Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Computer Aided Geometric Design 22 (2005) 444–465

COMPUTER AIDED GEOMETRIC DESIGN

www.elsevier.com/locate/cagd

Mesh scissoring with minima rule and part salience

Yunjin Lee [a], Seungyong Lee [a,*], Ariel Shamir [b], Daniel Cohen-Or [c], Hans-Peter Seidel [d]

[a] *POSTECH, Republic of Korea*
[b] *The Interdisciplinary Center, Israel*
[c] *Tel Aviv University, Israel*
[d] *MPI Informatik, Germany*

Available online 23 May 2005

Abstract

This paper presents an intelligent scissoring operator for meshes. Unlike common approaches that segment a mesh using clustering schemes, here we introduce a method that concentrates on the contours for cutting. Our approach is based on the *minima rule* and *part salience theory* from the cognitive theory. The minima rule states that human perception usually divides a surface into parts along the concave discontinuity of the tangent plane. The part salience theory provides factors which determine the salience of segments. Our method first extracts features to find candidate contours based on the minima rule. Subsequently, these open contours are prioritized to select the most salient one. Then, the selected open contour is automatically completed to form a loop around a specific part of the mesh. This loop is used as the initial position of a 3D geometric snake. Before applying a snake, we measure the part salience of the segments obtained by the completed contour. If conditions for the salience are not met, the contour is rejected. Otherwise, the snake moves by relaxation until it settles to define the final scissoring position. In this paper, we focus on a fully automatic scissoring scheme; nevertheless, we also report on semi-automatic user interfaces for intelligent scissoring which are easy to use and intuitive.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Mesh segmentation; Mesh partitioning; Part-type segmentation; Geometric snake

* Corresponding author.
E-mail address: leesy@postech.ac.kr (S. Lee).

0167-8396/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.cagd.2005.04.002

1. Introduction

In the last decade we have witnessed tremendous developments in techniques for modeling, reconstructing, and visualizing 3D meshes for various applications. With the advent of advanced scanning technology, meshes are becoming larger and more complicated. To repair, manipulate, or modify meshes, there is a need for interactive and intelligent tools that assist and enable simpler mesh editing. One such basic operator is the *scissoring* operator, which extracts sub-parts and pieces from existing meshes. In addition, many mesh related algorithms, such as parameterization, compression, morphing, and matching, use mesh partitioning as an initial stage. This means mesh partitioning has become a key ingredient for many mesh manipulation applications (Shamir, 2004).

Most previous work on mesh partitioning use clustering of similar mesh elements or components and then refine the border between these parts to find the segmentation (Katz and Tal, 2003; Garland et al., 2001; Inoue et al., 2001; Lévy et al., 2002). In contrast, our approach directly targets the cutting contours by providing a scissoring operator, which separates a given mesh into two disjoint pieces along a closed contour lying on the mesh. This idea is based on the *minima rule* from the cognitive theory. The minima rule states that human perception usually divides an object into parts along the concave discontinuity of the tangent plane (Hoffman and Richards, 1984; Hoffman and Signh, 1997). In various tests, it has been shown that humans take more notice to such discontinuity on the shape outline than to the coherency of different shape parts. Therefore, our scissoring operator searches for contour candidates that follow minimum negative curvatures. Later selected candidates are completed to form cutting loops, and their positions are refined using geometric snake movements.

Our scissoring operator is carried out using four basic steps:

1. *Feature contour extraction*: Finding and selecting a *feature contour* on the mesh (not necessarily closed).
2. *Loop completion*: Completing the feature contour to a closed *loop*.
3. *Part salience test*: Rejecting the loop if the conditions of *part salience* are not satisfied.
4. *Snake movement*: Using the loop as the initial positioning of a *geometric snake*, and evolving the snake to its final position for cutting.

The difference between automatic and semi-automatic scissoring in our approach is based on the amount of user guidance in the first and second steps. The initial feature contour on the mesh can either be indicated manually by the user or chosen automatically based on feature extraction. Completing the contour to a closed loop can either be done using guidance from the user or by an automatic procedure which directs the contour to loop around mesh parts. This approach provides the ability to define a range of possible tools from a fully automatic partitioning tool to an easy-to-use semi-automatic scissoring tool.

The basic idea of mesh scissoring with the minima rule was presented in our previous work (Lee et al., 2004). However, since the feature contour selection is based on heuristics, the scissor might cut insignificant parts, which sometimes induces user intervention to reject the scissoring results. This limitation hinders the scissoring process from being fully automatic and we put emphasis on a semi-automatic scissoring tool in (Lee et al., 2004). A central contribution of this paper is that we can overcome the limitation by incorporating the part salience test and consequently achieve fully automatic scissoring of a mesh into meaningful parts.

Our part salience test builds on the theory of *salience of visual parts* proposed by Hoffman and Singh (1997). According to their theory, the salience of a part is a function of the object size relative to the whole object, the degree to which it protrudes, and the strength of its boundary. We developed quantitative definitions for these measures that can be applied to meshes, and in particular the scissored parts. After a selected contour has been completed to a closed curve around a mesh part, we test the part salience of the segment that will be obtained by the scissoring with the curve. If the part salience conditions are not met, the curve is rejected and a new contour is selected and completed to a closed curve for scissoring.

In most cases, the partitioning is applied to a single model at a time. To efficiently process a large database of models, an automatic algorithm, such as proposed in this paper, would help the user a lot to achieve high quality results in a short time. Nevertheless, here as well, we report on semi-manual tools which are easy to use and intuitive. In this paper, in addition to the part salience test, we improve the performance of the loop completion step by adding the centricity to the weight terms for automatic completion. The centricity makes the automatic completion more robust especially when scissoring an elongated object.

Our scissoring process uses three fundamental geometric mesh attributes. The first is the curvature of the mesh and the rational behind using it lies in human perception and the minima rule. The second is the centricity of positions on the mesh, which separates main object parts from the peripheral. The last attribute deals with the length and smoothness of a scissoring cut. Using it assures smoother and shorter interfaces between segmented parts. As a result, our scissoring approach has the following properties:

- The approach is guided by fundamental mesh geometric attributes based on perception.
- The final position of the cut is smoother and presents a more meaningful boundary as a result of using the geometric snake.
- The scissoring operator generates salient segments in a fully automatic way.
- The approach enables a continuous range of tools between fully automatic and manual scissoring.

Our scissoring operator is a part-type segmentation method, which divides a mesh into meaningful parts along cutting contours (Shamir, 2004). Since extracted parts represent independent components of a model, the part-type segmentation can be used for skeleton extraction (Katz and Tal, 2003), 3D morphing (Shlafman et al., 2002), shape matching and reconstruction (Zuckerberger et al., 2002; Page et al., 2003b), modeling by example (Funkhouser et al., 2004), collision detection (Li et al., 2001), and so on.

2. Related work

Automatic partitioning of a mesh is typically performed by growing regions incrementally (Mangan and Whitaker, 1999; Kalvin and Taylor, 1996; Shlafman et al., 2002; Sorkine et al., 2002), or by merging regions (Garland et al., 2001; DeRose et al., 1998; Faugeras and Hébert, 1987). Hence, the boundaries between regions are implicitly defined by the regions themselves instead of explicitly using a scissoring operator. Following human perception and the minima rule, we use an opposite approach by extracting the boundaries first and defining the mesh sub-parts implicitly. This also enables a definition of smoother and natural looking region boundaries which are not constrained to lay on the mesh edges.

The approach presented in (Katz and Tal, 2003) refines the final cut in a fuzzy region between main parts. However, the approach still uses clustering with a threshold to determine the fuzzy region, while concentrating less on feature boundaries. Unlike our approach, it does not support intelligent manual operations and a cut is constrained to mesh edges. In (Li et al., 2001), a mesh is automatically decomposed by searching critical points of characteristic functions defined by volume features. Although the approach extracts the boundaries to determine components, it does not consider the features on the mesh surface and provides no interactive tool.

The minima rule has already been presented for segmentation of CAD models and meshes using region growing and the watershed algorithm (Page et al., 2003a). Due to the limitation of region growing, the technique cannot cut a part if the part boundary contains non-negative minimum curvatures. Similar to this paper, Page et al. (2003b) used the factors proposed in (Hoffman and Signh, 1997) to compute the salience of parts. However, contrary to our approach, they represented segmented parts as superquadric models and used the models to compute the part salience. In this paper, we present methods to directly compute the part salience on a mesh part.

Intelligent manual scissoring tools have been presented for image segmentation (Mortensen and Barrett, 1998; Mortensen and Barrett, 1995; Falcão et al., 1998). In this paper we extend these ideas to 3D mesh editing and manipulation. Such simple-to-use but intelligent tools are a must in newer applications of mesh editing such as modeling by example (Funkhouser et al., 2004), where the user extracts parts of a mesh in order to combine and paste them to other meshes. In the intelligent scissoring of (Funkhouser et al., 2004), the user paints strokes on a mesh to specify where the mesh should be segmented. The cost for searching a cut depends on the current view direction; the visible part of a cut contains the specified stroke and the other part is guided to traverse the back-side of the mesh. On the other hand, in this paper, the cost of a cut is determined by a feature contour and independent of the current view direction. Although considering the view information may help to make the cut completion more robust, our scheme has the merit of supporting fully automatic cutting without user interaction.

