Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

# Easy cross parameterization for articulated shapes*

Jian-wei HU[1,2], Li-gang LIU[†‡1], Guo-zhao WANG[1]

(*[1]Institute of Computer Graphics and Image Processing, Department of Mathematics, Zhejiang University, Hangzhou 310027, China*)

(*[2]Department of Mathematics, Huangshan University, Huangshan 245041, China*)

[†]E-mail: ligangliu@zju.edu.cn

Received Aug. 24, 2008; Revision accepted Feb. 17, 2009; Crosschecked June 10, 2009

**Abstract:** This paper presents a novel interactive system for establishing compatible meshes for articulated shapes. Given two mesh surfaces, our system automatically generates both the global level component correspondence and the local level feature correspondence. Users can use some sketch-based tools to specify the correspondence in an intuitive and easy way. Then all the other vertex correspondences could be generated automatically. The cross parameterization preserves both high level and low level features of the shapes. The technique showed in the system benefits various applications in graphics including mesh interpolation, deformation transfer, and texture transfer.

**Key words:** Sketch, Compatible mesh, Vertex correspondence, Cross parameterization
**doi:**10.1631/jzus.A0820615 **Document code:** A **CLC number:** TP391

## INTRODUCTION

Given any two polygonal meshes with similar topology it is possible to compute a compatible mesh which inherits the topological information from the first mesh and the geometrical information from the second one. The problem of computing such a compatible mesh is referred to as 'cross parameterization' which consists of building a one-to-one correspondence between meshes. Parameterization was first introduced to computer graphics for mapping textures onto surfaces (Bennis *et al*., 1991; Maillot *et al*., 1993). Over the last decade, cross parameterization has gradually become a lively topic of research and has been used frequently for many applications like 3D morphing, texture mapping, deformation transfer, and remeshing.

Obviously, in order to retain certain characteristics between the models at a human shape perception level, the correspondence between meaningful components of models is required. For instance, most of quadrupeds have a similar body part structure which consists of a head, a body, four legs, and a tail. Considering building a compatible mesh between the cheetah and the triceratops shown in Fig.1, it is desirable that the meaningful components such as the head, the legs, and the tail in both shapes are in correspondence. These meaningful corresponding components are required by many applications. For example, in morphing the mesh models are expected to transform one shape into another with a smooth transition.

Unfortunately, the concept of 'meaningful component' is highly subjective and hard to quantify, not to mention finding the meaningful component correspondence between two shapes automatically. Although it is quite easy for human beings to specify the component correspondence by saying something like "the head of the tiger corresponds to the triceratops' counterpart", computers are still incapable of completing this tasks unassisted. Therefore, when automatic methods fail to build correspondences or users have their special corresponding needs, the user interaction becomes very important and unavoidable for creating such a compatible mesh.

[‡] Corresponding author

[*] Project supported by the National Natural Science Foundation of China (No. 60773179), the joint grant of the National Natural Science Foundation of China and Microsoft Research Asia (No. 60776799), and the National Basic Research Program (973) of China (No. 2004CB318006)

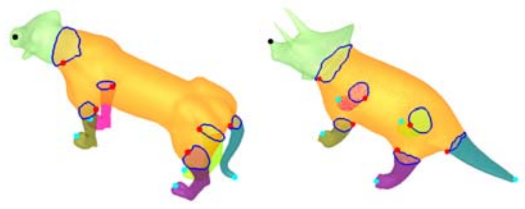

**Fig.1 Component and feature correspondences between two articulated models are generated fast in our system**
Corresponding components (component pairs) of these two models are shown in the same color. The feature points on the corresponding component pair are found to be in correspondence respectively by our approach automatically. Only the black point pair is specified by the user here

Existing cross parameterization tools are quite similar: specifying high-level correspondences is done by expressing such a requirement indirectly in terms of many low-level vertex correspondences. But specifying many points is often laborious and tedious for non-expert users. Although there are some interactive methods on morphing based on decomposition of mesh models (Shlafman *et al*., 2002; Zhao *et al*., 2003), these methods still require a tedious user interaction to perform this task.

Our goal is to build an easy-to-use interactive system that allows untrained users to create compatible mesh fast. The system allows users to easily and intuitively specify correspondence of meaningful components between two shapes that are consistent with user intention and human shape perception. Thus, our system is drastically simpler to use than generic interactive systems. For example, the component correspondence in Fig.1 was created with just eight mouse clicks and drags.

### Our approach

We propose a novel intuitive interface for cross parameterization by sketching freehand strokes on mesh surfaces with an automatic assistant process. The sketching based user interface is inspired by emerging sketching applications (Zeleznik *et al*., 1996; Igarashi *et al*., 1999; Ji *et al*., 2006; Karpenko and Hughes, 2006).

Our main goal is to reduce and simplify user interaction as much as possible within some automatic mechanism while providing users with some sketch tools to adjust the results. The process has four main steps: segmentation and skeleton extraction, component correspondence, feature correspondence, and parameterization. Each step involves several automatic algorithms, providing the users with an initial solution. Users are allowed to adjust the results of the first three steps if the outcomes are not that satisfying. Users simply need to draw some freehand sketches on both given meshes. Our system provides several types of sketch tools with different semantic cues, which will be introduced later.

Our system is efficient, easy to use and provides a great flexibility to the user. Many experiments show that our system builds correct correspondence of perceptual components precisely and efficiently while requiring little skill or effort from the user. By using a sketch-based interface and an automatic mechanism, we avoid the need for complex 3D interactions that can be cumbersome for novice users. Users can achieve relatively complex applications by simply drawing a few strokes on the screen.

