Interactive Mesh Fusion Based on Local 3D Metamorphosis

Takashi Kanai
Materials Fabrication Laboratory
RIKEN
kanai@postman.riken.go.jp

Hiromasa Suzuki Jun Mitani Fumihiko Kimura
Department of Precision Engineering
University of Tokyo
{*suzuki/mitani/kimura*}*@cim.pe.u-tokyo.ac.jp*

Abstract

This paper proposes a new mesh modeling scheme, called *mesh fusion*, based on three-dimensional (3D) mesh-based metamorphosis. We establish the attachment from a part of one mesh to a part of another with smooth boundaries, employing the traditional cutting and pasting operation in conjunction with a combination of meshes, applying the idea of 3D metamorphosis. We also offer an algorithm for adjusting two boundaries by using the combination of three geometrical operations *rigid transformation, scaling* and *deformation*. Our schematic offers a computation time swift enough that the user can create various shapes with interactive speed.

Key words: Geometric Modeling, Triangular Mesh, Cutting and Pasting, Metamorphosis, Shortest Path, Deformation.

1 Introduction

Models represented by triangular meshes are widely used in the Computer Graphics (CG) area. The purpose of our research is to develop a new method of generating a model from two existing models (or some portion thereof) within a frame of mesh-based geometric modeling. This paper focuses particularly on an operation, called *cutting and pasting*, in which a part of one mesh is attached to a certain region of an another. Cutting and pasting is a basic operation that is usually implemented on most two-dimensional (2D) drawing and painting software.

For 3D models, we should take into account that the smooth attachment of boundaries between a pasted model and its base is sometimes necessary. One possible way to resolve this issue is to apply some functions such as *blending* [10] along the boundary, after the union of two models is generated by *set operations* [18]. Many of the commercial 3D modeling systems for CAD/CAM or CG applications have a set operation function. However, very few modeling systems also have blending functions for arbitrary meshes.

Of course, we know that it is easy to apply these operations by using other forms such as meta-balls, soft objects, and so on [3]. In such representations, various shapes can be defined by a blend of primitives such as spheres or ellipsoids, or defined as some other function, maintaining the smooth attachment of boundaries. Unfortunately, these representations do not stand directly the mesh.

Some published research has focused on cutting and pasting of 3D models. Ranta et al. [21] propose a feature-based modeling scheme based on cutting and pasting and offer a basic algorithm for cutting and pasting between two solid primitives. Pedersen [20] offers a general method for cutting implicit surfaces. Chan et al. [4] propose a GUI-based approach for pasting a B-spline surface and for the interactive sliding of such a pasted surface on an another B-spline surface. However, these approaches are not intended to apply to arbitrary meshes.

Our scheme for cutting and pasting between two meshes, called *mesh fusion*, is a type of operation significantly different from that of previously proposed approaches. Our method is based on the idea of 3D mesh-based metamorphosis [13, 5, 6, 16, 9, 11, 12]. The basic procedure of this approach is divided into two steps. In the first step, face correspondences are established between two meshes by which each point on the face of the source mesh is mapped to a point on the face of the target mesh. This step is called *the correspondence problem*. The second step, called *the interpolation problem*, generates a smooth transition by interpolating corresponding points from the source to the target positions using those correspondences. In our scheme, we use Kanai et al.'s method [11] to address the correspondence problem.

Our major contribution is the development of several novel methods for resolving the interpolation problem. From a cutting and pasting point of view, it is necessary that shapes near the boundary gradually shift from one pasted mesh to the other, and retain, as closely as possible, their original shape in regions apart from the boundary. We resolve this issue by modifying the *Fusion Control Function* (FCF), which is defined as a parametric B-spline curve.

Further, to establish a smooth attachment of boundaries, we propose an algorithm based on three geomet-

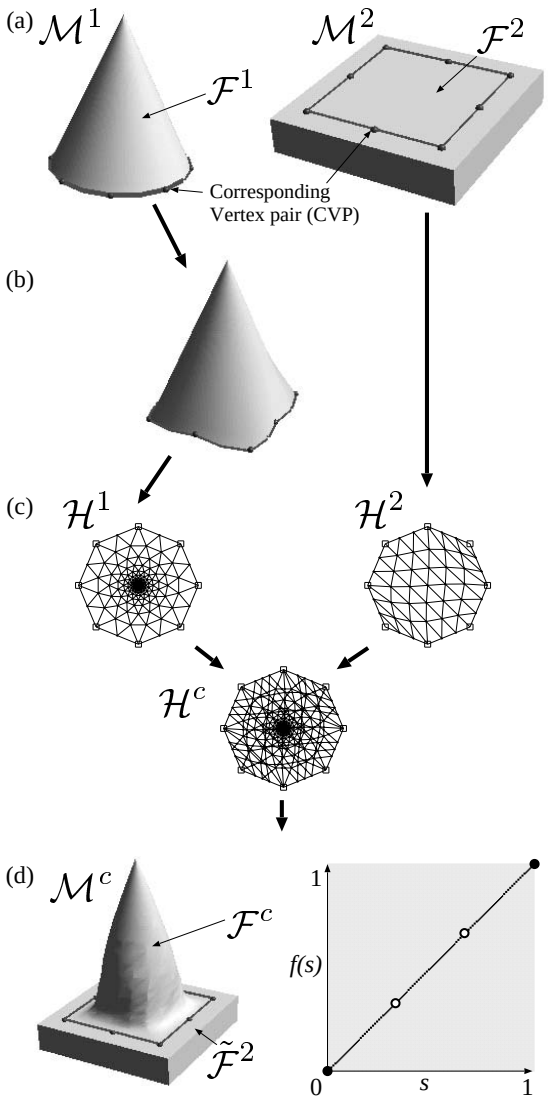


Figure 1: An Overview of mesh fusion.

rical operations, *rigid transformation, scaling* and *deformation*, for adjusting the shape of two boundaries. We can efficiently combine these three operations so that the deformation of boundaries is minimized.

