EUROGRAPHICS '99 / P. Brunet and R. Scopigno
(Guest Editors)

Volume 18 (*1999*), *Number 3*

Interactive Multiresolution Editing of Arbitrary Meshes

Seungyong Lee †

Department of Computer Science and Engineering
Pohang University of Science and Technology (POSTECH)
Pohang, 790-784, Korea

Abstract

This paper presents a novel approach to multiresolution editing of a triangular mesh. The basic idea is to embed an editing area of a mesh onto a 2D rectangle and interpolate the user-specified editing information over the 2D rectangle. The result of the interpolation is mapped back to the editing area and then used to update the mesh. We adopt harmonic maps for the embedding and multilevel B-splines for the interpolation. The proposed mesh editing technique can handle an arbitrary mesh without any preprocessing such as remeshing. It runs fast enough to support interactive editing and produces intuitive editing results.

1. Introduction

In computer graphics, triangular meshes are widely used in representing shapes. Mesh editing allows a user to control the shape of a mesh by adjusting the vertex positions of the mesh. It is a powerful tool for generating a desired shape from a given mesh. However, if a user can edit only one vertex at a time, it would be tedious and time consuming to modify the shape of a mesh with several thousand faces. Hence, there is a necessity for an effective mesh editing technique that can handle multiple vertices at a time in an intuitive way.

In this paper, we present a novel approach to triangular mesh editing that satisfies the following requirements.

- *Direct manipulation*: A user can edit a triangular mesh by picking and dragging parts of the mesh: specifically, a set of vertices. This contrasts with conventional deformation techniques such as free-form deformations[1], which modify object shapes by manipulating a control lattice. Various editing primitives are provided such as a chain of vertices, a closed sequence of vertices, and a scattered set of vertices. The result of editing reflects all the editing information specified at the vertices in the editing primitive.
- *Multiresolution editing*: When a vertex is edited, its neighboring vertices may be influenced to various degrees. In principle, this is the same as the multiresolution editing that can be provided by wavelets[2] or subdivision meshes[3]. The scale of the influence is controlled by an intuitive and continuous parameter.
- *Editing area control*: A user can exactly specify the editing area of the mesh, where only the vertices in the area may be adjusted by an editing operation. Editing area control is different from the multiresolution editing mentioned above. A small scale editing only has influences on the vertices that are nearby to the editing primitive, not all the vertices in an editing area. On the largest scale, an editing operation will modify the positions of all the vertices in an editing area.
- *Arbitrary mesh handling*: Meshes with arbitrary structures can be handled without any preprocessing. This contrasts with an editing technique based on subdivision[3], which requires a given mesh to have subdivision connectivity.
- *Intuitive response*: The result of an editing operation satisfies what a user intended by moving an editing primitive. In the updated mesh, the new positions of the vertices in the editing primitive are interpolated and smoothly propagated to the other vertices.
- *Interactive editing*: An editing operation is efficient enough to provide a user with an interactive response. The mesh is updated several times a second when an editing primitive is dragged.

The basic idea of mesh editing in this paper is to embed a specified editing area onto a 2D rectangle and interpolate the user-specified editing information over the 2D rectangle. We adopt a harmonic map[4] to obtain an embedding that

† leesy@postech.ac.kr, http://www.postech.ac.kr/∼leesy

minimizes metric distortion. If several editing operations are applied to an editing area, the harmonic map is constructed only once when a user specifies the editing area. When an editing primitive is selected and modified, the information is mapped to the corresponding vertices in the 2D rectangle. Multilevel B-spline interpolation[5] is then used to propagate the mapped information to the other vertices in the 2D rectangle. The interpolated editing information is finally mapped back to the vertices in 3D to generate an updated mesh.

Among the above requirements, editing area control and arbitrary mesh handling are made possible by the embedding of an editing area onto a 2D rectangle. We can easily implement and strictly enforce editing area control by considering only the vertices of an editing area in the embedding. Once the specified editing area is homeomorphic to a 2D disk, which is not a severe restriction, our mesh editing technique can be applied to a mesh with an arbitrary structure. Multilevel B-spline interpolation is a key component that provides direct manipulation and multiresolution editing. It interpolates user-specified editing information over the 2D embedding of an editing area by constructing surfaces that pass through scattered data points. By changing the size of the coarsest control lattice in multilevel B-splines, we can control the influence area of an editing primitive. Intuitive response comes from the favorable properties of harmonic embedding, which minimizes metric distortion, and multilevel B-splines, which smoothly interpolate scattered data points. Interactive editing is possible due to the fast speed of multilevel B-spline interpolation. Harmonic embedding may not be done at an interactive rate for an editing area with many triangles. However, it does not prohibit interactive editing because a harmonic map is constructed only once when the editing area is specified.

The remainder of this paper is organized as follows. Section 2 reviews the related work. In Section 3, we outline the mesh editing technique proposed in this paper. Sections 4 and 5 cover the key steps in the proposed mesh editing technique, which are 2D embedding and editing information interpolation, respectively. Section 6 shows mesh editing examples. Section 7 concludes this paper.

2. Related Work

2.1. 3D Deformations

Deformation is a powerful modeling tool that changes the shape of an existing object to create a complicated object. Mesh editing can be considered as a kind of deformation because it changes the shape of a mesh.

There has been much research on deformation techniques, including Barr's global and local deformations[6], free-form deformations[1] and its extensions[7, 8, 9, 10], axial deformations[11], and wires[12]. These techniques define transformation functions in 3D space by manipulating deformation primitives such as control lattices and curves. The shape of an object is changed by applying the transformation function to the vertices of the object.

