Multiresolution Interpolation Meshes

Takashi Michikawa* Takashi Kanai Masahiro Fujita Hiroaki Chiyokura
Keio University, Faculty of Environmental Information
Endo 5322, Fujisawa, Kanagawa, 252-8520, JAPAN
E-mail: *{miti, kanai, t98821mf, chiyo}@sfc.keio.ac.jp*
http://graphics.sfc.keio.ac.jp/MIMesh/

Abstract

In this paper, we propose a new multiresolution-based shape representation for 3D mesh morphing. Our approach does not use combination operations that caused some serious problems in the previous approaches for mesh morphing. Therefore, we can calculate a hierarchical interpolation mesh robustly using two types of subdivision fitting schemes. Our new representation has a hierarchical semi-regular mesh structure based on subdivision connectivity. This leads to various advantages including efficient data storage, and easy acquisition of an interpolation mesh with arbitrary subdivision level. We also demonstrate several new features for 3D morphing using multiresolution interpolation meshes.

Keywords*: 3D mesh morphing, interpolation mesh, multiresolution representation, subdivision connectivity, normal map, multiresolution editing.*

1. Introduction

Three-dimensional (3D) morphing (or metamorphosis), which establishes the smooth transition between two or more existing 3D objects, is now one of the major research topics in computer graphics and its applications. In this paper, we focus on mesh morphing, that is, morphing between two or more triangular polygon-based objects.

Mesh morphing techniques may be seen as a two-step process that involves finding one-to-one correspondence between two or more meshes, and defining interpolation paths for each pair of corresponding points on the meshes to calculate in-between shapes. Most researches in mesh morphing [16, 1, 10, 23, 14] construct an *interpolation mesh* to establish the correspondence in the first step. It has an ordinary vertex/edge/face connectivity of a mesh and two or more 3D positions at each vertex.

*Now at Oracle Corp. Japan.

The primary issue that should be considered is that the construction of such an interpolation mesh is based on combination operations of both connectivity and geometry between multiple meshes. Connectivity combination operations, in particular, cause some problems. First, it requires several unstable numerical calculations. Second, it typically produces a mesh with a larger number of faces than an original mesh. Third, it has an irregular connectivity including sharp, long and narrow triangles. Hence, its rendering quality is poor. Another issue is the difficulty of interpolation control using such an interpolation mesh.

In this paper, we address the above problems by proposing a *multiresolution interpolation mesh*, an interpolation mesh with multiresolution representation. For constructing such a mesh, we do not need to use combination operations between two meshes. The basic idea for the construction is to apply subdivision fitting processes to create an interpolation mesh that approximates multiple original meshes. The use of a multiresolution interpolation mesh has the following advantages: First, it has a semi-regular mesh structure obtained by repetitive 4-to-1 splits from a base interpolation mesh; thus, it is possible to store data efficiently. Second, it has a coarse-to-fine hierarchical structure; thus, an interpolation mesh with arbitrary subdivision level can be extracted easily. This is especially helpful for the rendering process. Third, we can keep the number of faces the same extent as that of an original mesh using a local subdivision fitting scheme. Moreover, we show that various new types of interpolations, for example, normal map morphing, interpolation path editing and multi-target morphing, can be established by using multiresolution interpolation meshes.

2. Related Work

The survey of 3D morphing can be referred to in [21, 9]. Approaches for establishing 3D morphing are classified into two categories: volumetric morphing and mesh morphing.

Mesh morphing to transform the source mesh to the target involves two problem steps. The first step is to establish a correspondence from each point of the source mesh to a point of the target. Using this correspondence, the next step creates a series of intermediate objects by interpolating corresponding points from the source positions to the target positions. These steps are called *correspondence problem* and *interpolation problem*, respectively [16].

Most approaches that address the correspondence problem use combination operations to establish face correspondences between two given meshes. In combination operations, we usually construct an interpolation mesh that is represented by a logical sum of two meshes. It involves the connectivity of the two meshes. Each vertex of the interpolation mesh has two 3D positions.

The construction of the interpolation mesh is based on the mapping from given meshes or a part of these to a common reference shape. Combination operations are actually performed on these shapes. For example, Kent et al. [16] and Alexa [1] used a sphere to project two polyhedral shapes. Lazarus and Verroust [20] introduce skeletons for cylinder-like objects. Kanai et al. [13], Gregory et al. [10] and Zöckler et al. [28] utilized mesh parameterization techniques such as harmonic mapping [6] to imbed a certain region of a mesh to a two-dimensional (2D) convex polygon.

However, the combination operation for constructing the interpolation mesh has some problems. Firstly, it produces a mesh that has a large number of faces. In [14], the number of faces becomes roughly 2.5 to 10 times larger than that of an original mesh. Secondly, it requires several numerical computations including intersections of line or curve segments. As a result, we need to design or choose algorithms to prevent computation failure and to keep the consistent connectivity of the combined mesh. Clearly, combination operations between meshes with a larger number of faces are more difficult to perform. Although Lee et al. [23] have developed an efficient and robust algorithm for the combination operation using MAPS [22], it still remains a problem that the combined mesh has a large number of faces.

Some issues need to be resolved when a combined interpolation mesh is used for the interpolation problem. One is that its connectivity usually becomes irregular including many sharp, long and narrow triangular polygons. This has a strongly harmful influence on the rendering of in-between shapes. Another issue is that an interpolation mesh has an "ordinary" mesh structure. Thus interpolation control per vertex is difficult. In some other researches, interpolation control [15, 2] per sub-region of a mesh has been done. However, more sophisticated interpolation control of each vertex in a sub-region is also difficult.

As mentioned above, it is difficult to control shape transformation using an interpolation mesh created by the combination operation based approaches. This is because these approaches cannot always be designed for the interpolation. It should be noted that the interpolation mesh itself is an important data structure to establish real-time shape deformations. Recently, activities for the standardization have been initiated. Alexa et al. [3] proposed "The Morph Node" as an extension node of VRML. XVL data format by Lattice Technology Inc. [26] also prepared a similar node. Furthermore, the recently developed graphics processor unit GeForce3 by nVIDIA Inc. has the hardware support for shape deformation in the form of an interpolation mesh by using one of the new functions, the *vertex shader*.