It is important for the user to be able to define or modify shapes in the mesh modeling with an interactive speed. All the algorithms used or proposed in our scheme are fast enough to satisfy this criterion of interactivity of the mesh modeling.

2 Mesh fusion using 3D metamorphosis

Given two triangular meshes $\mathcal{M}^1, \mathcal{M}^2$, we define a *tile* as a sub-region of these meshes $\mathcal{F}^1 \subseteq \mathcal{M}^1, \mathcal{F}^2 \subseteq \mathcal{M}^2$. We assume that a tile is topologically equivalent to a disk. We also define $\partial\mathcal{F}^1, \partial\mathcal{F}^2$ as the boundary of $\mathcal{F}^1, \mathcal{F}^2$.

Figure 1 shows an overview of our scheme. Mesh fusion is achieved by the following four steps. First, we define tiles $\mathcal{F}^1, \mathcal{F}^2$ in two meshes $\mathcal{M}^1, \mathcal{M}^2$. These extractions are achieved by selecting the same number of vertices on each mesh. In Figure 1(a), spheres on each mesh denote some selected vertices. Second, an algorithm is applied to $\mathcal{F}^1$ so that $\partial\mathcal{F}^1$ and $\partial\mathcal{F}^2$ are fitted for smooth attachment (Figure 1(b)). We describe the relevant details in Section 3. Third, we create convex polygons in 2D space $\mathcal{H}^1, \mathcal{H}^2$ as parameterizations of $\mathcal{F}^1, \mathcal{F}^2$, a combined polygon $\mathcal{H}^c$, and then a *combined tile* $\mathcal{F}^c$ (Figure 1(c)). $\mathcal{F}^c(\mathcal{H}^c)$ is a supermesh that has the combined graph structure of $\mathcal{F}^1(\mathcal{H}^1)$ and $\mathcal{F}^2(\mathcal{H}^2)$. Finally, the final shape $\mathcal{M}^c$ is created by operating a FCF (Figure 1(d)). $\mathcal{M}^c$ is represented as the form $\mathcal{F}^c \cup \tilde{\mathcal{F}}^2$ ($\tilde{\mathcal{F}}^2 \equiv \mathcal{M}^2 \setminus \mathcal{F}^2$).

2.1 Grouping as a tile

In this subsection, we describe the details of grouping faces from a mesh $\mathcal{M}$ as a tile $\mathcal{F}$. The grouping process includes two important sub-processes: the vertex correspondence process and the cutting process.

In the vertex correspondence process, the user selects a vertex from each of two meshes respectively to make a vertex correspondence. We call each selected vertex *corresponding vertex* (CV), and call a pair of vertices *corresponding vertex pair* (CVP) (Figure 1(a)). CVP is defined for the rough specification of attaching two boundaries of tiles, and is used as the reference for methods described in Section 2.2 and Section 3. We continue these selecting operations for some CVPs so that a desired sub-region on each of two meshes is surrounded by a closed loop of CVs.

In the cutting process, a sub-region from a mesh is defined according to a closed loop. To achieve this definition, we need to *cut* a mesh by curves. A curve should be on the mesh and its end points are two neighboring CVs, in a closed loop. We regard the calculation of such a curve to constitute finding the shortest path between two vertices. However, exactly locating the shortest path on the mesh is time-consuming work, and is not necessary for our purposes [19].

Instead, we use a method similar to that of Lanthier et al. [15] for calculating a path approximate to the shortest path between two CVs on a mesh. The difference between our method and that of Lanthier et al. is that we adopt an iteration-based method. A shortest path is calculated by selective refinements of sub-graphs which are composed in part of vertices and edges in $\mathcal{M}$ along with additional vertices and edges. Finding a shortest path can be achieved solely by the search of sub-graphs and does not require numerically heavy computations such as the 3D rotations of faces.

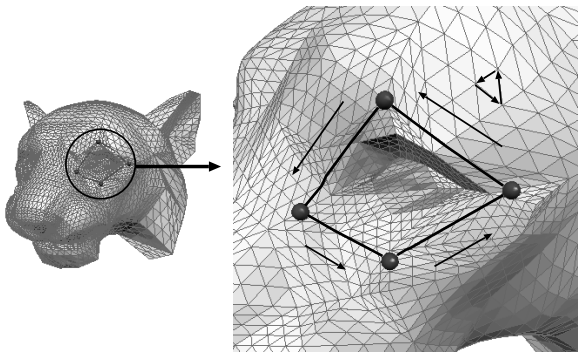
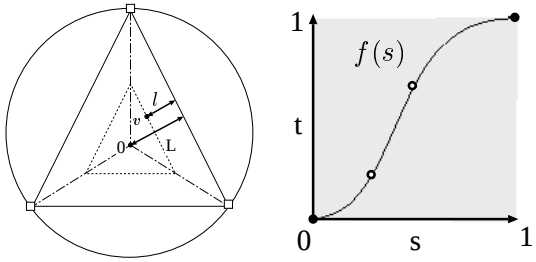


Figure 2: Selecting CVs as a counter-clockwise direction.



(a) (b)

Figure 3: Fusion control function: (a) The relationship between an embedding and a unit circle for the parameterization. (b) A non-uniform cubic B-spline function as our FCF.

Some vertices or edges of shortest paths found by an above algorithm may not be originals of $\mathcal{M}$. If an edge is not an original edge, a face under such an edge is divided. We call each edge of these shortest paths a *boundary edge*, and each vertex of paths a *boundary vertex*.