Snakes were presented as active contour models for semi-automatic detection of features in an image (Kass et al., 1988). Active contour models for images were extended to extract features from 3D surfaces. In (Milroy et al., 1997), the snake position is updated directly on a 3D surface. Feature curve detection on a 3D mesh by a minimal path (Cohen and Kimmel, 1997) between the source and destination points is presented in (Andrews, 2000). We follow the approach proposed in (Lee and Lee, 2002) where the snake's updated position is determined by energy minimization on a 2D embedding plane, which is computationally efficient. Using snakes as the final stage of the scissoring tool frees the user from tedious adjustments and smoothing of the cut.

3. Overview

In this section, we provide an overview of the mesh scissoring process (see Fig. 1), and highlight the differences between automatic and semi-automatic use of our approach. The following sections provide details on each of the stages.

Feature contour extraction. The first stage of finding candidate contours begins by computing the minimum curvature value for each vertex of the mesh. After proper normalization, these values are used as *feature values* on the vertices. We use thresholding and thinning to obtain several *graph structures* of a feature skeleton and then extract contour curves from the graphs. In this stage, we can also obtain a

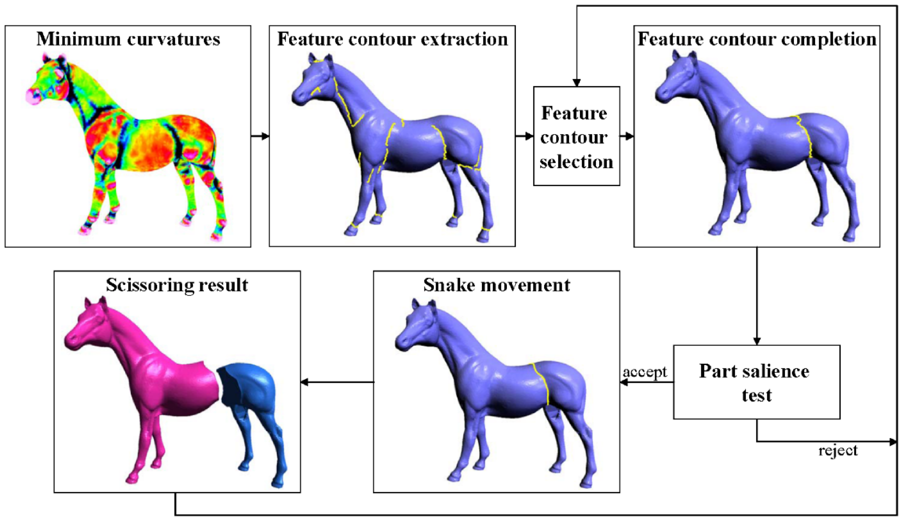


Fig. 1. Overview of the scissoring process.

candidate contour manually, which is provided by a simple gesture of drawing a 2D line segment on the viewing window.

Feature contour selection. In the second stage, we select a contour which serves as the initial *feature contour* in the succeeding scissoring stages. For automatic scissoring, we choose the best contour based on two criteria: the length of the contour and its centricity on the mesh. Subsequently, when the mesh partitioning is done, the process of choosing the best contour can be repeated recursively on each part separately for multiple-parts segmentation. Another option for designating a *feature contour* is using manual selection. A map of the automatically extracted feature contours is used to guide the user for selecting natural cutting positions.

Feature contour completion. In the third stage, the feature contour is completed to form a closed loop around a specific part of the mesh. For automatic completion, we find the weighted shortest path between the two endpoints of the contour. The weights are used both to direct the path to go over the other side of the mesh and to attract it to mesh features, instead of finding the simple shortest path. For semi-automatic completion, a line drawing gesture similar to an interface making a candidate contour can be used to create a full loop around the mesh. This line along with the center of projection defines a plane which cuts through the mesh. For delicate situations, the user can designate a sequence of points on the mesh which completes the loop.

Part salience test. The fourth stage determines whether segments obtained from the completed contour are significant enough or not. Automatic selection or completion might cause less meaningful segmentation by cutting off a piece which is not salient. For semi-automatic use, a user can make a decision by observing the shape of the loop on the mesh. To determine it automatically, we measure the salience of the segments which the scissoring operation is about to yield. The salience is estimated from the relative areas, protrusions, and strength of features on the loop.

Snake movement. The last stage of scissoring uses the closed loop as an initial position of a geometric snake. To attract the cut to mesh features, the external energy of the snake uses the mean curvature field over the mesh. The snake's internal energy controls the length and smoothness of the cut. The snake is evolved until it is settled, defining the final smooth scissoring cut. The cut does not necessarily follow the mesh edges and it can partition triangles. This creates a smoother and more natural boundary between mesh parts.

4. Feature extraction

The minima rule states that human vision tends to define areas of minimum negative curvatures, i.e., concave shape areas, as interfaces separating between object parts (Hoffman and Richards, 1984). Following this rule, we define a feature value on each vertex by the minimum curvature value, which is calculated by the tensor field computation used in (Alliez et al., 2003). Since the ranges of minimum curvature values are too diverse among different meshes (e.g., the range of a horse model is from -5.4 to 4.8 and the range of a hand model is from -0.3 to 0.19), we normalize the values. If $\kappa(v)$ is the minimum curvature value at a vertex v, the normalized value is $c_f(v) = (\kappa(v) - \mu)/\sigma$, where μ is the mean and σ is standard deviation of $\kappa(v)$ over all vertices of the mesh. We assign the normalized value $c_f(v)$ to each vertex v of the mesh as the feature value (see Fig. 2(a)). We use the mean and standard deviation for normalization instead of simple scaling (dividing by the maximum value), since the curvature value distribution is not uniform in the range. This allows us to use similar values independent of a specific mesh in the succeeding thresholding stage.

Similar to (Hubeli and Gross, 2001), we use hysteresis thresholding on the values of c_f to define high feature areas. In our experiments, the upper bound for the hysteresis is usually -1.2 and lower bound is -0.8. This means we choose high negative-curvature values which designate concave parts. In the original paper (Hubeli and Gross, 2001), the thresholding is performed on the edges, since the feature curvature values are computed for edges. In our approach, we obtain a set of vertices that pass the thresholding. Fig. 3 shows the thresholding results with the same bounds -1.2 and -0.8. We can see that the same value works well for different meshes due to the normalization of curvature values.

By connecting the vertices that pass the thresholding, we construct regions on the mesh surface defined by triangles that contain these vertices (see Fig. 2(b)). Next, the skeleton of each region is extracted by peeling vertices from the boundary of the region towards the inside. As a result of the peeling, we obtain a set of graph structures of feature skeletons (see Fig. 2(c)).

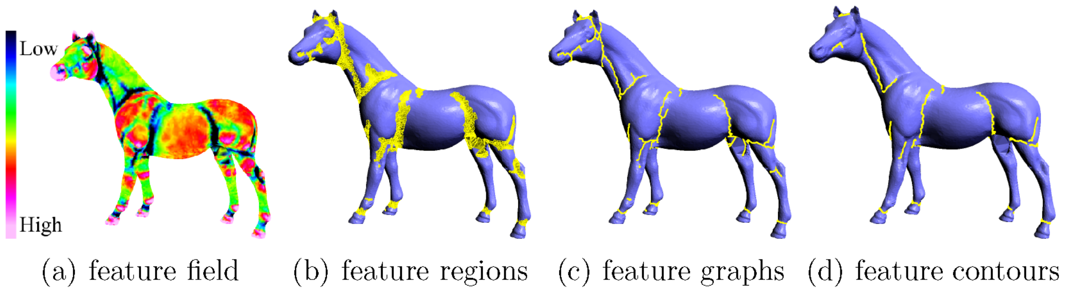


Fig. 2. Stages of feature contour extraction.

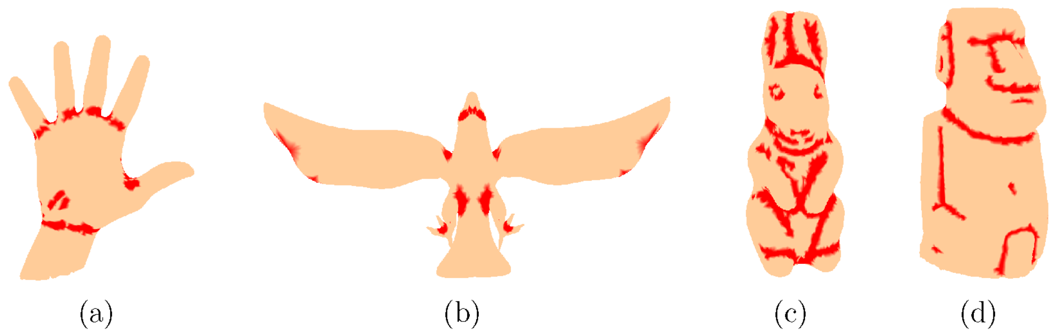


Fig. 3. Hysteresis thresholding.

