harmonic field and identifies the best isoline cut. Although solving the linear system once could be super-linear using modern sparse linear solvers, for an interactive application, it is still slow if each updating requires refactorization of a new linear system. We observe that each time new strokes invoke an update of the linear system, it only affects the positional constraints, while the Laplacian matrix remains unchanged. In particular, consider the following equation:

$$\mathbf{A}^\mathbf{T}\mathbf{A} = [\mathbf{L}^\mathbf{T}\ \mathbf{Q}^\mathbf{T}]\begin{bmatrix}\mathbf{L}\\ \mathbf{Q}\end{bmatrix} = \mathbf{L}^\mathbf{T}\mathbf{L} + \mathbf{Q}^\mathbf{T}\mathbf{Q}, \tag{5}$$

with $\mathbf{Q} = \mathbf{WP}$ and note that $\mathbf{Q}^\mathbf{T}\mathbf{Q} = \mathbf{G}^\mathbf{T}\mathbf{G}$ where $\mathbf{G}$ is a diagonal matrix with entities:

$$\mathbf{G}_{ij} = \begin{cases} w & \text{if } i = j \text{ and } i \in \{U \cup V\} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Thus, each update of the user constraints bring changes to matrix $\mathbf{G}$ only, which contains merely $2c$ non-zero elements. By precomputing the Choleskey factorization $\mathbf{F}^\mathbf{T}\mathbf{F}$ of $\mathbf{A}^\mathbf{T}\mathbf{A}$, only $\mathbf{F}$ needs to be factorized for each updating. This can be solved quite efficiently using some numerical factorization downdating and updating techniques [XZCOX09]. We use the CHOLMOD package [Dav08] which contains an implementation of fast Cholesky factorization downdating/updating technique.

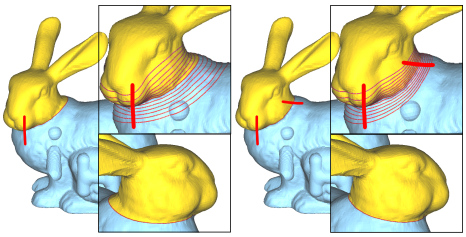

Figure 4: User can refine a cutting boundary to follow the geometry more closely by drawing additional strokes.

Although the dynamic updating scheme guarantees interactive rates, the initial factorization process may still take several seconds for large models. We further improve user's real-time experience by adding a preprocessing step. Upon loading a mesh model, we compute a harmonic field with two randomly selected mesh vertices as the boundary constraint. This process factorizes the linear system once so that the dynamic updating scheme can be immediately applied when the user draws a part-brush stroke.

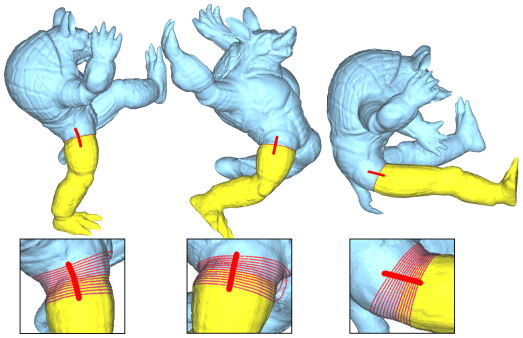

Figure 5: Insensitivity of our part-brush to pose variation.

3.1.2. Boundary Mollification

Our part-brush produces smooth boundaries. However, with a single cross-boundary stroke, the cutting boundary may not closely follow the shape geometry (Figure 4). Instead of using some graph cut algorithms or geometric snakes to enforce a cut boundary that respects the local geometric features, we resort to a strategy consistent with our user interface. Specifically, we let the user draw additional strokes to locally refine a boundary if it deviates from the user's expectation. More strokes means adding more user constraints to our system, enabling more control of the cutting boundary. Figure 4 right shows an example of boundary refinement (see also the accompanying video).

Since the cutting isoline passes through mesh faces, to

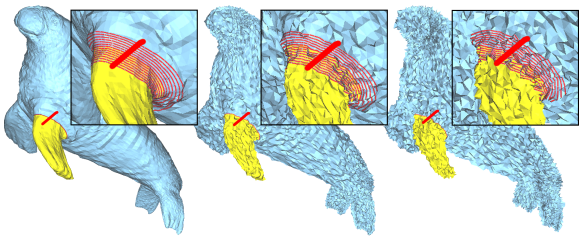

Figure 6: Our part-brush is robust against noise. From left to right, the walrus model is coupled with 0.1, 0.5 and 1.0 mean edge-length Gaussian noise, respectively.

