Efficient Fitting of Subdivision Surfaces by Simplified Newton Method

Masatake Higashi[1], Daisuke Mori, Shinji Seo and Masakazu Kobayashi

[1]Toyota Technological Institute, (higashi, sd08431, sd04045, kobayashi)@toyota-ti.ac.jp

ABSTRACT

In this paper, we present an efficient iterative fitting method of a Loop subdivision surface along with an approximation method of mesh data. We propose a simplified Newton method which uses errors at one neighbor control points and obtains the control mesh in order (n) for input points. We show the effectiveness of the method by calculations of some examples.

Keywords: fitting, subdivision surface, iterative calculation.
DOI: 10.3722/cadaps.2010.821-834

1 INTRODUCTION

To represent industrial products composed by free-form surfaces like automotive bodies and electrical appliances, tensor product surfaces such as non-uniform B-spline surfaces and Bézier patches have been widely used [8]. They can represent high-quality surfaces, but to connect them each other with continuity is rather difficult. Hence subdivision surfaces for arbitrary meshes are getting to be used in the field of computer graphics [7] because of their availability to any irregular meshes and automatic satisfaction of continuity [4]. In recent years, the set of tools available for manipulating subdivision surfaces has been growing steadily. Then, the use of subdivision-based representations for styling and conceptual design is being studied [19].

A triangular subdivision surface has been introduced by Loop [13] and subdivision rules were presented. Next, Hoppe et al. [11] proposed a method to automatically determine the topological type of the surface including the presence and location of sharp features, then sharp feature control was introduced by Bierman et al. [1]. A key ingredient of their method is a new subdivision surface scheme that allows the modeling of surface features such as corners, boundaries, creases, and darts. They also proposed a reconstruction method of subdivision surfaces optimizing the energy function after detecting sharp features. Since then, a lot of studies on fitting subdivision surfaces have been done as described in Section 2.

They have succeeded in obtaining exact least squares surfaces, but there remain two important problems. First, they need certain amount of calculation time: It requires about an hour for hundreds of thousands points and several minutes for tens of thousands points. Second, it is not easy to apply them to shapes of industrial products because they lack in controllability of surfaces keeping class-A smoothness. Hence we focus our research on the following two items.

- Efficient fitting method to obtain results in an interactive time
- High-quality surface satisfying boundary conditions at sharp features [10]

In this paper, we propose a method for the first item. The objective of our study is to obtain solutions within an interactive response time for tens of thousand input points. Hence the method approximates Hessian of the Newton method with constant values by grouping and averaging input points for the vertices of the simplified mesh.

2 RELATED WORK

In the method of Hoppe et al. [11], an initial dense mesh is generated from a set of unorganized points and is then decimated to fit the target shape via optimization of an energy function. Finally, a smooth subdivision surface is obtained from the mesh again via optimization. In [15], Marinov and Kobbelt perform closest point search on Loop's surface by combining Newton iteration and nonlinear minimization based on precise calculation of basis functions and surface points, followed by adaptively reconstructing the control mesh with respect to the L^∞ metric. Recently, square distance minimization has been proposed [16, 5, 6]. It can get quick convergence. In these methods, the geometric error between the fitting surface and the target shape needs to be measured or minimized, where the error function is defined by the summation of the squared distance between a point on the fitting surface and its closest point on the target shape.

Since the calculation of the least square method takes so much time, some simplified methods have been proposed. To obtain quick simplification of mesh, quadric error metrics (QEM) is applied by Garland and Heckbert [9]. Suzuki et al. [17] applied simplified fitting for approximating an input mesh and Litke et al. [12] introduced a quasi-interpolation of Catmull-Clark subdivision surfaces. These simplified methods tried to obtain quick solutions but sacrificed the quality of approximation. Maekawa et al. [14] introduced a geometrical iteration method for getting a high-quality fitted mesh.

3 ITERATIVE FITTING OF MESH

To obtain a control mesh from inputs points, we have to solve simultaneous equations for variables that are control vertices. Each equation represents a relation between an input point and control vertices around it: Limit points of control vertices are set to be equal to input points. The relation between limit points and control vertices is shown in Figure 1. When a number of input points is large such as tens or hundreds of thousands, the coefficient matrix becomes huge and computation time is too large to solve. Hence it is better to introduce an iterative method to solve the equations. Maekawa [14] introduced a method which calculates displacements by making perpendicular lines to tangents plane, so it takes rather a larger time. We propose an iterative method which gets quick and good convergence of a control mesh by using errors at one-ring neighbor vertices instead of using errors only at a corresponding vertex.

We use one-ring neighbor vertices for calculating a central control vertex, because the effects of two-ring neighbor vertices are negligible small. Let input point be s and corresponding control vertex be v and their one-ring neighbors be s_i and v_i as shown in Fig. 1. An equation which updates a new vertex is introduced as the sum of errors at one-ring neighbor vertices:

$$v' = v + \Delta v, \tag{1}$$

$$\Delta v = \lambda_1 (s - v^\infty) + \lambda_2 \sum_{i=1}^{m} (s_i - v_i^\infty).$$

Here, v^∞ and v_i^∞ are limit points for control vertices v and v_i, respectively. Coefficients λ_1 and λ_2 are obtained calculating an inverse matrix of the matrix representing the relation between input points and control vertices within one-ring. Weights for outside of one-ring are set zero. The matrix is shown below for a regular vertex whose valence is 6.

