Prop-Cut: A mesh cutting method based on Tikhonov regularization

Hongxin Zhang
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, China
Email: zhx@cad.zju.edu.cn

Juan Wang
Graphics Architecture Department
AMD Inc.
Shanghai, China
Email: juanwang@amd.com

Abstract

We present an interactive mesh cutting method in this paper, which is based on a formulation of semi-supervised learning. Users first assign foreground and background seed faces on a given triangular mesh by sketch lines. Then the system computes a scalar field across the mesh. After that a coarse segmentation boundary is computed out respect to a specific iso-value, which leads to a refined boundary by tracing the isoline in the scalar field. Our proposed methods are easy for implementing. The presented computing framework can not only do segmentation for single static mesh models using shape information, but also do segmentation for dynamic mesh models based on deformation information. By integrating the proposed mesh cutout tool, we also demonstrate a simple sketch-based mesh editing system. In our system the cutting results can be further deformed, morphed, or cut-and-pasted.

1. Introduction

It is traditionally a difficult and often expensive process to create digital geometric models in the applications of graphics, engineering, and digital games. Till now, a large amount of geometric processing is still done manually to manipulating vertices and faces directly.

Recently, example based modeling techniques [1] become the new trend in geometric modeling. As an ideal user interface, non-professional users can easily generate vivid geometric models by simply cutting meaningful components from known ones and merging them together. This process, in some sense, is called as the cut-and-paste operation, which is borrowed from the original word process applications.

In this paper we focus on developing a simple mesh cut-and-paste operation based on sketching user interface. Our major motivation is that pen-based user interfaces are very intuitive for design activities and become popular especially for many portable devices. As a lot of approaches [2], [3], [4] can efficiently paste different mesh parts together, in this paper we mainly present a generic framework for cutting meaningful mesh components from a single mesh or a mesh sequence maintaining different poses.

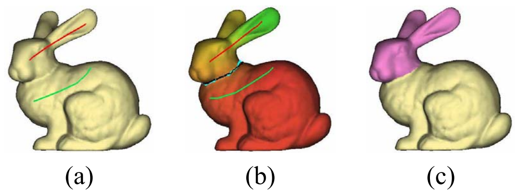

Figure 1. Cut out Bunny's head. (a) Users draw two sketch curves on the mesh to mark the foreground (in red) and the background part (in green). Then a scalar field across the mesh is calculated, shaded from red to green. (b) According to k-means cluster algorithm, an initial boundary is obtained and painted in black. The cyan boundary is the result of boundary refinement. (c) The two segmented parts are obtained by tracing isoline.

Human perception is complicated and is easy to be confused. Although it is quite easy for a model designer to figure out the boundary between the hand and the arm part in a character mesh model within sufficient time, it is still very difficult and is almost impossible for a digital computer to accomplish such cognition mission. However, researchers still believe that there exist several machine learning mechanisms that can partly solve these sole missions assisting by human. As we will discuss later in this paper that mesh cutting can be viewed as a semi-supervised learning task.

Many applicable mesh segmenting [5] or cutting methods [3] are based on a graph-cut procedure, for graph-cut can optimize segmenting boundary globally. However, one disadvantage of global graph-cut is that geodesic distance computing is required as pre-processing step, which is inefficient when handling meshes maintaining relative large amount of vertices.

In this paper we propose the *prop-cut* approach, which is a global cutting method avoiding to compute pair-wise geodesic distance. We calculate weights between every two adjacent faces. And then a belief value for each faces is learned after users draw several free hand curves representing background and foreground parts. These weights measure dihedral angle between adjacent faces or the deformation information obtained from a sequence of models

978-1-4244-3701-6/09/$25.00 ©2009 IEEE

as well. And according to these geometry-aware weights, label values can be well propagated by merely performing the Tikhonov regularization.

The main contributions of this paper are as follows:

- We formulate mesh cutting out operation as a semi-supervised learning task.
- We propose an interactive mesh cutting method based properties propagation.
- We extend our prop-cut method to mesh sequences maintaining different poses, and explore the approach to merge deformable mesh components together.

1.1. Related work

There is a great deal of work on mesh editing and its application in the literature. Here we only review the most relevant work, including mesh segmentation, semi-supervised learning and sketch-based user interface.