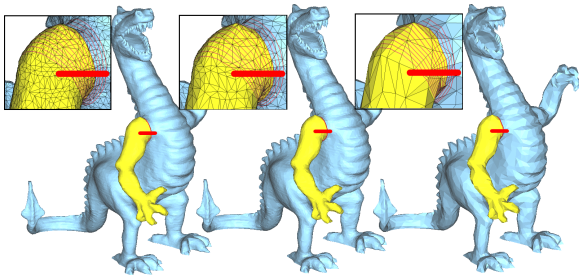

Figure 7: Segmentation results of our part-brush applied to the raptor model with different tessellations. From left to right: 25k vertices, 12k vertices, 3k vertices.

keep the cutting boundary taut and smooth, we need to subdivide these faces along the cutting isoline. However, as any modification in mesh connectivity results in a change of the linear system affecting the real-time performance, we display the subdivided faces but use the original mesh in the linear system.

3.1.3. Insensitivity to Pose, Noise, Tessellation and Stroke Variation

Segmentation using our part-brush has several good properties. Our carefully designed shape metric $\triangle_i$ captures the volume variation, thus is robust against isometric deformation, noise and different tessellation. Moreover, the generated harmonic field is always smooth even in the presence of isometric-deformation, noise and coarse tessellation because the cotangent weighted Laplacian depends only on the angles of the triangles. Figure 5 shows that similar segmentation results are obtained for the Armadillo model in different poses. The user draws a stroke at similar locations for all the three Armadilo models and our system identifies consistent cutting isolines for all of them. Figure 6 and 7 demonstrate the insensitivity of our part-brush against noise and tessellation, respectively. Observe that all the cutting boundaries are smooth and have similar locations even for models with noise or coarse tessellations.

Our part-brush is also insensitive to variation in user's stroke. Figure 8 shows a cactus model segmented with slightly different user strokes. The four strokes vary in orientation, length and start-end positions. Yet, similar segmentation results are obtained in all these cases.

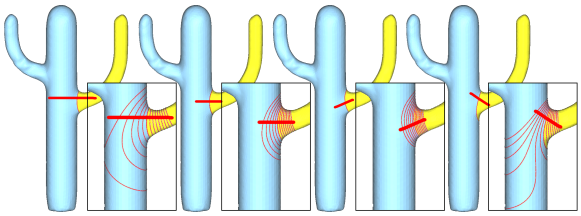

Figure 8: Similar segmentation results are obtained despite variation in the user's stroke.

3.2. Patch-brush

Our part-brush works well for segmenting part-components, but not for segmenting flat regions with sharp corners or creases. This limitation stems from the continuous nature of the harmonic field, making the extracted isolines fail to strictly capture sharp features which are piecewise smooth. Therefore we introduce the patch-brush to complement the usage of the part-brush.

By observing that flatter patch components usually contain sharp features which form local extrema, we build our patch-brush based on a region growing algorithm. Similar to [JLCW06, WPP*07], our patch-brush uses a greedy algorithm to grow two groups of competitors. Each pair of adjacent faces i and j is associated with a merge-cost $cost_{ij}$. Like the part-brush, our system also supports multiple-strokes for the patch-brush. Initially, the first group contains all the starting faces corresponding to the user strokes and the second group contains all the ending faces. Then we let each group grow step by step by absorbing its neighboring ungrouped faces. In each step, the ungrouped face with the smallest $cost_{ij}$ is absorbed by its neighboring group. The competition process stops when all the faces in the mesh are grouped.

Like many greedy algorithms, the optimal solution could fall into a local minimum, resulting in an undesirable cutting boundary. When this happens, the user can draw additional strokes to add more initial faces, which is similar to [JLCW06, WPP*07]. Note that the above group competition algorithm can be applied locally instead of the whole mesh. In practice, we always assign a new group label to all the faces of the newly cut-out mesh part. Then the next competition process is applied only to the local mesh region where the strokes lies.