In contrast, the mesh editing technique proposed in this paper allows a user to directly manipulate the vertices of a mesh, which enables fine control of the edited vertices. Also, the editing information is propagated to the other vertices over the surface of the mesh, not through 3D space. This property provides easy control of the modified area of a mesh. For example, when we want to edit a front part of a mesh, any part in the back should not be influenced even though both parts are nearby in 3D space.

2.2. Multiresolution editing

There have been several approaches to multiresolution editing of an object. Forsey and Bartels presented a hierarchical editing technique for B-spline surfaces based on local refinements of control lattices[13]. Wavelets provide hierarchical basis functions that enable multiresolution editing if an object can be represented by the basis functions[2, 14]. When a given mesh has been constructed by applying recursive subdivision to a simple mesh, the subdivision hierarchy can be used for multiresolution editing of the given mesh[3, 4, 15]. The resolution modeling system proposed by Cignoni *et al*. includes a simple multiresolution mesh editing feature[16].

Although these techniques support multiresolution editing, they have limitations on the representation or structure of the edited objects. Forsey and Bartels' method can only handle B-spline surfaces. To apply a wavelet-based technique, an object representation must be converted to a hierarchical representation with wavelet bases. This conversion may not be simple when an object has a complex structure. Multiresolution editing based on subdivision hierarchy can be used only when a given mesh has subdivision connectivity. There are remeshing techniques that convert an arbitrary mesh to a mesh with subdivision connectivity[4, 15]. However, remeshing changes the connectivity and positions of vertices, which results in an approximation of a given mesh. Mesh editing supported by the resolution modeling system also requires a preprocessing that converts a given mesh into a multiresolution representation.

In contrast, the approach proposed in this paper can handle an arbitrary mesh without any preprocessing such as wavelet analysis or remeshing. The only constraint of the approach is that the editing area should be homeomorphic to a 2D disk, which is not of great consequence in practical application.

2.3. Direct manipulation

Hsu *et al*. presented a direct manipulation technique for free-form deformations[17]. Welch and Witkin proposed a variational approach to directly manipulate B-spline surfaces with scattered points or curves[18]. To provide a user with

direct manipulation, these techniques respectively compute the pseudoinverse of a possibly large matrix or a numerical solution of energy minimization. When a large number of vertices are picked and dragged, these techniques may not produce updates at an interactive rate. Also, an editing area with an arbitrary boundary cannot be specified with these techniques.

In multiresolution editing based on wavelets or subdivision, large scale editing is done by manipulating upper level coefficients or polygons in the hierarchical representation. Editing of these scale coefficients or polygons has influence on all the finest level vertices that inherit their positions from the coefficients or polygons. This approach does not allow a user to work with the vertices at the finest level when large scale editing is desired. Also, a user cannot specify an editing area with an arbitrary boundary, for example, that crosses the influence area of an upper level polygon.

Kobbelt *et al.*[19] proposed a novel approach to interactive editing that overcomes the drawbacks of the techniques based on wavelets and subdivision. The approach allows a user to specify an arbitrary editing area in a mesh and to use any vertices in the area for editing at a large scale. To obtain the result of an editing operation, the thin plate energy minimization is performed by using multilevel discrete fairing.

In terms of user interface, our mesh editing technique is similar to that of Kobbelt *et al*. However, our technique is based on harmonic embedding and multilevel B-splines, which are simpler and easier to implement than multilevel discrete fairing. Also, in our technique, multiresolution editing is supported in a more intuitive way by providing a continuous scale parameter.

3. 3D Multiresolution Editing

3.1. Editing process

When a mesh is given, the editing process is as follows. Fig. 1 summarizes the process.

1. The user selects a closed editing area by specifying a sequence of vertices on the boundary of the area (Section 3.2).
2. The vertices in the editing area are embedded onto a 2D rectangle while minimizing metric distortion (Section 4).
3. If desired, the user can change the local coordinate frames at the vertices in the editing area (Section 3.4).
4. If desired, the user can change the scale parameter that controls the influence area of an editing primitive (Section 3.6).
5. The user selects an editing primitive that consists of a set of vertices in the editing area (Section 3.3).
6. The user modifies the positions of the vertices in the editing primitive (Section 3.3).
7. For the vertices in the editing primitive, the modified positions are converted to 3D offset vectors by using local frames (Section 3.5). The 3D offset vectors are then mapped to the corresponding points in the 2D embedding of the editing area.
8. The mapped 3D offset vectors are interpolated over the 2D embedding by generating three smooth surfaces (Section 5).
9. At all vertices in the editing area, 3D offset vectors are obtained from the surfaces and new positions are computed by using local frames (Section 3.5).
10. The updated mesh is displayed with the new positions of vertices in the editing area.

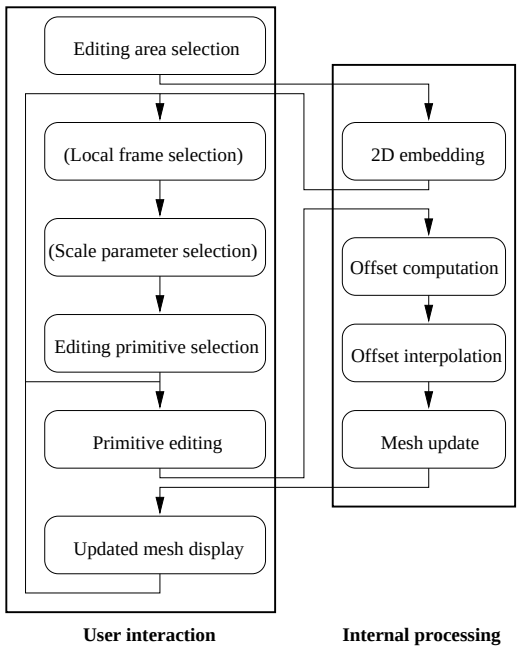


Figure 1: *The editing process.*

The user can interactively edit a mesh with a selected editing primitive by iterating steps 6 through 10. Also, the user can apply several editing operations to an editing area by iterating steps 3 through 10. In this case, 2D embedding of the editing area is done only once when it is specified. If the user wants to work with a different editing area, a new editing process is required that starts from step 1.