Mesh segmentation approaches can be roughly classified into two categories, i.e., *patch-type* and *part-type*. Patch-type segmentation is often applied for texture mapping, building charts and remeshing [6]. While part-type segmentation divides a mesh into meaningful parts without restricting the part topology. Several approaches have been proposed by different authors to automatically segment mesh into meaningful components, e.g., the minima rule [5] and watershed-based scheme [7]. It is easy to see strong relations among many part-type segmentation approaches, image segmentations and point-sets clustering in machine learning. For instance, Yamauchiy et al. [8] proposed to use mean shift for mesh segmentation. And later this method is extended for deforming mesh models [9], [10].

Mesh cutting methods, e.g. the intelligent scissoring tool [11] or easy mesh cutting [12], is quite different from automatic segmentations. User interactions are involved in these applications to reduce the computing cost for caring confusions and ambiguities.

Semi-supervised learning (SSL) is a branch of machine learning [13]. More precisely mesh cutting task is a semi-supervised binary classification but clustering is an unsupervised learning task. Standard classifier training utilizes only labeled data (feature/label pairs). However labeled data is often hard to be acquired because they need experienced human annotators, while unlabeled data may be relatively easy to collect. The goal of semi-supervised learning is to train better classifiers from both labeled and unlabeled data. Graph-cut is one of the graph-based SSL methods, and which is successfully applied in image/video cutout [14] and mesh cutting applications.

In recent years, sketching interfaces become wildly accepted for various design activities ranging from image matting to architecture modeling. They tend to provide users an environment simulating paper-ink interface that does not hinder creative thinking. Sketch-based interaction has been successfully used in graph cut image segmentation [14]. And now sketching interfaces are widely accepted to generating [15] and/or deforming 3D meshes [16], [17] by sketching curves.

2. Prop-Cut

In this section we will describe the main mathematical formulation of our approach.

2.1. Preliminaries

Let $\mathcal{M}$ be a 2-manifold surface embedded in $\mathbb{R}^3$, and $\mathcal{K}; \mathbf{X})$ be a triangular mesh representing the discrete geometry of $\mathcal{M}$, where $\mathcal{K} = \{\mathcal{V}, \mathcal{E}, \mathcal{F}\}$ encodes the connectivity of the simplicial complex containing the vertices $\mathcal{V} = \{i | 1 \leq i \leq m\}$, edges $\mathcal{E} = \{ \ i, j) | i, j \in \mathcal{V}\}$ and triangles $\mathcal{F} = \{ \ i, j, k) | \ i, j), \ j, k), \ k, i) \in \mathcal{E}\}$ of the mesh, and $\mathbf{X}$ represents the positions of the mesh vertices. The same symbol $\mathcal{M}$ will also refer to the triangulated mesh surface. A segmentation of mesh $\mathcal{M}$ is the set of sub-meshes induced by a partition of $\mathcal{K}$ into k disjoint subsets. In mesh cutting context, k is simply equal to 2.

As we will consider geometric models that have different poses, the term *multi-mesh* is used to represent mesh sequence data. In this paper, a multi-mesh refers to a mesh sequence consists of several meshes that share a single tropologic connectivity. Mathematically, let $\mathcal{M} = \ \mathcal{K}; \mathbf{X}^0, \mathbf{X}^1, \ldots, \mathbf{X}^N)$ be a multi-mesh, where $\mathcal{K}$ has the same meaning for a single static mesh. The symbol $\mathbf{X}^0$ represents the positions of the mesh vertices in rest pose, and $\mathbf{X}^i$ stands for deformed poses.

2.2. Prop-cut for a single mesh

Given a single static mesh $\mathcal{M}$ with n triangle faces, users need to assign the foreground and background triangles first. Then these faces on $\mathcal{M}$ are labeled as

$$b_i = \begin{cases} 1, & \text{foreground}, \\ -1, & \text{background}. \end{cases} \tag{1}$$

Without loosing generality, we assume that the first k faces are labeled. And we need to precondition the data by mean subtracting first. That is we take

$$\mathbf{b} = \ b_1 - \bar{b}, b_2 - \bar{b}, \ldots, b_k - \bar{b}, 0, \ldots, 0), \tag{2}$$

where $\bar{b} = \sum_i b_i$. To learn the value that how much a triangle face should be background or foreground, we leverage following propagation procedure to produce a scalar field over mesh $\mathcal{M}$.

