Intelligent Mesh Scissoring Using 3D Snakes

Yunjin Lee
POSTECH & MPI Informatik

Seungyong Lee
POSTECH & MPI Informatik

Ariel Shamir
The Interdisciplinary Center

Daniel Cohen-Or
Tel Aviv University

Hans-Peter Seidel
MPI Informatik

Abstract

Mesh partitioning and parts extraction have become key ingredients for many mesh manipulation applications both manual and automatic. In this paper, we present an intelligent scissoring operator for meshes which supports both automatic segmentation and manual cutting. Instead of segmenting the mesh by clustering, our approach concentrates on finding and defining the contours for cutting. This approach is based on the minima rule*, which states that human perception usually divides a surface into parts along the contours of concave discontinuity of the tangent plane. The technique uses feature extraction to find such candidate feature contours. Subsequently, such a contour can be selected either automatically or manually, or the user may draw a 2D line to start the scissoring process. The given open contour is completed to form a loop around a specific part of the mesh, and this loop is used as the initial position of a 3D geometric snake. The snake moves by relaxation until it settles to define the final scissoring position. This process uses several fundamental geometric mesh attributes, such as curvature and centricity, and enables both automatic segmentation and an easy-to-use intelligent-scissoring operator.*

1. Introduction

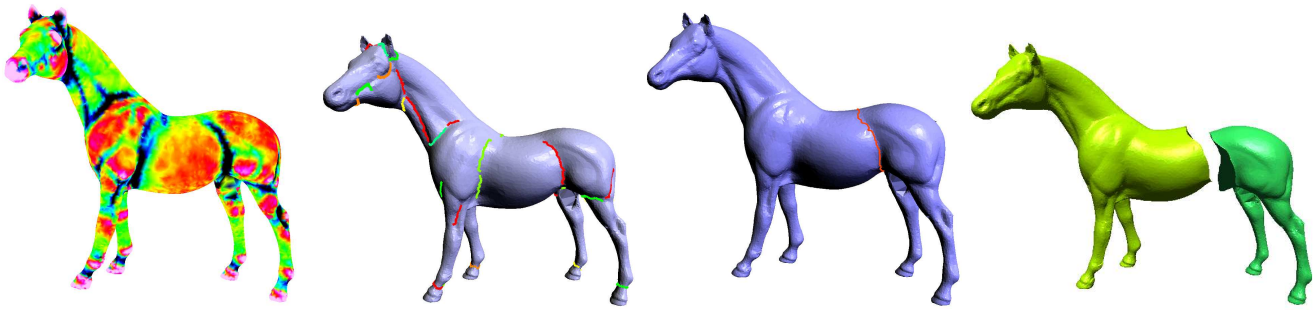
Three dimensional boundary meshes are used to define geometric objects in many applications in graphics. Meshes are created either using complicated modeling softwares or by scanning physical models. With the advent of advanced scanning technology, meshes are becoming larger and more complicated. In order 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. Typically a fully automatic partitioning algorithm is not as important as providing an intelligent tool to assist the user in manual scissoring. In this paper, we present a scissoring operator which supports both automatic and semi-automatic partitioning of meshes and allows easy and intuitive scissoring of meshes.

Scissoring is an operator that separates a given mesh into two disjoint pieces along a closed contour lying on the mesh. Our scissoring operator is carried out using three 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. **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 hints from the user or by an automatic procedure which guides the contour to loop around mesh parts. This approach provides the ability to define a range of possible tools beginning from a fully automatic partitioning tool to an easy-to-use semi-automatic scissoring tool.

Most previous work on mesh partitioning use clustering of similar mesh elements or components and then refine the borders between clusters to find the segmentation [6, 11, 16, 14]. Our scissoring approach is different in that it directly targets the cutting contours. This idea is based on the *minima rule* from human vision theory. The minima rule states that human perception usually divides an object into parts along the concave discontinuity of the tangent plane [8, 9]. In various tests, it has been shown that



(a) (b) (c) (d)

Figure 1. Overview of the scissoring process: (a) feature extraction - minimum curvatures of vertices on a mesh; (b) feature contour extraction and selection; (c) completing the selected contour to a loop and applying snake movement; (d) scissoring result.