3.2. Editing area

An editing area is a closed region on a mesh that is homeomorphic to a 2D disk. The user can specify an editing area by picking a sequence of vertices on the mesh. To determine the boundary of the editing area, we compute the geodesics between adjacent vertices in the sequence that pass through the edges of the mesh. The interior of the editing area is defined by using the ordering of vertices in the boundary loop and obtained by a graph search. The user can also extend an

editing area by picking several faces that are adjacent to the editing area.

3.3. Editing operation

An editing primitive consists of a set of vertices in an editing area. There is no requirement on the structure of the vertex set. This freedom enables a variety of editing primitives, including a single vertex, an open or closed sequence of vertices, and a connected or scattered set of vertices.

To edit a mesh, the user can manipulate an editing primitive in two modes. In the first mode, the user can drag an editing primitive, which changes the positions of all vertices in the primitive simultaneously. In this case, the mesh is updated while dragging and the user can interactively edit the mesh. The second mode allows the user to change the positions of vertices in an editing primitive independently, handling the vertices one by one. This mode is useful for fine control of a mesh shape. For example, some parts of an editing primitive can be fixed while the other parts are changed. In the second mode, the mesh update happens when the user completes the manipulation of an editing primitive and pushes an apply button.

3.4. Local frames

A local coordinate frame is attached to each vertex in an editing area. At the vertices in an editing primitive, local frames are used to obtain 3D offset vectors from position changes by an editing operation. After the offset vectors are interpolated over all vertices in the editing area, local frames are used to update the vertex positions with the interpolated offsets.

There are many ways to define a local frame at a vertex. In this paper, we provide the user with two simple options. One option is to use the global coordinate frame of a given mesh as the same local frame at all vertices. The other option allows vertices to have different local frames based on normal vectors. In this case, each vertex becomes the origin of its local frame. Although there may be a more elegant method, we simply compute the normal vector at a vertex by averaging the plane normals of the adjacent triangles. However, the normal vector at a vertex determines only one axis of the local frame. The other two axes of local frames should be consistent among adjacent vertices. To define consistent local frames based on normal vectors, we first pick a vertex v and assign a local frame that consists of the normal vector $\vec{n}$ and two orthonormal vectors $\vec{x}$ and $\vec{y}$ in the plane perpendicular to $\vec{n}$. The local frame at a vertex v' is then obtained by applying a single rotation to vectors $\vec{x}$, $\vec{y}$, and $\vec{n}$. The rotation is determined so that $\vec{n}$ is transformed to align with the normal vector $\vec{n}'$ at v'.

3.5. 3D offset vectors

In an editing operation, the positions of vertices in an editing primitive are changed by the user. These position changes are converted to 3D offset vectors by using the local frames of the vertices. Let $\vec{x}$, $\vec{y}$, and $\vec{n}$ be three axis vectors of the local frame at a vertex v. Let $\Delta\vec{p}$ be the displacement of vertex v by an editing operation. The 3D offset vector $\Delta\vec{d}$ at v is derived by projecting the displacement $\Delta\vec{p}$ to the three axes of the local frame at v. That is,

$$\Delta\vec{d} = (\Delta x, \Delta y, \Delta n) = (\vec{x}\cdot\Delta\vec{p}, \vec{y}\cdot\Delta\vec{p}, \vec{n}\cdot\Delta\vec{p}). \quad (1)$$

The offset vectors computed by Eq. (1) at the vertices in an editing primitive are interpolated over the editing area as will be explained in Section 5. The interpolation supplies an offset vector for each vertex in the editing area. The updated position $\vec{p}_{\text{new}}$ of a vertex v is then determined by the interpolated offset vector $\Delta\vec{d}$ with the current position $\vec{p}_{\text{old}}$ and the local frame of v. That is,

$$\vec{p}_{\text{new}} = \vec{p}_{\text{old}} + \Delta x\,\vec{x} + \Delta y\,\vec{y} + \Delta n\,\vec{n}.$$

3.6. Scale parameter

In an editing operation, the scale parameter allows the user to intuitively control the influence area of an editing primitive. It has a value between zero and one. With the value zero, only the vertices in the editing primitive may change positions in the updated mesh. When the scale parameter is one, all vertices in the editing area are influenced by an editing operation. In this case, the editing effects are diminished based on the distance from the editing primitive. For a scale parameter value other than zero and one, an editing primitive has influence on nearby parts in the editing area, where the influence area is proportional to the value. Regardless of the scale parameter, the boundary of the editing area remains fixed in the updated mesh. As will be explained in Section 5, the scale parameter is implemented by controlling the size of the coarsest control lattice used in multilevel B-spline interpolation.

4. 2D Embedding of Editing Area

Once the user selects an editing area, it is embedded onto a 2D plane. There are several techniques that can be used to embed a region of a mesh onto a 2D plane. The 3D morphing technique by Kent *et al*. includes a physically-based method to project a polyhedron onto its convex hull[20]. Maillot *et al*. have proposed an energy minimization approach to embed a triangular mesh onto a 2D texture space[21]. To obtain a parameterization on an arbitrary mesh, Eck *et al*. have used harmonic maps to embed regions of the mesh onto 2D polygons[4]. Duchamp *et al*. have presented an efficient technique to compute harmonic maps by using hierarchical preconditioning[22]. Lee *et al*. have proposed MAPS, which smoothly projects the vertices of a mesh onto a simplified mesh[15].