2.2.1. Propagation. The propagation objective is actually running a regression procedure on a dual graph $\mathcal{G}$ of $\mathcal{M}$ that is constructed by regarding each triangle as graph node and connect edges if two triangles share on edge. Our goal is to obtain a value field with smooth transition according to the given face values. Therefore, a least squares optimization is performed on $\mathcal{M}$ to find $\bar{\mathbf{f}}$ such that:

$$\min_{\mathbf{f}=(f_1,\ldots,f_n),\sum_i f_i=0} \frac{1}{k}\sum_i \ f_i - b_i)^2 + \gamma \mathbf{f} S \mathbf{f}^\top \tag{3}$$

where f_i means the value of face i and $\gamma \geq 0$ is an optimization parameter. It is worth noting that the first part of Equation (3) is employed as soft constraints, and the second part is acted as a smoothness term. As our learning processing is performed on a graph, a good choice of the smoothness matrix S is:

$$S = L^p, p \in \mathbb{N} \tag{4}$$

where L is a Laplacian matrix over $\mathcal{G}$. More precisely, given a similarity matrix $W = \ w_{ij})$ of $\mathcal{G}$, the Laplacian matrix $L = D - W$ can be calculated with $D = \text{diag}\ \sum_j w_{1j}, \sum_j w_{2j}, \ldots, \sum_j w_{nj})$. For a single static mesh $\mathcal{M}$, the *similarity weight* between face i and j can be defined as

$$w_{ij} = \exp\ \rho|\mathbf{N}_i \cdot \mathbf{N}_j|) \tag{5}$$

with $\mathbf{N}_i$ and $\mathbf{N}_j$ denoting the unit normal vector of face i and j, respectively. Parameter ρ (usually =5) controls the influence of the dihedral angle between adjacent faces for the cutting results. Larger ρ will lead to more detail sensitive results.

We apply *Tikhonov regularization* to solve above optimization problem [18]. If we denote by $\mathbf{1} = \ 1, 1, \ldots, 1)$ the vector of all ones, the solution can be given in the form

$$\bar{\mathbf{f}} = \ k\gamma S + I_k)^{-1}\ \mathbf{b} + \mu\mathbf{1}) \tag{6}$$

Matrix I_k is a diagonal matrix of multiplicities

$$I_k = \text{diag}\ n_1, n_2, \ldots, n_k, 0, \ldots, 0), \tag{7}$$

where n_i is the number of occurrences of face i among the labeled face. In Equation (6), coefficient μ is chosen such that the resulting vector $\bar{\mathbf{f}}$ is orthogonal to $\mathbf{1}$. It follows that

$$\mu = -\sigma\ A^{-1}\mathbf{b})/\sigma\ A^{-1}\mathbf{1}) \tag{8}$$

where $\sigma\ \mathbf{f})$ is defined as $\sigma : \mathbf{f} \to \sum_i f_i$, and $A = k\gamma S + I_k$. Finally, we have propagated label $\bar{\mathbf{f}}$ for each triangle on the surface.

To obtain a piecewise linear scalar field over mesh $\mathcal{M}$, a propagation value is assigned on each vertex by averaging the label of its adjacent triangles. In Figure 1(b) we use gradually changed color to illustrate the diffused value of each face.

2.2.2. Initial cutting boundary. Once we have all propagated face values, we apply one dimensional k-means clustering (with two centers) for all triangles using propagation values to determine a cutting threshold. Therefore a rough segmentation boundary is generated along those mesh edges. This cutting boundary gives us a hint where a good boundary should be, although it is jaggy in usual.

2.2.3. Boundary refinement by isoline tracing. In order to make a fair mesh cutting, we need to find an optimized cutting boundary across mesh $\mathcal{M}$. Since we already have propagated values $\mathbf{v} = \ v_1, v_2, \ldots, v_n)$ on all its vertices, a refined boundary can be obtained by tracing a specified isoline of the piecewise linear scalar field. We adopt following method to find such isoline. The algorithm starts from an arbitrary seed vertex. There are two special cases in tracing the isoline.