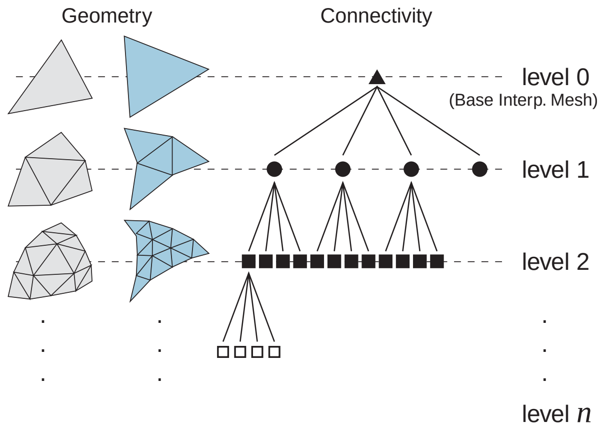


Figure 1. Quad-tree structure of MIMesh.

More recently, Ohbuchi et al. [24] proposed an interpolation control of 3D morphing based on multiresolution representation. Unfortunately, intermediate shapes must be extracted from a shape in four-dimensional space; thus, they do not take the form of an interpolation mesh.

3. Multiresolution Interpolation Mesh

Interpolation mesh has the same shape representation as an ordinary mesh that is composed of a vertex/edge/face graph structure. The difference is that each vertex has several 3D positions, attributes (colors, normals, texture coordinates and so on) and, if needed, interpolation paths between these positions. If you use these paths, 3D morphing can be carried out by shifting the 3D position of each vertex along a path, without changing its connectivity.

The *multiresolution interpolation mesh* (MIMesh) proposed here is a multiresolution version of the above interpolation mesh. It has a semi-regular mesh structure defined by regularly subdivided faces from a base mesh (named *base interpolation mesh*). We use 4-to-1 split to subdivide a face into sub-faces; thus, our MIMesh has subdivision connectivity. Faces of MIMesh are stored in a quad-tree data structure, as shown in Figure 1. In this structure, faces of the base interpolation mesh are stored in the root node. Each node has links to four child nodes, and each child node stores one of four sub-faces. We can easily obtain the interpolation

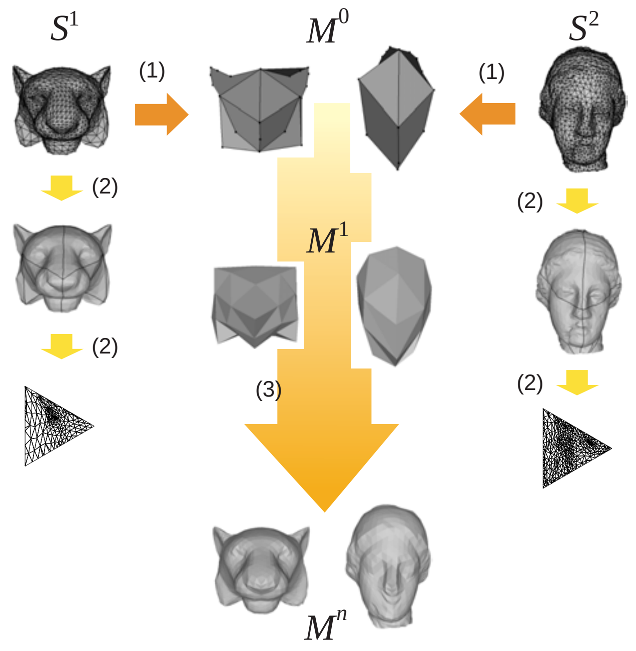


Figure 2. Overall framework of our MIMesh construction approach: (1) A base interpolation mesh M^0 creation from input meshes S^1, S^2. (2) Mesh partition and parameterization. (3) Subdivision fitting to construct hierarchical interpolation meshes $M^1, \ldots, M^n$.

mesh at an arbitrary subdivision level by traversing such a quad-tree structure. Other elements such as vertices, several 3D positions, attributes and interpolation paths are stored in the array structure.

Figure 2 illustrates an overview of our MIMesh construction approach. Our approach can be naturally extended to the case of more than three meshes, but we will discuss here the simple case in which the number of input meshes is two. The basic procedure of our construction approach is divided into the following three steps: (1) A base interpolation mesh M^0 is manually created by the user from input meshes S^1 and S^2 (Section 3.1). (2) Each input mesh is partitioned into several regions (we call them *patches* later) according to the faces of the base interpolation mesh. To each patch we apply mesh parameterization to assign a 2D parameter value to each vertex of input meshes (Section 3.2). (3) We apply a subdivision fitting algorithm to create hierarchical interpolation meshes $M^1 \ldots M^n$, where n denotes a subdivision level (Section 3.3).

Our MIMesh construction approach is strongly inspired by the remeshing algorithm of Guskov et al. [11]. The difference between [11] and ours is that our approach is designed for fitting two or more input meshes simultaneously. In addition, the first two steps of the construction process discussed later are similar to the approach of [14].

More recently, Praun et al. [25] proposed an approach quite similar to ours. The main difference is that we propose several techniques to realize an efficient real-time 3D morphing, for example, the normal map encoding and the multiresolution path editing, etc. (Section 4).

Notation: We denote topological elements such as a mesh, a face, an edge and a vertex by italic S, f, e and v, respectively. The 3D position of a vertex is represented by bold face $\mathbf{v} \in \mathbf{R}^3$ and the 2D position in the parametric domain is represented by bold face italic $\boldsymbol{v} \in \mathbf{R}^2$. A scalar value is represented by italic k.

3.1. Creating Base Interpolation Mesh

In the primary step for constructing MIMesh, the user manually creates a rough polyhedral surface for the base interpolation mesh. The user selects a vertex from each input mesh to create a vertex of the base interpolation mesh. The 3D positions of selected vertices in input meshes are assigned to such a created vertex. Next the user creates faces of the base interpolation mesh M^0 to be a rough approximation of each input mesh. We have developed a system for creating the base interpolation mesh.