When all of the shortest paths have been calculated, a mesh is divided into two sub-regions. To determine which of the two sub-regions is a tile, we make a certain rule for selecting CVs: Let a face of $\mathcal{M}$ be an outer direction if it has a counter-clockwise cycle of vertices. We have to select CVs in a direction the same as those shown in Figure 2. With such a rule, boundary edges can all be oriented; that is, start/end vertices of an edge can be determined. Then the left faces of these oriented edges are the faces of a tile. These faces can be gathered by a greedy-like algorithm.

There are two reasons why we adopt the above grouping method. One reason is to relieve the user's task for the grouping. In our method, the user's task consists only of picking several vertices on the mesh. The other reason is that the boundaries of the tiles are smooth. Selecting all the vertices to generate a boundary, via the method of Gregory [9], is rather cumbersome work for the user. Furthermore, we note from some experiments that those boundaries may be milled, which would negatively effect or hinder the smooth attachment of boundaries described in a later section.

If a more sophisticated selection of the boundary is needed, a B-spline curve based approach for calculating paths, proposed by Krishnamurthy et al. [14], seems to be more specific than ours, although we have not implemented it. Unfortunately, a path generated by Krishnamurthy's approach does not pass on a mesh. For our purposes, a slight modification of the approach would be necessary.

2.2 Finding face correspondences

From two tiles, face correspondences are generated for the fusion. We use Kanai et al's method [11] for achieving face correspondences. We overview a central portion of this method. The basic idea is to construct face correspondences by combining two tiles in a common domain.

First, we embed each tile to a regular n-gonal region $\mathcal{H}$, named an *embedding*, lying on a unit circle in 2D space whose center is at the origin. n CVs are positioned on a circle as vertices of a regular n-gon. The rest of the vertices in $\partial\mathcal{F}$ are positioned on edges of n-gon in the same order so that the ratios of mapped edge lengths in $\partial\mathcal{H}$ are equal to those of their original edge lengths in $\partial\mathcal{F}$. Positions of internal vertices in $\mathcal{H}$ are located by an approximate solution of harmonic mapping proposed by Eck et al. [7]. Second, a *combined embedding* $\mathcal{H}^c$ is created from those two embeddings $\mathcal{H}^1, \mathcal{H}^2$. After calculating intersection points between edges of two embeddings, faces of $\mathcal{H}^c$ are constructed from original vertices, subdivided edges of $\mathcal{H}^1, \mathcal{H}^2$ and some additional vertices found by intersection computations. Finally, a *combined tile* $\mathcal{F}^c$ is created from $\mathcal{H}^c$. Each vertex in $\mathcal{F}^c$ has two 3D positions; one is on a face of $\mathcal{F}^1$, the other on a face of $\mathcal{F}^2$. A position of a vertex in $\mathcal{F}^c$ is calculated by a linear interpolation between those of two vertices.

The final shape $\mathcal{M}^c$ is a logical sum of $\mathcal{F}^c$ and $\tilde{\mathcal{F}}^2$. To join $\mathcal{F}^c$ and $\tilde{\mathcal{F}}^2$ topologically, faces in $\tilde{\mathcal{F}}^2$, neighbor to $\partial\tilde{\mathcal{F}}^2$, are divided according to the edges of $\mathcal{F}^c$.

2.3 Controlling the fusion

In this subsection, we present a new method for establishing mesh fusion. From a cutting and pasting point of view, it is desirable that shapes near the boundary of $\mathcal{F}^c$ are $\mathcal{F}^2$, while those apart from the boundary are $\mathcal{F}^1$. From a metamorphosis point of view, it is desirable that $\mathcal{F}^c$ has an intermediate shape between $\mathcal{F}^1$ and $\mathcal{F}^2$.

Now, we propose a method that meets both of these criteria. It is noted that $\mathcal{H}^c$ is mapped onto a circle. Based on this property, we define a *fusion control function* (FCF) that can change the ratio of the fusion between two positions of each vertex in $\mathcal{F}^c$.

Figure 3(a) shows the relationship between a unit circle and $\mathcal{H}^c$. Let $s = 1 - l/L$ be a parameter of a vertex v in $\mathcal{H}^c$, where L denotes the minimum distance from the origin of a circle to the boundary (one edge of a regular n-gon), and where l denotes the minimum distance from v to the boundary. s has a value between 0 and 1. s of vertices near the boundary approach 0, and s of vertices near the origin approach 1. s of all vertices are calculated only once, as soon as $\mathcal{H}^c$ is created.

Figure 3(b) shows a FCF $f(s)$. FCF is defined as a non-uniform cubic B-spline curve interpolating four points in 2D space. We define that these points are free to move under the following conditions: (1) Each of the two end points is constrained as $f(s) = 0, f(s) = 1$. (2) Each interval at the s axis between two neighboring points is not less than 0. (3) $f(s)$ is constrained as $0 \leq f(s) \leq 1$. It is not always necessary for a FCF to be monotonic.

Let $\mathbf{v}^c$ be a 3D position of a vertex v in $\mathcal{F}^c$, and let $\mathbf{v}^1, \mathbf{v}^2$ be positions of corresponding vertices on $\mathcal{F}^1, \mathcal{F}^2$, respectively. Then $\mathbf{v}^c$ is calculated as the following equation:

$$\mathbf{v}^c = f(s)\mathbf{v}^1 + (1 - f(s))\mathbf{v}^2. \quad (1)$$

It seems possible to define other functions that satisfy these three conditions, though we have defined a FCF as described above. Particularly on B-spline based curve definitions, if a further detailed specification of $\mathcal{F}^c$ is needed, it is possible to add free points to our FCF.