First, the end point q of the isoline may coincide with a vertex of the input mesh. In this case, we must first examine the 1-ring neighbor vertices of this vertex, if any neighbor vertex also has the same scalar value, we simply prolong the isoline to this vertex (from $\overrightarrow{pq}$ to $\overrightarrow{qr}$, as illustrated in Figure 2(a)). If we cannot find a suitable adjacent vertex, then we examine each of the triangles incident on this vertex. Within each triangle, we can thus compute exactly where the isoline should cross a particular edge. If there are more than one triangle available, we advance the isoline through the triangle in which we can move the furthest on mesh (from $\overrightarrow{pq}$ to $\overrightarrow{qr}$, as shown in Figure 2(b)).

Second, the end point q of the flow line may lie along an edge of the mesh. The isoline will have already crossed one of the triangles incident on this edge, and we wish to extend it across the other incident triangle. We first check the vertices of this neighboring triangle, if any vertex of this triangle has the same scalar value, we simply extend the flow line to this vertex (from $\overrightarrow{pq}$ to $\overrightarrow{qr}$, as shown in Figure 2(c)). If all vertices are invalid, we thus compute exactly where the flow lines should cross a particular edge of the neighboring triangle (from $\overrightarrow{pq}$ to $\overrightarrow{qr}$, as shown in Figure 2(d)).

2.2.4. Boundary refinement by localized graph-cut. When handling a model with a small number of faces, the traced isoline gives us a satisfying boundary. However, when we segment a complex model with many geometry details, mere isoline will not reflect the geometry information. A localized graph-cut is performed to remedy this problem. First the 1-ring neighborhood around faces of the isoline is selected. Then a dual graph for all selected faces is built by using faces as graph nodes and connecting adjacent face nodes. Since only a small number of triangles are involved, the graph cut step consumes to get the segmentation boundary.

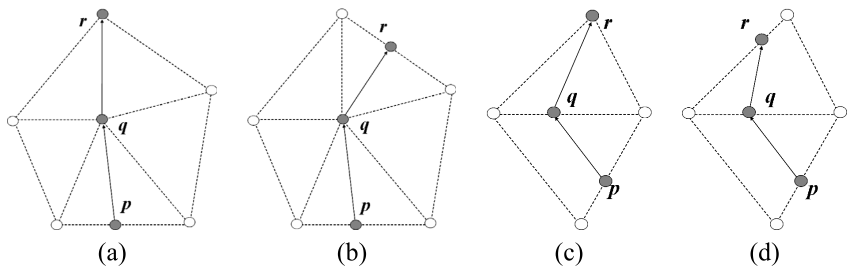

Figure 2. Image (a) and (b) demonstrate tracing isoline intersects a vertex on the mesh. Image (c) and (d) demonstrate tracing isoline intersects an edge on the mesh.

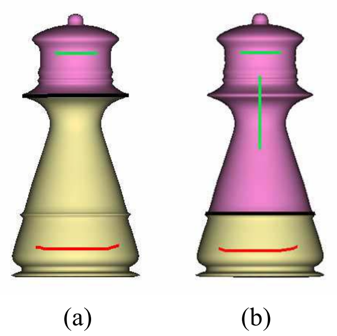

Figure 3. Illustrating of the boundary ambiguity problem. This problem can be simply avoided by drawing additional sketch curves.

2.2.5. User customized boundary. Although above two refinement strategies ensures to generate the cutting boundary as precisely as possible, there still exist undesired cases, especially some ambiguity boundaries. In our system, we provide two ways for user to modify the cutting boundary. Users are allowed to define specific isoline by simply tuning the threshold. This function can help user to shrink or enlarge a region of interesting. Second, user can draw additional strokes to update the cutting result. Therefore the ambiguity problem of segmentation can be avoided as shown in Figure 3.

2.3. Prop-cut for multi-mesh

A sequence of models can always give us more information for separating models into several meaningful parts. The models in a sequence can form a deformation space. Segmenting the static model can be extended to segmentation of the sequence models. The steps for sequence model segmentation are as follows:

1) Load the deforming model sequence, and compute the deform distance between every two adjacent faces.
2) Users sketch the foreground and background seed faces.
3) Our system begins to compute values for each face by solving the optimization problem similar to Equation (6).
4) Find the initial cutting boundary by k-means clustering.
5) Refine the boundary if necessary.