There are two reasons for creating the base interpolation mesh manually. One is that vertex/face correspondences created here imply the shape features of input meshes. For example, the user may wish to transform a tiger's nose to a venus's nose. It is difficult to specify such user-oriented correspondences automatically. The other reason is that the base interpolation mesh can be used for roughly checking the final morphing result. The design of the base interpolation mesh has a large effect on the partitioning of input meshes and is therefore an important process for the user to understand such feature correspondences intuitively.

It is desirable that the user creates the base interpolation mesh to be a simplified version of each input mesh. However, automatic construction of such a base mesh by using mesh simplification techniques (for example, [8]) is difficult, because a better approximation for one mesh cannot always be better for other meshes.

3.2. Mesh Partition and Parameterization

The next step in the MIMesh construction process is the partition of each mesh into several patches and to apply mesh parameterization to each patch. Each patch corresponds to a face of the base interpolation mesh.

First, we determine the boundary curves of the patches. Each boundary curve corresponds to an edge of the base interpolation mesh and should be on an input mesh. To obtain boundary curves, we calculate the approximate short-

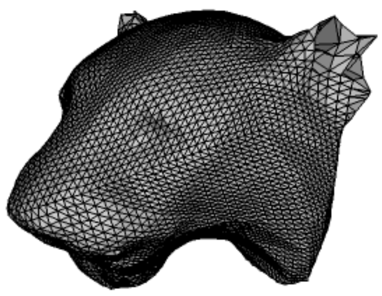


Figure 3. Subdivision fitting result without using mesh parameterization.

est paths using [12]. When all of the shortest paths have been calculated, input meshes are ready to be partitioned into patches. If a boundary curve passes into a face of an input mesh, we cut such a face to sub-faces and triangulate them. All faces of each patch can be gathered by a simple greedy-like algorithm.

In the mesh partition process, we need to consider the possibilities that boundary curves are *invalid*, namely, two boundary curves cross each other, or share some edges in an input mesh. This is mainly due to that the boundary curves are too long. We address this problem by means of an interactive curve modification. If such a case occurs, we insert some *mid-points* to shorten a boundary curve, then recalculate them. By using a mid-point, a boundary curve can be divided into two curves. Note that these points are used only for arranging boundary curves and are not affected by the connectivity of the base interpolation mesh.

For each patch, we apply mesh parameterization to assign a 2D parameter to a vertex. There are two reasons for using mesh parameterization. One is for the robust calculation of the subdivision fitting algorithm discussed in Section 3.3. If mesh parameterization is not applied, the fitting results will typically have some flipping triangles and self-intersections, as shown in Figure 3. The other is that they are used for texture mapping, as discussed in Section 4.1.

We use Floater's shape-preserving mapping algorithm [7] for mesh parameterization. In [7], an internal vertex p_i in the parametric domain is represented by a convex combination of its 1-ring neighbor vertices p_j:

$$\boldsymbol{p}_i = \sum_j \lambda_{i,j}\boldsymbol{p}_j, \quad \lambda_{i,j} \geq 0, \quad \sum_j \lambda_{i,j} = 1, \qquad (1)$$

where $\lambda_{i,j}$ denotes a scalar weight between p_i and p_j, calculated by using a geodesic polar map [27]. The main advantage of Floater's weight is that it always has a positive value. This guarantees that the linear system derived from Equation (1) can be solved robustly.

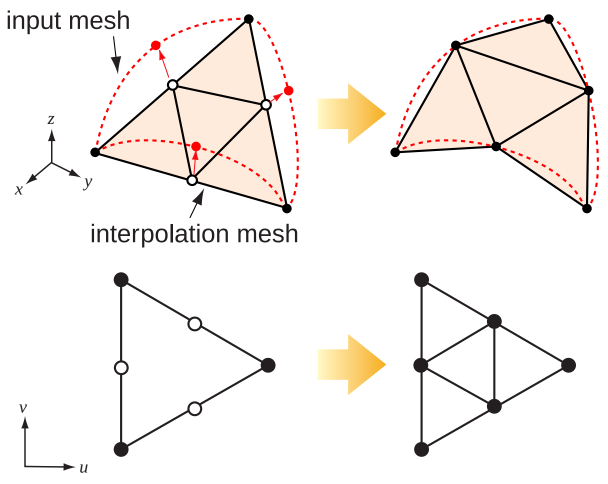


Figure 4. Uniform subdivision fitting process.

3.3. Subdivision Fitting

The final step in the MIMesh construction process is the subdivision fitting from the base interpolation mesh to approximate input meshes. Figure 4 illustrates a uniform subdivision fitting process for each face. To obtain the interpolation mesh at subdivision level $i+1$ from that at level i, the following two steps are processed.

First, we apply 4-to-1 split for all faces at level i. Each face is subdivided into four sub-faces. At the same time, we assign a 2D parameter value to each created vertex. We use the midpoint of two end vertices of an edge. Next, we find the 3D positions of each created vertex of the interpolation mesh so that which are on the faces of input meshes. This can be processed by using the inverse mapping of mesh parameterization described in Section 3.2. When a vertex p is included in a face of an input mesh $f = (p_i, p_j, p_k)$ on the parametric domain, the 3D position $\mathbf{p}$ is calculated by a barycentric coordinate (α, β, γ):

$$\mathbf{p} = \alpha\mathbf{p}_i + \beta\mathbf{p}_j + \gamma\mathbf{p}_k. \qquad (2)$$

Before the calculation of $\mathbf{p}$, we need to find a face f in which p is included. This problem can be, in general, regarded as a *point location problem* in the parametric domain, and can be processed in $O(\log n)$-time for each vertex. Then, the calculation for the uniform subdivision fitting at each level is processed in $O(m \log n)$-time, where m is the number of created vertices and n is the number of faces in the interpolation mesh at that level.