## PREVIOUS WORK

### Sketch-based user interfaces

Sketch-based interfaces have emerged as an approach to enhancing user interaction for various design activities. They are expected to provide flexible and informal interaction between computers and users that does not hinder creative thinking. Sketch-based interaction has been successfully used in image segmentation (Boykov and Jolly, 2001; Li *et al*., 2004), objects creating and editing (Igarashi *et al*., 1999; Karpenko and Hughes, 2006), shape deformation (Kho and Garland, 2005; Nealen *et al*., 2005), and mesh cutout (Ji *et al*., 2006).

### Interactive mesh correspondence

A common approach to finding a correspondence between two given meshes is to allow users to specify vertex matching pairs and use them in guiding the computation of the complete vertex correspondence between two meshes. Lee *et al*.(1999) allowed users to specify vertex pairs on two original meshes and solved the correspondence problem through simplified meshes which have fewer vertexes but preserve the features of the original meshes. Kanai *et al*.(2000) employed coarse meshes overlaid on the source meshes to let users specify mesh-to-mesh feature correspondence. Alexa (2000) allowed users to specify scattered feature correspondences and then aligned the selected features which are projected to a spherical surface.

A more general approach is to build vertex correspondence based on patch decomposition. A user needs to specify a feature net (i.e., a set of features) and their connectivity by assigning feature vertex pairs and then identifying identical inter-patch connectivity between meshes. Gregory *et al*.(1998) decomposed mesh surfaces according to the user specified feature nets over them. Shlafman *et al*.(2002) proposed an algorithm for establishing a correspondence of polyhedral models based on decomposing the input models. They decomposed the models into pieces and built correspondence piece by piece. Praun *et al*.(2001) created consistent parameterizations by partitioning the meshes based on the connectivity of the simplicial complex and parameterizing each patch onto the respective simplicial complex face. The user needs to specify corresponding vertices and the simplicial complex. Kraevoy and Sheffer (2004) built mesh correspondence by not requiring the simplicial complex to be specified a priori. Zhao *et al*.(2003) proposed an interactive morphing framework to empower users to control the whole morphing process, but it is difficult for an untrained user. Schreiner *et al*.(2004) constructed consistent parameterizations using a symmetric, stretch-based relaxation procedure. All these works lack some guiding mechanism to reduce and simplify user interaction.

### Cross parameterization

In a cross parameterization or inter-surface mapping setup, the parameter domain for one mesh is another surface mesh. Many methods use mapping to a common base complex to obtain a cross parameterization (Praun *et al*., 2001; Kraevoy *et al*., 2003; Kraevoy and Sheffer, 2004; Schreiner *et al*., 2004). Several methods use an energy-driven approach for cross parameterization, where one mesh is directly attracted towards the other (Allen *et al*., 2003; Sumner and Popović, 2004; Matsui *et al*., 2007). It is important to point out that Bronstein *et al*.(2006) proposed an automatic method to perform surface matching based on Riemannian geometry, called ‘generalized multidimensional scaling’ (GMDS), where an isometry-invariant embedding was used to compute an intrinsic geometric representation of the surface. Recently, Mateus *et al*.(2008) proposed an automatic method for establishing dense correspondences between the points associated with two articulated objects using both spectral matching and unsupervised point registration. This method has several advantages such as the capability of dealing with partially missing surfaces; however, it focuses on isometry-invariant surfaces and lacks the capability of matching shapes with different intrinsic structures.

## SYSTEM OVERVIEW

A mesh exists as a collection of 3D points and their connection, and contains no explicitly defined high-level information. To decompose a mesh into a collection of primitives is a popular way to manipulate it. We refer to each primitive as a component. The set of points shared by two adjacent components is termed a boundary. Figs.2c~2d show the above definitions.

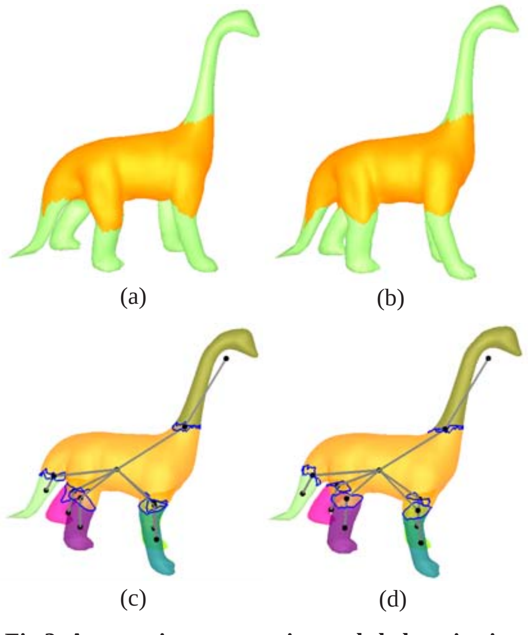

**Fig.2 Automatic segmentation and skeletonisation**
(a) and (b) are the results of the automatic partitioning method proposed in Shapira *et al*.(2008) with the parameter $\lambda$=0.3 and 0.45, respectively; (c) and (d) are the segmentation and skeleton results of (a) and (b), respectively