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 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.

Our scissoring approach presents several nice properties:

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

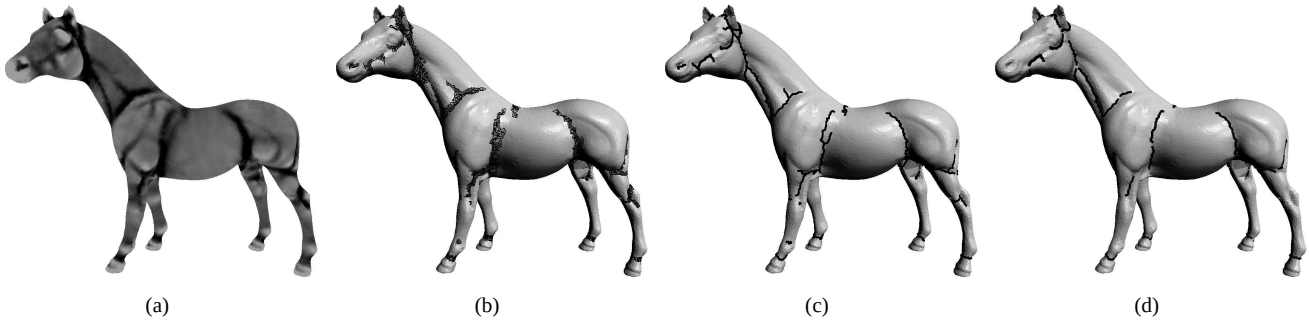
2. Related Work

Automatic partitioning of meshes is typically performed by growing regions incrementally [12, 18, 24, 25], or by merging regions [6, 23]. 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 interfaces which are not constrained to lay on the mesh edges.

The minima rule has been used for segmenting CAD models and meshes but with the watershed algorithm [22]. Due to the limitation of the watershed algorithm, the technique cannot cut a part if the part boundary contains non-negative minimum curvatures. The approach presented in [14] 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 also does not support intelligent manual operations and a cut is constrained to mesh edges. In [17], 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.

Intelligent manual scissoring tools have been presented for image segmentation in [20, 21, 4]. 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 [5], where the user extracts parts of a mesh in order to combine and paste them to other meshes.

Snakes were presented as active contour models for semi-automatic detection of features in an image [13, 26]. Active contour models for images were extended to extract features from 3D surfaces. In [19], the snake position is updated directly on a 3D surface. Feature curve detection on a 3D mesh by a minimal path [3] between the source and destination points is presented in [2]. We follow the approach proposed in [15] where the snake's updated position is determined by energy minimization on a 2D embedding plane,



(a) (b) (c) (d)

Figure 2. Stages of feature contour extraction: (a) minimum curvature feature field defined over a mesh; (b) regions of high feature energy; (c) feature graphs created by thinning the regions; (d) feature contours extracted from the graphs.

which is computationally efficient. Using a snake 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 usage of our approach. The following sections will provide details on each of the different stages.

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 the *feature values* on the vertices. We use thresholding and thinning to obtain several *graph structures* of feature skeletons and then extract single contours. For automatic scissoring, we choose the best contour based on two criteria: the length of the contour and its centricity on the mesh. The selected contour serves as the initial *feature contour* in the succeeding scissoring stages. 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. Nevertheless, the user is not restricted to select such feature contours. A simple gesture of drawing a 2D line segment on the viewing window can be used to guide the scissoring. This line along with the center of projection defines a plane which cuts through the mesh.

In the second stage, the feature contour is completed to form a closed loop around a specific part of the mesh. When 2D line drawing is used, we can use the cutting plane to create a full loop around the mesh. For delicate situations, the user can designate points on the mesh which complete the loop. However, when no user guidance is given, this can also be done automatically. 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.