Figure 4 demonstrates a simple example as a guideline for using a FCF. In this figure, a part of a cone is attached to a part of a sphere, where both boundaries have the same circle shape. Figure 4(a) shows the result with the case that FCF is a linear function. $\mathcal{F}^c$ gradually shifts from $\mathcal{F}^2$ to $\mathcal{F}^1$ with this setting. Figure 4(b) shows the result with $f(s) \approx 1$ except near $s = 0$. Cutting and pasting can be achieved by this setting. Figure 4(c) shows the result with $f'(s) \approx 0$. Shapes near the boundary of $\mathcal{F}^c$ are those of $\mathcal{F}^2$ with this setting. Figure 4 (d) shows the result with $f(s) = 0.5$, which a intermediate shape between $\mathcal{F}^1$ and $\mathcal{F}^2$ is generated in all regions of $\mathcal{F}^c$.

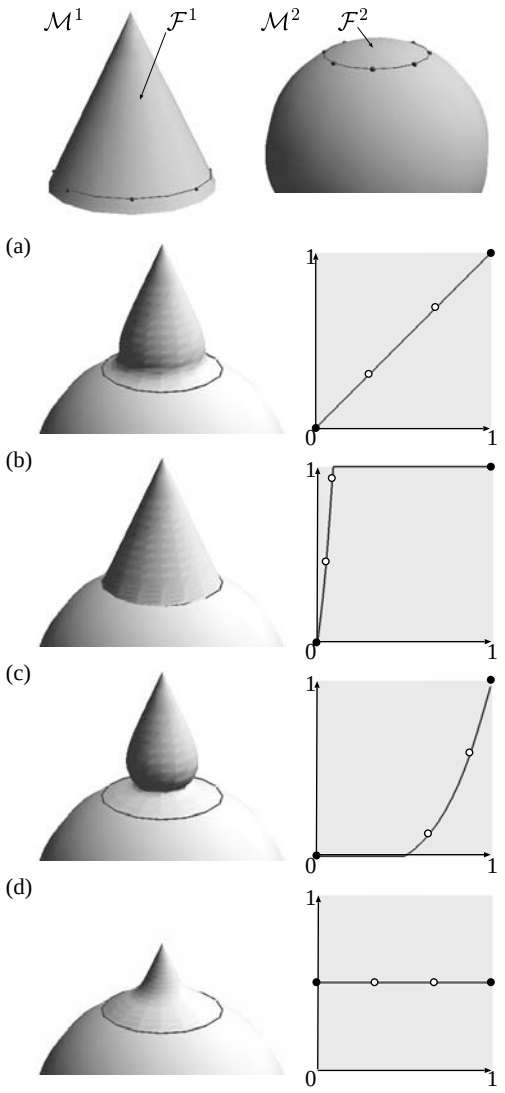


Figure 4: Various shapes using a FCF.

3 The adjustment of boundaries for the fusion

By the method described thus far, two boundaries $\partial\mathcal{F}^c, \partial\tilde{\mathcal{F}}^2$ are joined *topologically*. Here, we want to attach two boundaries with different shapes for the general case. In this section, we describe three geometrical operations and propose an algorithm for the adjustment of two boundaries *geometrically*. In each operation, we utilize CVP for the reference of the adjustment because the use of $\partial\mathcal{F}$ itself (that is, the use of boundary edges in $\partial\mathcal{F}$) requires time consuming computations.

Two important conditions are necessary for joining two boundaries *smoothly*. First, the shapes of two boundaries are coincident; second, two tangent directions normal to each boundary are coincident. We deal with the former issue in this section. The latter issue is discussed in Section 5.

3.1 Rigid Transformation

For the adjustment without changing an original shape, we use a rigid transformation so that $\partial\mathcal{F}^1$ adjusts to $\partial\mathcal{F}^2$. Let $\mathbf{p}_i^1, \mathbf{p}_i^2 (i = 1 \ldots n)$ be each position of CVP in $\mathcal{M}^1, \mathcal{M}^2$. A rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{c}$ are found by minimizing the following energy function:

$$E_{trans} = \sum_i^n \left| \mathbf{p}_i^2 - (\mathbf{R}\mathbf{p}_i^1 + \mathbf{c}) \right|^2. \quad (2)$$

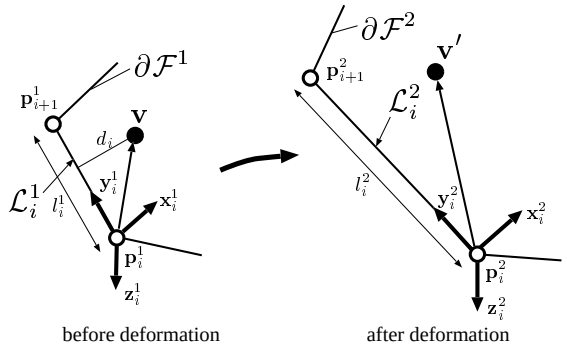


Figure 5: The deformation of a vertex in $\mathcal{F}^1$.

We solve this least square problem by using an algorithm based on SVD (Singular Value Decomposition) [1].

3.2 Scaling

For an adjustment corresponding to a minor change in an original shape, we use a scale transformation. A scale value k is found by minimizing the following energy function:

$$E_{scale} = \sum_{i}^{n} \left|\mathbf{p}^2_i - k\mathbf{p}^1_i\right|^2. \tag{3}$$

We solve an equation $\partial E_{scale}/\partial k = 0$ for obtaining k.

3.3 Deformation

For an adjustment corresponding to a more significant change in an original shape, we use a deformation. It is desirable here that $\partial\mathcal{F}^1$ is deformed so that $\mathbf{p}^1_i$ is coincident to $\mathbf{p}^2_i$. We use a line-based deformation approach similar to 2D image or 3D volumetric deformation approaches described in a previous study by Beier et al. and Lerios et al. [2, 17]. With this technique, we define a potential field based on the mapping from $\partial\mathcal{F}^1$ to $\partial\mathcal{F}^2$. Under this field, all vertices in $\mathcal{F}^1$ are deformed.