The key in designing the group competition algorithm lies with defining a good cost function $cost_{ij}$. Several surface-based cost functions have been introduced [JLCW06, WPP*07, BMB09, LZSCO09], but most of which are either vertex-based or costly to compute. We define a simple face-based cost function which performs well in all our experiments:

$$cost_{ij} = (1 + \|\vartheta_{ij}\|)\|\vec{n}_i - \vec{n}_j\|, \tag{7}$$

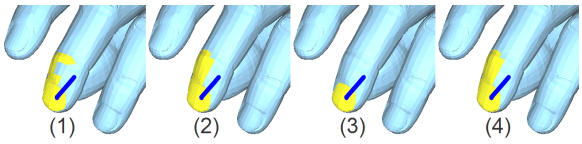

Figure 9: Combining the two terms (in Equation 7) of our face-based metric gives better results than the single terms. Using (1) term I only, (2) term II only, (3) terms I & II and (4) the face-based metric in [BMB09].

where

$$\vartheta_{ij} = \frac{\vec{n}_i \cdot \vec{e}_{ij}}{\|\vec{e}_{ij}\|}, \tag{8}$$

with $\vec{e}_{ij}$ denoting the edge connecting the centers of face i and face j. The term ϑ_{ij} is a discrete approximation of the normal curvature in the direction of $\vec{e}_{ij}$ in the dual domain and the term $\|\vec{n}_i - \vec{n}_j\|$ is an approximation of curve length on a unit Gaussian sphere [WPP*07]. Both favor sharp features. Since the term ϑ_{ij} tends to be relatively small, we add 1 to ϑ_{ij} to balance the influence of the two terms. We observe that patch components may be either slightly protruding (e.g., the 'H' character in Figure16) or sinking (e.g., the 'i' character in Figure16). Thus we take the absolute value of ϑ_{ij} so as to give similar weights to both convex and concave edges. As demonstrated in Figure 2, our patch-brush succeeds in cutting out both characters. Figure 9 shows that the combination of the two terms in Equation 7 produces better results than the single terms alone. In Section 5, we compare with previous vertex-based surface metrics, and show that our face-based metric produces better segmentation results (Figure 16).

Normal Filtering We observe that all region growing methods are sensitive to noise, so is ours. Noise or fine features, as in the Armadillo model, introduce local minima at which the region growing algorithms could get stuck, resulting in undesirable boundaries (Figure 10 top row). To make our face-based metric more robust against surface fine details or noise, we filter the surface normal field before applying our group competition algorithm. Specifically, we replace $\vec{n}_i$ and $\vec{n}_j$ in Equation 7 with an updated $\vec{n}_i'$ and $\vec{n}_j'$ respectively,

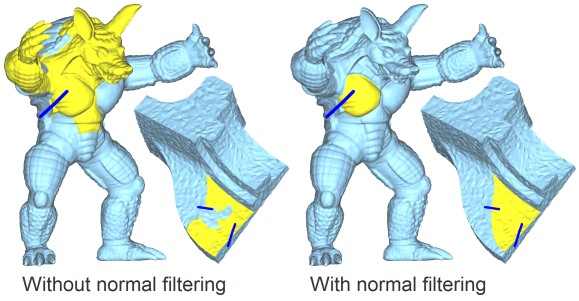

Figure 10: Normal filtering improves the segmentation results even when the models are coupled with noise (the fandisk).

where

$$\vec{n}_i' = normalize(\sum_{j \in NII(i)} w_{ij}\vec{n}_j). \tag{9}$$

Here $NII(i)$ denotes the neighborhood of face i defined as all the faces that share common vertices with face i and w_{ij} is a Gaussian function $w_{ij} = e^{-\frac{\|\vec{n}_i - \vec{n}_j\|^2}{2\sigma^2}}$. Such a weighting scheme gives smaller influence to face normals that have larger difference with face i (i.e., preserve features) while ignoring small normal differences that might be introduced by noise. Again, σ is the kernel width which controls the fall-off of the Gaussian norm. Theoretically, σ should be tuned properly according to the noise level in order to obtain the best results. However, we found that a fixed parameter $\sigma = 0.35$ performs well in all our experiments. Figure 10 shows the improved segmentation results with normal field filtering. Note that the normal filtering step can be done as preprocessing.

Although the normal filtering improves the performance of our patch-brush, it is still not easy to segment complex geometry features such as hair since the region-growing algorithm could still get stuck at some local minima. In these cases, it often requires multiple user strokes to control the boundary locally as shown in Figure 11.