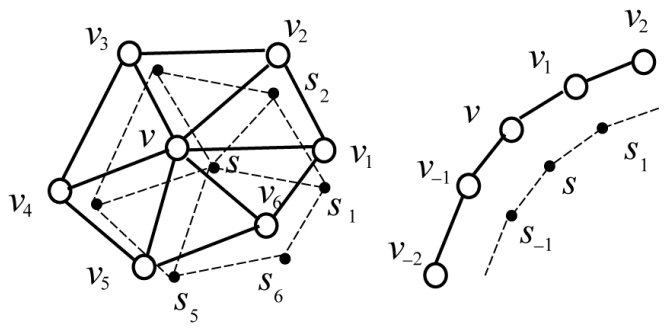


Fig. 1: Iterative calculation of vertices.

$$\begin{pmatrix} s \\ s_1 \\ \vdots \\ s_6 \end{pmatrix} = \begin{pmatrix} \frac{6}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \\ \frac{1}{12} & \frac{6}{12} & \frac{1}{12} & 0 & 0 & 0 & \frac{1}{12} \\ & & \vdots & & & & \\ \frac{1}{12} & \frac{1}{12} & 0 & & \cdots & & \end{pmatrix} \begin{pmatrix} v \\ v_1 \\ \vdots \\ v_6 \end{pmatrix}. \tag{2}$$

The inverse matrix of this one for updating a vertex is

$$\begin{pmatrix} \Delta v \\ \Delta v_1 \\ \vdots \\ \Delta v_6 \end{pmatrix} = \begin{pmatrix} \frac{16}{7} & -\frac{2}{7} & -\frac{2}{7} & -\frac{2}{7} & -\frac{2}{7} & -\frac{2}{7} & -\frac{2}{7} \\ -\frac{2}{7} & \frac{151}{70} & -\frac{23}{70} & \frac{1}{10} & \frac{1}{70} & \frac{1}{10} & -\frac{23}{70} \\ & & \vdots & & & & \\ \frac{1}{12} & \frac{1}{12} & 0 & & \cdots & & \end{pmatrix} \begin{pmatrix} \Delta s \\ \Delta s_1 \\ \vdots \\ \Delta s_6 \end{pmatrix}.$$

Only the first line is important for the iterative calculation for obtaining a control vertex. From this we get coefficients:

$$\lambda_1 = \frac{16}{7} = 2.2857, \quad \lambda_2 = -\frac{2}{7} = -0.2857.$$

In Suzuki et al. [17], they don't use λ_2 and set λ_1 to 1.

Similarly, we obtain coefficients for vertices with the other valences as shown in Tab. 1. And for crease vertices, we can calculate matrix similarly using neighbor vertices and obtain coefficients. The increment for a regular crease vertex is

$$\Delta v = \frac{45}{26}(s - v^{\infty}) - \frac{12}{26}\sum_{i=1,-1}(s_i - v_i^{\infty}) - \frac{3}{26}\sum_{i=2,-2}(s_i - v_i^{\infty})$$

These coefficients can be calculated beforehand, so we only calculate linear summation with constant coefficients in each iteration for each control point.

valence	3	4	5	6	7	8
λ_1	4.0000	3.0692	2.5668	2.2857	2.1158	2.0057
λ_2	–1.0000	–0.6023	–0.3974	–0.2857	–0.2193	–0.1766

Tab. 1: Coefficients for iterative calculation.

The update is repeated until Δv is less than the given tolerance. The calculation is $O(n)$ for a number of input points, and the number of iterations of the executed examples is about ten for a small tolerance.

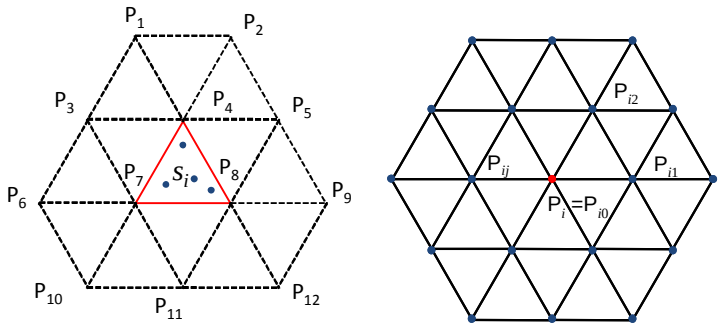
4 APPROXIMATION OF MESH DATA BY SIMPLIFIED NEWTON METHOD

4.1 Optimization of the L^2 Metric

Next, we examine the problem of efficiently finding the best L^2 approximation of a given set of samples S = { $s_1, \cdots, s_M$ } by a Loop subdivision surface D with fixed number (N) of control points and connectivity. Since S is a discrete set, the L^2 error is expressed by the following sum:

$$L^2(\{s_i\}, D) = \sum_{i=1}^{M} (\boldsymbol{s}_i - \boldsymbol{D}(t_i))^2 \tag{3}$$

where $\{t_i\}$ are the parameter values assigned to the samples $\{s_i\}$ with respect to some parameterization of the surface D. The most common way is to use barycentric coordinates with respect to the triangles of the base mesh C_0, i.e., $t_i =< f_i, (u_i, v_i, w_i) >$, where $f_i \in T_0$ indicates the patch to which the sample s_i is mapped and (u_i, v_i, w_i), $u_i+v_i+w_i=1$ define the barycentric coordinates of t_i within this triangle.



(a) (b)

Fig. 2: Relation of control points of triangle. (a) A central patch influenced by 12 control points. (b) Influence of the central control point to surrounding patches.

Given a fixed correspondence $s_i \leftrightarrow t_i$, the problem of minimizing (3) is solved in the *least squares* sense by finding that solution P_0 which minimizes the L^2 residuum of the over-determined linear system

$$A\,P_0 = S. \tag{4}$$

In order to compute the matrix $A = [\phi_j(t_i)]_{M\times N}$ for arbitrary parameterizations $\{t_i\}$, we have to evaluate the basis functions $\{\phi_1, \cdots, \phi_N\}$ which define D at $\{t_1, \cdots, t_M\}$. Solving the system (4) gives us the optimal position of the control vertices P_0 of D. The sparsity of the matrix A depends on the support of the basis functions ϕ_j. In the case of Loop subdivision surfaces, each patch (corresponding to one triangle of the base mesh) is affected by 12 control vertices on average; For a regular mesh, it is affected by exactly 12 control points as shown in Fig. 2a. Subdivision surface D for patch f_j is represented by control points P_i affecting patch f_j and barycentric coordinates $t_k \in f_j$:

$$\boldsymbol{D}(u_k, v_k, w_k) = \sum_{i=1}^{12} \boldsymbol{P}_i \phi_i(u_k, v_k, w_k) \tag{5}$$