Local Subdivision Fitting

The uniform subdivision fitting scheme described above, however, has a problem that the number of faces is unneces-

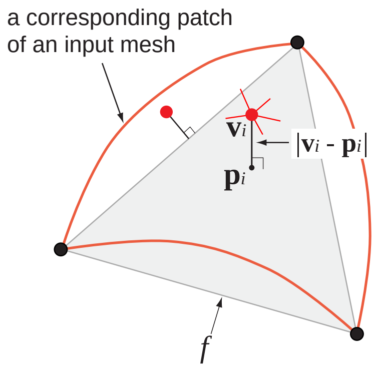

a corresponding patch of an input mesh
$\mathbf{v}_i$
$|\mathbf{v}_i - \mathbf{p}_i|$
$\mathbf{p}_i$
f

Figure 5. Error function for a face of MIMesh.

sarily increased because those in regions with high approximation accuracy are also subdivided. To address this problem, we introduce the local subdivision fitting scheme. We modify a fitting algorithm so that MIMesh with less number of faces can have high approximation accuracy. The basic idea is that we measure an approximation error from each vertex of an input mesh to a face of MIMesh in advance, and we subdivide the face that has a high approximation error.

We now define an error function $E(f)$ for each face f of MIMesh, as shown in Figure 5. For each vertex $\mathbf{v}_i$ of a patch, we find a point $\mathbf{p}_i$ in a face of MIMesh, in which $|\mathbf{v}_i - \mathbf{p}_i|$ is minimum Euclid distance. An error function of f for one mesh S^1 is defined as the maximum Euclid distance for all vertices of a patch that correspond to a face of MIMesh as follows:

$$E^1(f) = \max\{|\mathbf{v}_i - \mathbf{p}_i|\}, \quad i = 1...n, \tag{3}$$

where n denotes the number of vertices in a patch. We also define $E^2(f)$ for the other mesh S^2. An error function $E(f)$ used for the evaluation is defined as the maximum of *normalized* $E^1(f)$ and $E^2(f)$, that is, the value divided by the diagonal length of a bounding box $B(\mathcal{S}^1)$ $\left(B(\mathcal{S}^2)\right)$ of each input mesh, respectively:

$$E(f) = \max\left(\frac{E^1(f)}{B(\mathcal{S}^1)}, \frac{E^2(f)}{B(\mathcal{S}^2)}\right). \tag{4}$$

The reason why we normalize is that $E^1(f)$ and $E^2(f)$ are dependent on the size of each mesh. If S^1 is much larger than S^2, $E^1(f)$ will be larger than $E^2(f)$. This produces an unbalanced MIMesh in which only the geometry of S^1 is considered.

The local subdivision fitting algorithm has the following steps:

1. Insert all faces of the base interpolation mesh M^0 to a list $\mathcal{L}$.

2. Calculate $E(f)$ described in Equation (4) for each face f of $\mathcal{L}$. If $E(f) \leq \epsilon$, delete f from $\mathcal{L}$. ϵ denotes a user-specified threshold.

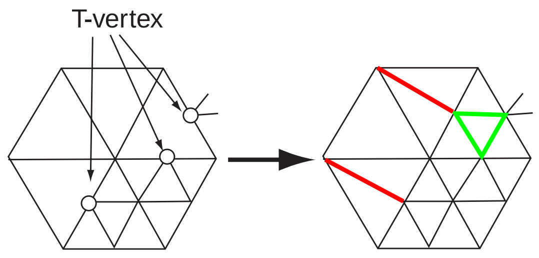


Figure 6. Adaptive subdivision scheme to delete T-Vertices.

3. For each face in $\mathcal{L}$, subdivide to four sub-faces by 4-to-1 split. Calculate 3D positions for each created vertex. Delete f and insert four sub-faces to $\mathcal{L}$.

4. Apply an adaptive subdivision scheme to delete T-vertices, as shown in Figure 6.

5. Repeat 2, 3 and 4 until $\mathcal{L}$ is empty.

T-vertex appears along the faces at which different subdivision levels meet. To delete T-vertices, we use red-green triangulations [19]. In addition to an ordinary 4-to-1 split (green), we apply a triangle bisection (red). If the number of T-vertices is one in a face, we apply a red triangulation. If the number of T-vertices is two, we first apply a green triangulation to this face, and then apply a red triangulation to its neighboring face.

3.4. Experimental Results

We evaluate our MIMesh construction approach in this section. Figure 7 shows the results of constructing MIMesh from two input meshes, venus (24,000 faces) and tiger (8,064 faces). In Figure 7, (a) shows two input meshes S, and (b) shows the base interpolation mesh created by the user from them. We created the base interpolation mesh so that corresponding vertices are uniformly distributed on the whole shape. Furthermore, more vertices are added to distinctive regions (for example, ears, noses), so that a less subdivided interpolation mesh with higher approximation accuracy can be obtained. Figure 7 (c) shows MIMesh to which uniform subdivision fitting is applied. Here, left is subdivision level 2 (M^2), and right is level 4 (M^4). Figure 7 (d) shows MIMesh to which local subdivision fitting is applied at different ϵ (0.01, 0.001).

Table 1 indicates mesh sizes (the number of faces), approximation errors and calculation times for the construction of MIMesh shown in Figure 7. We use IRI-CNR Metro tool [5] for the evaluation of approximation errors. We show here the percentage of mean square error (L^2-norm) for a diagonal length of a bounding box of a mesh. It can be seen in this table that the number of faces of the interpolation