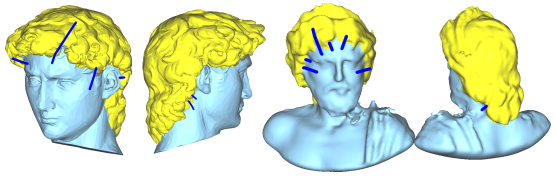

Figure 11: Complex geometric features such as hair often require multiple strokes to segment. In these two examples, the hair is separated from the head using 6 and 7 patch-brush strokes respectively.

4. Experimental Results

We have applied our system to segment a vast variety of models, including the recently introduced 3D Segmentation Benchmark of Chen et al. [CGF09] and other models with complex structures as well as CAD models. For all the models we tested, our interactive segmentation tools can effectively fulfill the user's intention. It usually takes less than 1 second to execute a cut and no more than half a minute to finish segmenting a model. In our experiments, the user's usage of the two types of brush generally agree with their design purposes. In some cases, both brushes can be used to achieve similar cutting results, e.g., legs of table, arms of octopus and base of the Eros head shown in Figure 12. So in these cases the user is free to use either brush. We found that our patch-brush is well-suited for segmenting CAD models since many of which contain flat patches as demonstrated in Figure 1 and Figure 13.

Princeton Segmentation Benchmark. Figure 12 shows our segmentation results for the Princeton Segmentation Benchmark [CGF09]. This benchmark contains 380 models in 19

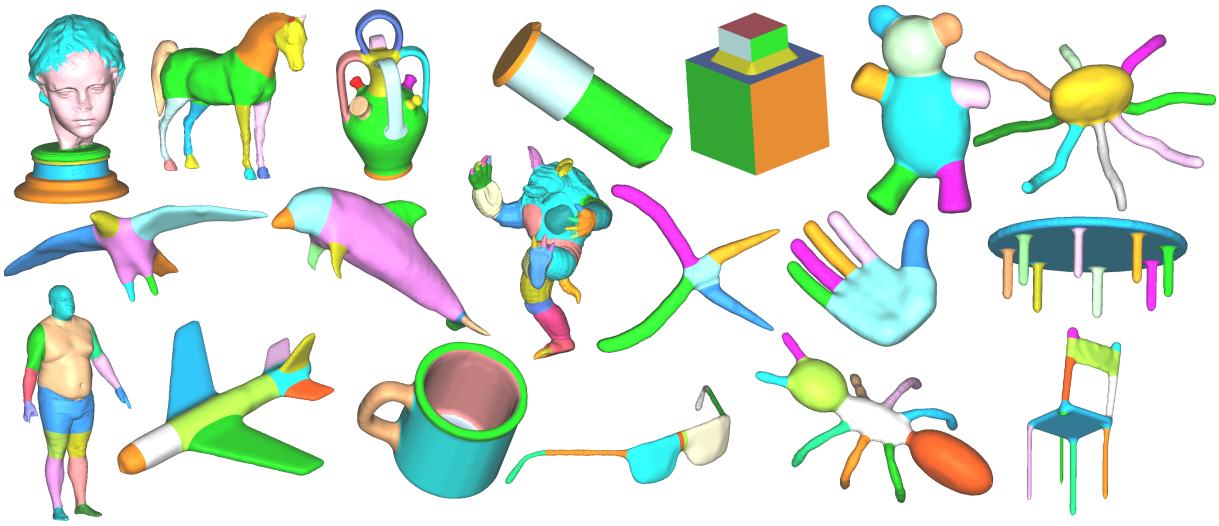

Figure 12: Results of applying our method to Princeton Benchmark models. Most cuts are executed using single-stroke mode except the face of Eros, left arm of teddy, upper and bottom parts of cup and some tunnels in teapot. Observe that our boundaries follow the geometry features.