To minimize Eqn. (3), we differentiate it by control point $\boldsymbol{P}_i$ and set each equation to zero.

$$E_i = \frac{\partial L^2}{\partial \boldsymbol{P}_i} = \sum_{k=1}^{M} \phi_i(t_k)(\boldsymbol{D}(t_k) - \boldsymbol{s}_k) = 0 \tag{6}$$

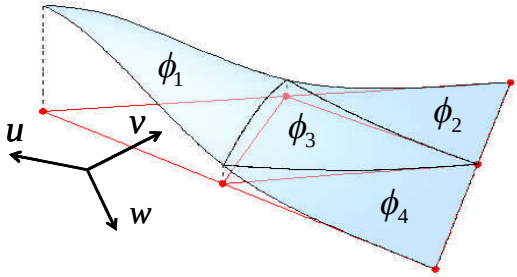


Fig. 3: Basis function for triangular spline.

Since control point $\boldsymbol{P}_i$ affects within two-ring neighbors, it affects 24 patches for a regular mesh as shown in Fig. 2b and the number of summation in Eq. (6) corresponds to that of the points in these patches.

$$\begin{aligned} E_i &= \sum_{k=1}^{M} \phi_i(t_k)(\boldsymbol{D}(t_k) - \boldsymbol{s}_k) \\ &= \sum_{k=1}^{M} \left(\boldsymbol{P}_{i0}(\phi_{i0}(t_k))^2 + \sum_j \boldsymbol{P}_{ij}\phi_{i0}(t_k)\phi_{ij}(t_k) - \boldsymbol{s}_k\phi_{i0}(t_k) \right) \\ &= \boldsymbol{P}_{i0}\sum_k (\phi_{i0}(t_k))^2 + \sum_j \boldsymbol{P}_{ij}\sum_k \phi_{i0}(t_k)\phi_{ij}(t_k) - \sum_k \boldsymbol{s}_k\phi_{i0}(t_k) \end{aligned} \tag{7}$$

Here, function ϕ_{ij} is a basis function for control points $\boldsymbol{P}_{ij}$, which calculates surface points for parameter t_k. $\boldsymbol{P}_{ij}$ is a control point within two-neighbors of control point $\boldsymbol{P}_i$ as shown in Fig. 2b. Fig. 3 shows shapes of 4 functions out of 12. Four functions are calculated from box spline [2, 3] as follows:

$$\begin{aligned} \phi_1(u,v,w) &= \frac{1}{12}(1+4u+6u^2-4u^3-u^4+2w+6uw-6u^2w-2u^3w-12uw^2-4w^3+4uw^3+2w^4) \\ \phi_2(u,v,w) &= \frac{1}{12}u^3(u+2w) \\ \phi_3(u,v,w) &= \frac{1}{12}(1-2u+2u^3-u^4+2w-6uw-6u^2w-2u^3w-4w^3+4uw^3+2w^4) \\ \phi_4(u,v,w) &= \frac{1}{12}(2u^3-u^4-2u^3w) \end{aligned} \tag{8}$$

The other functions are represented by symmetrically substituting barycentric parameters: *u*, *v*, and *w*, in the above four functions.

4.2 Procedures of Calculation

We have to solve a linear system of a huge matrix concerning Eqn. (6) and its elements consist of weighting functions corresponding to input points and against so many control points, 12 points on the average. It is desirable to omit some calculations if the convergence of iterations is assured. Therefore, we introduce a group of input points and calculate control points using the average error values for the points in the group as well as the average function values.

The procedure of calculating control points by the least square method is as follows.

1. Generation of an initial simplified mesh by the QEM method [9].

The calculation is repeated until the number of mesh points is less than designated one, *N*. The grouping of input points corresponding to the mesh points is obtained at the same time.

2. Calculate the initial control mesh by the fitting method in Section 3.
3. Calculate L^2 error and if the difference in iterated process is less than the given tolerance then quit.

   The distances from input points to the corresponding perpendicular surface points are calculated along with their parameter values.
4. Update each control point using average values of functions and errors for the neighbor groups of input points, and repeat step 3 and 4.

The calculation in step 4 is described in Section 4.3. It does not calculate the Hessian of the Newton method, neither does update the control mesh using inverse matrix of Hessian H, which is a sparse matrix consisting of rows corresponding to $\boldsymbol{P}_i$:

$$\left(\sum_k (\phi_{i0}(t_k))^2, \sum_k \phi_{i0}(t_k)\phi_{i1}(t_k), \cdots, \sum_k \phi_{i0}(t_k)\phi_{in}(t_k) \right). \tag{9}$$

4.3 Simplified Calculation of Newton Method

The calculation of elements of H needs the sum of weighting functions for the input points affected by $\boldsymbol{P}_i$. The functions are quartic polynomials $\phi_i(t_k)$ of parameter $t_k =< f_i, (u_k, v_k, w_k) >$. The shape of the function around $\boldsymbol{P}_i$ is flat near the center and almost zero near the control points of 2-ring neighbors of $\boldsymbol{P}_i$ as shown in Fig. 3. Hence we neglect the function for the 2-neighbor control points. Further, we use the average of the function values with respect to each group of input points instead of calculating the sum of exact function values. By considering the average of error distances of input points in each group, we can omit even the calculations of the functions themselves. Accordingly, we obtain linear equations for control points $\boldsymbol{P}_i$s from the simplified Hessian matrix.

The groups of input points are determined in the simplification process in the QEM method. The points included in one group G_i are those omitted by joining them to one mesh point. Then, they are located almost in the Voronoi region of the control point $\boldsymbol{P}_i$ as shown in Fig. 4.

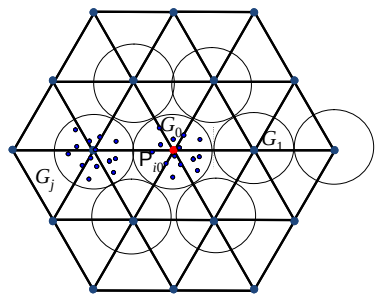


Fig. 4: Group of input points.