The input to our system is a pair of mesh models of articulated shapes and the output is a correspondence between these two meshes. Our method includes four steps which allow human interactions. In each step, users' specifications are needed only when the results are not that satisfying. First, the framework calls the principal component analysis (PCA) method to reorient and normalize the given meshes. The PCA method both builds a covariance matrix by treating a model as a collection of points and computes normalized eigenvectors and eigenvalues of the covariance matrix. Each eigenvector describes a principal mode of variation along the set, the corresponding eigenvalue indicating the importance of this mode in the shape space scattering. If the orientations are not reasonable, for example, the head of a horse mesh and the tail of a tiger mesh share the same direction, users can easily change it by dragging the mouse. After alignment, the input models are normalized by comparing the diagonals of the models' bounding box. Second, both meshes are segmented into meaningful patches automatically. If some components are not meaningful, users can modify the results by either dragging the scroll bar or using a sketch cut tool proposed in Ji *et al*.(2006) with some improvements to obtain a better result. Then skeletons are extracted based on the segmentation information. Third, our framework builds correspondences of components between the two meshes by using a graph-match algorithm. When the postures of the given meshes are slightly different, this match may fail. In this case we provide users a brush tool to change corresponding components directly. Fourth, multifarious methods are applied to make correspondence of feature points and boundary points. Also, users are allowed to mark these low-level point correspondences by simply clicking the mouse. Finally, in order to establish a feature-preserving compatible mesh, a parameterization method is involved. The workflow of our system is illustrated in Fig.3.

## HIGH-LEVEL COMPONENT CORRESPONDENCE

Given two meshes $M_1$ and $M_2$, users can specify their components and the correspondences over these components. The result of the component correspondence process is the correspondences among all components in $M_1$ and $M_2$.

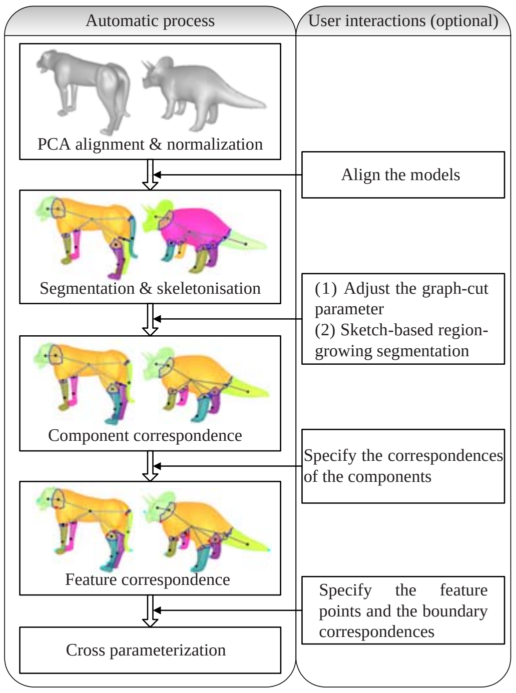
Automatic process
User interactions (optional)
PCA alignment & normalization
Align the models
Segmentation & skeletonisation
(1) Adjust the graph-cut parameter
(2) Sketch-based region-growing segmentation
Component correspondence
Specify the correspondences of the components
Feature correspondence
Specify the feature points and the boundary correspondences
Cross parameterization

**Fig.3 Overview of our system**
Our system is able to perform each of the steps automatically. Some simple user interaction is needed only when users are not satisfied with the results at some step

### Meaningful segmentation

In order to segment the meshes into a set of meaningful components automatically, there are two recent related works (Kraevoy *et al.*, 2007; Shapira *et al.*, 2008) which can be used here. We adopt the partition scheme presented in Shapira *et al*.(2008) since this method is easy to implement and can obtain good results as well. The partition scheme relies on a volume-based shape function called the shape diameter function (SDF) which remains largely oblivious to pose changes of the same object and retains similar values in analogue parts of different objects. Although the method can do a good job with most models, sometimes users may want to adjust the result.

The meaningful segmentation algorithm is composed of three steps:

1. Calculate an SDF value for each face of the input meshes. The SDF value is an approximation of mesh volume information.

2. A Gaussian mixture model (GMM) is used to fit $k$ Gaussians to the histogram of SDF values. This is achieved using the expectation-maximization (EM)

algorithm. Each of the Gaussians is considered as a cluster. The result of this process is a vector of length *k* describing its probability of each face to be assigned to one of the SDF clusters.

3. In order to optimize the boundaries, the algorithm employs an alpha expansion graph-cut algorithm which minimizes the energy functional in Eq.(1), built from the data term $e_1$ and the smoothness term $e_2$:

$$E(x)=\sum_{f\in F}e_1(f,x_f)+\lambda\sum_{\{f,g\}\in N}e_2(x_f,x_g), \tag{1}$$

$$e_1(f,x_f)=-\log(P(f\mid x_f)+\varepsilon), \tag{2}$$

$$e_2(x_f,x_g)=\begin{cases}l(f,g)(1-\log(\theta(f,g)/\pi)), & x_f\neq x_g,\\ 0, & x_f=x_g,\end{cases} \tag{3}$$

where $x_f$ (respectively $x_g$) is the cluster assigned to face *f* (respectively *g*); *F* is the set of mesh facets; *N* is a set of adjacent face pairs in the mesh; $\lambda$ is a parameter defining the degree of smoothness; $l(f, g)$ is the length of the edge between *f* and *g*; $\theta(f, g)$ is the dihedral angle between facets *f* and *g*; $\varepsilon$ and $\pi$ can be found in Shapira *et al*.(2008). From the experimental results, we believe that the parameter $\lambda$ can affect the partitioning result directly since it balances the data term against the smooth term. Thus, we provide users a scroll bar to change the value of $\lambda$. With benefits from the independence and the speediness of the graph-cut algorithm, the modified result can be illustrated almost in real time (Figs.2a and 2b show the example of segmenting a dinosaur model with $\lambda$=0.3 and 0.45, respectively).

The parameter $\lambda$ is a trade-off weight between the data term and the smooth term. When we increase the value of $\lambda$, the boundaries become smoother and the number of components decreases. To make the system flexible, we provide a scroll tool to allow users to adjust the value of $\lambda$ to control the segmentation results. For articulated meshes with similar shapes, this algorithm is likely to give the same number of components. But it may fail when the structures of the input meshes are different, such as a horse model without the tail and a dinosaur model (Fig.4). In this case, we provide users an interactive tool based on sketch (described in the next subsection) so that the numbers of components of the input meshes are the same.