Figure 5 shows our deformation approach for a vertex in $\mathcal{F}^1$. Before the deformation, we define a line $\mathcal{L}_i$ between two neighboring CVs, $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ (bold lines in Figure 5). Let $\mathbf{v}$ be an original position of a deformed vertex. First, $\mathbf{v}$ is transformed to $\mathbf{u}$, a position on the local coordinate system $(\mathbf{x}^1_i, \mathbf{y}^1_i, \mathbf{z}^1_i)$ defined in $\mathcal{L}^1_i$. $\mathbf{u}$ is scaled to $\mathbf{u}'$ by the ratio of lengths of two corresponding lines, $\mathcal{L}^1_i$ and $\mathcal{L}^2_i$. Then, a position $\mathbf{v}'_i$ of the deformation at each line is calculated by the inverse transformation to the world coordinate system. A final position $\mathbf{v}'$ is calculated by a weighted sum of these positions. These steps can be summarized by the following equations:

$$\mathbf{u}_i = \begin{pmatrix} \mathbf{x}^1_i & \mathbf{y}^1_i & \mathbf{z}^1_i \end{pmatrix}^{-1} (\mathbf{v} - \mathbf{p}^1_i), \tag{4}$$

$$\mathbf{u}'_i = \frac{l^2_i}{l^1_i}\mathbf{u}_i, \tag{5}$$

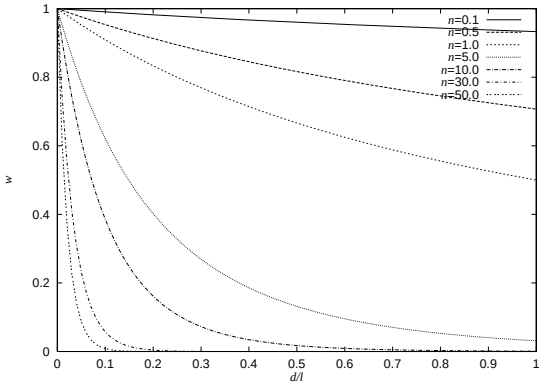


Figure 6: A weight function w_i with various values of n.

$$\mathbf{v}'_i = \begin{pmatrix} \mathbf{x}^2_i & \mathbf{y}^2_i & \mathbf{z}^2_i \end{pmatrix} \mathbf{u}'_i + \mathbf{p}^2_i, \tag{6}$$

$$\mathbf{v}' = \frac{1}{\sum_i w_i} \sum_i w_i \mathbf{v}'_i, \tag{7}$$

where l^1_i, l^2_i denote each length of $\mathcal{L}^1_i, \mathcal{L}^2_i$.

The local coordinate system of each line $\mathcal{L}_i$ is defined using the following steps. First, we define a starting point $\mathbf{p}_i$ of $\mathcal{L}_i$ as the origin, and define a unit vector from $\mathbf{p}_i$ to $\mathbf{p}_{i+1}$ as $\mathbf{y}_i$. Next, to make an orthogonal coordinate system, $\mathbf{z}_i$ is created by a cross product of a unit vector from a barycentric of CVs to $\mathbf{p}_i$ and $\mathbf{y}_i$. Finally, $\mathbf{x}_i$ is created by a cross product of $\mathbf{y}_i$ and $\mathbf{z}_i$.

A weight function w_i, which decides the effect of each line $\mathcal{L}_i$ in the deformation, is defined as follows:

$$w_i = \left(1 + \frac{d_i}{l^1_i}\right)^{-n}, \tag{8}$$

where d_i denotes the distance between $\mathbf{v}$ and $\mathcal{L}^1_i$. w_i is defined so that the larger line or the shorter d_i causes a larger value. Figure 6 shows graphs of a function w_i with various values of n. From these graphs, we can derive that the larger values of n tend to have a more local effect for $\mathbf{v}$; that is, w_i is rapidly decreased with the larger d_i/l^1_i. We select $n = 5$ from our various experiments.

3.4 An algorithm for adjusting boundaries

We apply the following algorithm to $\mathcal{F}^1$ before finding face correspondences described in Section 2.2:

STEP1 Apply a rigid transformation once. If an average of the distance between a transformed CV in $\mathcal{F}^1$ and a corresponding CV in $\mathcal{F}^2$ is within a threshold ϵ, then go to **STEP4**.

STEP2 Apply a scaling and a rigid transformation in order. If an average distance as calculated in **STEP1** is within ϵ, then go to **STEP4**. If $k \approx 1$, then go to **STEP3**. Otherwise, repeat **STEP2**.

STEP3 Apply a deformation.

STEP4 Terminate.

A basic concept of this algorithm is "we wish to deform an original shape as little as possible". **STEP1** is applied first for the adjustment of two boundaries that have the same shape and size. If two boundaries are adjusted in this step, the remaining steps do not need to be applied. **STEP2** is applied for the adjustment of two boundaries that have the same shape but different sizes. One problem arises that this step can be successfully finished, because this step is an iterative step. In all of our examples, it can be iterated within five repetitions. **STEP3** is applied only once so that the positions of the corresponding two CVs are coincident. We found that the amount of the deformation of a tile is slight, minimized by the previous steps.

It is very important to note that the above algorithm can play a role only when there is little modification of shapes near the boundary. In the case that two tiles have quite different boundary shapes, and thus a rather large deformation of a tile is needed, it is not appropriate to apply this algorithm. If we set any value to n in Equation (8), it is difficult to achieve total control of the shape by using our deformation approach, though it is simple enough to adjust shapes near the boundary. In that case, the effect of the deformation of lines $\mathcal{L}$ would not reach those shapes apart from the boundary when n has the larger value, and the original shape of a tile would in large degree be lost when n has the smaller value.

Our current solution for this problem is to bend a tile $\mathcal{F}^1$ dramatically so that the boundary of $\mathcal{F}^1$ is approximately along that of $\mathcal{F}^2$, using a method such as FFD (Free-Form Deformation) [22]. In this bending process, it is not necessary to adjust shapes or sizes of two boundaries exactly, because the above algorithm can accomplish this without such troublesome work.