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 settles with a minimum energy, 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 object parts. Following this rule, we first calculate the curvature values on the vertices of the mesh using the tensor field computation used in [1]. Next, we assign the normalized minimum curvature value to each vertex of the mesh as a feature value (see Fig. 1(a)). If $\kappa(v)$ is the minimum curvature value at a vertex v, we assign $c_f(v) = (\kappa(v) - \mu)/\sigma$, where μ is the mean and σ is standard deviation of $\kappa(v)$ over all vertices of the mesh. We use this approach 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 specific meshes in the succeeding thresholding stage.

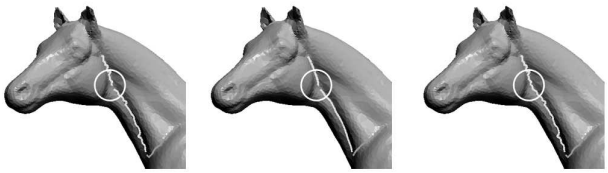


Figure 3. Connecting small contours (left) having similar major directions after smoothing (middle) to longer ones (right).

Similar to [10], we adopt hysteresis thresholding on the values of c_f to define high feature areas. We usually use -1.2 for the upper bound of the hysteresis and -0.8 for the lower bound. This means we choose high negative-curvature values which designate concave parts. Note that in the original paper [10], 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 (see Fig. 2(a)). By connecting such vertices, 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 (see Fig. 2(c)).

As a result of these steps, we obtain a set of graph structures of feature skeletons. 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. In order to merge close and similar feature contours and create longer ones, we first smooth the contours and obtain their major directions. We then connect two contours which have close endpoints and similar directions. See Fig. 3 and Fig. 2(d) for the process and the result, respectively.

5. Feature Contour Selection

In order to perform the scissoring, a single specific feature contour must be chosen to start the process. Using the map of candidate contours as shown in Fig. 2(d), the user can choose a specific contour to perform manual scissoring. Another option for the user is simply to draw a 2D guiding line on the screen over the mesh. This line is then projected over the front-faces of the mesh to define the feature contour.

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. The *centricity* of a vertex v is defined as the average geodesic distance ($\text{agd}(v)$) from v to all other vertices [7]. Let m represent the maximum average geodesic distance among all mesh vertices, $m = \max_v(\text{agd}(v))$. We define the normalized centricity as $c(v) = 1 - \text{agd}(v)/m$. 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 of its two endpoints), we define $P(\gamma) = \sum_{e\in\gamma} l(e) \cdot c(e)$, and choose $\gamma_{\max} = \text{argmax}_\gamma P(\gamma)$.

After a mesh has been partitioned, the current centricity values of vertices are no longer valid in the resulting distinct parts. 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 the true geodesic distances with a subset of vertices.

Let $\mathcal{U}$ be a uniformly sampled subset of the set of all mesh vertices $\mathcal{V}$. We calculate and store the geodesic distance for each pair $u_i, u_j \in \mathcal{U}$. This gives a matrix of size $|\mathcal{U}|\times|\mathcal{U}|$ of true geodesic distances. Consequently, in the average geodesic distances calculation, to obtain the geodesic distance $g(v_a, v_b)$ from vertex v_a to vertex v_b, we find the closest vertices u_a and u_b to them in $\mathcal{U}$, respectively, and approximate the distance by:

$$g(v_a, v_b) \quad \cong \quad g(v_a, u_a) + g(u_a, u_b) + g(u_b, v_b)$$

After scissoring, the set of vertices $\mathcal{V}$ is partitioned into two sets, $\mathcal{V}_0$ and $\mathcal{V}_1$ (and some new vertices are created in each one). $\mathcal{U}$ must be separated into two subsets, $\mathcal{U}_0 \subset \mathcal{V}_0$ and $\mathcal{U}_1 \subset \mathcal{V}_1$, and the matrix into two sub-matrices. Nevertheless, the entries in the sub-matrices need not be recalculated. Only some vertices whose closest points in $\mathcal{U}$ fall in the other part must find new closest points (e.g., $v_a \in \mathcal{V}_0$ but $u_a \in \mathcal{U}_1$).