Eck *et al*. have mentioned that the method of Kent *et al*. produces considerable metric distortion[4]. The method of

Maillot *et al.* may require heavy computation to obtain the solution of energy minimization. MAPS is based on mesh simplification and projects a region of a mesh onto 3D polygons, not a 2D plane. Hence, in this paper, we choose harmonic maps as the tool to embed an editing area onto a 2D plane. Recently harmonic maps have also been used for 3D morphing among two polyhedra[23, 24].

4.1. Harmonic maps

To implement 2D embedding of an editing area, we adopt the piecewise linear approximation of a harmonic map proposed by Eck *et al.*[4] Let $h : A \rightarrow P$ be a mapping from the vertices in a region A of a mesh onto a 2D polygon P. The energy of mapping h is defined by

$$E(h) \;=\; \frac{1}{2} \sum_{(v_i,v_j)\in edges(A)} \kappa_{ij} ||h(v_i) - h(v_j)||^2.$$

An interior edge (v_i, v_j) in A is incident to two faces, (v_i, v_j, v_{k_1}) and (v_i, v_j, v_{k_2}). The spring constants κ_{ij} is then given by

$$\kappa_{ij} \;=\; \frac{l_{ik_1}^2 + l_{jk_1}^2 - l_{ij}^2}{a_{ijk_1}} + \frac{l_{ik_2}^2 + l_{jk_2}^2 - l_{ij}^2}{a_{ijk_2}},$$

where l_{ij} is the length of an edge (v_i, v_j) and a_{ijk} is the area of a face (v_i, v_j, v_k). The lengths and areas are measured in A.

When the mapping from the boundary vertices of A to the boundary of P is fixed, there exists a unique mapping h that minimizes the energy $E(h)$. We can obtain the unique mapping by solving a sparse linear system for its values at the interior vertices of A. The unique mapping is called the harmonic map from A to P though strictly speaking it is only a piecewise linear approximation[4]. The harmonic map tends to produce an embedding of A onto P that minimizes metric distortions of the edges in A.

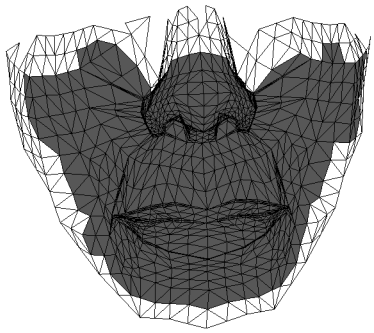
Eck *et al.* mentioned that self-intersections may occur in the 2D mesh that is generated by a harmonic map from A to P [4]. In this case, the 2D mesh contains foldovers, where parts of the mesh fold upon another parts. These foldovers may introduce an unexpected result when editing information is interpolated over the 2D mesh and mapped back to 3D. However, the self-intersections occur extremely rarely[4] and even when they really happen, the foldovers are usually confined to small regions so that the editing result is still intuitive. Hence, in this paper, we do not check the self-intersections in the 2D mesh.

4.2. Boundary mapping

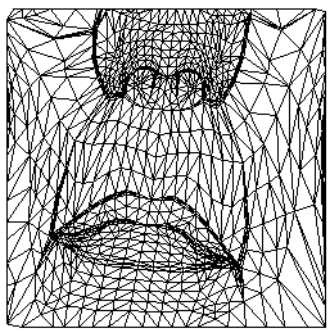
In this paper, an editing area A is embedded onto a rectangle R by a harmonic map. This relates with the multilevel B-spline interpolation used in Section 5 that requires a rectangular domain. Let ∂A and ∂R be the boundaries of A and R, respectively. To obtain a harmonic map from A to R, we need to determine the mapping from ∂A to ∂R.

If the user has picked four vertices on a mesh to specify the editing area A, those vertices are mapped to the corners of ∂R. The remaining vertices on ∂A between the picked vertices are then positioned on the corresponding edges of ∂R, where the distances between them on ∂A are proportional to those on ∂R. When three or more than four vertices have been picked to specify A, we simply choose one of the picked vertices and map it to a corner of ∂R. The vertices on ∂A are then positioned on ∂R so that their distances on ∂A from the chosen picked vertex are proportional to the corresponding distances on ∂R.

Fig. 2 shows an example of the 2D embedding. In Fig. 2(a), the shaded region is an editing area selected on a mesh. Fig. 2(b) shows the embedding of the editing area onto a 2D rectangle.



(a) an editing area



(b) the embedding of (a)

Figure 2: *An editing area and its embedding to a rectangle.*

5. Interpolation of Editing Information

Suppose that an editing area A has been specified and embedded onto a rectangle R. In an editing operation, the user changes the positions of vertices v_i in an editing primitive. At each vertex v_i, the position change is converted to a 3D offset vector, $\Delta d_i = (\Delta x_i, \Delta y_i, \Delta n_i)$, by using the

local frame. The 3D offset vectors Δd_i are then mapped to the points p_i in R that correspond to vertices v_i. To propagate the 3D offsets Δd_i to the other points in R, we construct three smooth surfaces that respectively interpolate Δx_i, Δy_i, and Δn_i. In this section, we only consider the interpolation of Δx_i. Δy_i and Δn_i can be handled in the same way.