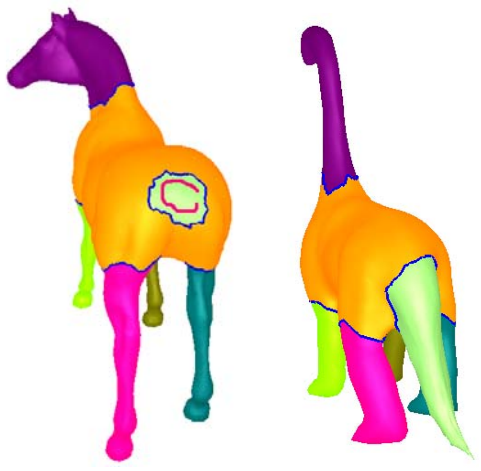

**Fig.4 Lasso tool in our system**
(a) Users draw a loop sketch on the back of the horse model; (b) The region bounded by the boundary is the specified new 'component' and is in correspondence with the tail part of the dinosaur model

### Sketch-based segmentation tools

As mentioned above, the graph-cut parameter can affect the segmenting result globally. That means if users change the value of the parameter $\lambda$, all boundaries of the model are slightly modified simultaneously. The problem is, if users want to modify one boundary and keep the others unchanged, they cannot achieve this by changing the parameter value. In these cases, we use a region-growing scheme based on a sketch tool and let users design their own boundaries in an easy and efficient manner. The region-growing algorithm starts with some seeds (faces) specified by user's sketch, and then the algorithm adds new neighboring faces to form the segmented patches iteratively. What users need to do is to draw two sketches in different components whose boundary needs to be adjusted and our tool re-segments this region immediately (Fig.5). It starts with different seed vertices on marker sketches simultaneously and grows according to an improved isophotic metric. This is similar to the work of Ji *et al*.(2006). But their scheme does not work well when the two meaningful components are smoothly connected and the boundary between them is perceptually unclear. Our algorithm is better and different from theirs in three aspects, as elaborated below.

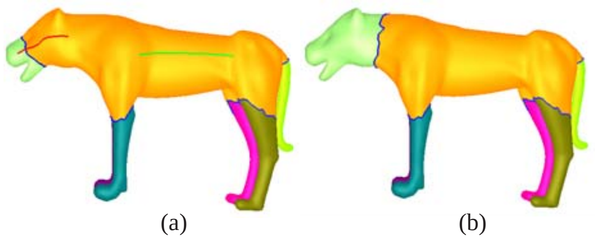

**Fig.5 The sketch-based user interface for adjusting the boundary of the component**
(a) The user interface: the two sketched lines are drawn by the user; (b) The adjusted result

1. Symmetric sketch hints

After the user draws a sketch on the screen, the sketch is projected over front faces of the mesh (Ju *et al.*, 2007). As the user has the perceptual concept on the component, it is assumed that he/she draws the sketch from a perfect view direction. For that case, we can create more seeds in growing the patches by projecting the sketch over back faces of the mesh. Note that we select only the nearest back faces from the view point if there is more than one.

2. Boundary improvement

The initial boundaries of components are not smooth after the segmentation process. In order to match the boundaries as accurate as possible in the next step, these boundaries need to be improved. The boundary between two components is optimized by the 'snake' algorithm used in the technique of 'easy mesh cutting' (Ji *et al.*, 2006). We further improve the boundary by reducing the sum of volumes of the convex hulls of the components (Kraevoy *et al.*, 2007). The boundary improvement algorithm tests whether the sum of the hull volumes of the components can be reduced by removing a vertex from one component and adding the vertex and all the incident triangles to the other component. Reducing this sum implicitly reduces the overlap, since the sum double counts the volume of the overlapping region. The boundaries between the two neighboring components are typically fairly straight, when the overlap between hulls is minimal. The procedure is repeated as long as the sum of volumes can be reduced.

3. Lasso and brush tool

When users want to specify a new region that has not been previously detected by the automatic method, we provide the user a lasso tool to specify a disk-like region on the mesh that 'grows' a component agreeing with human perceptions. For example, if the user wants to build a compatible mesh between a horse model and a dinosaur model but the horse model does not have a tail part, the user just draws a loop like stroke near the back of horse, as seen in Fig.4. Then the loop stroke will be closed and smoothed automatically and the region bounded by the stroke is the specified new 'component' of the horse model.

## Graph matching

Once the components and the skeletons of the given meshes are built, the system has to find the component correspondences between the two meshes. When this procedure is to be performed by a computer automatically without the assistance of a human expert, a useful way of representing the knowledge is through graphs. In this work, we consider the skeleton as a graph. Our system constructs a skeleton of a mesh by assigning a bone to each component. It is not required that the bone for a component accurately represent the component shape. Each component possesses a centroid called the component origin. The skeleton is a tree composed of two alternating kinds of points: the component origins and the centers of mass of the boundaries between two adjacent components, referred to as boundary centers. Once the decomposition procedure is finished, the decomposition tree is traversed and a skeleton (tree of joints) is generated. Our system finds a component with a maximal number of neighbors and defines the root of the skeleton as its origin. After that, we use a width-first graph traversal scheme to extract a skeleton tree, as shown in Figs.2c~2d.