6. Feature Contour Completion

Let γ be the contour selected. 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 that form a loop from one endpoint of γ to the other. However, this method is tedious and error-prone.

A loop around the mesh can also be defined using a 2D line drawing over the mesh. We define a cutting plane which must pass 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.

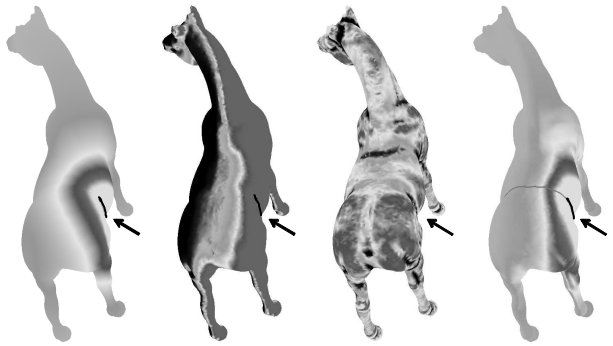


Figure 4. Visualization of the three functions used to complete an open contour (marked by the arrow) to a loop: (a) $c_d(v)$ drops as we get farther from the contour; (b) $c_n(v)$ is lower as the normals point in the other directions than the contour; (c) $c_f(v)$ is the feature energy (min curvature); (d) combination of all functions and the minimum path found.

Nevertheless, in order to perform fully automatic scissoring or when the 2D line drawn and projected over the mesh is used just over the front faces for semi-automatic scissoring, we need an automatic contour closing method which completes γ to a closed loop around the mesh.

6.1. Automatic contour closing

The basic problem of completing an open contour to a closed loop is that the path from one endpoint to the other of the contour 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 three functions that guide the search towards the other side of the mesh and through mesh features.

Let v be a mesh vertex. We define

$$c_d(v) \quad = \quad \sum_{v_i \in \gamma} \frac{1}{g(v, v_i)}$$

$c_d(v)$ is the sum of inverse distances from vertex v to all vertices on γ. Hence this function will be high in the vicinity of the contour γ, and drops as we get farther away (see Fig. 4(a)).

The second function $c_n(v)$ is lower for normals which face opposite directions of γ (see Fig. 4(b)). 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. Assume n_v is the normal at vertex v, we define

$$c_n(v) \quad = \quad \begin{cases} 1 & \text{if } n_c \cdot n_v \leq \cos(\alpha) \\ \frac{n_c \cdot n_v + 1}{\cos(\alpha) + 1} & \text{otherwise} \end{cases}$$

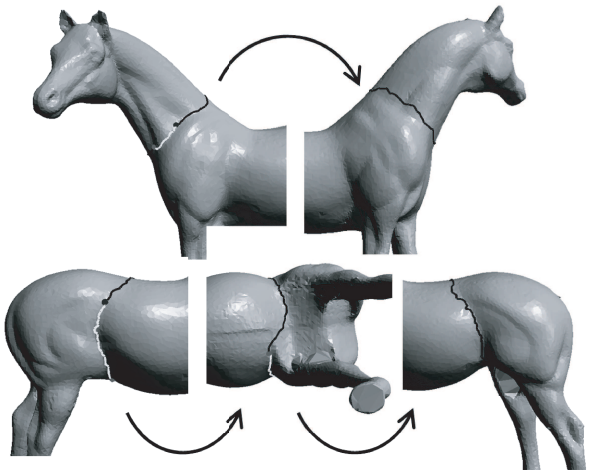


Figure 5. Examples of paths created by completing open contours to closed loops on the neck and around the waist of the horse.

The third function $c_f(v)$ guiding the path towards mesh features is the same one used to define the feature contours (see Fig. 4(c)).