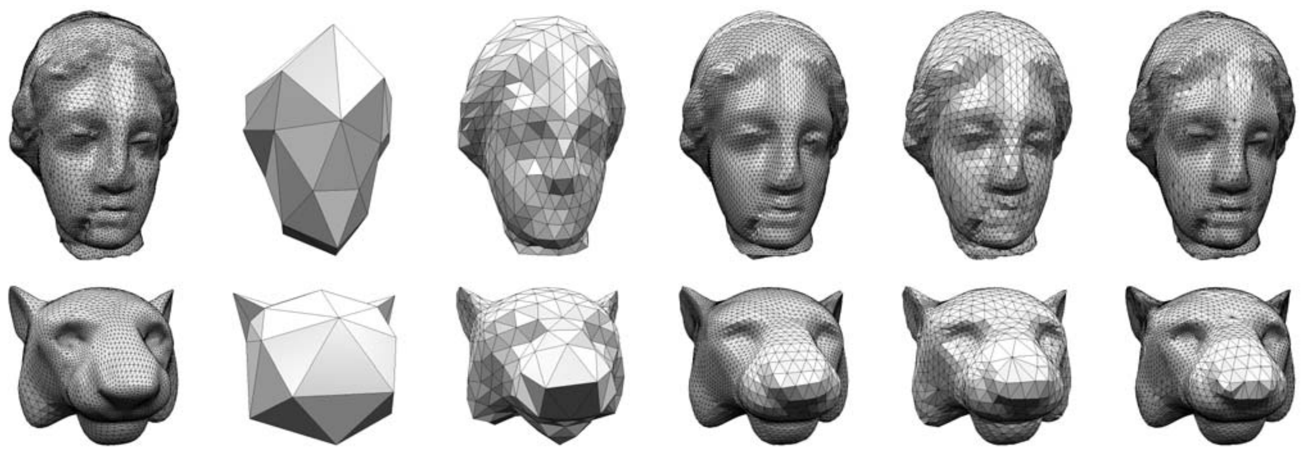


Figure 7. Subdivision fitting results: (a) Input meshes (venus, tiger). (b) A base interpolation mesh created by the user. (c) Uniform subdivision fitting results. Left: M^2, Right: M^4. (d) Local subdivision fitting results. Left: $\epsilon = .01$, Right: $\epsilon = .001$.

	size (#faces)						error (%)				
	S	M^0	M^2	M^4	$\epsilon = .01$	$\epsilon = .001$	M^0	M^2	M^4	$\epsilon = .01$	$\epsilon = .001$
venus	24,000	54	864	13,824	5,048	19,784	3.81	0.50	0.09	0.09	0.04
tiger	8,064						3.59	0.72	0.12	0.15	0.07

	time (sec.)				
	pre.	M^2	M^4	$\epsilon = .01$	$\epsilon = .001$
	51.2	1.4	32.3	20.9	84.1

Table 1. Statistical summary of Figure 7: mesh sizes (number of faces), approximation errors and calculation time.

mesh with $\epsilon = 0.01$ is 60% fewer than that of M^4, while both approximation errors of these two meshes are almost equal. This proves that our local subdivision fitting scheme produces a good approximation with decreasing the number of faces. It can also be seen from Figure 7 (d) that faces especially in the higher curvature regions are selectively subdivided.

On the other hand, it should be noted that our local subdivision fitting scheme also produces redundantly subdivided faces for one mesh. That is to say, faces that have low approximation errors in a certain region of one mesh are forcibly subdivided because faces in a corresponding region of the other mesh have high approximation errors. This is due to the fact that an error function in our fitting algorithm adopts the maximum approximation error of either face in two meshes.

The calculation time is measured on an AT-compatible PC (PentiumIII 1GHz CPU, 512MB Memory) environment. Note that our code is not fully optimized and needs more improvement. In Table 1, the value indicated by "pre." includes the time for calculating the shortest paths to partition input meshes, for grouping meshes and for mesh parameterization of all patches. Shortest path computation is very costly among these three processes, because 81 paths for each mesh must be computed. Other calculation times shown in this table are the cost for the subdivision fitting process from the base interpolation mesh M^0. In general, a local subdivision scheme can be more costly than a uniform subdivision scheme. This is due to an additional minimum distance computation for each vertex in the former.

4. 3D Morphing Using Multiresolution Interpolation Meshes

MIMesh is a very powerful tool for 3D morphing. To prove this, we discuss in this section three new interpolation schemes, normal map morphing, interpolation path editing using multiresolution representation, and multi-target morphing.

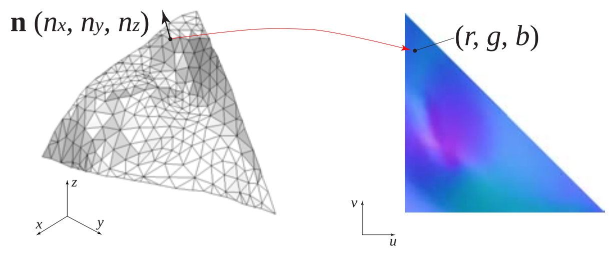


Figure 8. Normal map creation.

4.1. Using Normal Map

MIMesh is only an approximation of input meshes; thus, their exact and detailed geometries cannot be recovered perfectly. However, this issue can be resolved to a certain extent by texture mapping based on using normal maps for visual effects.

Normal map is an image that represents quantized normals of surfaces or meshes and has the possibility to be an alternative for visualizing highly detailed geometry. In contrast, there is another technique called *bump mapping*. It utilizes a height map to represent a bumpy surface; however, the conversion from a height map to a *normal perturbation map* is needed [17]. If per-vertex normals of an input mesh are obtained, it is better to use normal maps for exact visual effects. Recently, Cignoni et al. [4] proposed a method for obtaining normal maps by sampling normals of an original mesh to its simplified mesh.

In our MIMesh, the interpolation mesh at each subdivision level shows one-to-one correspondence to each of the input meshes. That is, each vertex of a patch has a 2D parameter calculated by mesh parameterization as described in Section 3.2, and each vertex of MIMesh has a 2D parameter in the same parameter space as well. Therefore, it is easy to apply texture mapping to MIMesh using these parameters and normal maps extracted from input meshes.

Our method for creating normal maps is simple and direct. As shown in Figure 8, we first calculate a per-vertex normal $\mathbf{n}$ (n_x, n_y, n_z) for each vertex of a patch. Each coordinate of this normal is within the limits of [-1, 1]. We quantize these coordinates to integers of a range [0, 255] to create an RGB pixel (r, g, b). We define a triangular region on 2D image space and put a pixel into it. Each remaining RGB pixel in the region is calculated by barycentric interpolation of its corresponding face of a patch. Finally, a set of pixels for all patches of an input mesh are packed into a single rectangular image. For 3D morphing, we interpolate not only between geometries but also between two normal maps.