The solution to a graph-matching problem is defined as the optimum of some objective function which measures the similarity between matched nodes and edges. This objective function is called the fitness function (Bengoetxea, 2002), which is a graduated assignment algorithm for graph matching. Since our given meshes are reoriented and normalized, here we simply use the Euclidean distance function as the fitness function. We reformulate the problem as finding the maximum cardinality and minimum weight matching in a bipartite graph. In such a formulation, there is a connection called edge between each node in one graph and each node in the other. The weight of the edge represents the Euclidean distance between the two nodes' 3D space positions. In solving a maximum cardinality, minimum weight matching involves choosing a subset of the edges which provide a one-to-one mapping in the bipartite graph. The sum of edge weights (distance) is small, and cardinality (matching size) is high. However, this algorithm cannot handle the problem that the hierarchical structures of the two graphs are not obeyed (Bengoetxea, 2002). In that case, two nodes with different numbers of adjacent nodes may match each other. To preserve the hierarchical (ancestral) relationships in the two graphs, we have combined the above bipartite algorithm with a depth-first search. The approach can be thought of as a coarse to fine strategy, in which matching at higher levels of the tree is used to constrain matching at lower levels (Sundar *et al*., 2003), and furthermore, since the numbers of nodes with the same valence in both graphs in our problem are the same, we constrain the above algorithm to find a one-to-one mapping between all nodes of each graph. The algorithm efficiently finds matches between the trees, starting at the root of the tree (the node with the maximum valence), and proceeding down through the subtrees in a depth-first fashion. At each level of the subtrees, we define the fitness measure as

$$\sum_{i,j=1}^{N_k} d(n_i^1, n_j^2), \tag{4}$$

where $n_i^1$ and $n_j^2$ are the nodes at the *k*th level in trees $T_1$ and $T_2$ respectively, $d(a, b)$ is the distance between nodes $a$ and $b$, and $N_k$ is the number of the nodes at the *k*th level in each tree.

The above algorithm works for most mesh models. When the algorithm fails, our framework provides a brush tool to specify the component correspondences between models. When a component of a mesh is marked to correspond with one of the other mesh, the corresponding nodes and edges are removed from the two graphs, and then the two graphs are matched again.

## LOW-LEVEL FEATURE CORRESPONDENCE

At the local level, we consider two types of point correspondence: feature point correspondence and boundary point correspondence. In addition, users can specify and match two points of a component pair to control the resulting compatible mesh.

### Feature point correspondence

For each pair of components, the system is able to find and add pairs of local feature points. At least one pair of features is needed here to avoid building correspondences between a region around a feature and a region far from a feature. We define the distance between two vertices as the length of the shortest path between them along the mesh edges and the distance between two vertex sets as the minimum distance between their vertices for the rest of this section.

In Zhao *et al*.(2003), the vertex farthest away from the boundary of its component is computed and treated as the tip vertex. Two such tip vertices in two corresponding components are paired during their morphing process. Because these tip vertices do not contain feature information, sometimes they do not lie on the sharp feature; it is not coincident with human perception. Fig.6 shows that our strategy is better.

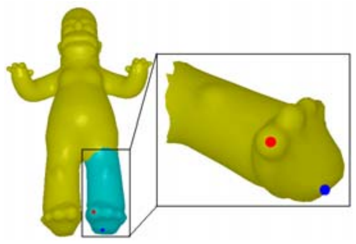

**Fig.6 Tip vertex**
The point on the sole is the tip vertex found by using the method of Zhao *et al*.(2003). Considering feature information, our method chooses the one on the toe, which is closer to human perceptual knowledge

Let $C$ be a component, $v_i$ a vertex on it, and $\Omega$ its boundary. We compute the distance $d_i$ between vertex $v_i$ and the boundary $\Omega$. In order to combine the distance information with feature information, a saliency value $S(v_i)$ is calculated for each vertex $v_i$ using the mesh saliency approach proposed by Lee *et al*.(2005). Then $S(v_i)$ is normalized to $\bar{S}(v_i) \subset (0, 1]$ and the feature value $f_i$ of each vertex $v_i$ is formulated as

$$f_i = \sum_i d_i + \gamma \bar{S}(v_i), \tag{5}$$

where $\gamma$ is the weight parameter. We set $\gamma$=0.4 in our system, which has performed well for all the examples shown in the paper. Our system chooses the point with the maximum feature value as the feature point of component $C$. The point is used for boundary point correspondence described in the following.

**Boundary point correspondence**

Two corresponding boundaries $\Omega_1$ (belongs to $C_1$) and $\Omega_2$ (belongs to $C_2$) of a component pair ($C_1$, $C_2$) are two feature vertex loops. There should be some correspondences between them so that the framework knows how they are aligned during parameterization. Our system builds the correspondences on $\Omega_1$ and $\Omega_2$ as follows.

In order to align the boundary we use the skeleton information. Considering two connected component with one common boundary $\Omega$, we extended the graph describing each skeleton by adding some new nodes and call the matching algorithm again so that the new graph can be matched properly while keeping the node correspondences of the original graphs unchanged. For a boundary pair (see, for example, Fig.7, the head boundaries of a cheetah model and a triceratops model), we formulate the number of new nodes as $M$=min($M_1$, $M_2$), where $M_1$ and $M_2$ are the numbers of points in $\Omega_1$ and $\Omega_2$ respectively. For example, if $M_1$<$M_2$ (so $M$=$M_1$), we choose all the $M$ points as new nodes in $\Omega_1$. For $\Omega_2$, $M$ points are chosen averagely. Then the new skeleton graphs are built by adding these new nodes to the original boundary. For example, Fig.7 illustrates the matching result of our algorithm.