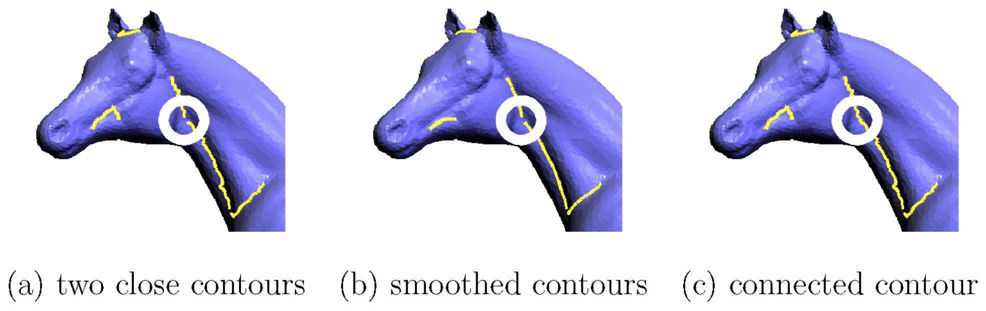


Fig. 4. Contour connection.

To create feature contours, we disconnect the skeleton graphs at vertices where more than two feature edges meet. This creates a set of non-branching feature contours, some of which are too short. We remove the useless short contours, and merge close and similar feature contours to create longer ones (see Fig. 2(d)). The similarity between two close contours is measured by the angle between their major directions around end points. We first smooth the contours by applying energy minimization loops of a geometric snake, proposed in (Lee and Lee, 2002) and also used in Section 8. Second, their major directions are obtained by averaging edge directions around the end points. We then connect two contours which have close endpoints and similar directions (see Fig. 4).

Another option for creating candidate contours is simply to draw a 2D guiding line on the screen over the mesh. This line is projected over front- and back-faces of the mesh to define a closed loop or only front-faces to define an open contour. Fig. 5 shows the case of cutting two wings of a feline model using contours created by drawing a 2D line. The left wing can be simply cut by the loop obtained by projecting a line over front- and back-faces. On the other hand, the loop around the right wing passes over the body of the feline. For that case, we can create a contour by projecting a line only over front-faces and complete it by the automatic technique presented in Section 6. Fig. 5(c) shows the result.

5. Feature contour selection

In order to perform the scissoring, a single specific feature contour must be chosen from the candidates obtained in the feature extraction stage. It is important to determine the selected order of contours because

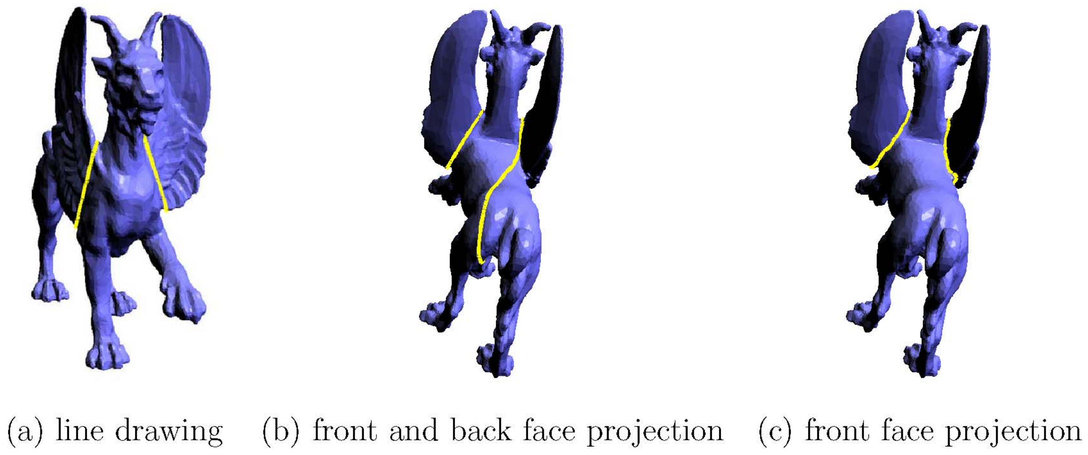


Fig. 5. Contours created by user specification: (a) 2D line drawing over the mesh; (b) contours created by projecting onto front & back faces; (c) contours created by projecting only onto front faces and applying automatic curve completion.

our approach prevents the boundaries of segments from crossing each other, which limits the region to be cut in subsequent scissoring.

In automatic scissoring, the most salient feature contour must be chosen automatically. We use two criteria to define the best contour: the length of the contour and its centricity because we want to divide a mesh into as large parts as possible. The *centricity* of a vertex v is defined as the average geodesic distance ($\mathrm{agd}(v)$) from v to all other vertices (Hilaga et al., 2001). Let m represent the maximum average geodesic distance among all mesh vertices, $m = \max_v(\mathrm{agd}(v))$. We define the normalized centricity as $c(v) = 1 - \mathrm{agd}(v)/m$. For each candidate feature contour γ, which is a sequence of edges, we define its priority as the sum of products of all normalized centricity of its edges by their lengths. More specifically, if $l(e)$ is the length of an edge e and $c(e)$ is the normalized centricity of e (the average value of its two endpoints), we define $P(\gamma) = \sum_{e\in\gamma} l(e)\cdot c(e)$, and choose $\gamma_{\max} = \mathrm{argmax}_\gamma\, P(\gamma)$. Fig. 6 shows scissoring results by automatic selection. We can observe that the longest contour nearest to the center is selected first.

After a mesh has been partitioned, the current centricity values of vertices are no longer valid in the resulting segments. Hence, we should recompute the centricity values after each partitioning when we perform recursive partitioning on a mesh for multiple-parts segmentation. To accelerate the computation, we approximate centricity values using a subset of vertices.

We first apply hierarchical face clustering (Garland et al., 2001) to a given mesh and obtain a simple graph G from the clustering result, as shown in Fig. 7. In graph G, vertices are the corner vertices in the clustering where more than two clusters meet and edges correspond to cluster boundaries. The edge lengths are determined as the 3D distances between the end vertices. We obtain the centricity values for the vertices of G by computing the shortest paths through the edges of G. From the centricity values in G, we can interpolate the centricity values of vertices in the given mesh. Let v be a vertex in the mesh. We compute the distances d_k from v to the corner vertices v_k of the cluster that contains v. The centricity value of v is determined by:

$$c(v) = \frac{\sum_k w_k \cdot c(v_k)}{\sum_k w_k},$$

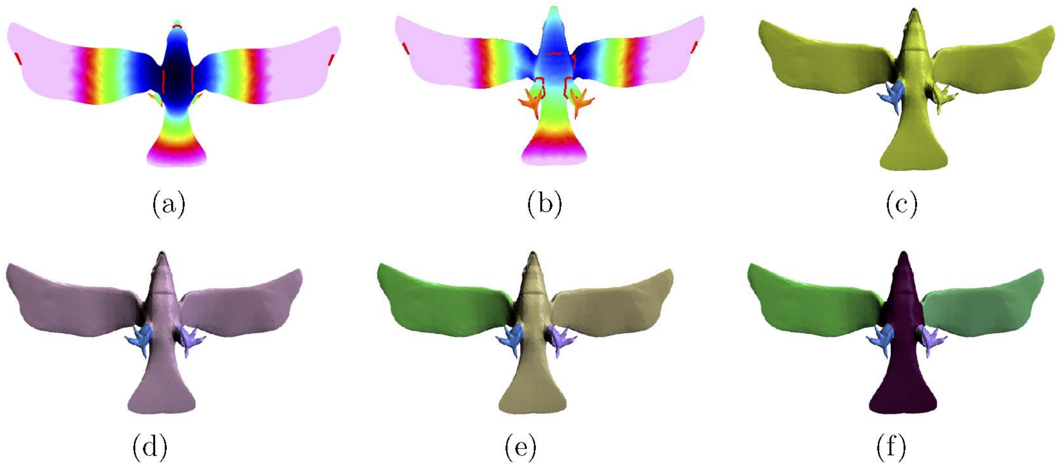


Fig. 6. Scissoring results with automatic selection: (a), (b) the distribution of centricity and extracted feature contours; (c)–(f) sequential scissoring results.

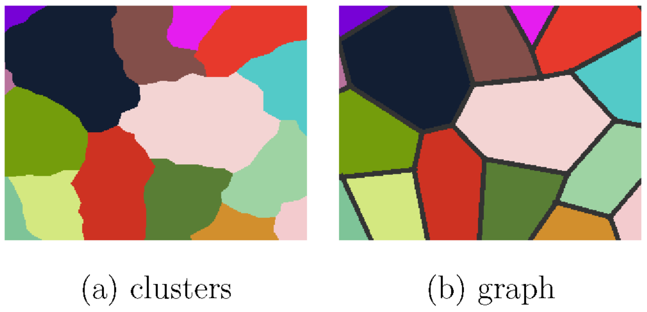


Fig. 7. Graph obtained from clustering.

where $w_k = 1/d_k$. When a scissoring operation is performed, graph G is partitioned into two subgraphs G_1 and G_2. We re-calculate the centricity values for the vertices in G_1 and G_2 and interpolate the values to determine the updated centricity values of vertices in the segments resulting from the scissoring.