Figure 9 demonstrates a visual comparison between MIMesh with only geometry, and with both geometry and texture. Figure 9 (a) shows an input mesh, and (b) shows

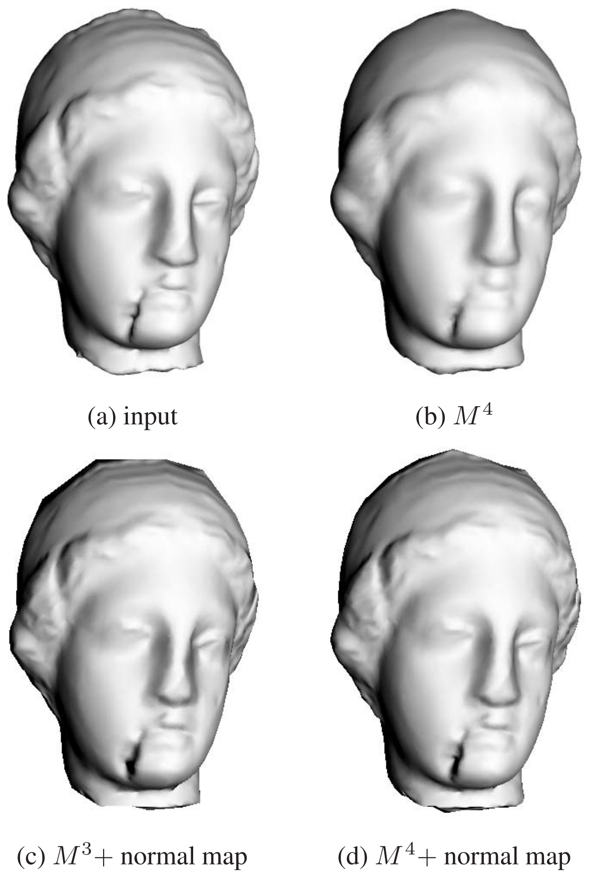


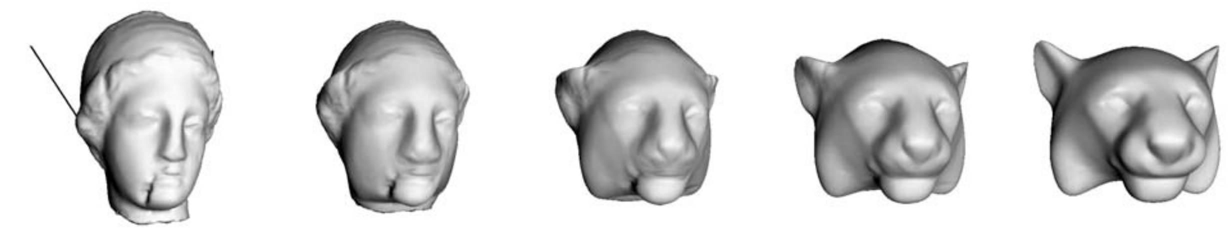
Figure 9. Texture Mapping of MIMesh using normal map.

only the geometry of the interpolation mesh M^4. Figures 9 (c) and (d) show the results of applying normal maps to the interpolation mesh at levels 3 (M^3) and 4 (M^4), respectively. It can be seen from this figure that we can obtain visually more similar results from an interpolation mesh with normal maps than from only the geometry of M^4, while the former uses the coarser geometry of M^3. We can obtain an interpolation mesh at arbitrary subdivision level by applying the same texture from MIMesh by traversing a quad-tree structure. This is effective for the LOD control of display.

4.2. Using Multiresolution Representation

In this section, we discuss two advantages of our MIMesh due to its multiresolution representation.

One advantage is the efficiency of data storage. Faces are not necessary in the data format of MIMesh. In the ordinary mesh format such as VRML, faces (for example, several vertex ids) must be explicitly specified. In MIMesh, faces are consistently managed in subdivision connectivity, and can be created automatically in the applications. Our data


(a)

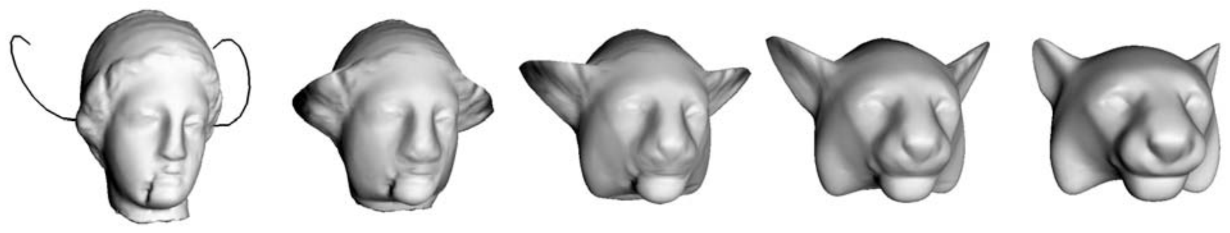

(b)

Figure 10. Interpolation path control using multiresolution representation.

MIMesh (up to M^4)	485
Normal map images	187 + 139
venus	818
tiger	260

Table 2. Comparison of data storage. (kbyte)

format of MIMesh consists of an ordinary mesh format (vertices and faces) of the base interpolation mesh, file names of normal map images, a header including per-patch attributes (for example, a set of interpolation paths), and a set of 3D positions for vertices needed as a keyframe interpolation. Typically, most of data are occupied by the 3D positions. A 2D parameter for each vertex is calculated automatically in the applications.

Table 2 shows the comparison of data storage in Figures 7 and 8. The MIMesh used here is a uniform version up to M^4 stored in ASCII format. Each normal map has 512×512 image size and is stored by compressed TGA format. Input meshes are saved in standard Wavefront OBJ ASCII format. It can be seen from Table 2 that the sum of data size needed for 3D morphing (MIMesh and two normal map images) is 50% less than the sum of data size of input meshes.