Given offsets Δx_i at scattered points p_i in R, we need to derive a real value function f defined on R such that $f(p_i) = \Delta x_i$. This is the well known scattered data interpolation problem. There are several approaches to the problem: Shepard's method, radial basis functions, thin plate splines, and finite element methods. For more information, the reader is referred to the survey by Franke and Nielson[25].

In this paper, we use multilevel B-spline interpolation[5] to obtain the function f from scattered offsets Δx_i. There are several reasons for this choice. First, the technique runs fast enough to support interactive editing. Second, it provides a simple way to implement multiresolution editing. Third, with this technique, we can easily fix the boundary of an editing area.

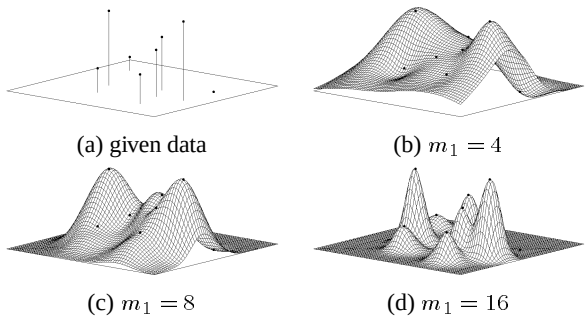
5.1. Multilevel B-spline interpolation

The offsets Δx_i at scattered points $p_i = (u_i, v_i)$ in a rectangle R can be considered as a set of 3D points $(u_i, v_i, \Delta x_i)$ over R. Multilevel B-spline interpolation generates a C^2-continuous surface that passes through the scattered 3D points[5]. The technique is based on the B-spline approximation that efficiently determines a control lattice by minimizing a local approximation error for each control point. It makes use of a coarse-to-fine hierarchy of control lattices to generate a sequence of B-spline surfaces whose sum approaches the desired interpolation surface. The sum of these surfaces is reduced into one equivalent B-spline surface by using B-spline refinement.

The shape of the resulting surface is affected by the size of the coarsest control lattice in the hierarchy. Fig. 3 shows an example. Let $(m_1+3)\times(m_1+3)$ be the size of the coarsest control lattice. For the data points given in Fig. 3(a), Figs. 3(b), (c), and (d) show the changes in the surface shape with different values of m_1. When m_1 is small, the effects of data points are blended together to yield a smooth surface shape. With a large m_1, the influences of data points are limited to small neighborhoods.

5.2. Influence area control

When we use multilevel B-splines to generate an interpolation function f over a rectangle R, the influence area of an offset Δx_i is determined by the size of the coarsest control lattice. The influence area directly relates with the influence area of an editing primitive in an editing operation. Although the lattice size can be used to intuitively control the influence area, it is an integer and does not support a continuous scale parameter between zero and one. To provide the user with a continuous scale parameter, we use the linear interpolation of functions f that are generated with two consecutive sizes of the coarsest control lattice.



(a) given data (b) $m_1 = 4$ (c) $m_1 = 8$ (d) $m_1 = 16$

Figure 3: *Surface shape changes caused by different sizes of the coarsest control lattice.*

Let l be the number of levels in a control lattice hierarchy, where Φ_1 and Φ_l are the coarsest and finest control lattices, respectively. Let $(m_k+3)\times(m_k+3)$ be the size of a control lattice Φ_k. In this paper, we use $m_1 = 1$ and $m_{k+1} = 2m_k$ for the sizes of control lattices in the hierarchy. Let f_k be the interpolation function that is generated by using multilevel B-splines with a subset of the control lattice hierarchy from Φ_k to Φ_l. We also define a function f_{l+1} that has values Δx_i at scattered points where offset vectors are assigned and is zero elsewhere.

Now we divide the interval $[0,1]$ into l uniform subintervals. Then, a function f_k is matched to the value $s_k = \frac{l-k+1}{l}$ of the scale parameter, for $k = 1, 2, \ldots l+1$. For the parameter values other than s_k, functions f_k are linearly interpolated. That is, for a parameter value s such that $s_k \le s \le s_{k+1}$,

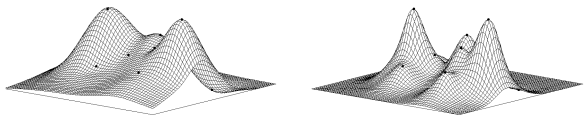
$$f_s \;=\; (1-t)\cdot f_k + t\cdot f_{k+1},$$

where $t = \frac{s-s_k}{s_{k+1}-s_k}$.

Fig. 4 shows examples. We have used a control lattice hierarchy with six levels to generate the surfaces in Fig. 3. Then, the surfaces in Figs. 3(b), (c), and (d) correspond to the scale parameter values $s = \frac{2}{3}, \frac{1}{2}, \frac{1}{3}$, respectively. Fig. 4(a) shows the surface f_s for $s = \frac{7}{12}$, which is the average of the surfaces in Figs. 3(b) and (c). In Fig. 4(b), the scale parameter s is $\frac{5}{12}$.

5.3. Boundary fixing

In an editing operation, the boundary of an editing area A is required to remain fixed. Let R be the rectangle on which the area A has been embedded. Let f be the function that interpolates scattered offsets Δx_i over R. To support the fixed

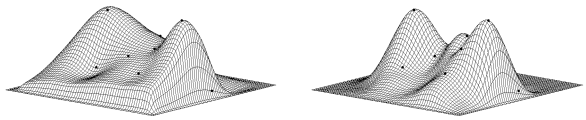


(a) $s = \frac{7}{12}$ (b) $s = \frac{5}{12}$

Figure 4: *Surfaces with different scale parameters.*

boundary of area A, the value of f should be zero at the boundary of R.