In our implementation, the area of a cluster is constrained not to exceed δA, where A is the total area of the mesh surface. In the experiments, we used $\delta = 0.01$, which generated about 300 vertices for the graph G. For such a graph G, we can compute the centricity values of vertices very fast. Since clusters are nearly flat, the edge lengths in G determined by 3D distances between vertices give good approximations of the exact distances through the mesh surface. Although the centricity for G is computed with the shortest paths through edges, not the true geodesics on the mesh surfaces, the clusters have relatively small sizes and we can obtain good approximation for the centricity. In Fig. 8, we compare the centricity derived by the method in (Hilaga et al., 2001) with our approximation. Figs. 8(b) and 8(c) show that our method gives a good approximation for the whole mesh before scissoring. In Figs. 8(d) and 8(e), the approximation of updated centricity has artifacts around cluster boundaries but is good enough to automatically select the best contour.

In our previous work (Lee et al., 2004), we also used a subset of vertices to accelerate the centricity computation when a scissoring operation is performed. However, in that case, we store the geodesic

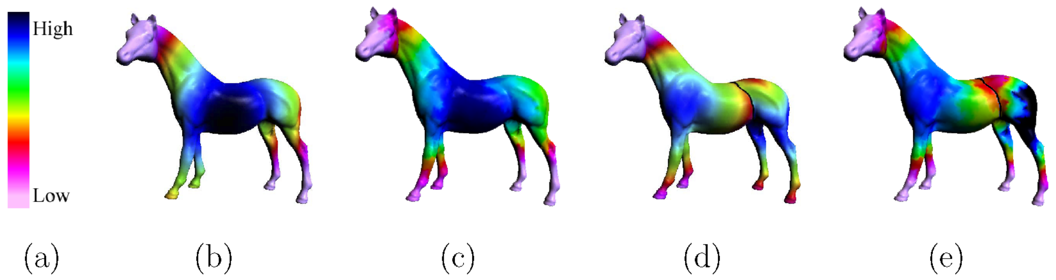


Fig. 8. Centricity update: (a) color table for centricity values; (b) centricity computed by the method in (Hilaga et al., 2001); (c) our approximation of (b); (d) updated centricity after scissoring, computed by the method in (Hilaga et al., 2001); (e) our approximation of (d). (For interpretation of the references in color in this figure legend, the reader is referred to the web version of this article.)

distances between sample vertices and reuse the distances to compute the updated centricity of all vertices. Although the stored distances reduce the computational overhead, the approach still calculate the geodesic distances between all pairs of vertices to obtain the centricity values. In addition, there were some artifacts near the sampled vertices because the geodesic distances for a vertex are obtained from the nearest sampled vertex. On the other hand, our new method does not have a memory overhead to store the geodesic distances between sample vertices. The computation is much faster because the geodesic distances are calculated only among the vertices of graph G. The interpolation from the vertices of G generates a smooth distribution of the approximate centricity values.

In manual scissoring, the user can choose a specific contour using the map of candidate contours, such as shown in Fig. 2(d). When the user wants to cut specific parts of a mesh sequentially, for examples, legs or foots of a horse, manual selection would be useful.

6. Feature contour completion

Let γ be a selected contour. Most of the time γ is not a closed contour, and there is a need to complete it to form a closed loop around the mesh. This can be done by designating a set of vertices around the mesh which form a loop from one endpoint of γ to the other. However this method is tedious and error-prone. In this section, we suggest methods to complete a contour to a closed loop automatically or semi-automatically.

In order to perform fully automatic scissoring or when the 2D line drawn on the screen is projected just over the front-faces in semi-automatic scissoring, we need an automatic contour closing method which completes γ to a closed loop around the mesh. The basic problem of completing an open contour to a closed loop is that the path from one endpoint to the other must be directed to go over the other side of the mesh instead of the natural shortest path, and to follow the mesh features as much as possible. To obtain this, we use a combination of four functions that guide the search towards the other side of the mesh and through mesh features.

Distance function. In order to make a loop to pass through the other side of the mesh, we first define distance function $\eta_d(v)$ which measures the distances from γ to other vertices. The function $\eta_d(v)$ is

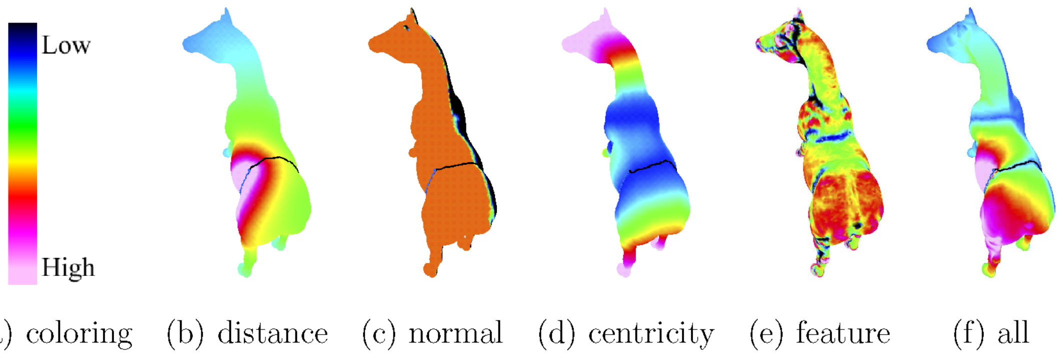


Fig. 9. Visualization of the four functions used to complete an open contour to a loop: (a) color table for values of cost functions; (b)–(d) the path is completed using each function; (e) $\eta_f(v)$ is the feature energy; (f) the minimum path is found with combination of all functions. (For interpretation of the references in color in this figure legend, the reader is referred to the web version of this article.)

designed to be high in the vicinity of the contour γ, and drops as we get farther away (see Fig. 9(b)). Let v be a mesh vertex. We define

$$\eta_d(v) = \sum_{v_i \in \gamma} \frac{1}{d(v, v_i)}.$$

$\eta_d(v)$ is the sum of inverse distances from vertex v to all vertices on γ.

Normal function. The second function $\eta_n(v)$ is lower for normals that face opposite directions of γ (see Fig. 9(c)), which also help search the path toward the other side of the mesh. Let n_γ be the center vector of the normal cone of all vertex normals of $v_i \in \gamma$ and α the angle of this cone. Let n_v be the normal vector of a vertex v. We define

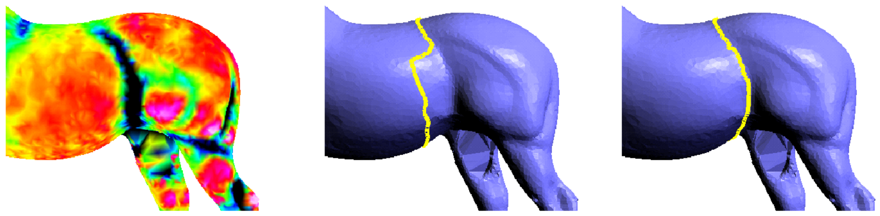
$$\eta_n(v) = \begin{cases} 1 & \text{if } n_\gamma \cdot n_v \geqslant \cos(\alpha), \\ \frac{n_\gamma \cdot n_v + 1}{\cos(\alpha)+1} & \text{otherwise.} \end{cases}$$

In this equation, $\eta_n(v)$ has the highest value ($= 1$) if the normal n_v belongs to the normal cone. Otherwise, $\eta_n(v)$ has a lower value as n_v is apart from the normal cone.

Centricity function. The third function $\eta_c(v)$ guides the loop to be perpendicular to the medial axis of a mesh shape. The user often intends to cut a mesh along boundaries which is perpendicular to the medial axis. Fig. 13 shows an example. To achieve this goal, we use the centricity over a mesh instead of the medial axis because it is difficult to find a medial axis in a robust and stable way. In Fig. 4 of (Hilaga et al., 2001), contours which have the same centricity seem to cut a mesh perpendicularly to the axis. We define

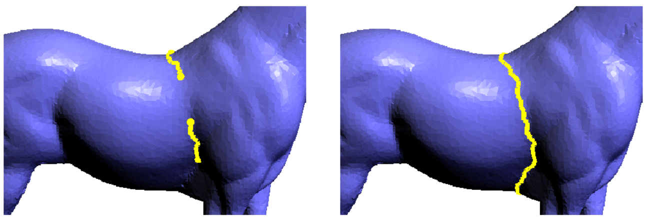
$$\eta_c(v) = \sum_{v_i \in \gamma} \left| c(v_i) - c(v) \right|,$$

where $c(v)$ is the normalized centricity of a vertex. The function $\eta_c(v)$ is higher for vertices that have different centricity values from the vertices on γ (see Fig. 9(d)).