For an edge e, $l(e)$ denotes the length of e. The cost functions, $c_d(e)$, $c_n(e)$, and $c_f(e)$, 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

$$f(e) \quad = \quad l(e) \cdot c_d(e)^{w_1} \cdot c_n(e)^{w_2} \cdot c_f(e)^{w_3}$$

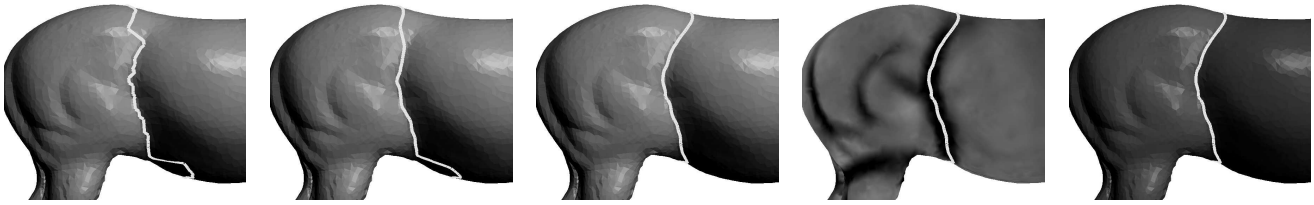
where w_i are used to control the strengths of these functions. We usually set all w_i to 1.0. Fig. 4(d) and Fig. 5 show examples of the paths created by completing open contours to closed loops.

7. 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;

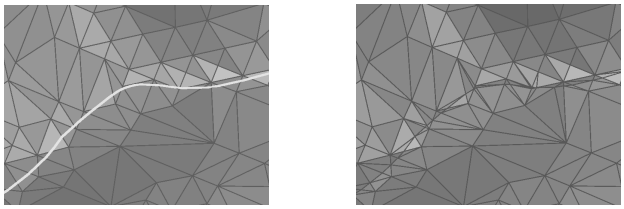
$$E_{snake}(s) \quad = \quad \int (E_{spline}(s) + E_{mesh}(s))dt$$

The snake external energy E_{mesh} is designed to capture nearby features and its internal energy E_{spline} to smooth



(a) (b) (c) (d) (e)

Figure 6. Snake relaxation procedure: (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.



(a) (b)

Figure 7. (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.

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 [15], we use a local parameterization that embeds faces around a snake onto a 2D plane. With this parameterization, we can use the same equation of an image snake for the energy minimization. The snake position is incrementally updated by repeatedly solving linear equations until a minimum is reached. See [15] for the details of snake movement.

The final position of a snake defines the scissoring position where a cut is made (see Fig. 6). Since snake sample points can lay at anywhere on the mesh surface, the snake contour may pass through triangles of a mesh. These triangles are subdivided along the snake to achieve smoother interface between the mesh parts (see Fig. 7).

8. Experimental Results

All following experiments were carried on an Intel(R) Pentium(R) M processor 1.5 Ghz with 512 Mb memory. Fig. 8 shows automatic partitioning results. These results were created by automatic extraction and completion of feature contours. The only manual intervention was performed in the contour selection phase. Some of the contours were selected manually, while others selected automatically had to be rejected in order to obtain better results (see Fig. 9). In more complex situations, the user can begin with automatic scissoring and then reject specific cuts and continue scissoring the mesh further in designated places. Fig. 10 shows examples.