Figure 7 demonstrates these steps of our algorithm for the adjustment of two boundaries. $\mathcal{F}^1$ involves a set of solid characters which is attached to a bent solid plate. $\mathcal{F}^2$ is a rectangular region across one of side edges of a "Bottle" model. Both two boundary shapes are rectangles, but their aspect ratios are different. Figure 7(b) shows the result after **STEP1** and **STEP2**. Figure 7(c) shows the result of the fusion without applying **STEP3**. In this figure, we can find that the shape near the boundary has strong roughness. This is because that two boundaries are not adjusted geometrically while these are joined topologically. Figure 7(d) shows the result of the fusion with applying **STEP3**. We can observe that a set of solid characters is stretched so as to match the boundary of a pasting region of the bottle, and that two boundaries are adjusted with geometrical smoothness as a desirable effect by the deformation.

	Short. Path (Sec.)		Adj. (Sec.)	Corres. (Sec.)
	$\mathcal{M}^1$	$\mathcal{M}^2$		
Fig. 7	2.95	2.77	0.19	0.75
Fig. 8(b)-(c)	0.45	7.05	0.08	0.87
Fig. 8(d)-(e)	0.33	7.05	0.05	0.53
Fig. 9(a)	3.67	0.87	1.03	0.52
Fig. 9(b)	2.32	2.93	0.63	3.53

Table 1: Computational times taken in our examples.

4 Implementation

We have implemented a prototype system on a graphics workstation OCTANE SI (MIPS R10000 175MHz CPU). Our system prepares two display modes. The user can change each mode by manipulating a toggle switch. One is for the extraction of tiles, which have two screens: one displays $\mathcal{M}^1$, the other displays $\mathcal{M}^2$. In this mode, the user can select vertices for creating CVs, and can modify CVs by using a mouse. The other is for the control of a FCF, which also has two screens: one displays $\mathcal{M}^c$, the other displays a FCF. If a large deformation of $\mathcal{F}^1$ is needed, $\mathcal{M}^1$ is bent in advance using a FFD operation on a commercial modeling system (for example, "Deforming with Lattices" operation on SOFTIMAGE 3D$^{\mathrm{TM}}$).

5 Results and discussion

Figure 8 demonstrates two mesh fusion examples. One is a fusion between a region defined at a nose of a "mannequin" model and a cone (8(b)-8(c)). The other is a fusion between the same region of a mannequin and a hemisphere (8(d)-8(e)). In this figure, a FCF in Figure 4(b) is used in 8(b) and 8(d), a FCF in 4(c) is used in 8(c), and a FCF at 4(d) is used in 8(e). Each nose in 8(b) and 8(d) is different from its original cone or hemisphere, because these original shapes are deformed for the adjustment of boundaries. It is noted that the shape of a cone only appears in the top of a nose as shown in 8(c). This is because that the tangent normal to the boundary of an original nose ($\mathcal{F}^2$) has a large effect to the result.

Figure 9 shows two examples of a cutting and pasting. Figure 9(a) shows an example which the shape near the front of a "Delorian" model has been replaced to that of a "Volkswagen" model. Figure 9(b) shows an example which the front half of a "Helicopter" model has been replaced by the head of an "Eagle" model. In both examples, a FCF at Figure 4(b) has been used.

Table 1 shows computation times of the examples described in Figure 7-9 which is divided into three terms: the calculation of all approximate shortest paths per each mesh (Short. Path), the adjustment of two boundaries (Adj.) and the search of face correspondences (Corres.) .

	CVPs	$\mathcal{M}^1$		$\mathcal{M}^2$		$\mathcal{F}^1$ ($\mathcal{H}^1$)		$\mathcal{F}^2$ ($\mathcal{H}^2$)		$\mathcal{F}^c$ ($\mathcal{H}^c$)		$\mathcal{M}^c$	
		v	f	v	f	v	f	v	f	v	f	v	f
Fig. 7	6	1,293	2,582	3,589	7,084	875	1,677	55	74	1,835	3,569	5,141	10,244
Fig. 8(b)-(c)	8	434	864	6,769	13,472	393	760	404	721	2,072	4,041	8,592	16,906
Fig. 8(d)-(e)	8	242	480	6,769	13,472	121	216	452	805	1,682	3,249	8,182	16,059
Fig. 9(a)	7	6,775	13,502	1,300	2,580	521	1,003	62	103	1,502	2,961	2,759	5,470
Fig. 9(b)	5	5,625	11,167	5,188	10,332	1,541	3,045	1,158	2,244	8,325	16,548	12,425	24,666

Table 2: The number of vertices v, faces f, and CVPs in our examples.

Although the time taken about the generation of CVPs by the user is not included in this table, it is the work only picking several vertices, then it takes only two or three minutes in each example. Table 2 shows the number of vertices and faces in $\mathcal{M}^1$, $\mathcal{M}^2$, $\mathcal{F}^1$, $\mathcal{F}^2$, $\mathcal{F}^c$, $\mathcal{M}^c$, and the number of CVPs.

It can be shown in Table 1 that each algorithm described in this paper is accomplished with alacrity, and the user can modify $\mathcal{F}$ interactively. Although modification of a CVP requires re-calculation of all of the algorithms in our system, these operations can be carried out without time consuming work. Each time a free point of a FCF on the graph is moved, our system calculates the positions of all vertices in $\mathcal{F}^c$ using Equation (1). The calculation is fast enough to modify a FCF and to visualize the result of $\mathcal{F}^c$ interactively.