We denote the average of the function values ϕ_i for $\boldsymbol{P}_i$, for the input points included in G_j by $\bar{\phi}_i^{\,j}$, then we get

$$\bar{\phi}_i^{\,j} = \frac{1}{M_j} \sum_{k \in G_j} \phi_i(t_k). \tag{10}$$

Similarly we denote the average of ϕ_{ij} by $\bar{\phi}_{ij}$, using $\bar{\phi}_{ij}^{\,l}$ which is the average of $\bar{\phi}_{ij}$ for the points in the group G_l as

$$\bar{\phi}_{ij} = \sum_l \bar{\phi}_{ij}^l, \qquad \bar{\phi}_{ij}^l = \frac{1}{M_l} \sum_{k \in G_l} \phi_{i0}(t_k)\phi_{ij}(t_k). \tag{11}$$

Then, we get $\bar{E}_i$, the average value of errors, by calculating the product of the average values.

$$\bar{\boldsymbol{E}}_i = \sum_{j=0} \left(\bar{\phi}_i^{\,j} \cdot \frac{1}{M_j} \sum_{k \in G_j} (\boldsymbol{D}(t_k) - \boldsymbol{s}_k) \right). \tag{12}$$

The equation for control point $\boldsymbol{P}_i$ (=$\boldsymbol{P}_{i0}$) is

$$\bar{\phi}_{i0}\boldsymbol{P}_{i0} + \bar{\phi}_{i1}\boldsymbol{P}_{i1} + \cdots + \bar{\phi}_{in}\boldsymbol{P}_{in} = \bar{\boldsymbol{E}}_i. \tag{13}$$

Hence the row elements of H become as follows using these values:

$$\begin{pmatrix} \cdots, \bar{\phi}_{i0}, \bar{\phi}_{i1}, \cdots, \bar{\phi}_{in}, \cdots \end{pmatrix} \begin{pmatrix} \\ \boldsymbol{P}_i \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \bar{\boldsymbol{E}}_i \\ \\ \end{pmatrix}. \tag{14}$$

We can approximate the surface using the average values of the functions in each group, if the number of the input points in a group is large and distribution of the input points is well-balanced. Then, we need not calculate the function values, but we use estimated average values from the functions. We obtain average values for $\bar{\phi}_i^l$ and $\bar{\phi}_j^l$ for the groups of center and 1-ring neighbor control points, respectively. The values for $\bar{\phi}_{ij}$ are approximated as the product of the two average values of the functions:

$$\bar{\phi}_{ij} = \sum_l \bar{\phi}_{ij}^l \approx \sum_l \bar{\phi}_i^l \cdot \bar{\phi}_j^l. \tag{15}$$

The functions for valences 5, 6, and 7, which are obtained using the method proposed by Bolz and Shröder, are shown in Fig. 5 for an inner mesh point. For a regular mesh, the value of $\bar{\phi}_i^{\,j}$ is the same to $\bar{\phi}_j^0$, but for non-regular meshes, it is not symmetric at the vertex whose valence is not 6. Red circles show the vertices whose valences are not 6. In Fig. 5, we show values $\bar{\phi}_j^0$ at the non-regular vertex. The average values are shown in Tab. 2. Here, the values in the parentheses show the values of the parameter (u, v, w) at the control points as shown in Fig. 5.

For the mesh points on sharp features such as crease and corner, we use other types of weighting functions as shown in Fig. 6. The middle row of the figure shows a weighting function of a control point on a sharp feature and the upper row shows that of inner control points of the feature. The lower row is functions at a corner. Red lines show sharp features and red circles show control points with different values of valence.

We obtain equations of control points using an inverse matrix whose elements are values of $\bar{\phi}_{i0}$s and $\bar{\phi}_{ij}$s in Tabs. 2 and 3 as similarly as in Sec. 3. $\boldsymbol{P}_i$ is calculated from average errors of one neighbor control points.

$$\boldsymbol{P}_i' = \boldsymbol{P}_i + \Delta\boldsymbol{P}_i, \qquad \Delta\boldsymbol{P}_i = \sum_j \lambda_j \bar{\boldsymbol{E}}_j. \tag{16}$$

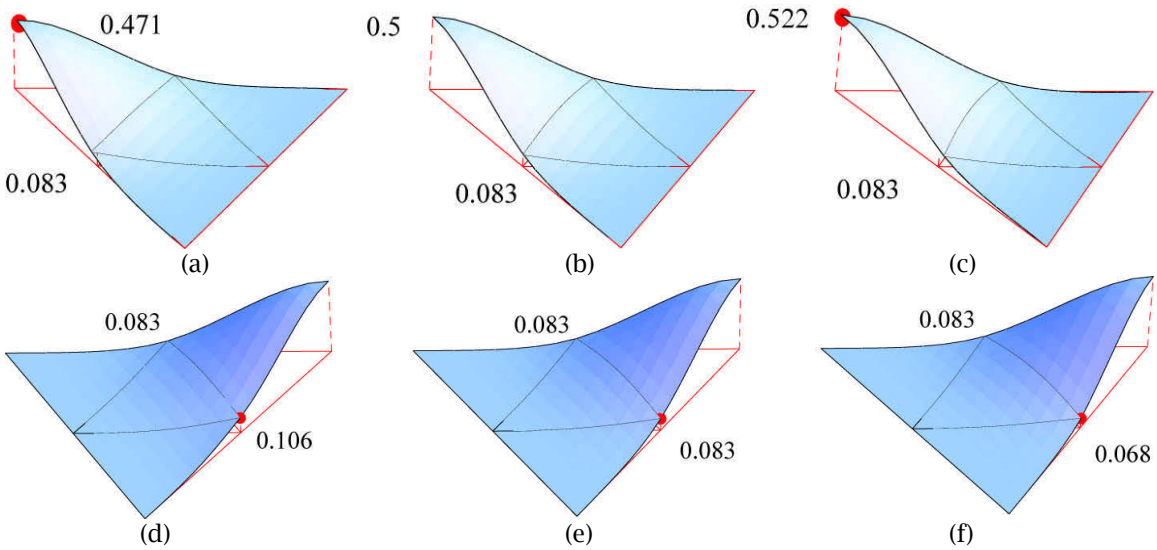


Fig. 5: Weighting functions for different valences. Upper row is for ϕ_0^i and lower row is for ϕ_j^i. The valence of (a) and (d) is 5, that of (b) and (e) is 6 and that of (c) and (f) is 7.