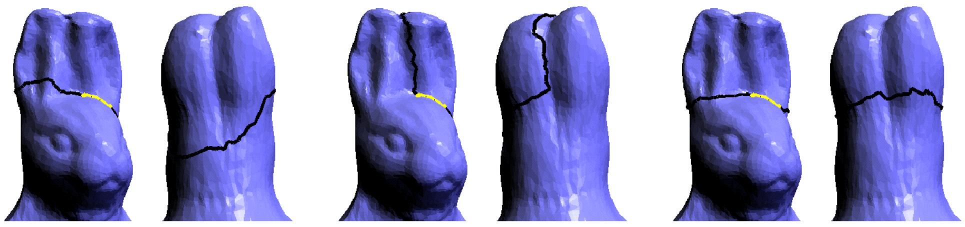
(a) feature function η_f (b) without η_f (c) with η_f

Fig. 10. Effect of feature function.



(a) two contours (b) completed contour

Fig. 11. Feature connection in the contour completion.



(a) $w_d = 2$ (b) $w_n = 10$ (c) $w_d = 1$, $w_n = 1$

Fig. 12. Combination of distance and normal functions.

As described in Section 5, we update the centricity values of vertices after a scissoring is performed. However, to compute the function $\eta_c(v)$, we use the original centricity value computed for the given mesh before any scissoring operation is performed. The user would be more interested in the cuts perpendicular to the medial axis of the whole mesh than specific parts resulted from scissoring.

Feature function. The fourth function $\eta_f(v)$ guiding the path towards mesh features is the same one used to define the feature contours (see Fig. 9(e)), but it is normalized between 0 and 1. With the feature function $\eta_f(v)$, the loop passes around features as shown in Fig. 10. In addition, the function enables two feature contours far from each other to be connected in the completion stage (see Fig. 11).

For an edge e, $l(e)$ denotes the length of e. The cost functions, $\eta_d(e)$, $\eta_n(e)$, $\eta_f(e)$, and $\eta_c(e)$, are defined as the averages of the values at the end vertices of edge e. To find the path from one end vertex to the other of γ, we search for the shortest path using the edge cost

$$f(e) = l(e) \cdot \eta_d(e)^{w_d} \cdot \eta_n(e)^{w_n} \cdot \eta_f(e)^{w_f} \cdot \eta_c(e)^{w_c},$$

where cost functions $\eta_i(e)$ are normalized between 0 and 1 and w_i are used to control the strengths of these functions. We usually set w_d, w_n, and w_f to 1.0 and w_c to 0.4, which are values determined by experiments.

In Figs. 9(b)–(d), we can see the contour may be successfully completed to pass through the other side of the mesh by each cost function. However, despite the costs to compute all four functions, it is advantageous to use them all instead of a single one. As shown in Fig. 12, when we use only one of the cost functions, it is difficult to find proper parameters to make a loop go over the other side of the mesh. In Figs. 12(a) and 12(b), several trials were needed to find proper parameter values for the single cost function. Also, large values of parameters may cause unexpected paths in the contour completion. On the other hand, by using the combination of the cost functions, we can obtain a nice path with the default parameter values in Fig. 12(c).

The centricity function $\eta_c(e)$ strongly affects the path of a completed contour. If the boundaries which a mesh is cut along are expected to be perpendicular to the medial axis of the mesh, the function improves the completion results considerably (see Fig. 13). Otherwise, it may cause a path to be too dependent on the distribution of the centricity. Depending on the input meshes, the user can choose that $\eta_c(e)$ will not be applied simply by setting w_c to zero.

A loop around the mesh can also be defined using a 2D line drawn over the mesh. We define a cutting plane which passes endpoints of γ and whose orientation is similar to the plane defined by the 2D line and the view direction. The normal vector N of the cutting plane should be perpendicular to both the line direction L and the view direction V as much as possible. Therefore, we minimize $L \cdot N + V \cdot N$ and use a Lagrange multiplier to constrain a cutting plane to pass the two endpoints; that is, we minimize

$$f(N) = L \cdot N + V \cdot N + \lambda\big(N \cdot (P_1 - P_0)\big),$$

where λ is a Lagrange multiplier and P_0 and P_1 are the positions of two endpoints of γ.

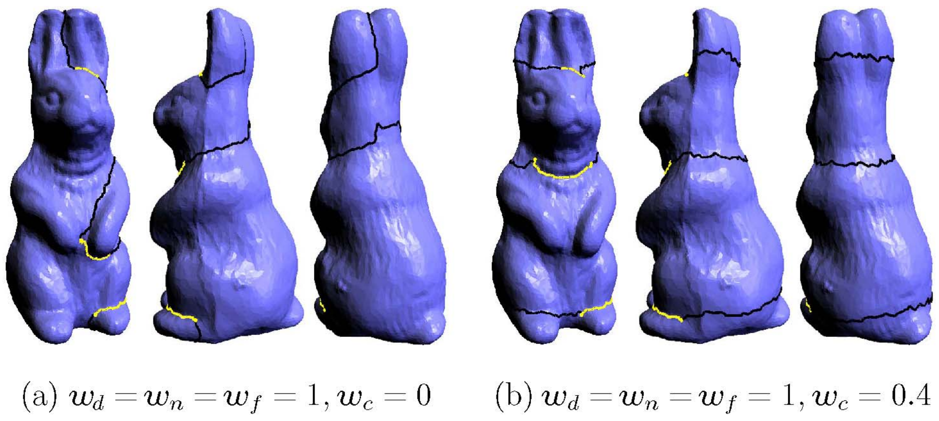

(a) $w_d = w_n = w_f = 1, w_c = 0$
(b) $w_d = w_n = w_f = 1, w_c = 0.4$

Fig. 13. Effect of centricity function.

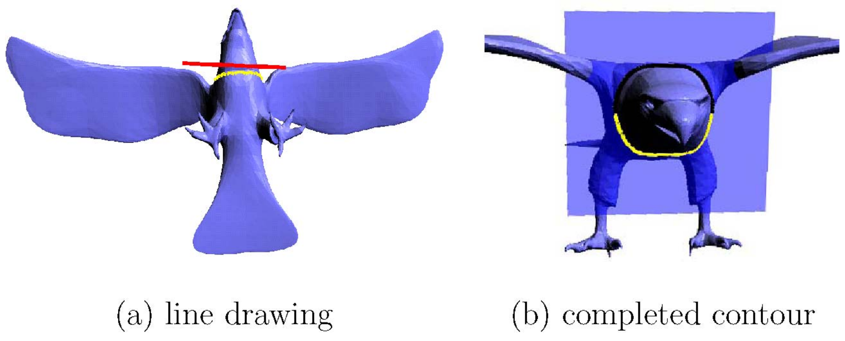


Fig. 14. Semi-automatic contour completion.

Fig. 14 shows an example of manual guidance with 2D line drawing. Although a red line drawn by a user does not pass the endpoints, the cutting plane not only passes the endpoints but its normal also seems perpendicular to the view direction and the line direction as much as possible (for interpretation of the references in color, the reader is referred to the web version of this article).

7. Part salience test

In the mesh scissoring process, a completed feature contour almost determines the shape of a scissoring cut because the final position of the snake is not much different from the initial one. However, automatic selection and completion can sometimes result in less meaningful contours, which produces no salient segments. In Fig. 15, two open contours have similar lengths and centricity, which means either of them can be chosen first. In order to select the contour in Fig. 15(b) first, instead of the one in Fig. 15(a), we need a criterion to check whether segments obtained from the completed contour are significant enough or not.

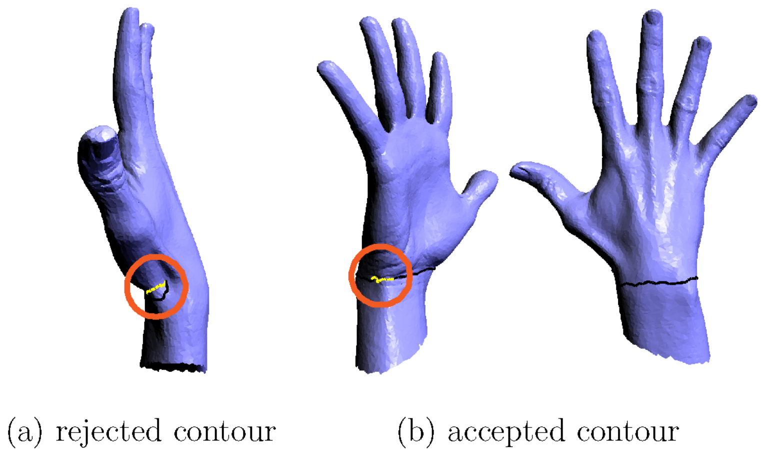


Fig. 15. Completion of two selected contours with similar lengths and centricity: (a) the area of the segment is too small to partition; (b) The area of the segment is large enough to partition.