One merit of this system is that the user need not be vigilant about the smoothness of boundaries. In particular, the deformation process described in Section 3.3 does not require that equations for the continuity of the tangent direction normal to boundaries, such as the cross boundary derivatives of free-form surfaces, be included in Equation (4)-(7); the smoothness can be controlled by a FCF. For example, the tangent direction normal to $\partial\tilde{\mathcal{F}}^2$ is reflected to that of $\partial\mathcal{F}^c$ if a FCF is designed so as to be $f'(0) \approx 0$ as shown in Figure 4(c). However, such the smoothness is not always required, as shown in Figure 4(b). Our FCF can control various kinds of attachments.

We have proposed combine-based approach to uniformly treat the fusion of meshes which includes the cutting and pasting operation. Because of this, it is potentially problematic that the number of faces in $\mathcal{F}^c$ is dramatically increased. This results in the bad triangulation of the combined region of the final result, and hence the deterioration of its rendering quality can occur. For only the purpose of cutting and pasting, it is clear that the number of faces need not be increased anywhere other than near the boundaries.

One measure that might be taken is to apply a mesh simplification approach such as [8] after a FCF is modified. However, it seems that a direct display of such large meshes is more effective for purposes of interactivity, and the addition of a simplification approach comes at a rather high computational cost. If we apply a mesh simplification, it is desirable to apply a fast method to be in keeping with the goal of interactive performance; a method capable of establishing a local control of simplifications according to the values of a FCF.

6 Conclusion and Future Work

We have introduced a new mesh modeling scheme based on local 3D metamorphosis. It establishes not only a cutting and pasting operation with the smooth attachment of boundaries for two meshes, but combines the meshes with each other. To establish smooth attachment, we have also presented an algorithm for the adjustment between the boundaries of two meshes. We have demonstrated through use of examples, and have shown that each algorithm is fast enough to generate various shapes by modifying CVPs or a FCF interactively.

Our scheme allows users to easily establish a cutting and pasting operation. We are very interested in an extension of our scheme that would exploit this new-found ease, and encourage applications to computer-aided exploration of design possibilities in the service of industrial design.

7 Acknowledgement

The "mannequin" model is from Graphics and Imaging Laboratory, University of Washington. The "Delorian", "Tiger", "Bottle", "Volkswagen", "Helicopter" and "Eagle" models are courtesy of Viewpoint DataLabs. We wish to thank Prof. Daniel Cohen-Or, Tel-Aviv University, for making us aware of references regarding a SVD based rigid transformation algorithm.

A portion of this research was supported by The Ookawa Foundation for Information and Telecommunication.

8 References

[1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, Sept. 1987.

[2] T. Beier and S. Neely. Feature-based image metamorphosis. In *Computer Graphics (Proc. SIG-*

GRAPH 92), pages 35–42. ACM Press, New York, 1992.

[3] J. Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers, San Francisco, Calif., 1997.

[4] L. K. Y. Chan, S. Mann, and R. Bartels. World space surface pasting. In *Proc. Graphics Interface '97*, pages 146–154. Morgan Kaufmann Publishers, San Francisco, Calif., May 1997.

[5] D. T. Chen, A. State, and D. Banks. Interactive shape metamorphosis. In *Proc. 1995 Sympo. on Interactive 3D Graphics*, pages 43–44. ACM Press, New York, 1995.

[6] D. DeCarlo and J. Gallier. Topological evolution of surfaces. In *Proc. Graphics Interface '96*, pages 194–203. Morgan Kaufmann Publishers, San Francisco, Calif., May 1996.

[7] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Computer Graphics (Proc. SIGGRAPH 95)*, pages 173–182. ACM Press, New York, 1995.

[8] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Computer Graphics (Proc. SIGGRAPH 97)*, pages 209–216. ACM Press, New York, 1997.

[9] A. Gregory, A. State, M. Lin, D. Manocha, and M. Livingston. Feature-based surface decomposition for correspondence and morphing between polyhedra. In *Proc. Computer Animation '98*, pages 64–71. IEEE CS Press, Los Alamitos, Calif., June 1998.

[10] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A K Peters, Natick, Massachusetts, 1993.

[11] T. Kanai, H. Suzuki, and F. Kimura. Three-dimensional geometric metamorphosis based on harmonic maps. *The Visual Computer*, 14(4):166–176, 1998.

[12] T. Kanai, H. Suzuki, and F. Kimura. Metamorphosis of arbitrary triangular meshes with user-specified correspondence. *IEEE Computer Graphics and Applications*, 1999. to appear.

[13] J. R. Kent, W. E. Carlson, and R. E. Parent. Shape transformation for polyhedral objects. In *Computer Graphics (Proc. SIGGRAPH 92)*, volume 26, pages 47–54. ACM Press, New York, 1992.

[14] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Computer Graphics (Proc. SIGGRAPH 96)*, pages 313–324. ACM Press, New York, 1996.

[15] M. Lanthier, A. Maheshwari, and J.-R. Sack. Approximating weighted shortest paths on polyhedral surfaces. In *Proc. 13th ACM Sympo. on Computational Geometry*, pages 274–283. ACM Press, New York, June 1997.

[16] F. Lazarus and A. Verroust. Metamorphosis of cylinder-like objects. *J. Visualization and Computer Animation*, 8(3):131–146, July–Sept. 1997.

[17] A. Lerios, C. D. Garfinkle, and M. Levoy. Feature-Based volume metamorphosis. In *Computer Graphics (Proc. SIGGRAPH 95)*, pages 449–456. ACM Press, New York, 1995.

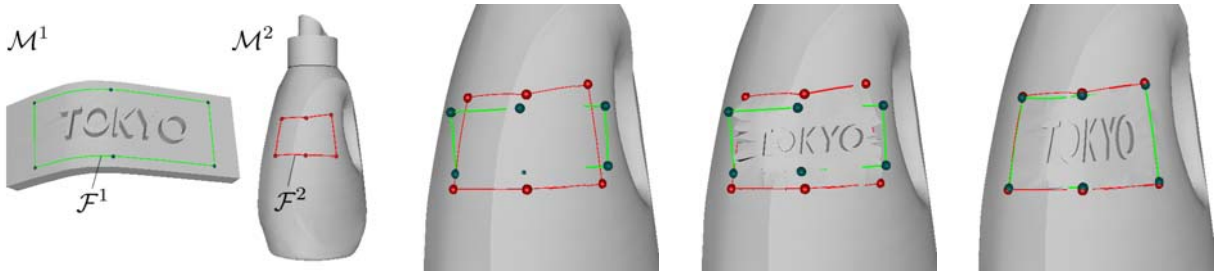
[18] M. Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, Maryland, 1988.