The function f obtained by multilevel B-spline interpolation is a uniform cubic B-spline function defined with the finest control lattice Φ_l [5]. Let $(m_l + 3) \times (m_l + 3)$ be the size of Φ_l. Let ϕ_{ij} be the value of the ij-th control point for $i, j = -1, 0, ..., m_l + 1$. Details of the control lattice placement can be found in reference[5]. We can simply make the values of f be zero at the boundary of R by assigning zero to boundary rows and columns of lattice Φ_l. That is, $\phi_{ij} = 0$ for $i = -1, 0, 1, m_l - 1, m_l, m_l + 1$ or $j = -1, 0, 1, m_l - 1, m_l, m_l + 1$. The resulting f may not interpolate an offset Δx_i near the boundary of R, but it is still C^2-continuous. Fig. 5 shows the modified versions of the surfaces in Figs. 3(b) and (c), which have fixed boundaries.



(a) $m_1 = 4$ (b) $m_1 = 8$

Figure 5: *Surfaces with fixed boundaries.*

6. Examples

Fig. 6 illustrates the editing process that has been performed to generate the mesh editing examples in Fig. 7. In Fig. 6(a), the gray region is the selected editing area. Fig. 6(b) shows the vertices selected as an editing primitive, where they are colored in black. Figs. 6(c) and (d) show the initial and moved positions of the editing primitive, respectively.

Fig. 7 shows the results of the mesh editing process given in Fig. 6. These results demonstrate the effects of the scale parameter and local frames on the updated mesh. Fig. 7(a) shows the original mesh. In Fig. 7(b), the nose of the mesh has been changed, where the scale parameter s is 0.75 and the same local frame is used for all vertices. When the scale parameter s is decreased to 0.25, we obtain the updated mesh in Fig. 7(c), which has a narrower nose than that of Fig. 7(b). Fig. 7(d) shows the result when s is 0.5 and the local frames based on normal vectors are used. In this case, the editing operation has moved the vertices in the editing area along different directions.

Fig. 8 shows another mesh editing examples generated by using the original mesh in Fig. 7(a). In Figs. 8(a) and (b), mesh editing has been used to pull out the left eye and to open the mouth of the face, respectively.

Table 1 summarizes the sizes of the editing areas and editing primitives used for generating the examples in Figs. 7 and 8. It also contains the processing time for harmonic embedding and editing operation, which has been measured on a PC (Pentium II 400MHz). The processing time for editing operation includes all the steps that are required to obtain a updated mesh when a user moves an editing primitive: 3D offset computation at an editing primitive, offset interpolation, and vertex position updates with interpolated offsets. The performance values in Table 1 show that the technique proposed in this paper runs fast enough to support interactive mesh editing.

7. Conclusions

In this paper, we have presented a novel approach to mesh editing that supports direct manipulation, multiresolution editing, editing area control, arbitrary mesh handling, intuitive response, and interactive editing. The approach is based on harmonic maps and multilevel B-spline interpolation. A harmonic map is used to embed an editing area onto a 2D rectangle. User-specified editing information at several vertices is propagated all over the editing area by applying multilevel B-spline interpolation.

Future work mainly relates with the 2D embedding of an editing area. The intuitiveness of an editing result can be enhanced by an embedding technique that preserves the ratios of edge lengths in the interior of an editing area. A harmonic map is guaranteed to have this property only at the boundary of an editing area.

Acknowledgments

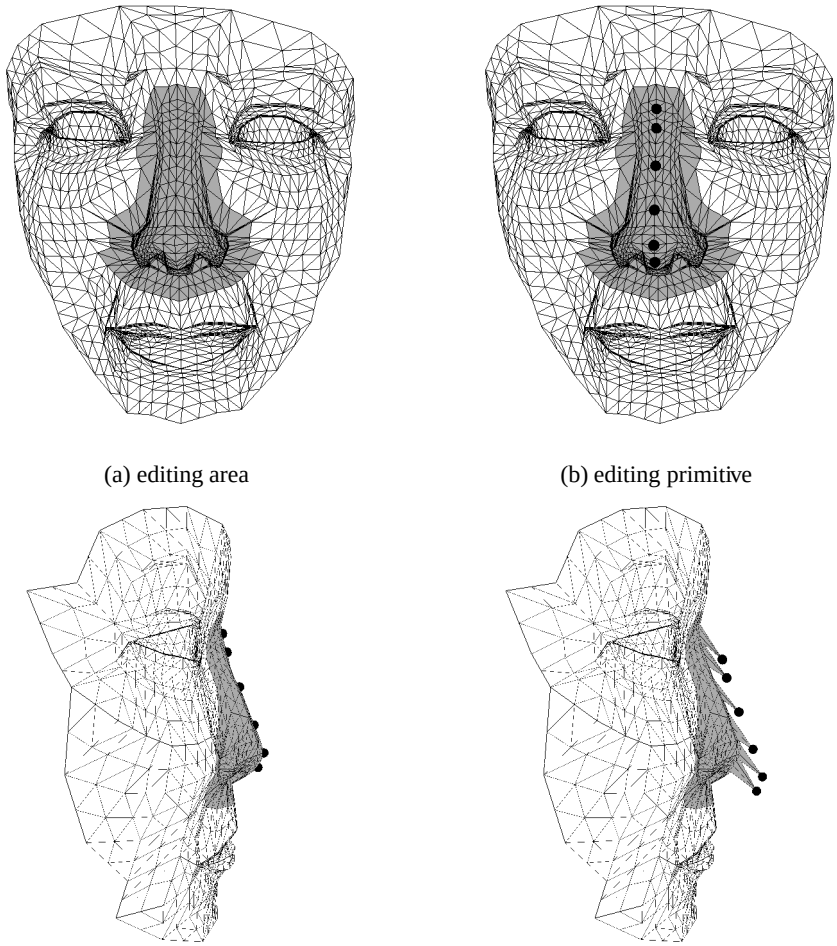
This work was supported in part by Ministry of Information and Communication under grant 2NI9890701. The author is grateful to Takashi Kanai for providing his harmonic map implementation. The author also owes thanks to Junho Kim and Kyuman Jeong for help with implementation and examples.