Valence	$\bar{\phi}_i^0$	$\bar{\phi}_i^j$	$\bar{\phi}_j^0$	$\bar{\phi}_{i0}$	$\bar{\phi}_{ij}$
4	0.350 (0.436)	0.100 (0.083)	0.170 (0.141)	0.162	0.119
5	0.380 (0.471)	0.100 (0.083)	0.120 (0.106)	0.194	0.106
6	0.400 (0.500)	0.100 (0.083)	0.100 (0.083)	0.220	0.100
7	0.420 (0.522)	0.100 (0.083)	0.080 (0.068)	0.236	0.094
8	0.450 (0.539)	0.100 (0.083)	0.070 (0.058)	0.262	0.092

Tab. 2: Average values of weighting functions

Valence	$\bar{\phi}_i^0$	$\bar{\phi}_i^j$	$\bar{\phi}_i^j$ (crease)	$\bar{\phi}_j^0$	$\bar{\phi}_{i0}$	$\bar{\phi}_{ij}$
3	0.540 (0.667)	0.220 (0.083)	0.290 (0.167)	0.150 (0)	0.508	0.482
4	0.540 (0.667)	0.220 (0.083)	0.290 (0.167)	0.150 (0)	0.557	0.570
5	0.540 (0.667)	0.220 (0.083)	0.290 (0.167)	0.150 (0)	0.605	0.658
3 (corner)	0.850 (1.000)	0.300 (0.083)	0.370 (0.167)	0.150 (0)	1.086	0.462

Tab. 3: Average values of weighting functions on sharp features

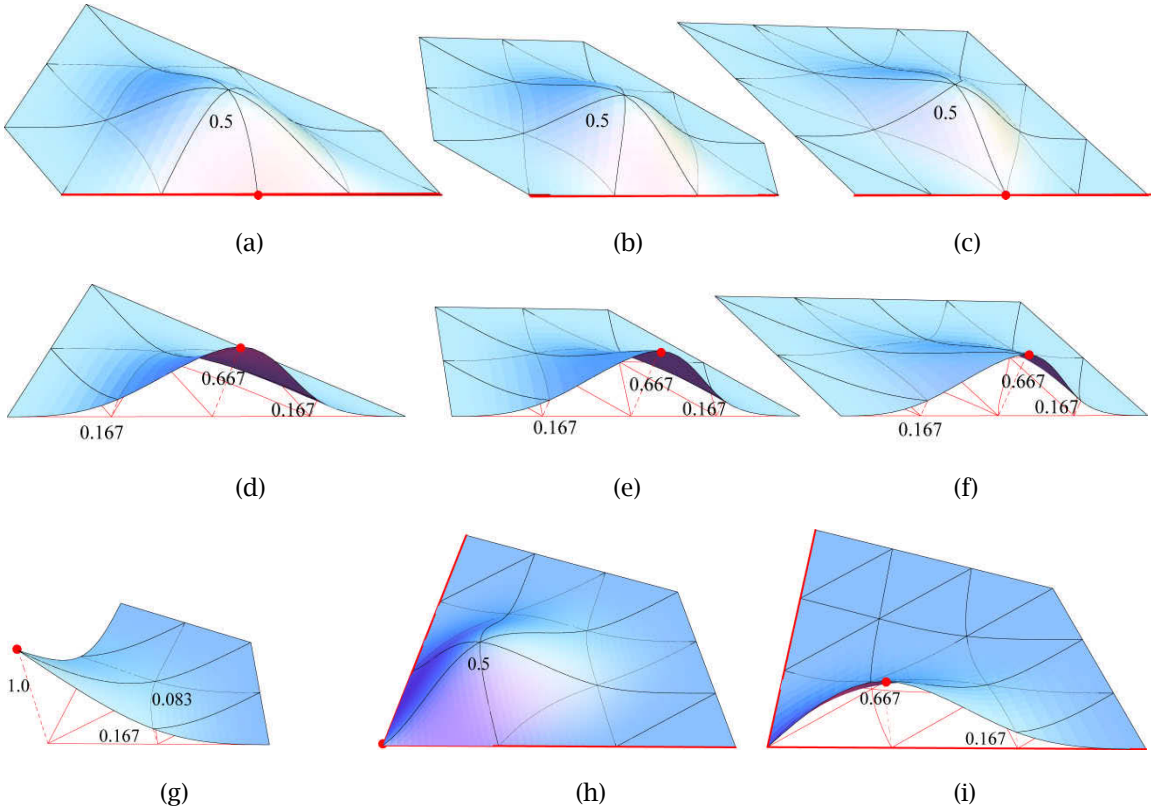
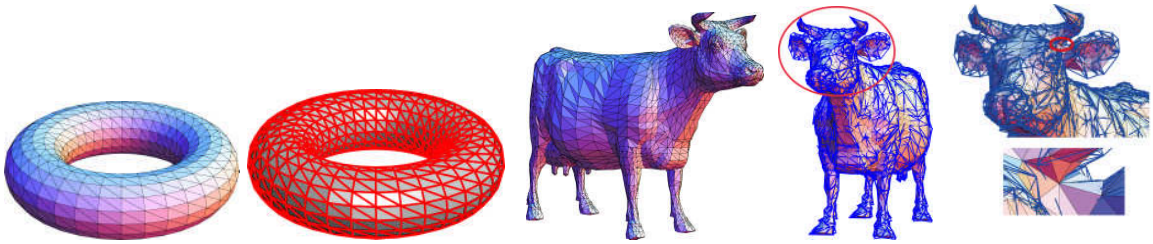


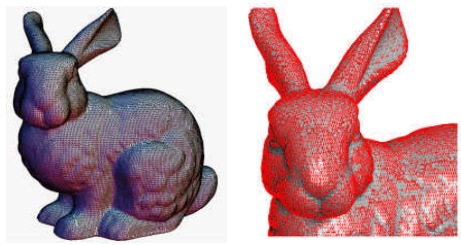
Fig. 6: Weighting functions on a crease or a boundary with different valences. The middle row is for ϕ_i on a crease and the upper row is for ϕ_j on an inner point. The valence of (a) and (d) is 3, that of (b) and (e) is 4 and that of (c) and (f) is 5. The lower row is functions at a corner. (g) is for ϕ_i , and (h) and (i) are for ϕ_j .