[19] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM J. Computing*, 16(4):647–668, 1987.

[20] H. K. Pedersen. Decorating implicit surfaces. In *Computer Graphics (Proc. SIGGRAPH 95)*, pages 291–300. ACM Press, New York, 1995.

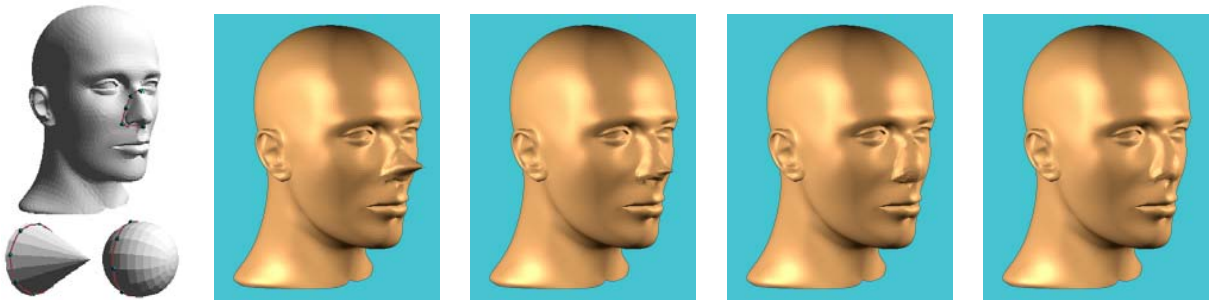
[21] M. Ranta, M. Inui, F. Kimura, and M. Mäntylä. Cut and paste based modeling with boundary features. In *Proc. 2nd Sympo. on Solid Modeling and Applications*, pages 303–312. ACM Press, New York, 1993.

[22] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Computer Graphics (Proc. SIGGRAPH 86)*, pages 151–160. ACM Press, New York, 1986.



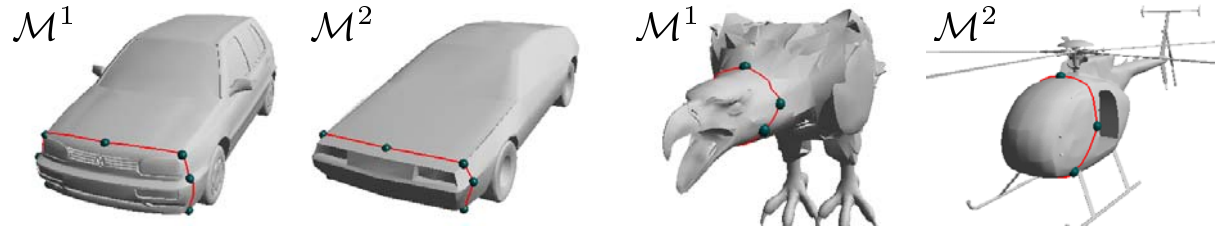
(a) (b) (c) (d)

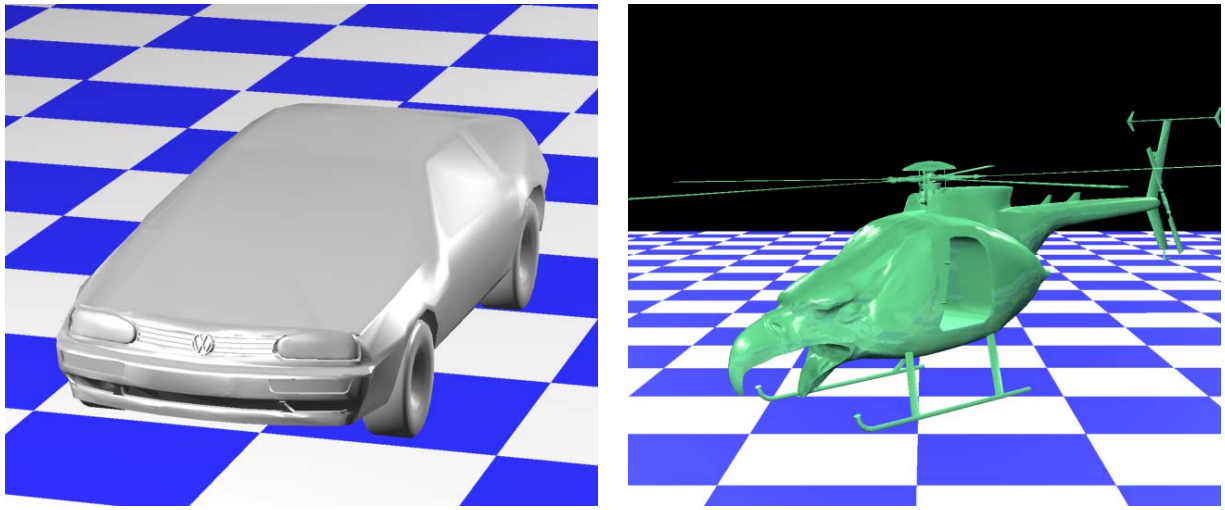
Figure 7: The adjustment of two boundaries using our algorithm.



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

Figure 8: Various fusion results about a nose of “mannequin”.





(a) (b)

Figure 9: Fusion results as the example of cutting and pasting: (a) “Delowagen”. (b) “Washicopter” (“Washi” in Japanese means “Eagle”).