It is easy to see that, comparing with the from single static mesh cases, the only difference of our cutting out method for multi-mesh is in Step 1. For the sake of measuring the face similarity over a multi-mesh, there are two aspects which have to be taken into account, i.e., intrinsic geometric information as well deformation information. We hence introduce following several notions. First, we define function $Len\ \mathcal{T}_i, \mathcal{T}_j)$ as a distance based measurement of two faces $\mathcal{T}_i$ and Tr_j. $Len\ \mathcal{T}_i, \mathcal{T}_j)$ is actually approximated by the distance between the center point of two adjacent faces, or a relatively huge number M otherwise. Second, we measure the deformation distance between $\mathcal{T}_i$ and $\mathcal{T}_j$, denote as $Deform\ \mathcal{T}_i, \mathcal{T}_j)$. We employ the method in [10] to compute deformation distances. Finally, the similarity weight between two adjacent faces $\mathcal{T}_i$ and $\mathcal{T}_j$ in a multi-mesh $\mathcal{M}$ is defined as

$$w_{ij} = \ 1-\alpha)\frac{Len\ \mathcal{T}_i, \mathcal{T}_j)}{\text{avg } Len)} + \alpha\frac{Deform\ \mathcal{T}_i, \mathcal{T}_j)}{\text{avg } Deform)} \qquad (9)$$

where we normally set $\alpha = 0.9$, and avg $Len)$ as well as avg $Deform)$ denote to compute the average value of $Len\ \mathcal{T}_i, \mathcal{T}_j)$ and $Deform\ \mathcal{T}_i, \mathcal{T}_j)$ for all adjacent triangle pairs, respectively.

As we need to compute the transformation matrix between the rest pose and each deformed pose for every face on mesh $\mathcal{M}$, it may consume a lot of computing power. To accelerate the processing speed, we compute the transformation matrices as pre-processing step and store the results first. When we load the sequence models, we then only need to access the results directly.

3. Sketch based mesh editing system

In this section, we describe our novel framework of mesh (and/or multi-mesh) cutting-and-pasting. First we present the sketch-based user interface and mesh editing operators. And later we demonstrate the results obtained by our prototype system and several discusses are provided.

3.1. User interface

Our prototype modeling system mainly consists of mesh cutting, deformation, morphing and pasting functions. Users can accomplish almost all operations by drawing strokes or points.

Users first load mesh models (or multi-mesh models), and can navigate freely to choose appropriate view positions and angles (and also poses for multi-mesh models). Then users can mark several green and red strokes by mouse or digital pen to indicate foreground and background respectively. This marking user interface is similar to [12], [14]. Since our system can generate cutting boundaries in interactive speed, users can see the cutting result on screen and decides if additional strokes need to be marked or erase some failure strokes.

Once appropriate mesh components are selected by users, they are merged together by pointing two corresponding points for each of those cutting boundaries. We implemented a variant of [4] for the pasting function. Please refer to our supplemented video demo.

In some cases, it may produce visual unpleasant results when merely assemble a new object by mesh components from very different sources. So deforming pasted part can be applied to remedy this problem. For single static mesh sources, we adopt a sketch based deformation approach similar to [16]. Geometry details are fairly preserved by such deformation techniques.

3.2. Results

We show several examples demonstrating the usefulness and flexibility of our approach in this section. All the examples presented in this paper were made on a 2.4GHz Pentium IV computer with 1GB memory.

Table 1 lists the running time of performing three types of mesh cutting algorithms on several static mesh examples. Note that we also implemented a hierarchical graph-cut (HG-Cut) which first perform graph-cut on the simplified model and then perform localized graph-cut again to refine the cutting boundary on the original mesh. As we can see, HG-Cut is about five times faster than the original graph-cut. Our method is even faster than the HG-Cut, although our prop-cut merely works on the original resolution. The major reason is that our method can avoid computing geodesic distance globally, while it is an essential step in graph-cut algorithms.