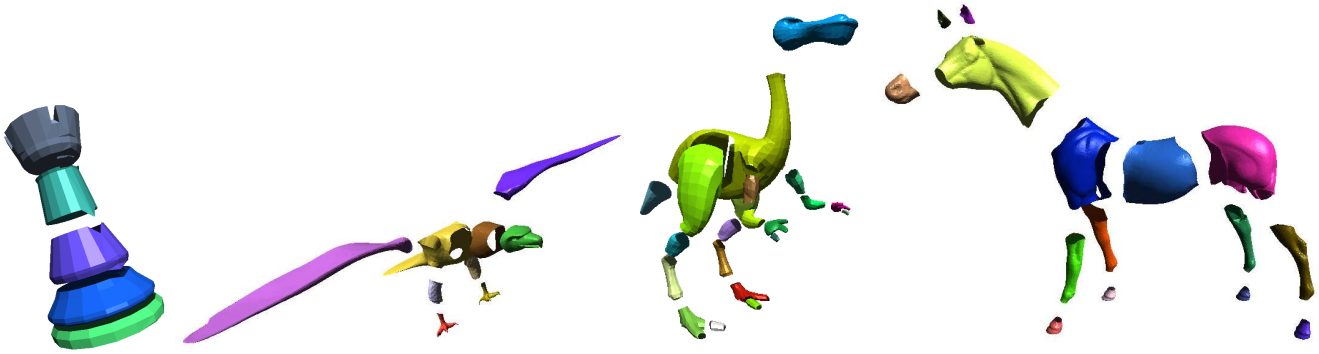
Manual scissoring can disregard the feature contours altogether. Using the 2D slicing approach, any designated part of the mesh can be removed. However, intelligent scissoring assures the cut would be smooth and follow natural shape features as much as possible. Fig. 11 shows examples.

Table 1 shows timing statistics for different meshes used in our experiments. As can be seen, after some preprocessing for feature extraction and geodesic distance computation, the method runs at interactive speed even on large meshes. Hence, using this approach, an interactive tool is provided where semi-automatic as well as automatic scissoring can be applied easily.

9. Summary and Future Work

We have presented an approach for 3D mesh scissoring which can be used for both automatic partitioning and manual cutting. Instead of clustering mesh elements, our scissoring approach targets the cutting position using feature extraction. This enables an easy-to-use partitioning tool which is more feasible to use than tedious cutting and more useful than fully automatic clustering approaches.

Most of the timing constraint for large meshes come from feature extraction and geodesic distance computation. The former must be used in any scissoring style, man-



(a) (b) (c) (d)

Figure 8. Mesh scissoring results using an automatic approach.

(a) (b) (c)

Figure 9. Some of the feature contours selected automatically (a) are rejected in order to achieve better results (b-c).

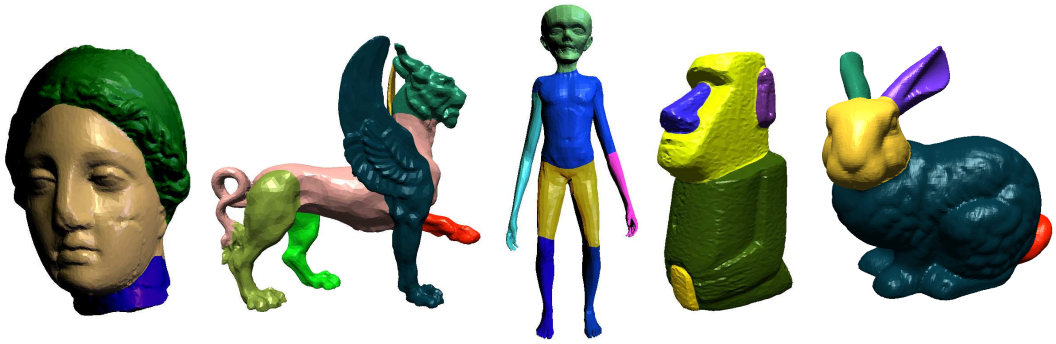


Figure 10. Mesh scissoring results using a semi-automatic approach.