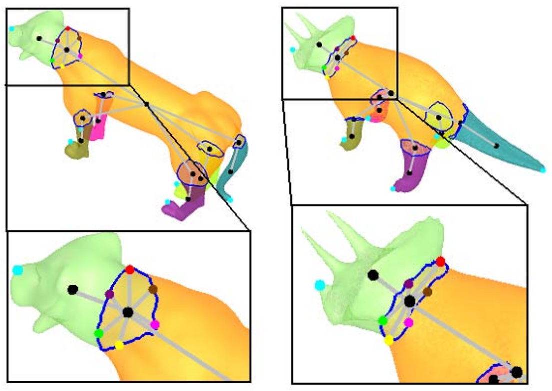

**Fig.7 Boundary correspondence**
The corresponding boundaries of two models are matched by using a graph match algorithm. The points in the same color are considered to be a pair. Here we show only six points in each boundary for a clearer illustration

**User specified low-level correspondence**

Though adding the feature point pairs and the boundary pairs is quite enough to generate a compatible mesh automatically, users may want to specify more feature pairs. Thus, we allow the user to use a brush tool to specify the low-level vertex correspondence. Each short stroke has a specified width representing a rough region that contains the vertex required by the user. From the user's perspective, the meaning of each stroke is 'I want to select a vertex within here'. From the system's perspective, the vertex with the largest value of mesh saliency (Lee *et al*., 2005) is the one specified by the user. In our system, the user can also use the brush tool to paint long strokes on the mesh to specify feature points.

## CROSS PARAMETERIZATION

From the key vertex correspondences between the source mesh and the target mesh, types of key vertex correspondences include user specified vertex pairs, boundaries vertex pairs, and feature vertex pairs automatically generated by the system. There are several methods for topological merging that can be used to create the complete vertex correspondence for each pair of patches. Since the articulated models have similar intrinsic characters to human body shapes, we adopt the method proposed in Allen *et al*.(2003). The approach formulates an optimization problem by using a set of error functions: data error, smoothness error, and marker error. With those point pairs in the two mesh models, the compatible mesh can be established correctly and fast.

## APPLICATIONS

We implemented a prototype system for the easy cross parameterization framework on a Pentium IV 3.0 GHz computer with 512 MB RAM. The most time-consuming step was the automatic segmentation step and other steps took only several milliseconds. Table 1 provides the computational time of the segmentation step for the models used in this study.

1. Mesh interpolation. After establishing the complete vertex correspondence between the source mesh and the target mesh, we adopt a linear

**Table 1 The running time of automatic segmentations for the examples used in this study**

| Model | Number of vertices | Number of faces | Time (s) |
|---|---|---|---|
| Triceratops | 2832 | 5657 | 2 |
| Cheetah | 5549 | 11 094 | 5 |
| Dinosaur | 10 189 | 20 374 | 12 |
| Armadillo | 165 954 | 331 904 | 163 |

interpolation to generate vertex trajectories. As the compatible meshes are well established, the interpolating sequence is meaningful, which means interpolating between the corresponding components from different models. In Fig.8, users need to add a new component near the horse back and in Fig.9, more feature points are specified by us at the head and the toes, so that the details and sharp features can be preserved in the resulting compatible mesh. Fig.10 shows that our method can deal with two models with quite different poses; in this example, we use several interactions including sketch-based segmentation tools, component corresponding tools, and user-specified low-level correspondence tools.

2. Deformation transfer. Since the segmenting algorithm (Shapira *et al*., 2008) is based on a volume-based shape function that maintains similar values in analogue parts of different objects, our system can build an excellent correspondence between two similar articulated mesh models. In Fig.11, we use our system to establish point correspondences and transfer the deformation of a cat to a lion by using the method provided in Sumner and Popović (2004). In this example, the challenging part is the thin tail of these two models. Nevertheless, most parts' correspondences can be done well with few user interactions.

3. Texture transfer. Fig.12 illustrates the texture transfer from a tiger model to a pig model. With some user interactions, the correspondences can be more accurate especially in the face and the tail regions.

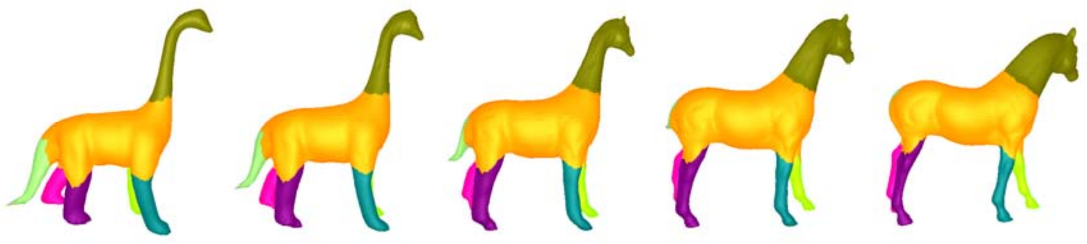
**Fig.8 Mesh interpolation between a dinosaur and a horse**

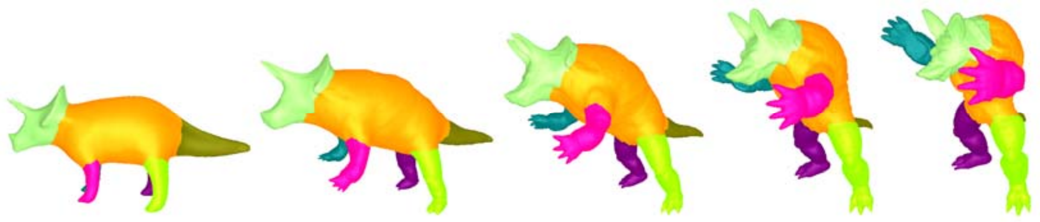
**Fig.9 Mesh interpolation between a triceratops and an armadillo**