5 EXAMPLES

First, we show a few examples of fitting subdivision surfaces for the measured data which consist of a great number of points. We pick up three shapes: a torus, a cow and Stanford bunny (see Figure 7a ~ c). The numbers of input points are 576, 2,904 and 34,835, respectively, and the sizes of the shapes are set to 1. The calculation is done on a PC with 3.0 GHz CPU and 2.0 GB memory. In the result of cow in Fig. 7b, the fitting was successful although the valence of the center vertex is 22. The calculation time for generation of data structures (DS) and fitting of points are shown in Tab. 4. The calculation was conducted for two values of tolerances for the distance between an input point and a limit point: 1.0×10^{-4} and 1.0×10^{-7}, where the number of iterations for 1.0×10^{-7} does not increase by three times. The time is less than half second for 35 thousands input points, which is faster more than 100 times than that for the similar number of input points in Maekawa's method [14].



(a) Torus (b) Cow



(c) Stanford bunny

Fig. 7: Input data and obtained control mesh.

Model	No. pts	Time for DS generation (s)	Time for calculation			
			Tol = 10^{-4}		Tol = 10^{-7}	
			No. iteration	Time (s)	No. iteration	Time (s)
torus	576	0.031	3	0.000	10	0.000
cow	2,904	0.093	4	0.016	11	0.032
bunny	34,835	0.656	4	0.141	11	0.407

Tab. 4: Fitting calculation.

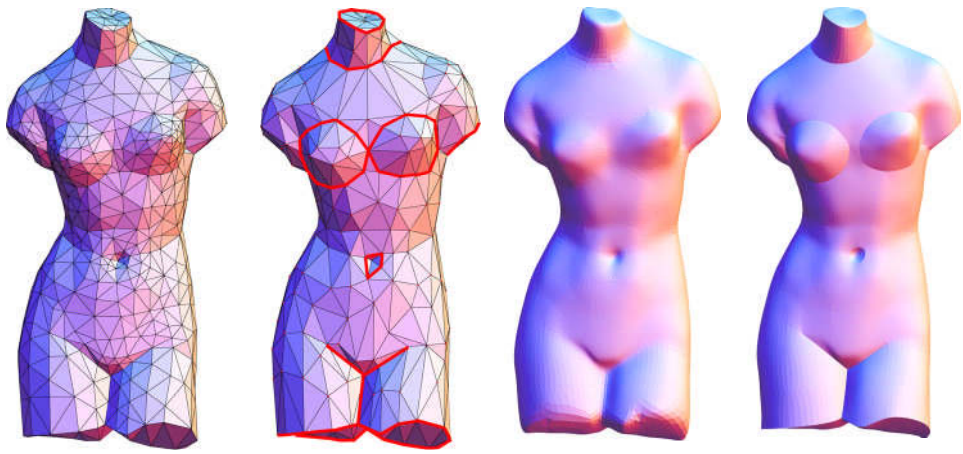
Next, we show examples of approximation calculation. We compare the results of our method (Simplified Newton method: S-Newton) with two methods; One is the steepest descent method (SDM) and the other is the Newton method. In the Newton method, we obtain control points from the inverse matrix of the Hessian matrix, and in SDM, we calculate $\boldsymbol{P}_i$ from E_i divided by the squared sum of ϕ_{i0} in Eqn. (7). Distances from input points to a subdivision surface are calculated using a sufficiently refined mesh from the intermediate control mesh. We used two-times subdivision in this experiment. Object shapes for calculations are a statue of Venus, torus and cow which are used for the above fitting; Their sizes are normalized to 1. The tolerance of the convergence for E is 10^{-3}.

Fig. 8 shows input mesh, simplified mesh by QEM and approximation results of the subdivision surface for a Venus statue. We specified sharp features of the control mesh as shown with red lines before the approximation calculation. Left figure of (c) is the result without specifying sharp features, and the right one is with sharp features which is approximated with clear creases. Fig. 9 shows an example of a torus. Figure (a) shows an input mesh and (b) control mesh obtained by QEM. QEM generated an unbalanced mesh for such an uneven shape with negative Gaussian curvature as pointed out by Maekawa et al. in [14], so the fitting result had small distortion as shown in (d). However, when we used a regularly compressed input, we obtained a balanced initial control mesh as shown in (c) and a good-quality result without undulations as shown in (e) and (f). This is shown by that the maximum error of input points is reduced from 0.017 for the initial control mesh to 0.00023 for the final one, although it is from 0.12 to 0.013 for the QEM control mesh. We need further consideration of a

method for getting an appropriate initial control mesh instead of QEM. Fig. 10 shows an example for a cow model. The number of the input points is reduced from 2,904 to 502.