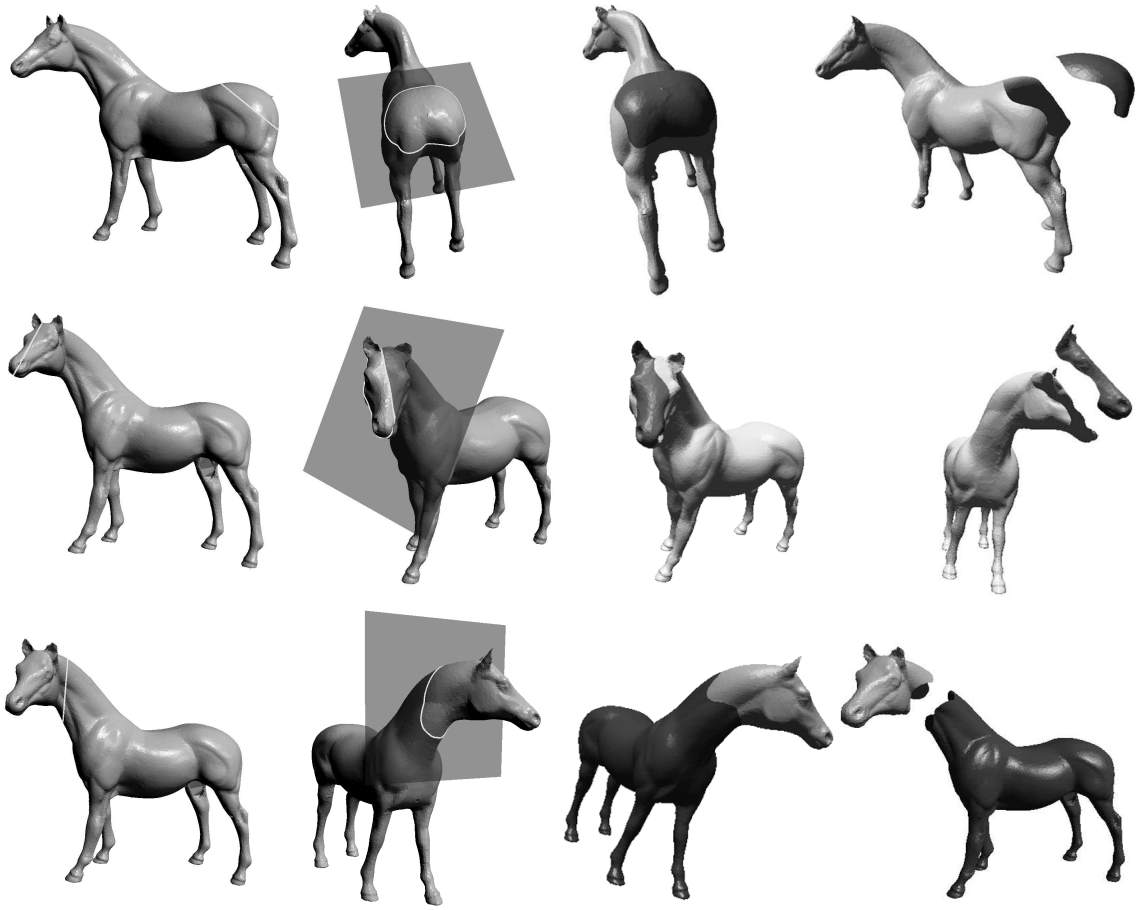


Figure 11. Manual mesh scissoring results: A 2D line is drawn on the mesh (left) and projected to create a slicing plane. This plane can be used to define the initial loop (middle left). Alternatively, the line projected on the front faces can be completed using our automatic completion method. Note 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).

ual or automatic, but can be pre-computed once for each mesh. The latter is only needed for automatic scissoring and hence we would like in the future to experiment our semi-automatic scissoring operator on very large mesh as well. Furthermore, we plan to incorporate our scissoring tool in a full mesh editing and manipulation system.

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 Korea Ministry of Education through the Brain Korea 21 program and the Game Animation Research Center.

References

[1] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Levy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Computer Graphics (Proc. of SIGGRAPH 2003)*, pages 485–493, 2003.

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

[3] L. D. Cohen and R. Kimmel. Global minimum for active contour models: A minimal path approach. *International Journal of Computer Vision*, 24(1):57–78, 1997.