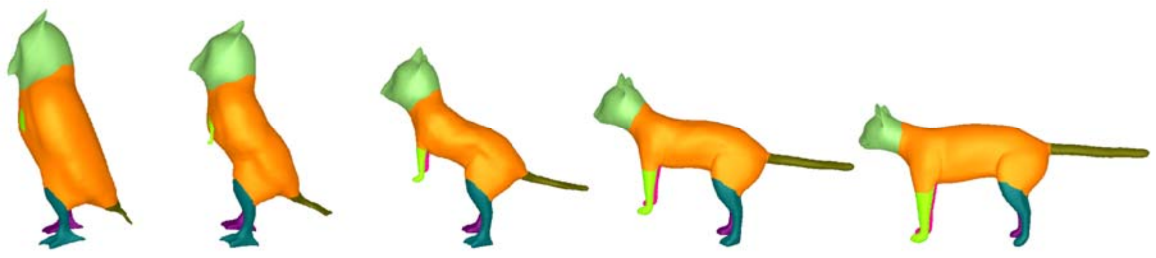
**Fig.10 Mesh interpolation between an owl and a cat**

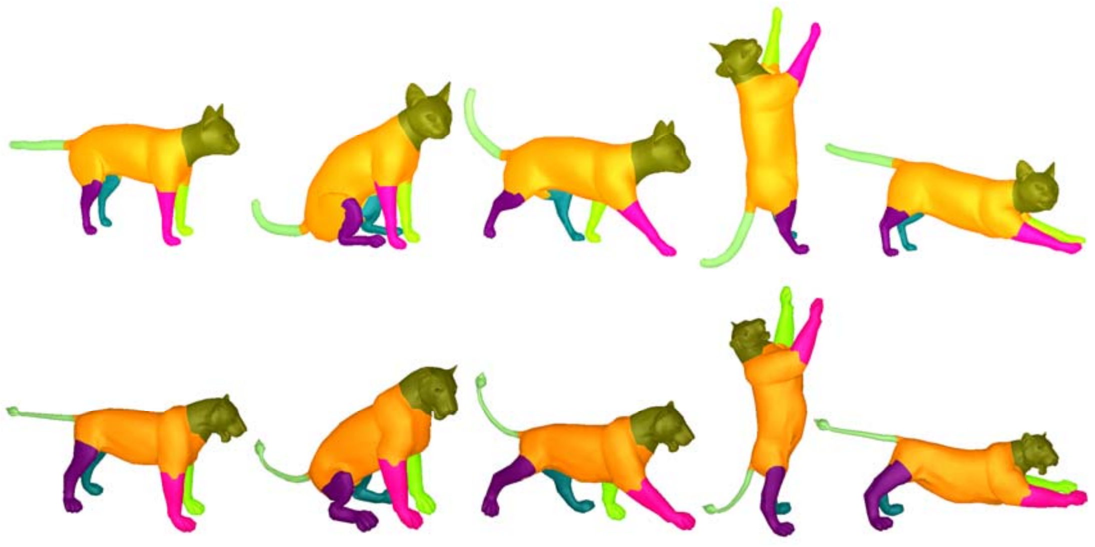

**Fig.11 Example of deformation transfer**

The deformation sequence of the cat model is given in the upper row. After building the component correspondence between the cat model and the lion model (the leftmost column), the deformation sequence of the lion model can be generated automatically

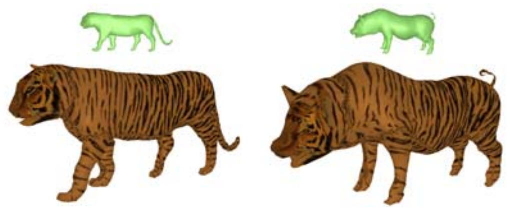

**Fig.12 Transferring the texture from the tiger model to a pig model in our system**

The vertex correspondence was automatically built in the system

## CONCLUSION

This paper presents a novel intuitive system for establishing a compatible mesh using a series of automatic mechanisms and a human interaction tool based on sketching. The system provides an automatic algorithm to reduce users' interaction as much as possible. Also, users can directly specify high-level correspondences by pairing components to represent their high-level requirements and low-level correspondences among local features within a pair of corresponding components. All these specifications can be easily done by drawing a few freehand strokes on the screen.

However, our method cannot deal with partial matching. The system may require many interactions if the input models are very different in shape and the automatic method fails. Several enhancements can be added to our algorithm. Now only genus-0 models can be handled, and we are investigating the matching scheme to deal with the case where a boundary is shared by more than one component. We can use the Gromov-Hausdorff distance (Bronstein *et al*., 2006) instead of the Euclidean distance in our graph matching step. We believe that our proposed system provides a good opportunity to use the sketching interface in complicated applications like mesh morphing.

## References

Alexa, M., 2000. Merging polyhedral shapes with scattered features. *The Vis. Comput.*, **16**(1):26-37. [doi:10.1007/PL00007211]

Allen, B., Curless, B., Popović, Z., 2003. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Trans. Graph.*, **22**(3):587-594. [doi:10.1145/882262.882311]

Bengoetxea, E., 2002. Inexact Graph Matching Using Estimation of Distribution Algorithms. PhD Thesis, University of the Basque Country, Basque Country, Spain.