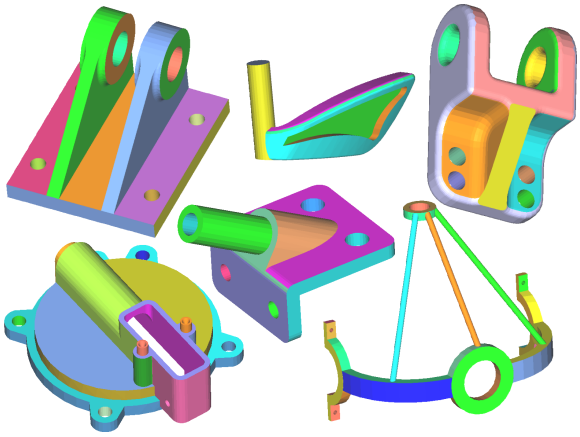

Figure 13: Results of applying our method to CAD models. All models are segmented using patch-brush using single stroke. Models are from National Design Repository, Drexel University.

Figure 14: Our tools can successfully cut out all the desired components for complex models, including those that are hard to detect automatically, such as suction cups and eyes of octopus.

classes. We have tested hundreds of these models, but show only one representative model from each class in Figure 12. When drawing strokes, we aim to achieve the segmentation results provided in [CGF09]. Our final cutting results visually agree with their results. However, in most cases, our cutting boundaries follow the shape geometry better. This is because their method forms cut by geodesic shortest path, making their boundaries disregard local geometry. Whereas in the our case, the isolines are smooth and we could draw additional strokes, enforcing the isolines to better respect the local geometry. For all the models shown in Figure 12, we use only a single stroke to cut out each segment except for the following: upper-part of cup (3 strokes), face of Eros (3 strokes), left hand of Teddy bear (2 strokes) and some tunnels in the teapot model (2-3 strokes). For all the models tested, our brushes suffice to obtain good segmentation results using at most 4 strokes to execute each cut except for some models in the head class (Figure 11). Nevertheless, our system is still able to cut out the face of Eros head (Figure 12) using three strokes of our patch-brush. Finally, our user interface is easier to use than theirs, which requires the user to rotate the model and click multiple times on the cutting boundary to obtain a cut.

Other Models Figure 13 show our segmentation results for

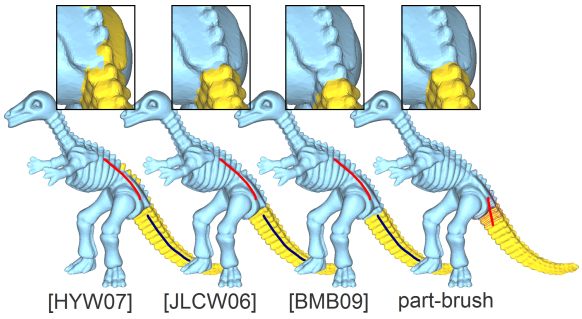

Figure 15: Comparison of our part-brush with [JLCW06, WPP*07, BMB09] without applying boundary refinement. For their methods, two strokes specifying the foreground and background parts are drawn. Note the jaggy cutting boundaries. For our method, a single cross-boundary stroke is drawn.

some CAD models. Note that, in all these examples, the cutting boundaries follow the sharp features. Besides, for all the CAD models tested, we use only a single stroke to execute a cut. Figure 14 shows the segmentation results of other complex models containing multiple loops or complex structures. Our system successfully cut along all the desired boundaries. In the dancing-children and Neptune models, we use a single patch-brush stroke to cut out the base plate. For the base plate of the fertilty model, we use the part-brush (2 strokes) to execute the cut because the bottom region of the model is very smooth. All the other cuts are executed using single stroke except for the head and upper-leg of Neptune (4 strokes and 2 strokes), legs of dancing-children (2-3 strokes) and eyes, suction cups and arms of octopus (2-3 strokes). For models with complex geometry features, precisely locating a cut usually requires several rotations and strokes. For example, it took about 2 minutes to precisely locate all cuts for the octopus and Neptune models.

5. Comparison with Previous Methods

With automatic segmentation methods Our extensive experimental results have demonstrated the discriminative power of the two brushes of our system. We believe that our tools are very useful for cutting out components that are hard for existing automatic segmentation methods to detect and segment. For example, our part-brush is able to cut out the suction cups on the octopus' arms (2 part-brush strokes) as well as its eyes (2 part-brush strokes) (Figure 14). Another example is the chest of the Armadilo model shown in Figure 10, where we use a single patch-brush stroke.