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

[5] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by exam-

Mesh	number of vertices	curvature extraction	geodesic distance	contour extraction	contour connecting	automatic closing	geodesic updating	snake movement
Bunny	34,834	11∼12	89∼90	1∼2	1∼1.5	3∼4	11∼12	3∼9
Horse	19,851	6∼7	25∼26	0.5∼0.6	1∼1.5	3∼4	4∼6	2∼7
Igea	18,004	4∼ 5	21∼22	0.6∼0.7	1∼2	1∼2	6∼15	2∼3
Eagle	14,618	37∼38	9∼10	0.4∼0.5	0.80	1.00	1∼3	2∼3
Hand	10,070	2∼3	4∼5	0.3∼0.4	0.40	0.6∼0.7	1∼2	2∼2
Feline	10,013	2∼3	4∼5	0.2∼0.3	0.6∼0.7	0.3∼0.7	1.0∼1.5	1∼2
Moai	10,002	1∼2	4∼5	0.2∼0.3	0.80	1.00	0.3∼1.0	0.5∼0.8
Alien	7,401	3∼4	2∼3	0.1∼0.2	0.6∼0.7	0.50	0.3∼0.6	1∼2
Dino	3,323	0.9∼1	1∼2	0.1∼0.2	0.70	2.00	0.2∼0.3	1∼2
Chess	1,514	0.3∼0.4	0.40	0.03	0.6∼0.7	0.20	0.03	1∼2

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

ple. *ACM Computer Graphics (Proc. of SIGGRAPH 2004)*, 23:to appear, 2004.

[6] M. Garland, A. Willmott, and P. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proc. 2001 ACM Symposium on Interactive 3D Graphics*, pages 49–58, March 2001.

[7] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for full automatic similarity estimation of 3D shapes. *ACM Computer Graphics (Proc. of SIGGRAPH 2001)*, pages 203–212, 2001.

[8] D. Hoffman and W. Richards. Parts of recognition. *Cognition*, 18:65–96, 1984.

[9] D. Hoffman and M. Signh. Salience of visual parts. *Cognition*, 63:29–78, 1997.

[10] A. Hubeli and M. Gross. Multiresolution feature extraction for unstructured meshes. In *Proc. IEEE Visualization 2001*, pages 287–294, 2001.

[11] K. Inoue, I. Takayuki, Y. Atsushi, F. Tomotake, and S. Kenji. Face clustering of a large-scale CAD model for surface mesh generation. *Computer Aided Design*, 33(3):251–261, March 2001. The 8th International Meshing Roundtable Special Issue: Advances in Mesh Generation.

[12] A. D. Kalvin and R. H. Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications*, 16(3):64–77, 1996.

[13] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[14] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Computer Graphics (Proc. of SIGGRAPH 2003)*, 22(3):954–961, 2003.

[15] Y. Lee and S. Lee. Geometric snakes for triangular meshes. *Computer Graphics Forum (Proc. Eurographics 2002)*, 21(3):229–238, 2002.

[16] B. Levy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Computer Graphics (Proc. of SIGGRAPH 2002)*, pages 362–371, 2002.

[17] X. Li, T. Toon, T. Tan, and Z. Huang. Decomposing polygon meshes for interactive applications. In *Proc. 2001 ACM Symposium on Interactive 3D Graphics*, pages 35–42, March 2001.

[18] A. P. Mangan and R. T. Whitaker. Partitioning 3D surface meshes using watershed segmentation. *IEEE Trans. Visualization and Computer Graphics*, 5(4):308–321, 1999.

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

[20] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. *ACM Computer Graphics (Proc. of SIGGRAPH '95)*, pages 191–198, 1995.

[21] E. N. Mortensen and W. A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5):349–384, September 1998.

[22] D. Page, A. Koschan, and M. Abidi. Perception-based 3D triangle mesh segmentation using fast marching watersheds. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '03) - Volume II*, pages 27–32, 2003.

[23] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. *ACM Computer Graphics (Proc. of SIGGRAPH 2001)*, pages 409–416, 2001.

[24] S. Shlafman, A. Tal, and S. Katz. Metamorphosis of polyhedral surfaces using decomposition. *Computer Graphics Forum (Proc. Eurographics 2002)*, 21(3):219–228, 2002.

[25] O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski. Bounded-distortion piecewise mesh parameterization. In *Proc. IEEE Visualization 2002*, pages 355–362, 2002.

[26] C. Xu and J. L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Trans. Image Processing*, 7(3):359–369, 1998.