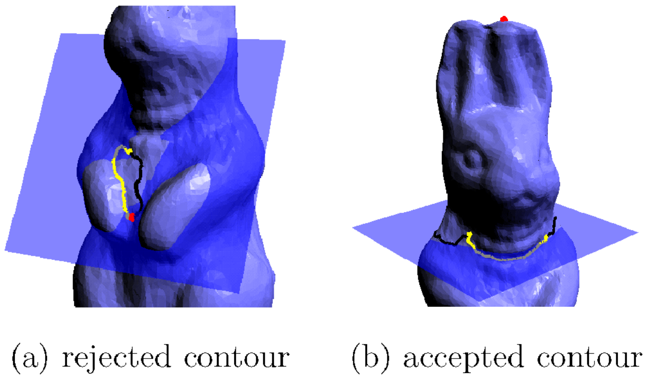


Fig. 16. Part salience test by protrusion: (a) S is too flat to partition; (b) S is protrudent enough to partition.

Hoffman and Singh (1997) proposed three factors that determine the salience of a part: the relative size, the boundary strength, and the degree of protrusion. We incorporate the test based on those three factors into our scissoring framework to automatically reject a closed contour that would not generate salient parts.

Area. We measure the area factor by the ratio of the area Σ of a segment before scissoring to the areas Σ_0 and Σ_1 of the two segments after scissoring. The area of a segment is computed by summing the areas of triangles in the segment. Let σ be the smaller value among Σ_0 and Σ_1. If $\sigma/\Sigma < \xi_{\text{area}}$, we reject the contour and select a new one in the contour candidates. For example, the contour in Fig. 15(a) can be rejected by this area factor. We usually set ξ_{area} to 0.05.

Protrusion. The protrusion factor is measured using a fitting plane obtained from the sample points on a contour. Let S be a segment before scissoring and let S_0 be the segment with the smaller area after scissoring. We compute the longest distance d_p from the vertices on S_0 to the fitting plane. Let r be the radius of the bounding sphere of segment S. If $d_p/r < \xi_{\text{prot}}$, we reject the contour. In Fig. 16, blue planes are the fitting planes of contours and red points indicate the farthest points from the planes (for colors see the web version of this article). The contour in Fig. 16(a) passes the area test but it is rejected by the protrusion test. On the other hand, the contour in Fig. 16(b) is accepted by both the area and protrusion tests. In our experiments, ξ_{prot} is usually set to 0.05.

Feature. Completed contours usually have strong features since contours are extracted from feature regions and contour completion considers featureness of edges. Accordingly, the rejection by weak features hardly occurs. Instead, in the part salience test with features, we prevent high feature contours from being rejected by the protrusion test. In the case of Fig. 17, the region enclosed by the contour is flat and the contour is rejected by the protrusion test. However, the region could be a good candidate for segmentation because it is surrounded by strong features. We call an edge on a contour a *feature edge* if the end vertices of the edge have passed the thresholding in the feature extraction step. If $l_f/l_c > \xi_{\text{feat}}$, where l_f is the sum of feature edge lengths and l_c is the length of the contour, we accept the contour regardless of the protrusion factor. ξ_{feat} is usually set to 0.8.

In addition to automatic part salience test, the user can select another contour or remove the current contour manually. Fig. 18(a) shows an example of a contour that can pass the part salience test but may

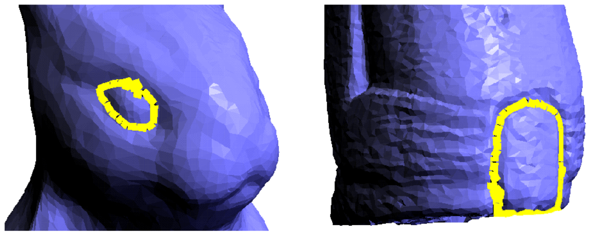


Fig. 17. Flat region enclosed by features.

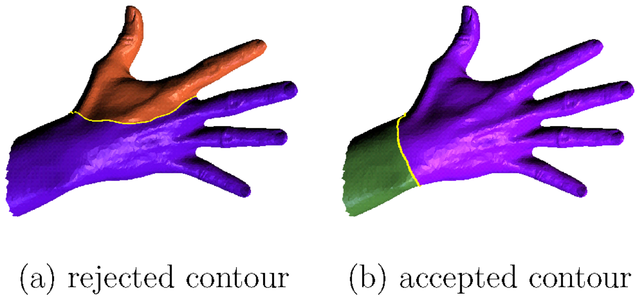


Fig. 18. Manual rejection: A user can cut a wrist by rejecting a contour in (a) and selecting a contour in (b).

not be desirable by the user. If the user wants to cut a wrist, as in Fig. 18(b), or fingers of a hand, the current contour in Fig. 18(a) can be manually rejected.

8. Snake movement

Once a feature contour is selected and closed to form a loop, a geometric snake is initialized using a sequence of sample points $s(i)$ which lay on the loop. The snake moves by minimizing an energy functional E_{snake} composed of internal and external parts;

$$E_{\text{snake}}(s) = \int \bigl(E_{\text{spline}}(s) + E_{\text{mesh}}(s)\bigr)\,\mathrm{d}t.$$

The snake external energy E_{mesh} is designed to capture nearby features and its internal energy E_{spline} to smooth its shape and shorten its length. The external energy is defined using the mesh curvature again. However, as opposed to the feature extraction stage, we now search for any nearby feature either concave or convex. Hence, we use the absolute values of the mean curvatures as the feature field.

To constrain a geometric snake on the mesh surface in the energy minimization process, as proposed in (Lee and Lee, 2002), we use a local parameterization that embeds faces around a snake onto a 2D plane. With this parameterization, we can use the same equation as an image snake for the energy minimization. The snake position is incrementally updated by repeatedly solving linear equations until a minimum is reached. See (Lee and Lee, 2002) for the details of snake movement.

The final position of a snake defines the scissoring position where a cut is made (see Fig. 19). Since snake sample points can lay at anywhere on the mesh surface, the snake contour may pass through

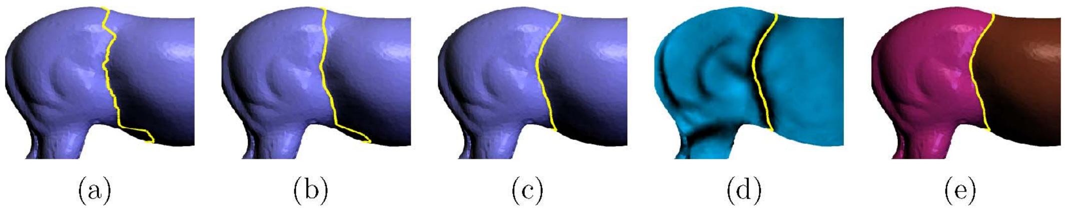


Fig. 19. Snake relaxation process: (a) initial positioning on a mesh based on a completed feature contour; (b) snake position after one iteration; (c) final snake position; (d) final position with feature energy; (e) scissoring with the snake.

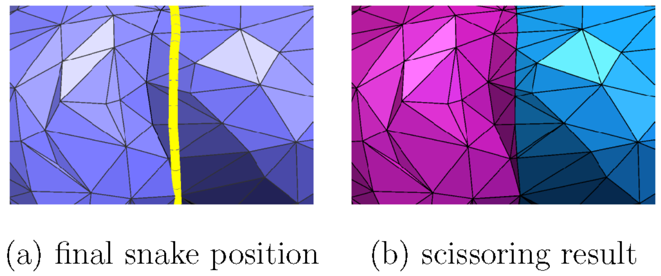


Fig. 20. (a) The final snake position does not necessarily follow mesh edges, creating a smoother region boundary. (b) Scissoring divides the mesh faces following the final snake position.

triangles of a mesh. These triangles are subdivided along the snake to achieve smoother interface between the mesh parts (see Fig. 20).

9. Experimental results

All following experiments were carried out on a 3.2 GHz Intel(R) Pentium(R) M processor with 1 Gb memory. Fig. 21 shows automatic partitioning results. These results were created by automatic extraction and completion of feature contours. Figs. 21(a)–(c) are fully automatic results, where the whole scissoring process was performed without any user intervention. For the results in Figs. 21(d)–(g), some contours that passed the part salience test were rejected by the user to obtain better results. However, since the part salience test can filter most contours with non-salient segments, the manual rejection is needed very rarely, only when there is real ambiguity such as shown in Fig. 18(a). For Figs. 21(b) and 21(e), the centricity function was applied to complete contours, while the function is not used for the other models in Fig. 21.

Fig. 22 shows results of semi-automatic scissoring of meshes. In more complex situations, the user can begin with automatic scissoring and then reject specific cuts and continue scissoring further in designated places (compare Fig. 21(c) and Fig. 22(a)). Figs. 22(c) and 22(d) show that our scissoring tool can segment a mesh whose genus is not zero. By scissoring the mesh along yellow and green cuts in sequence, we can separate the tail from the body and leg (for colors see the web version of this article).

Fig. 23 shows the scissoring results of models which have sharp concave creases, such as CAD models. For these models, it is better to constrain the boundaries of segments along mesh edges and our system provides such an option.