With interactive segmentation methods To demonstrate the performance of our segmentation framework over existing interactive tools, we compare our system with three recent interactive segmentation methods proposed in [JLCW06, WPP*07, BMB09]. We compare both of our brushes with their tools. Since our approach requires no post-processing, we do not apply boundary refinements to their results either. Figure 15 shows the comparison results

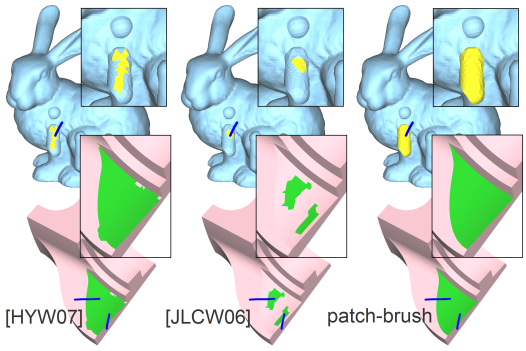

Figure 16: Comparison of our face-based metric and the vertex-based metrics of [JLCW06, WPP*07]. No boundary refinements are applied. For their methods, the two mesh vertices associated to the endpoints of each user stroke specify the foreground/background information. Our method uses the two mesh faces of each stroke. No normal filtering is applied. Observe that our cutting boundaries strictly follow the geometry features.

of our part-brush with all three methods. It shows that our part-brush successfully obtains a user desired cut while their cut boundaries are all jaggy [JLCW06, WPP*07] due to the sensitivity of their underlying algorithms towards fine details.

For the patch-brush, we compare our metric with previous metrics. In Figure 9, we compare it with the face-based metric defined in [BMB09]. Figure 16 shows a comparison with the vertex-based metrics used in [JLCW06, WPP*07]. Both figures demonstrate that our face-based metric performs better along sharp features. Observe that in Figure 16 the methods of [JLCW06, WPP*07] fail to cut out the character 'i' in the bunny model. This is because their metrics tend to locate cutting boundaries at local concave regions while our metric favors both convex and concave regions over flat regions.

6. Conclusions

We have presented an effective interactive system with an easy-to-use UI for mesh segmentation. The system provides easier user control over the cutting boundaries than previous interactive systems, and produces better segmentation results. Our main contribution is in the design of the part-brush. With part-brush, the cutting boundaries are isolines, which are a robust shape representation, giving several desirable properties. The part-brush provides sufficient discriminative power for cutting out protruding semantic parts. Complemented by the patch-brush designed for cutting flat regions with sharp boundaries, our system is capable of dealing with a large set of real world objects.

Our approach has its own limitations. Since our part-brush relies on the isolines of harmonic field, as a limitation, our system can not process non-manifold mesh surfaces which do not have good harmonic fields. Our current system is capable of dealing with meshes containing hundreds of thousands of vertices at interactive rates, however, for models

of larger size, solving the linear system would become a bottleneck even with advanced numerical solvers. Multi-resolution strategies may be used in such cases. Another limitation comes from the underlying region-growing algorithm of our patch-brush. The inherent sensitivity to local minima hinders the greedy algorithms from reaching the global optimum, requiring more strokes to segment complex geometric details such as hair. This problem may be alleviated by designing a more robust surface metric or a more powerful part-brush that makes the isoline acts like a geometric snake to automatically attach to the local geometry features.

Acknowledgement: We would like to thank the anonymous reviewers for their valuable comments. We also thank Hongbo Fu and Oscar Kin-Chung Au for insightful discussion, Ligang Liu for the code of Easy Mesh Cutting, and Pedro Sander for video narration. This work is supported by the Hong Kong Research Grant Council (Project No: GRF619908).

References

[AKM*06] ATTENE M., KATZ S., MORTARA M., PATANE G., SPAGNUOLO M., TAL A.: Mesh segmentation - a comparative study. In *SMI '06* (2006), p. 7.

[APP*07] AGATHOS A., PRATIKAKIS I., PERANTONIS S., SAPIDIS N., AZARIADIS P.: 3D mesh segmentation methodologies for cad applications. *Computer-Aided Design & Applications 4*, 6 (2007), 827–841.

[ATC*08] AU O. K.-C., TAI C.-L., CHU H.-K., COHEN-OR D., LEE T.-Y.: Skeleton extraction by mesh contraction. *ACM Trans. Graph. 27*, 3 (2008).

[BDBP09] BERRETTI S., DEL BIMBO A., PALA P.: 3D mesh decomposition using Reeb graphs. *Image Vision Comput. 27*, 10 (2009), 1540–1554.