Figures 1 and 4 are three examples demonstrating static mesh model cutting. It is worth noting that it is difficult to cut out the subparts using previous intelligent scissors [10] as discussed in [7]. As a global optimization approach,

Model	#verts	#faces	Prop-Cut	Graph-Cut	HG-Cut
Bunny	10k	20k	1.19	19.2	3.49
Venus	10k	20k	1.14	35.3	4.25
Gargoyle	36k	73k	4.65	369	9.22

Table 1. Time comparison between our prop-cut, graph-cut and hierarchical graph-cut (HG-Cut)

our prop-cut approach can achieve similar cutting quality to graph-cut approaches and is applicable for the cases that may be failed by using region growing methods, e.g. [7].

Figures 5, 6 and 8 demonstrate multi-mesh cutting-and-pasting. Since we use different weighting schemes, the propagation manner of multi-mesh cutting is quite different from the static cutting algorithm. It can be clearly observed in Figure 6 that the static cutting is sensitive to geometry features while the multi-mesh cutting is mainly caring of dynamic differences.

4. Conclusions and future work

We present a novel method for cutting out meaningful components from a single mesh or a mesh model with different poses using simple sketching interface. The free-hand strokes roughly mark out parts of interest and the background, as we formulate the mesh cutting problem as a semi-supervised learning task. Our system can segment the regions of interest interactively. The cutting boundary then can be optimized automatically. The system also provides flexible tools for users to edit the boundary. Our prop-cut approach is beneficial for interactive graphics applications.

Our presented approach still has much room for enhancements or extensions. First it will be helpful to design hierarchical version of prop-cut to accelerate the system response speed. And it is worthwhile to further investigate how to cut-and-paste dynamic mesh models for specific graphics applications. It would be also interesting to explore sketch-based mesh cutting methods which can incrementally adding sketch lines to refine the cutting results for very large meshes.

Acknowledgments

This work is supported in part by the National 863 Program of China (Grant No. 2006AA01Z335), and the National Natural Science Foundation of China (Grant No. 60505001).

References

[1] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin, “Modeling by example,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 652–663, 2004.

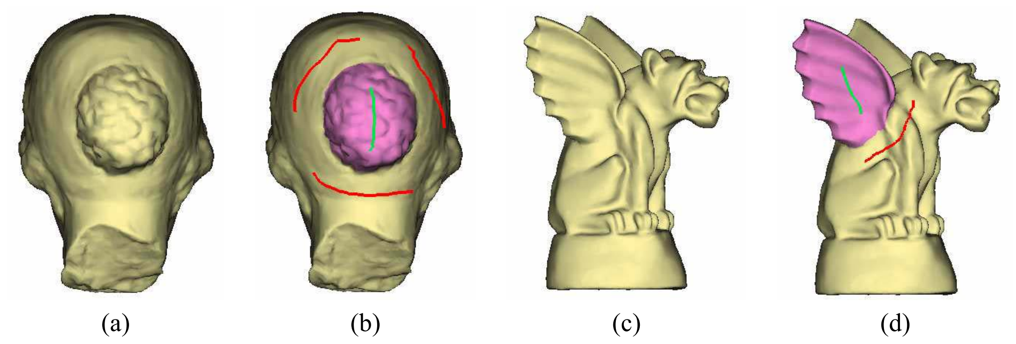
Figure 4. Illustrating performing prop-cut on static models. Image (b) demonstrate cutting a hair part from the Venus model (a). Image (d) demonstrate cutting a wing part from the Gargoyle model (c).

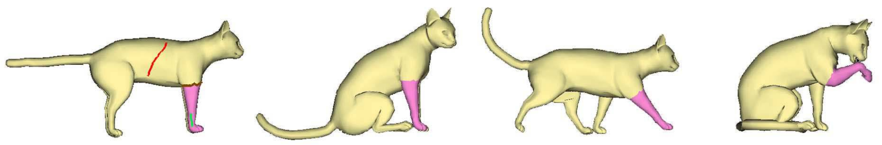
Figure 5. Cutting leg from a running cat model.