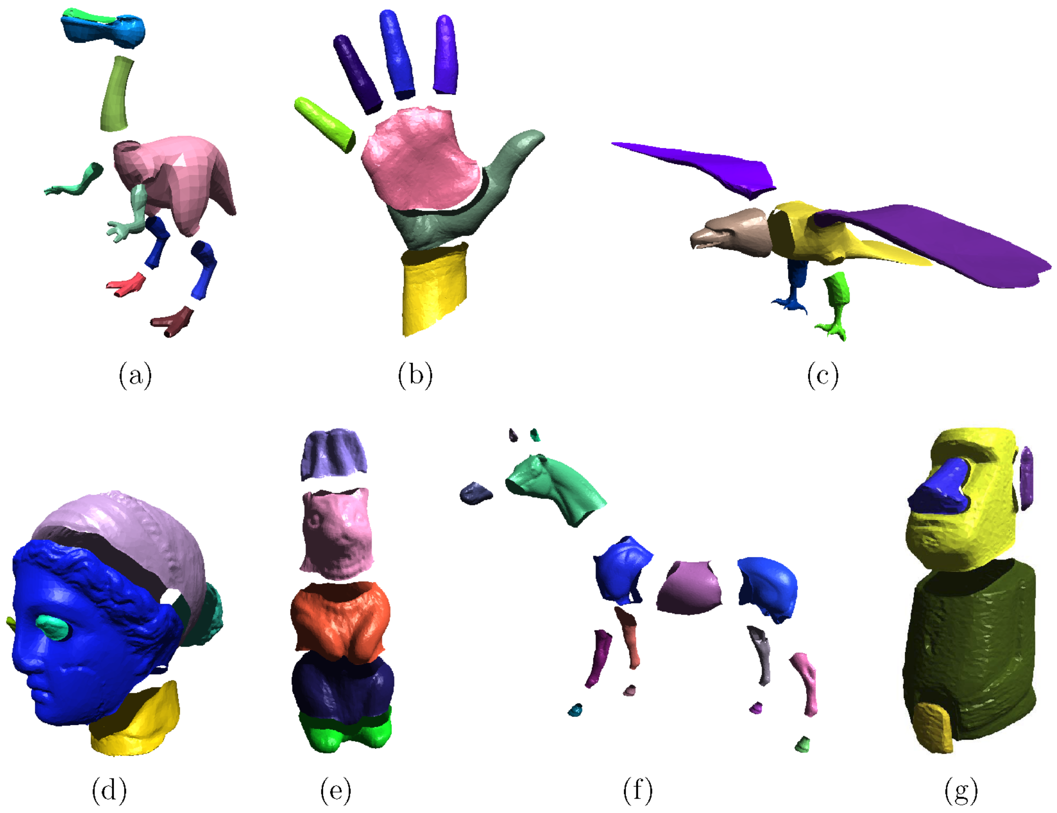


Fig. 21. Mesh scissoring results using an automatic approach.

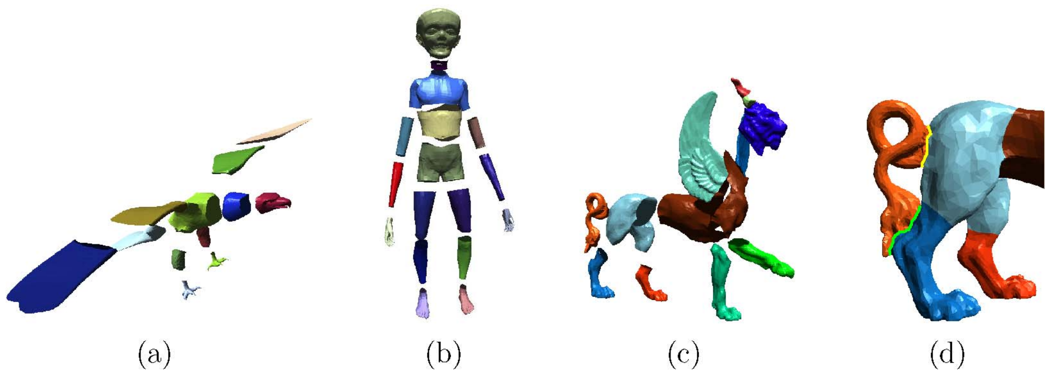


Fig. 22. Mesh scissoring results using a semi-automatic approach.

Fig. 24 shows manual scissoring with the 2D slicing approach. Fig. 25 is the final scissoring result. Contours created by drawing 2D lines can disregard the feature contours altogether. However, intelligent scissoring with automatic contour completion and a geometric snake assures the cut would be smooth and follow natural shape features as much as possible as shown in Fig. 24.

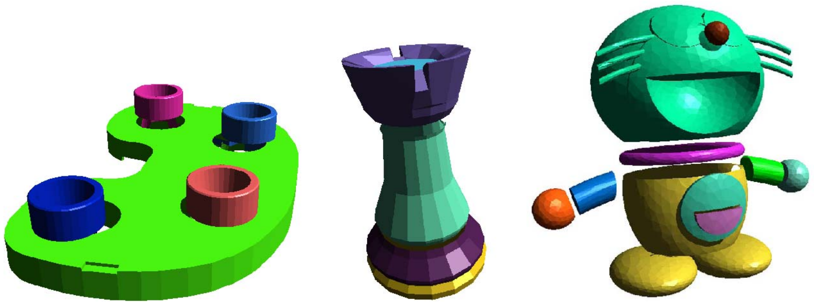


Fig. 23. Segmentation of models with concave creases.

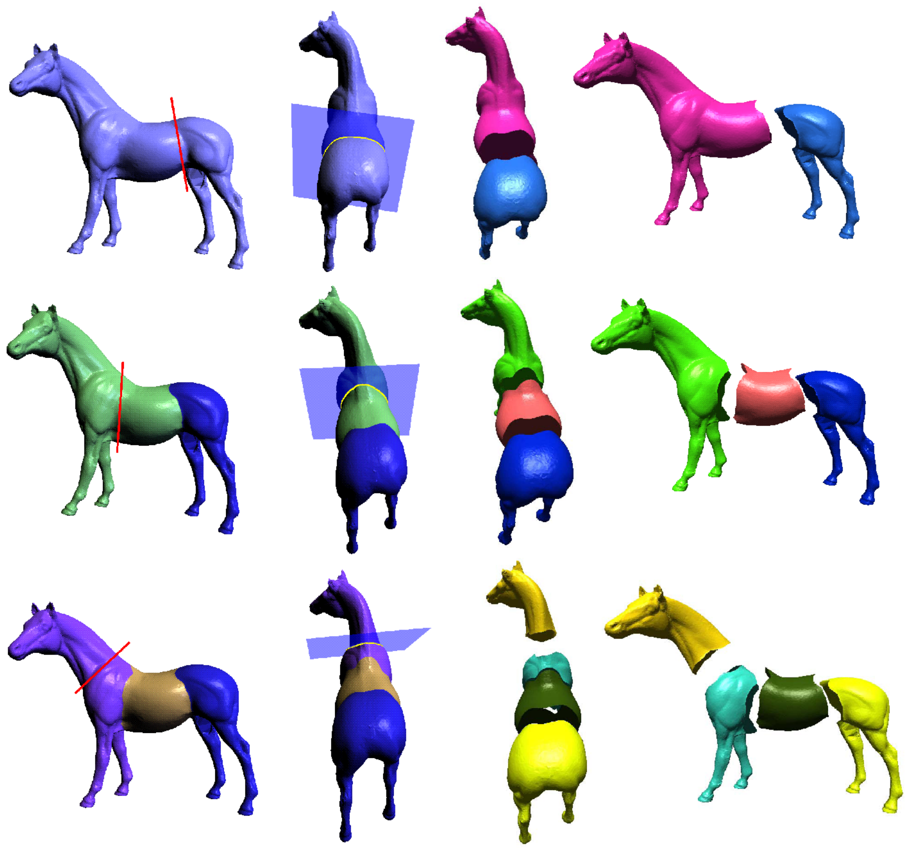


Fig. 24. Manual mesh scissoring examples: a 2D line is drawn on the mesh (left) and projected to create a slicing plane. This plane can be used to define the initial loop (middle left). Alternatively, the line projected on the front faces can be completed using our automatic completion method. Note that the initial slicing position is not a rigid constraint on the scissoring, hence the final cut can follow natural shape features (middle right and right).

Table 1 shows timing statistics on different meshes used in our experiments. After the pre-processing for feature extraction and geodesic distance computation, the method runs at an interactive speed even on large meshes. Hence, using this approach, an interactive tool is provided where semi-automatic as

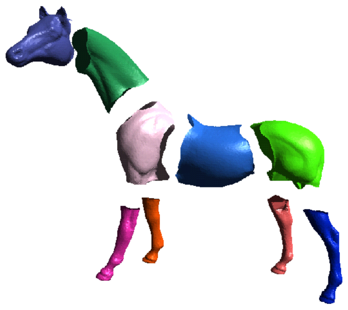


Fig. 25. The final result of manual scissoring in Fig. 24.