The other advantage is that we can edit interpolation paths using multiresolution representation. In MIMesh, the vertex $\mathbf{p}^i$ of an interpolation mesh at subdivision level i can be represented by using two end vertices $\mathbf{p}_s^{i-1}, \mathbf{p}_e^{i-1}$ of an edge at coarser level $i-1$ and a difference vector $\mathbf{D}_i$ (called *detail coefficient*) as follows:

$$\mathbf{p}^i = \frac{1}{2}\left(\mathbf{p}_s^{i-1} + \mathbf{p}_e^{i-1}\right) + \mathbf{D}_i. \tag{5}$$

From Equation (5), MIMesh can be alternatively defined as a set of detail coefficients and a base interpolation mesh. We can use this re-definition for multiresolution editing. As some other researches on multiresolution editing for single meshes [29, 18], modifications can be done at the coarsest level (a base interpolation mesh in our case), and meshes at higher levels can be recovered by sequentially adding detail coefficients.

Figure 10 demonstrates the results of the interpolation path editing in MIMesh. (a) shows the result that all paths are linear. In (b), the interpolation path for each vertex of the base interpolation mesh is defined as a cubic Bézier curve. For paths put on top of venus's ears, only two middle control points are modified and detail coefficients are added. Black lines on the left of Figures 10 (a) and (b) denote such paths.

4.3. Multi-target Morphing

By using uniform subdivision fitting, the number of faces of MIMesh depends only on that of the base interpolation mesh and the number of subdivision levels. Therefore, we can establish *multi-target morphing* for more than three input meshes in particular.

Figure 11 demonstrates the results of multi-target morphing, in which one more mesh (mannequin) is added to the result of two meshes shown in Figure 7. By using our system for creating a base interpolation mesh, several vertices of the third mesh correspond to those of the other two meshes in the same manner. We calculate 3D positions of interpolation meshes up to subdivision level M^4 in the uniform subdivision fitting process. Mesh size of new M^4 is 13,824, and is the same as that in the case of two meshes.

In Figure 11, we use a simple convex combination in-

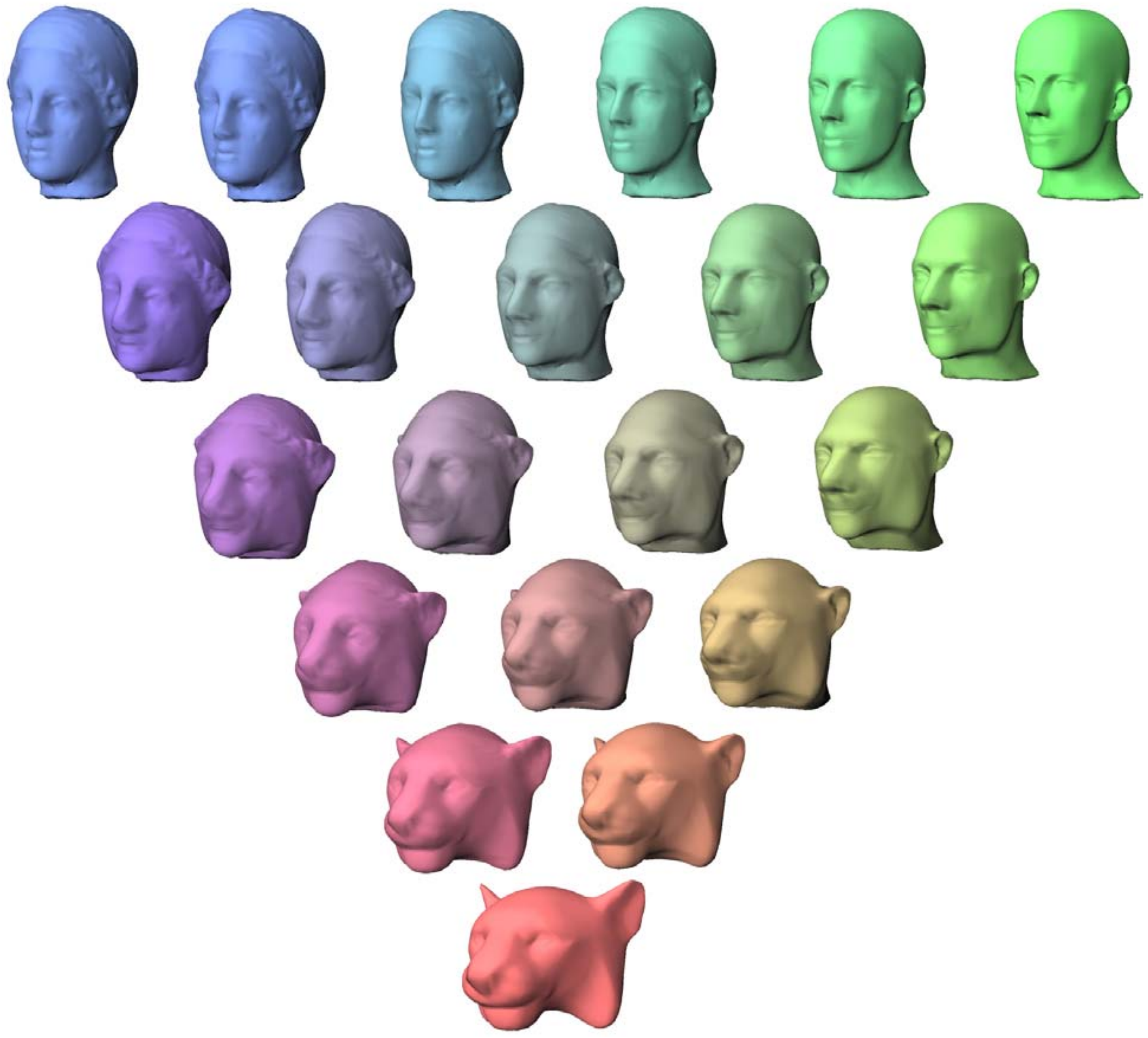


Figure 11. Multi-target morphing from three input meshes.

terpolation. For the interpolation of n meshes, 3D position $\mathbf{v}$ of an in-between mesh is calculated using the following equation:

$$\begin{aligned} \mathbf{v}(t_1, t_2, ..., t_n) &= \sum_{i=1}^{n} t_i \mathbf{v}^i, \quad \sum_{i=1}^{n} t_i = 1, \\ 0 \leq \quad t_i &\leq 1, \end{aligned} \tag{6}$$

where t_i denotes the weight for each 3D position.