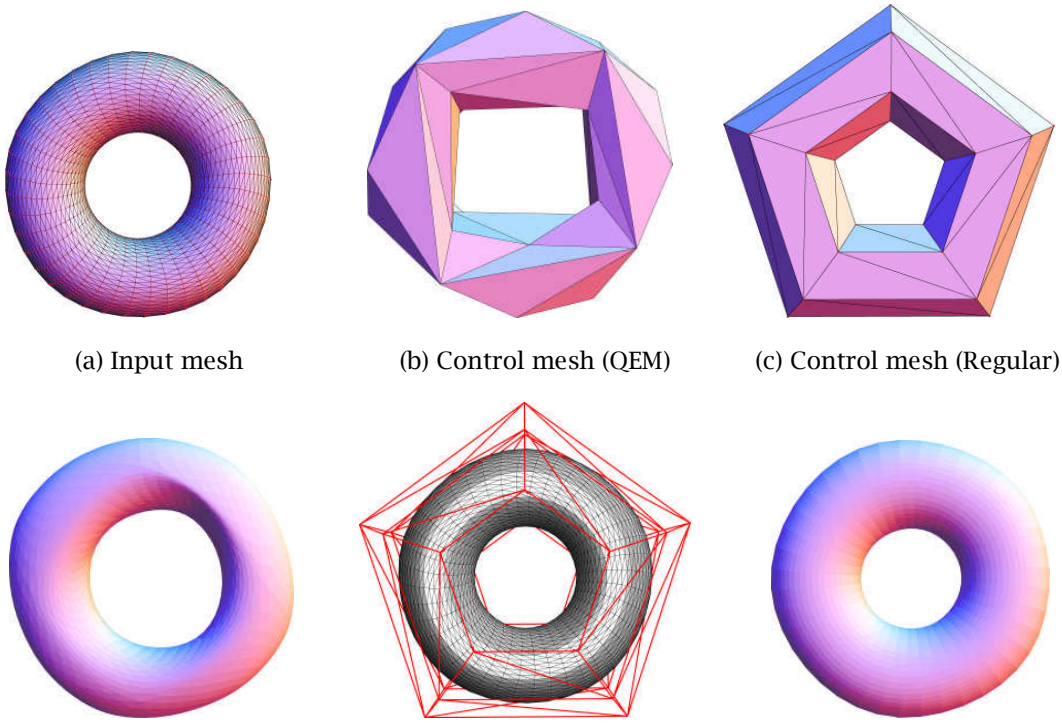
Fig. 11 shows the convergence of Venus, torus and cow for E (sum of squared distances) with respect to a number of iterations. Tab. 5 summaries the calculation results showing numbers of input points and control points, computation time, the sum of squared distances, and max and average error distances. The computation time is rather large because we executed the calculation by the program written in Mathematica for the confirmation of the algorithm.

For the number of iterations, our method is almost the same to the Newton method and half of SDM, and the convergence value of E is getting to the same when the ratio of control points to input points is getting larger. The squared sum E of the Newton method is smallest and that of our method is smaller than that of SDM. The maximum error of input points to the subdivision surface of our method is bigger than that of the Newton method, because we use estimated average values of the functions for input points. Since the calculation in Eqn. (16) is $O(N)$, the computation time of our method is smaller than those of the other methods and the ratio is getting much smaller according to the increase of the number of input points. Especially, the calculation of inverse matrix in the Newton method is getting large as $O(N^3)$ of control points although the time is small when the size of the sparse matrix is not big. We will confirm the efficiency by implementing a C program and performing experiments.



(a) Input mesh (b) Control mesh (c) Approximated surface
(1) Without creases (2) With creases

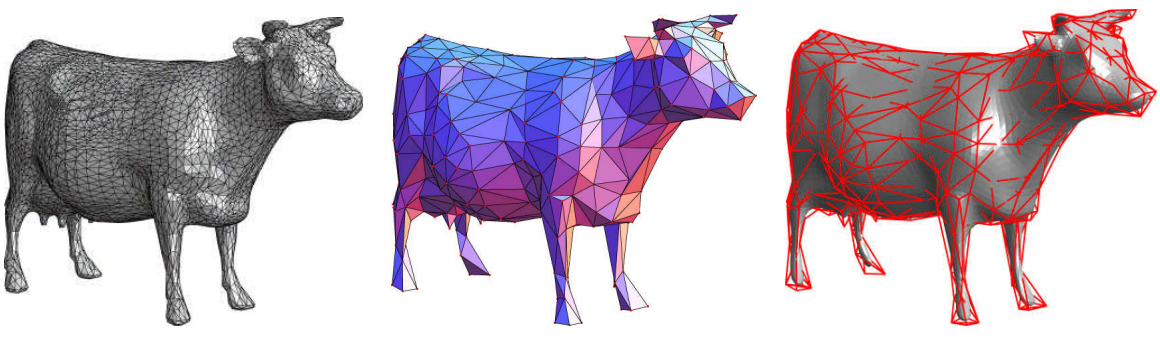
Fig. 8: Approximation of Venus statue. Red lines show specified creases.



(a) Input mesh (b) Control mesh (QEM) (c) Control mesh (Regular)

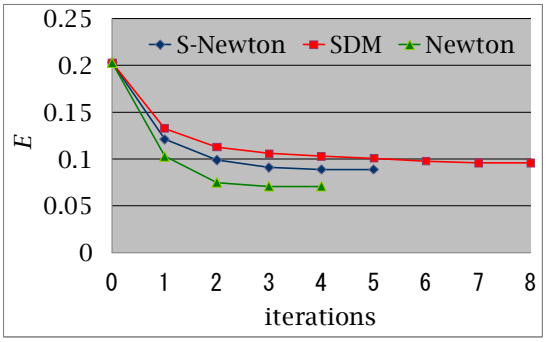
(d) Approximated surface (QEM) (e) Control mesh (f) Approximated surface (Regular)

Fig. 9: Approximation of torus.