Table 1
Typical timing statistics (in seconds) of different steps of our mesh scissoring algorithm

Mesh	# of vertices	Curvature extraction	Clustering	Centricity calculation	Contour extraction	Closing & salience test	Snake movement
CAD	20,570	9 ∼ 10	3 ∼ 4	0.3 ∼ 0.4	3 ∼ 4	4 ∼ 5	1 ∼ 2
Horse	19,851	6 ∼ 7	3 ∼ 4	0.3 ∼ 0.4	1.5 ∼ 2.5	1 ∼ 2	2 ∼ 7
Igea	18,004	4 ∼ 5	3 ∼ 4	0.3 ∼ 0.4	2 ∼ 3	1 ∼ 2	2 ∼ 3
Eagle	14,618	37 ∼ 38	2 ∼ 3	0.3 ∼ 0.4	1.5 ∼ 2.5	0.5 ∼ 0.8	2 ∼ 3
Hand	10,070	2 ∼ 3	1 ∼ 2	0.2 ∼ 0.3	1.0	0.4 ∼ 0.7	1 ∼ 2
Alien	7,401	3 ∼ 4	1 ∼ 2	0.15	1.0	0.3 ∼ 0.5	1 ∼ 2
Doll	3,756	0.4 ∼ 0.5	1.5	0.1	0.70	0.3 ∼ 0.5	0.5 ∼ 1.0
Dino	3,323	0.9 ∼ 1	0.5	0.1	1.0	0.20	1 ∼ 2
Chess	1,514	0.3 ∼ 0.4	0.03	0.04	0.70	0.20	1 ∼ 2

well as automatic scissoring can be applied easily. The computation time for updating centricity is similar to or less than centricity calculation for the whole mesh. Since the number of vertices in feline, rabbit, and moai models are around 1K, the computation time is almost the same as the hand model.

10. Summary and discussion

In this paper, we have presented an approach for 3D mesh scissoring which can be used for both automatic partitioning and manual cutting. Based on the minima rule, our scissoring approach targets the cutting positions determined by feature extraction. The part salience theory makes automatic scissoring more robust by automatically rejecting less meaningful segmentation. User intervention is rarely necessary, only when there is real ambiguity that automatic thresholding cannot figure out.

In our scissoring operator, most of the timing constraint for large meshes come from feature extraction and centricity calculation. Feature extraction is needed only once for each mesh and can be performed as preprocessing before the scissoring process starts. The centricity should be computed for the given mesh and updated for the segments resulting from scissoring. To accelerate the centricity computation, we presented an approximation technique using a subset of vertices. For a very large mesh, however, feature extraction and centricity calculation may still incur much computational overhead. To reduce the overhead, we can use a simplified mesh in our scissoring process, in a similar way to the approach proposed in (Katz and Tal, 2003). That is, we can perform most steps of scissoring, including feature extraction and centricity calculation, on a simplified mesh and map the resulting cuts onto the original mesh. Only the geometric snake step is performed on the original mesh to determine the final smooth cuts, using the cuts from a simplified mesh as the initial positions.

Acknowledgements

The bunny, horse and Igea, and feline models are courtesy of the Stanford Computer Graphics Laboratory, Cyberware, and the Caltech Multi-Res Modeling Group, respectively. This work was supported in part by the Korean Ministry of Education through the BK21 program, the Korean Ministry of Information and Communication through the ITRC support program, and the Israeli Ministry of Science and Education Grant Tashtiot Grant No. 01-01-01509.

References

Alliez, P., Cohen-Steiner, D., Devillers, O., Lévy, B., Desbrun, M., 2003. Anisotropic polygonal remeshing. In: Proc. of SIGGRAPH 2003, ACM Comput. Graph., 485–493.

Andrews, S., 2000. Interactive generation of feature curves on surfaces: A minimal path approach. Master's thesis, University of Toronto.

Cohen, L.D., Kimmel, R., 1997. Global minimum for active contour models: A minimal path approach. Internat. J. Comput. Vis. 24 (1), 57–78.

DeRose, T., Kass, M., Truong, T., 1998. Subdivision surfaces in character animation. In: Proc. of SIGGRAPH '98, ACM Comput. Graph., 85–94.

Falcão, A.X., Udupa, J.K., Samarasekera, S., Sharma, S., Hirsch, B.E., de A. Lotufo, R., 1998. User-steered image segmentation paradigms: Live wire and live lane. Graph. Models Image Process. 60 (4), 233–260.

Faugeras, O.D., Hébert, M., 1987. The representation, recognition, and positioning of 3-D shapes from range data. In: Kanade, T. (Ed.), Three-Dimensional Machine Vision. Kluwer Academic, Dordrecht, pp. 301–353.

Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D., 2004. Modeling by example. In: Proc. of SIGGRAPH 2004, ACM Comput. Graph. 23, 652–663.

Garland, M., Willmott, A., Heckbert, P.S., 2001. Hierarchical face clustering on polygonal surfaces. In: Proc. 2001 ACM Symposium on Interactive 3D Graphics, pp. 49–58.

Hilaga, M., Shinagawa, Y., Kohmura, T., Kunii, T.L., 2001. Topology matching for fully automatic similarity estimation of 3D shapes. In: Proc. of SIGGRAPH 2001, ACM Comput. Graph., 203–212.

Hoffman, D., Richards, W., 1984. Parts of recognition. Cognition 18, 65–96.

Hoffman, D., Signh, M., 1997. Salience of visual parts. Cognition 63, 29–78.

Hubeli, A., Gross, M., 2001. Multiresolution feature extraction for unstructured meshes. In: Proc. IEEE Visualization 2001, pp. 287–294.

Inoue, K., Itoh, T., Yamada, A., Furuhata, T., Shimada, K., 2001. Face clustering of a large-scale CAD model for surface mesh generation. In: 8th International Meshing Roundtable Special Issue: Advances in Mesh Generation. Computer-Aided Design 33 (3), 251–261.

Kalvin, A.D., Taylor, R.H., 1996. Superfaces: Polygonal mesh simplification with bounded error. IEEE Comput. Graph. Appl. 16 (3), 64–77.

Kass, M., Witkin, A., Terzopoulos, D., 1988. Snakes: Active contour models. Internat. J. Comput. Vis. 1 (4), 321–331.

Katz, S., Tal, A., 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. In: Proc. of SIGGRAPH 2003, ACM Comput. Graph. 22 (3), 954–961.

Lee, Y., Lee, S., 2002. Geometric snakes for triangular meshes. In: Proc. Eurographics 2002, Comput. Graph. Forum 21 (3), 229–238.

Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., Seidel, H.-P., 2004. Intelligent mesh scissoring using 3d snakes. In: Proc. Pacific Graphics 2004, pp. 279–287.

Lévy, B., Petitjean, S., Ray, N., Maillot, J., 2002. Least squares conformal maps for automatic texture atlas generation. In: Proc. of SIGGRAPH, ACM Comput. Graph., 362–371.

Li, X., Toon, T., Tan, T., Huang, Z., 2001. Decomposing polygon meshes for interactive applications. In: Proc. 2001 ACM Symposium on Interactive 3D Graphics, pp. 35–42.

Mangan, A.P., Whitaker, R.T., 1999. Partitioning 3D surface meshes using watershed segmentation. IEEE Trans. Vis. Comput. Graph. 5 (4), 308–321.

Milroy, M.J., Bradley, C., Vickers, G.W., 1997. Segmentation of a wrap-around model using an active contour. Computer-Aided Design 29 (4), 299–320.

Mortensen, E.N., Barrett, W.A., 1995. Intelligent scissors for image composition. In: Proc. of SIGGRAPH '95, ACM Comput. Graph., 191–198.

Mortensen, E.N., Barrett, W.A., 1998. Interactive segmentation with intelligent scissors. Graph. Models Image Process. 60 (5), 349–384.

Page, D.L., Koschan, A.F., Abidi, M.A., 2003a. Perception-based 3D triangle mesh segmentation using fast marching watersheds. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '03), vol. II, pp. 27–32.

Page, D.L., Abidi, M.A., Koschan, A.F., Zhang, Y., 2003b. Object representation using the minima rule and superquadrics for under vehicle inspection. In: 1st IEEE Latin American Conference on Robotics and Automation, pp. 91–97.

Shamir, A., 2004. A formulation of boundary mesh segmentation. In: 2nd International Symposium on 3D Data Processing, Visualization, and Transmission, pp. 82–89.

Shlafman, S., Tal, A., Katz, S., 2002. Metamorphosis of polyhedral surfaces using decomposition. In: Proc. Eurographics 2002, Comput. Graph. Forum 21 (3), 219–228.

Sorkine, O., Cohen-Or, D., Goldenthal, R., Lischinski, D., 2002. Bounded-distortion piecewise mesh parameterization. In: Proc. IEEE Visualization 2002, pp. 355–362.

Zuckerberger, E., Tal, A., Shlafman, S., 2002. Polyhedral surface decomposition with applications. Computer and Graphics 26 (5), 733–743.