[BMB09] BROWN S., MORSE B., BARRETT W.: Interactive part selection for mesh and point models using hierarchical graph-cut partitioning. In *GI '09* (2009), pp. 23–30.

[CGF09] CHEN X., GOLOVINSKIY A., FUNKHOUSER T.: A benchmark for 3D mesh segmentation. *ACM Trans. Graph.* (2009), 1–12.

[Dav08] DAVIS T.: *User Guide for CHOLMOD: a sparse Cholesky factorization and modification package*, 2008.

[DBG*06] DONG S., BREMER P.-T., GARLAND M., PASCUCCI V., HART J. C.: Spectral surface quadrangulation. *ACM Trans. Graph.* (2006), 1057–1066.

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. *ACM Trans. Graph.* (2004), 652–663.

[GF08] GOLOVINSKIY A., FUNKHOUSER T.: Randomized cuts for 3D mesh analysis. *ACM Trans. Graph.* (2008), 1–12.

[GG04] GELFAND N., GUIBAS L. J.: Shape segmentation using local slippage analysis. In *SGP '04* (2004), pp. 214–223.

[GWH01] GARLAND M., WILLMOTT A., HECKBERT P. S.: Hierarchical face clustering on polygonal surfaces. In *I3D '01* (2001), pp. 49–58.

[HRP*84] HOFFMAN D., RICHARDS W., PENTL A., RUBIN J., SCHEUHAMMER J.: Parts of recognition. *Cognition 18* (1984), 65–96.

[HS97] HOFFMAN D. D., SINGH M.: Salience of visual parts. *Cognition 63*, 1 (1997), 29–78.

[JLCW06] JI Z., LIU L., CHEN Z., WANG G.: Easy mesh cutting. *Computer Graphics Forum (Proc. of EuroGraphics 2006) 25*, 3 (2006), 283–291.

[KT03] KATZ S., TAL A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph.* (2003), 954–961.

[KT05] KATZ S. L. G., TAL A.: Mesh segmentation using feature point and core extraction. In *The Visual Computer* (2005), vol. 21, pp. 649 – 658.

[KT09] KAPLANSKY L., TAL. A.: Mesh segmentation refinement. In *PG '09* (2009), vol. 25, p. to appear.

[LKA06] LIEN J.-M., KEYSER J., AMATO N. M.: Simultaneous shape decomposition and skeletonization. In *SPM '06* (2006), pp. 219–228.

[LLS*04] LEE Y., LEE S., SHAMIR A., COHEN-OR D., SEIDEL H.-P.: Intelligent mesh scissoring using 3D snakes. In *PG '04* (2004), pp. 279–287.

[LZ07] LIU R., ZHANG H.: Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum (Proc. of Eurographics 2007) 26*, 3 (2007), 385–394.

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

[MDSB02] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and Mathematics III* (2002), 35–57.

[MJL08] MENG M., JI Z., LIU L.: Sketching mesh segmentation based on feature preserving harmonic field. *Journal of Computer-Aided Design & Computer Graphics (in Chinese) 20* (2008), 1146–1152.

[MPS*04] MORTARA M., PATANÈ G., SPAGNUOLO M., FALCIDIENO B., ROSSIGNAC J.: Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. In *SMA '04* (2004), pp. 339–344.

[MW99] MANGAN A., WHITAKER R.: Partitioning 3D surface meshes using watershed segmentation. *IEEE Trans. Vis. & Comput. Graph. 5*, 4 (1999), 308–321.

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

[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.

[STK02] SHLAFMAN S., TAL A., KATZ S.: Metamorphosis of polyhedral surfaces using decomposition. In *Computer Graphics Forum* (2002), pp. 219–228.

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

[XZCOX09] XU K., ZHANG H., COHEN-OR D., XIONG Y.: Dynamic harmonic fields for surface processing. *Computers and Graphics (SMI '09) 33* (2009), 391–398.

[ZRKS05] ZAYER R., RÖSSL C., KARNI Z., SEIDEL H.-P.: Harmonic guidance for surface deformation. *Computer Graphics Forum (Proc. of EUROGRAPHICS 2005) 24*, 3 (2005), 601–609.

[Zuc02] ZUCKERBERGER E.: Polyhedral surface decomposition with applications. *Computers and Graphics 26*, 5 (2002), 733–743.