References

1. T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," *ACM Computer Graphics (Proc. of SIGGRAPH '86)*, **20**(4), pp. 151–160 (1986).

2. A. Finkelstein and D. H. Salesin, "Multiresolution curves," *ACM Computer Graphics (Proc. of SIGGRAPH '94)*, pp. 261–268 (1994).

3. D. Zorin, P. Schröder, and W. Sweldens, "Interactive multiresolution mesh editing," *ACM Computer Graphics (Proc. of SIGGRAPH '97)*, pp. 259–268 (1997).
4. M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes," *ACM Computer Graphics (Proc. of SIGGRAPH '95)*, (1995).
5. S. Lee, G. Wolberg, and S. Y. Shin, "Scattered data interpolation with multilevel b-splines," *IEEE Transactions on Visulaization and Computer Graphics*, **3**(3), pp. 228–244 (1997).
6. A. H. Barr, "Global and local deformations of solid primitives," *ACM Computer Graphics (Proc. of SIGGRAPH '84)*, **18**(4), pp. 21–30 (1984).
7. S. Coquillart, "Extended free-form deformation: A sculpturing tool for 3D geometric modeling," *ACM Computer Graphics (Proc. of SIGGRAPH '90)*, **24**(4), pp. 187–196 (1990).
8. H. J. Lamousin and W. N. Waggenspack, "NURBS-based free-form deformations," *IEEE Computer Graphics and Applications*, **14**(6), pp. 59–65 (1994).
9. R. MacCracken and K. I. Joy, "Free-form deformatoins with lattices of arbitrary topology," *ACM Computer Graphics (Proc. of SIGGRAPH '96)*, pp. 181–188 (1996).
10. A. Rappoport, A. Sheffer, and M. Bercovier, "Volume-preserving free-form solids," *IEEE Trans. Visualization and Computer Graphics*, **2**(1), pp. 19–27 (1996).
11. F. Lazarus, S. Coquillart, and P. Jancène, "Axial deformations: An intuitive deformation technique," *Computer-Aided Design*, **26**(8), (1994).
12. K. Singh and E. Fiume, "Wires: A geometric deformation technique," *ACM Computer Graphics (Proc. of SIGGRAPH '98)*, (1998).
13. D. R. Forsey and R. H. Bartels, "Hierarchical B-spline refinement," *ACM Computer Graphics (Proc. of SIGGRAPH '88)*, **22**(4), pp. 205–212 (1988).
14. S. J. Gortler and M. F. Cohen, "Hierarchical and variational geometric modeling with wavelets," in *Proc. of Symposium on Interactive 3D Graphics*, pp. 43–54, (1995).
15. A. W. Lee, W. Sweldens, P. Schroder, L. Cowsar, and D. Dobkin, "MAPS: Multiresolution adaptive parameterization of surfaces," *ACM Computer Graphics (Proc. of SIGGRAPH '98)*, (1998).
16. P. Cignoni, C. Montani, C. Rocchini, and R. Scopigno, "Zeta: A resolution modeling system," *Graphical Models and Image Processing*, **60**(5), pp. 305–329 (1998).
17. W. M. Hsu, J. F. Hughes, and H. Kaufman, "Direct manipulation of free-form deformations," *ACM Computer Graphics (Proc. of SIGGRAPH '92)*, **26**(2), pp. 177–184 (1992).
18. W. Welch and A. Witkin, "Variational surface modeling," *ACM Computer Graphics (Proc. of SIGGRAPH '92)*, **26**(2), pp. 157–166 (1992).
19. L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel, "Interactive multi-resolution modeling on arbitrary meshes," *ACM Computer Graphics (Proc. of SIGGRAPH '98)*, (1998).
20. J. R. Kent, W. E. Carlson, and R. E. Parent, "Shape transformation for polyhedral objects," *ACM Computer Graphics (Proc. of SIGGRAPH '92)*, **26**(2), pp. 47–54 (1992).
21. J. Maillot, H. Yahia, and A. Verroust, "Interactive texture mapping," *ACM Computer Graphics (Proc. of SIGGRAPH '93)*, pp. 27–34 (1993).
22. T. Duchamp, A. Certain, A. Derose, and W. Stuetzle, "Hierarchical computation of PL harmonic embeddings," tech. rep., University of Washington, (1997).
23. T. Kanai, H. Suzuki, and F. Kimura, "3D geometric metamorphosis based on harmonic map," in *Proc. of Pacific Graphics '97*, pp. 97–104, (1997).
24. H. Bao and Q. Peng, "Interactive 3D morphing," *Computer Graphics Forum (Proc. of Eurographics '98)*, **17**(3), pp. 23–30 (1998).
25. R. Franke and G. M. Nielson, "Scattered data interpolation and applications: A tutorial and survey," in *Geometric Modelling: Methods and Their Application* (H. Hagen and D. Roller, eds.), (Berlin), pp. 131–160, Springer-Verlag, (1991).