5. Conclusions and Future Work

We have proposed a new multiresolution-based shape representation for 3D mesh morphing. Our approach does not use combination operations that caused several serious problems in previous mesh morphing. Therefore, we can obtain interpolation meshes robustly using two types of subdivision fitting schemes. The multiresolution interpolation mesh has a hierarchical semi-regular mesh structure based on subdivision connectivity. We have shown that this leads to various advantages, for example, efficient data storage and easy acquisition of an interpolation mesh with arbitrary subdivision level. We have also demonstrated several new features by using our multiresolution interpolation meshes, efficient rendering of 3D morphing using normal maps, multiresolution editing of interpolation paths, and multi-target morphing.

There are many future directions for us to make the most of our new representation for efficient 3D morphing. We are interested in the following two issues in particular: new special effects of the interpolation using the image processing approach of normal maps, and applications to non-triangular meshes including quadrilateral (for example, Catmull-Clark) subdivision surfaces.

Acknowledgement

The authors would like to thank the reviewers for useful comments. "Mannequin" model is from Graphics and Imaging Laboratory, University of Washington, "venus" model is from Cyberware Inc., and "tiger" is courtesy of Viewpoint DataLabs.

References

[1] M. Alexa. Merging polyhedral shapes with scattered features. *The Visual Computer*, 16(1):26–37, 2000.

[2] M. Alexa. Local control for mesh morphing. Shape Modeling International, to appear, 2001.

[3] M. Alexa, J. Behr, and W. Müller. The morph node. In *Proc. ACM VRML/Web3D 2000*, pages 29–34. ACM Press, New York, 2000.

[4] P. Cignoni, C. Montani, C. Rocchini, R. Scopigno, and M. Tarini. Preserving attribute values on simplified meshes by re-sampling detail texture. *The Visual Computer*, 15(10):519–539, 1999.

[5] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998.

[6] 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.

[7] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14:231–250, 1997.

[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] J. Gomes, L. Darsa, B. Costa, and L. Velho. *Warping and Morphing of Graphical Objects*. Morgan Kaufmann, 1999.

[10] A. Gregory, A. State, M. Lin, D. Manocha, and M. Livingston. Interactive surface decomposition for polyhedral morphing. *The Visual Computer*, 15(9):453–470, 1999.

[11] I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal meshes. In *Computer Graphics (Proc. SIGGRAPH2000)*, pages 95–102. ACM Press, New York, 2000.

[12] T. Kanai and H. Suzuki. Approximate shortest path on a polyhedral surface based on selective refinement of the discrete graph and its applications. In *Proc. Geometric Modeling and Processing 2000*, pages 241–250. IEEE CS Press, Los Alamitos CA, Apr. 2000.

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

[14] T. Kanai, H. Suzuki, and F. Kimura. Metamorphosis of arbitrary triangular meshes. *IEEE Computer Graphics and Applications*, 20(2):62–75, April 2000.

[15] T. Kanai, H. Suzuki, J. Mitani, and F. Kimura. Interactive mesh fusion based on local 3D metamorphosis. In *Proc. Graphics Interface '99*, pages 148–156. Morgan Kaufmann Publishers, San Francisco, 1999.

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

[17] M. J. Kilgard. A practical and robust bump-mapping technique for today's GPUs. Game Developer's Conference 2000 Course "Advanced OpenGL Game Development", 2000. http://www.nvidia.com/Developer.nsf.

[18] L. P. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Computer Graphics (Proc. SIGGRAPH 98)*, pages 105–114. ACM Press, New York, 1998.

[19] U. Labsik, L. P. Kobbelt, and H.-P. S. Robert Schneider. Progressive transmission of subdivision surfaces. *Computational Geometry*, 15(1–3):25–39, 2000.

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

[21] F. Lazarus and A. Verroust. Three-dimensional metamorphosis: A survey. *The Visual Computer*, 14(8-9):373–389, 1998.

[22] A. W. F. Lee, D. Dobkin, W. Sweldens, and P. Schröder. MAPS: Multiresolution adaptive parameterizaiton of surfaces. In *Computer Graphics (Proc. SIGGRAPH 98)*, pages 95–104. ACM Press, New York, 1998.

[23] A. W. F. Lee, D. Dobkin, W. Sweldens, and P. Schröder. Multiresolution mesh morphing. In *Computer Graphics (Proc. SIGGRAPH 99)*, pages 343–350. ACM Press, New York, 1999.

[24] R. Ohbuchi, Y. Kokojima, and S. Tahahashi. Blending shapes by using subdivision surfaces. *Computers and Graphics*, 25(1):41–58, 2001.

[25] E. Praun, W. Sweldens, and P. Schröder. Consistent mesh parameterizations. Computer Graphics (Proc. SIGGRAPH 2001) to appear, 2001.

[26] A. Wakita, M. Yajima, T. Harada, H. Toriya, and H. Chiyokura. XVL: A compact and qualified 3D representation with lattice mesh and surface for the internet. In *Proc. ACM VRML/Web3D 2000*, pages 21–24. ACM Press, New York, 2000.

[27] W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. In *Computer Graphics (Proc. SIGGRAPH 94)*, pages 247–256. ACM Press, New York, 1994.

[28] M. Zöckler, D. Stalling, and H.-C. Hege. Fast and intuitive generation of geometric shape transitions. *The Visual Computer*, 16(5):241–253, 2000.

[29] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Computer Graphics (Proc. SIGGRAPH 97)*, pages 259–268. ACM Press, New York, 1997.