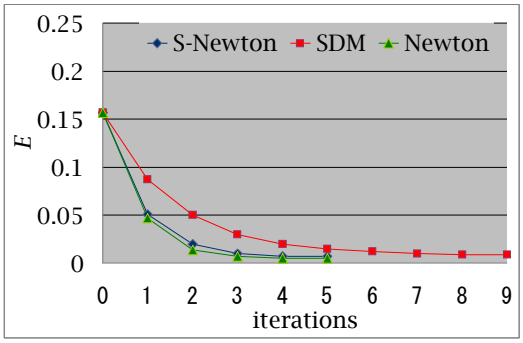


(a) Input mesh (b) Control mesh (c) Approximated surface

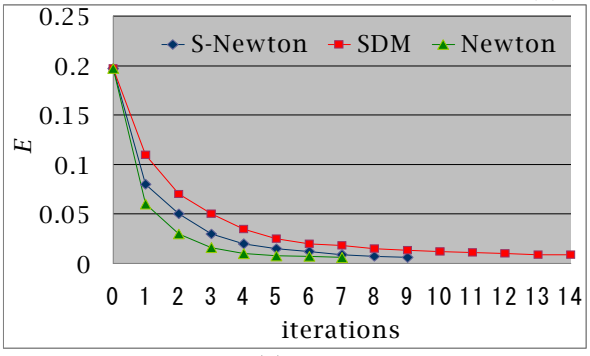
Fig. 10: Approximation of Cow.



(a) Venus



(b) Torus



(c) Cow

Fig. 11: Convergence of iterative calculation.

	Venus		No. input points: 711 No. control points: 352			Torus		No. input points: 576 No. control points: 60		
	No. iteration	E	Emax	Eave	Time (s)	No. iteration	E	Emax	Eave	Time (s)
S-Newton	5	0.089	0.0027	0.00032	8.95	5	0.0074	0.00023	0.000029	1.79
SDM	8	0.096	0.0037	0.00043	13.72	9	0.0091	0.00031	0.000041	3.65
Newton	4	0.071	0.0019	0.00021	25.12	5	0.0052	0.00014	0.000018	4.72

	Cow		No. input points: 2,904 No. control points: 502		
	No. iteration	E	Emax	Eave	Time (s)
S-Newton	9	0.0062	0.00041	0.000057	16.3
SDM	14	0.0088	0.00064	0.000079	22.1
Newton	7	0.0059	0.00036	0.000041	52.9

Tab. 5: Approximation results

6 CONCLUSION

We have proposed an efficient fitting and approximation method of a Loop subdivision surface for a triangular mesh. The method obtains satisfactory quality of approximation in a short computation time ($O(N)$), because the method is simplified from the Newton method using average distances of input points at one neighbor control points and it omits exact calculations of basis functions of subdivision surfaces by applying their average values for grouped input points.

Our future work includes

- to apply our methods to practical data by implementing C Program of our algorithm,
- to get the subdivision surface for the specified error.

For the latter item, we insert control points where the errors are larger than the given tolerance. We will easily get a control mesh using the group data obtained in the simplified process in the QEM.

This work was partly supported by KAKENHI (20560139).

REFERENCES

[1] Biermann, H., Levin, A., Zorin, D.: Piecewise Smooth Subdivision Surfaces with Normal Control, Proc. SIGGRAPH2000, 2000, 113-120.

[2] Boem, W.: The De Boor Algorithm for Triangular splines, in: Surfaces in Computer Aided Geometric Design, North-Holland Publishing Company, 1983, 109-120.

[3] Boem, W.: Generating the Bezier Points of Triangular Spline, in: Surfaces in Computer Aided Geometric Design, North-Holland Publishing Company, 1983, 77-91.

[4] Botsch, M., Pauly, M., Kobbelt, L., Alliez, P., Lévy, B., Bischoff, S., Rössl, C.: Geometric Modeling Based on Polygonal Meshes, SIGGRAPH07 Course a1, 2007.

[5] Cheng, K.-S.D., Wang, W., Qin, H., Wong, K.-Y.K., Yang, H. and Liu, Y.: Fitting Subdivision Surfaces to Unorganized Point Data Using SDM, Proc. 12th Pacific Conf. Computer Graphics and Applications (PG '04) , 2004, 16-24.

[6] Cheng, K.-S.D., Wang, W., Qin, H., Wong, K.-Y.K., Yang, H. and Liu, Y.: Design and Analysis of Optimization Methods for Subdivision Surface fitting, IEEE Trans. On Visualization and Graphics, 13(5), 2007, 878-890.

[7] DeRose, T., Kass, M., Truong, T.: Subdivision Surfaces in Character Animation, Proc. SIGGRAPH98, 1998,. 85-94.

[8] Farin, G.: Smooth Interpolation to Scattered 3D Data, in: Surfaces in Computer Aided Geometric Design, North-Holland Publishing Company, 1983, 43-63.

[9] Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics, Computer Graphics, 31, 1997, 209-216.

[10] Higashi, M., Inoue, H., Oya, T.: High Quality Sharp Features in Triangular Meshes, Computer-Aided Design & Applications, 4(1-4), 2007, 227-234.

[11] Hoppe, H., DeRose, T., Duchamp, T., Halstead, M.: Piecewise Smooth Surface Reconstruction, Proc. SIGGRAPH94, 1994, 295-302.

[12] Litke, N., Levin, A., Shröder, P.: Fitting Subdivision Surfaces, Proc. IEEE Visualization 2001, 2001. 319-324.

[13] Loop, C.: Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah 1987.

[14] Maekawa, T., Matsumoto, Y., Namiki, K.: Interpolation by geometric algorithm, Computer-aided Design, 39(4), 2007, 313-323.

[15] Marinov, M., Kobbelt,L.: Optimum Techniques for Approximation with Subdivision Surfaces, Proc. ACM Symposium on Solid Modeling and Application, 2004, 113-122.

[16] H. Pottmann, and M. Hofer, Geometry of the Squared Distance Function to Curves and Surfaces," Visualization and Math. III, Springer, 2003, 223-244.

[17] Suzuki, H. Takeuchi, S. Kimura, F. and Kanai, T.: Subdivision surface fitting to a range of points. In Proc. Pacific Graphics, 1999, 158-167.

[18] Warren, J., Weimer, H.: Subdivision Methods for Geometric Design: A Constructive Approach. Morgan Kaufmann, 2001.

[19] Zorin, D.: Modeling Multiresolution Subdivision Surface, SIGGRAPH06 Course, 2006.