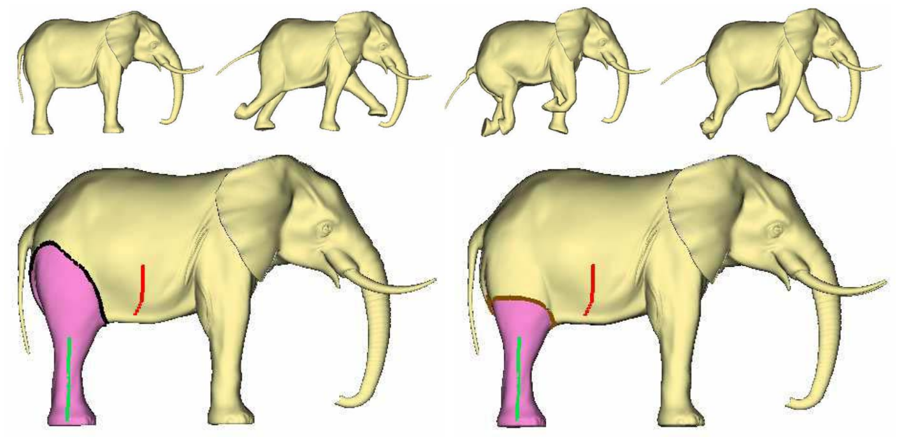
Figure 6. Cutting leg from a running cat model. The lower left result is obtain by performing static cutting, while the lower right one is obtained by performing multi-mesh cutting.

[2] X. Huang, H. Fu, O. K.-C. Au, and C.-L. Tai, “Optimal boundaries for poisson mesh merging,” in *ACM Solid and Physical Modeling Symposium*, 2007, pp. 35–40.

[3] A. Sharf, M. Blumenkrants, A. Shamir, and D. Cohen-Or, “Snappaste: an interactive technique for easy mesh composition,” *The visual computer*, vol. 21, no. 9, pp. 835–844, 2006.

[4] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum, “Mesh editing with poisson-based gradient field manipulation,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 644–651, 2004.

[5] S. Katz and A. Tal, “Hierarchical mesh decomposition using fuzzy clustering and cuts,” in *ACM SIGGRAPH ’03*, 2003, pp. 954–961.

[6] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, “Least squares conformal maps for automatic texture atlas generation,” in *ACM SIGGRAPH ’02*, 2002, pp. 362–371.

[7] A. Mangan and R. Whitaker, “Partitioning 3d surface meshes using watershed segmentation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 308–321, 1999.

[8] H. Yamauchiy, S. Lee, Y. Lee, Y. Ohtake, A. Belyaevy, and H.-P. Seidel, “Feature sensitive mesh segmentation with mean shift,” in *Shape Modeling International*, 2005, pp. 236–243.

[9] D. L. James and C. D. Twigg, “Skinning mesh animations,” in *ACM SIGGRAPH ’05*, 2005, pp. 399–407.