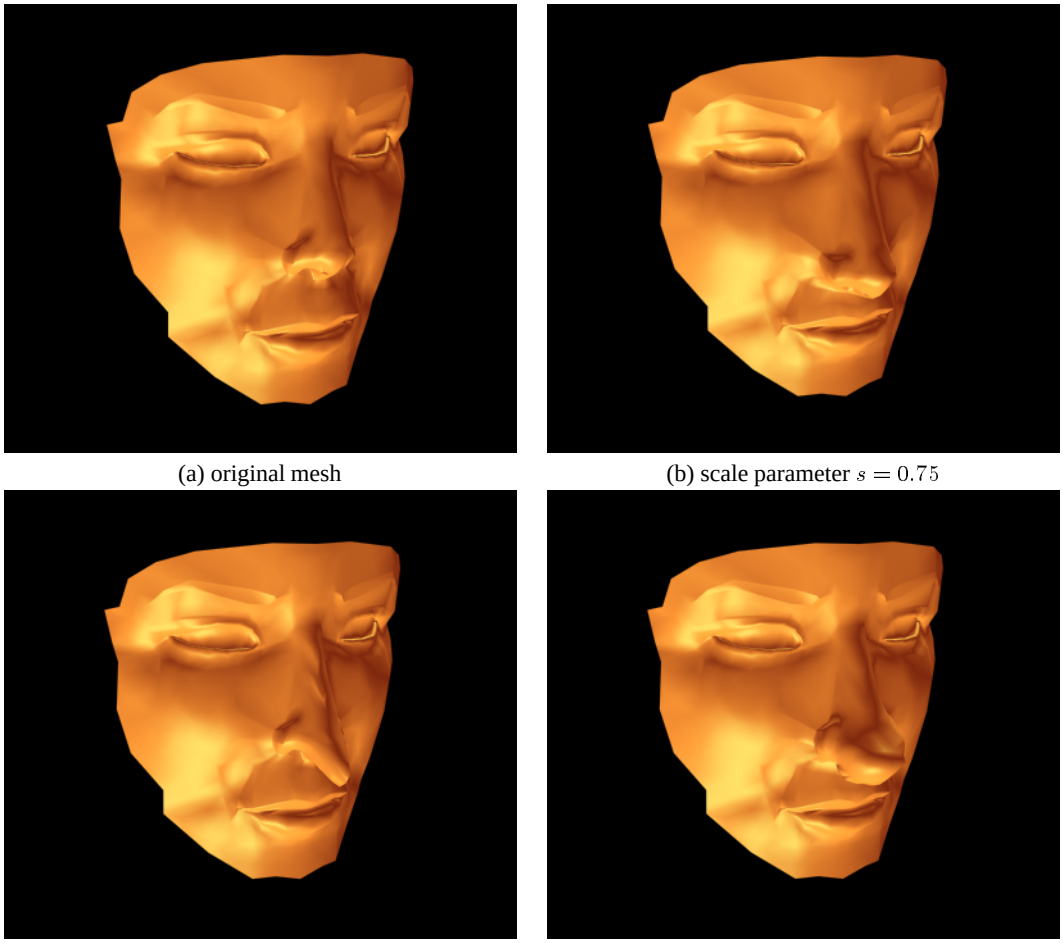
(a) editing area (b) editing primitive

(c) initial position of editing primitive (d) moved position of editing primitive

Figure 6: *The editing area and editing primitive used to generate the mesh editing examples in Figs. 7.*

	# of vertices in editing area	# of faces in editing area	# of vertices in editing primitive	processing time for harmonic embedding	processing time for editing operation
nose	530	992	6	266 millisecs	31 millisecs
eye	294	550	5	125 millisecs	30 millisecs
mouth	358	663	8	139 millisecs	32 millisecs

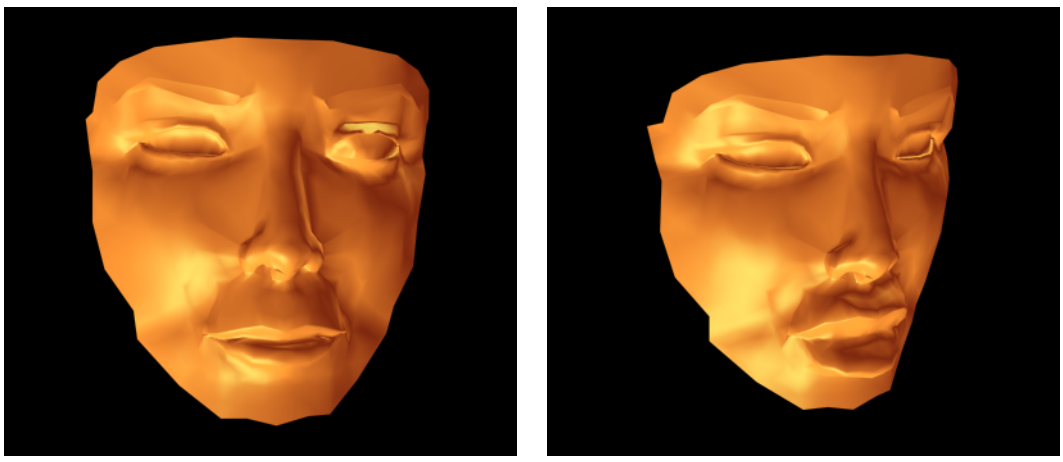
Table 1: *Performance values for the mesh editing examples in Figs. 7 and 8.*



(a) original mesh (b) scale parameter $s = 0.75$

(c) scale parameter $s = 0.25$ (d) local frames based on normal vectors

Figure 7: *Mesh editing examples, where the nose has been edited with different scale parameters and local frames.*



(a) (b)

Figure 8: *Mesh editing (a) to pull out the left eye and (b) to open the mouth.*