Bennis, C., Vézien, J.M., Iglésias, G., 1991. Piecewise surface flattening for non-distorted texture mapping. *ACM*

*SIGGRAPH Comput. Graph.*, **25**(4):237-246. [doi:10.1145/127719.122744]

Boykov, Y.Y., Jolly, M.P., 2001. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. Eighth Int. Conf. on Computer Vision, **1**:105-112. [doi:10.1109/ICCV.2001.10011]

Bronstein, A.M., Bronstein, M.M., Kimmel, R., 2006. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *PNAS*, **103**(5):1168-1172. [doi:10.1073/pnas.0508601103]

Gregory, A., State, A., Lin, M., Manocha, D., Livingston, M., 1998. Feature-based Surface Decomposition for Correspondence and Morphing between Polyhedra. Proc. Computer Animation, p.64-71. [doi:10.1109/CA.1998.681909]

Igarashi, T., Matsuoka, S., Tanaka, H., 1999. Teddy: A Sketching Interface for 3D Freeform Design. Proc. ACM SIGGRAPH, p.409-416. [doi:10.1145/1185657.1185772]

Ji, Z., Liu, L., Chen, Z., Wang, G., 2006. Easy mesh cutting. *Comput. Graph. Forum*, **25**(3):283-291. [doi:10.1111/j.1467-8659.2006.00947.x]

Ju, T., Zhou, Q.Y., Hu, S.M., 2007. Editing the Topology of 3D Models by Sketching. *ACM Trans. Graph.*, **26**(3): Article No. 42, p.1-9. [doi:10.1145/1276377.1276430]

Kanai, T., Suzuki, H., Kimura, E., 2000. Metamorphosis of arbitrary triangular meshes. *IEEE Comput. Graph. Appl.*, **20**(2):62-75. [doi:10.1109/38.824544]

Karpenko, O.A., Hughes, J.F., 2006. SmoothSketch: 3D free-form shapes from complex sketches. *ACM Trans. Graph.*, **25**(3):589-598. [doi:10.1145/1141911.1141928]

Kho, Y., Garland, M., 2005. Sketching Mesh Deformations. Proc. ACM Symp. on Interactive 3D Graphics, p.147-154. [doi:10.1145/1053427.1053452]

Kraevoy, V., Sheffer, A., 2004. Cross-parameterization and compatible remeshing of 3D models. *ACM Trans. Graph.*, **23**(3):861-867. [doi:10.1145/1015706.1015811]

Kraevoy, V., Sheffer, A., Gotsman, C., 2003. Matchmaker: constructing constrained texture maps. *ACM Trans. Graph.*, **22**(3):326-333. [doi:10.1145/882262.882271]

Kraevoy, V., Julius, D., Sheffer, A., 2007. Model Composition from Interchangeable Components. Proc. 15th Pacific Conf. on Computer Graphics and Applications, p.129-138.

Lee, A., Dobkin, D., Sweldens, W., Schröder, P., 1999. Multiresolution Mesh Morphing. Proc. ACM SIGGRAPH, p.343-350. [doi:10.1145/311535.311586]

Lee, C.H., Varshney, A., Jacobs, D., 2005. Mesh saliency. *ACM Trans. Graph.*, **24**(3):659-666. [doi:10.1145/1073204.1073244]

Li, Y., Sun, J., Tang, C.K., Shum, H.Y., 2004. Lazy snapping. *ACM Trans. Graph.*, **23**(3):303-308. [doi:10.1145/1015706.1015719]

Maillot, J., Yahia, H., Verroust, A., 1993. Interactive Texture Mapping. Proc. 20th Annual Conf. on Computer Graphics and Interactive Techniques, p.27-34. [doi:10.1145/166117.166120]

Mateus, D., Horaud, R.P., Knossow, D., Cuzzolin, F., Boyer, E., 2008. Articulated Shape Matching Using Laplacian Eigenfunctions and Unsupervised Point Registration. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, p.1-8. [doi:10.1109/CVPR.2008.4587538]

Matsui, S., Aoki, K., Nagahashi, H., Morooka, K., 2007. Cross-parameterization for Triangular Meshes with Semantic Features. Proc. 15th Pacific Conf. on Computer Graphics and Applications, p.457-460. [doi:10.1109/PG.2007.21]

Nealen, A., Sorkine, O., Alexa, M., 2005. A sketch based interface for detail preserving mesh editing. *ACM Trans. Graph.*, **24**(3):1142-1147. [doi:10.1145/1073204.1073324]

Praun, E., Sweldens, W., Schröder, P., 2001. Consistent Mesh Parameterizations. Proc. 28th Annual Conf. on Computer Graphics and Interactive Techniques, p.179-184. [doi:10.1145/383259.383277]

Schreiner, J., Asirvatham, A., Praun, E., Hoppe, H., 2004. Inter-surface mapping. *ACM Trans. Graph.*, **23**(3):870-877. [doi:10.1145/1015706.1015812]

Shapira, L., Shamir, A., Cohen-Or, D., 2008. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Vis. Comput.*, **24**(4):249-259. [doi:10.1007/s00371-007-0197-5]

Shlafman, S., Tal, A., Katz, S., 2002. Metamorphosis of polyhedral surfaces using decomposition. *Comput. Graph. Forum*, **21**(3):219-228. [doi:10.1111/1467-8659.00581]

Sumner, R.W., Popović, J., 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, **23**(3):399-405. [doi:10.1145/1015706.1015736]

Sundar, H., Silver, D., Gagvani, N., Dcikinson, S., 2003. Skeleton Based Shape Matching and Retrieval. Proc. Shape Modeling Int., p.130-139. [doi:10.1109/SMI.2003.1199609]

Zeleznik, R., Herndon, K., Hughes, J., 1996. Sketch: An Interface for Sketching 3D Scenes. Proc. ACM SIGGRAPH, p.163-170. [doi:10.1145/1185657.1185770]

Zhao, Y., Ong, H.Y., Tan, T.S., Xiao, Y., 2003. Interactive Control of Component-based Morphing. Proc. Eurographics/SIGGRAPH Symp. on Computer Animation, p.339-348.