[10] T.-Y. Lee, Y.-S. Wang, and T.-G. Chen, “Segmenting a

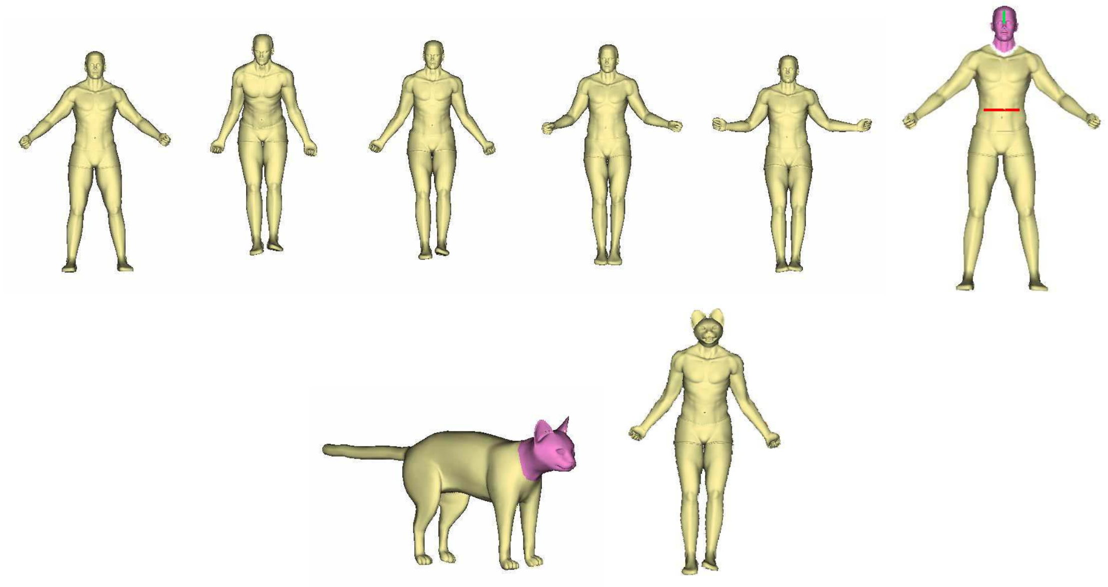

Figure 7. Pasting a cut head to a deforming human model.

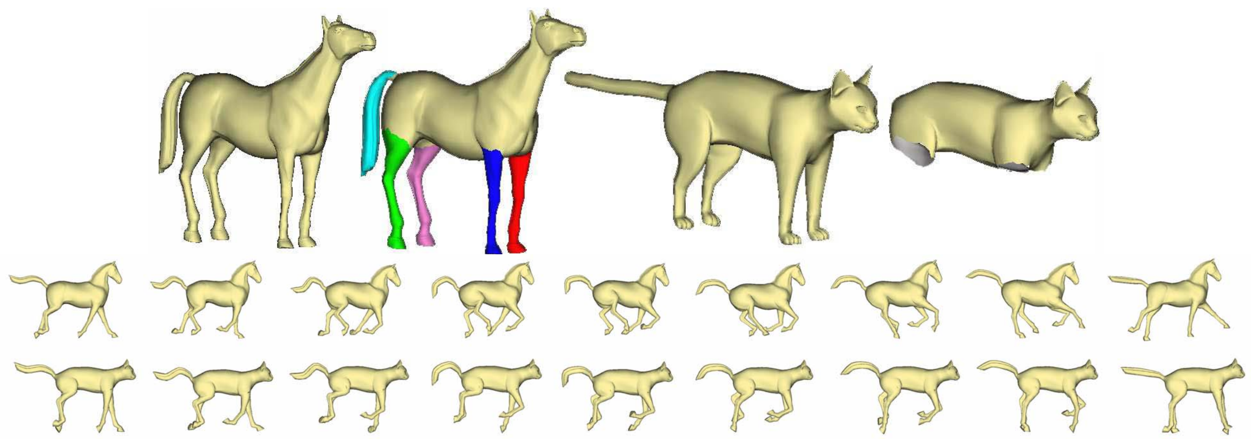

Figure 8. Illustrating of multi-mesh cutting and pasting. Legs and tails are separated for running horse and cat model. And a new funny motion sequence is generated by assembling the body of cat, legs and the tail of the horse model.

deforming mesh into near-rigid components," *The Visual Computer*, vol. 22, no. 9, pp. 729–739, 2006.

[11] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel, "Mesh scissoring with minima rule and part salience," *Computer Aided Geometric Design*, vol. 22, no. 5, pp. 444–465, 2005.

[12] Z. Ji, L. Liu, Z. Chen, and G. Wang, "Easy mesh cutting," *Computer Graphics Forum*, vol. 25, no. 3, pp. 283–291, 2006.

[13] X. Zhu, "Semi-supervised learning literature survey," Department of Computer Sciences, University of Wisconsin, Madison, Tech. Rep. 1530, 2005.

[14] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum, "Lazy snapping," in *ACM SIGGRAPH '04*, 2004, pp. 303–308.

[15] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: a sketching interface for 3d freeform design," in *ACM SIGGRAPH '99*, 1999, pp. 409–416.

[16] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or, "A sketch-based interface for detail-preserving mesh editing," in *ACM SIGGRAPH '05*, 2005, pp. 1142–1147.

[17] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum, "Large mesh deformation using the volumetric graph laplacian," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 496–503, 2005.

[18] M. Belkin, I. Matveeva, and P. Niyogi, "Tikhonov regularization and semi-supervised learning on large graphs," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004, pp. 1000–1003.