Poster Proceedings of Pacific Graphics 2008
T. Igarashi, N. Max, and F. Sillion
(Editors)

Volume 0 (2008), Number 0

Loop Subdivision Surface Fitting by Geometric Algorithms

Yu Nishiyama[†2] Masayuki Morioka[‡1] and Takashi Maekawa[§1]

[1]Yokohama National University, Japan
[2]Goldman Sachs Japan Co., Ltd, Japan

Abstract

This paper describes a method to approximate point sets by Loop subdivision surfaces based on geometric algorithms. We assume that the data points are given in triangular mesh of arbitrary topological type. The initial control mesh of the Loop subdivision surface is obtained by simplifying the input triangular mesh using QEM algorithm. Our algorithm iteratively updates the control mesh in a global manner based on a simple point-surface distance computation followed by translations of the control vertices along the displacement vectors. The main advantages of our approach compared to existing surface fitting methods are simplicity, speed, and generality. Computational results show that our algorithm runs at least six times faster than current state-of-the-art subdivision fitting methods. We demonstrate our technique with a variety of complex examples.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object ModelingCurve, surface, solid, and object representations

1. Introduction

Fitting surfaces to 3*D* cloud points generated by laser range scanning systems is one of the most important problems in CAD, geometric modeling and computer graphics [HJ99, CWQ*07]. We distinguish two types of fitting: interpolation and approximation [HL93, PT97]. In interpolation we generate an interpolating surface that passes through the data points. In approximation we generate an approximating surface that passes near the data points which minimizes the deviation of the surface from the data points. Consequently the number of control vertices and the number of data points is the same for interpolation, while the number of control vertices is in general much less than the number of data points for approximation. The most popular representation for fitting such data in geometric modeling is the tensor product B-spline surfaces. There is a tremendous amount of literature on fitting B-spline curves/surfaces to a set of data points, however it is not the scope of the present paper to review all the existing work. Good surveys of shape-preserving fitting techniques with B-spline surfaces are given in [HL93, PT97, HJ99, WARV02].

However, it is extremely difficult for tensor product surfaces to handle surfaces with arbitrary topology, and to maintain continuity conditions near the extraordinary points where the number of edges that meet at this point is not equal to four. A new class of surfaces called subdivision surfaces, which offers an alternative to the tensor product B-spline, has become very popular in movie production and game engines [DKT98].

In this paper we use the Loop subdivision surface [Loo87] to approximate the data points given in triangular mesh of arbitrary topological type, although the proposed algorithm can be easily extended to Catmull-Clark subdivision surfaces as well as Doo-Sabin subdivision surfaces. Our algorithm has the following features:

- **Simplicity:** Our algorithm iteratively updates the input mesh in a global manner based on a simple point-surface distance computation followed by translations of control vertices along the displacement vectors.
- **Speed:** Our algorithm runs at least six times faster than current state-of-the-art subdivision fitting methods.
- **Scalability:** Our algorithm quickly gets a coarse fit, while

† e-mail:d06gb150@ynu.ac.jp
‡ e-mail:d07gb167@ynu.ac.jp
§ e-mail:maekawa@ynu.ac.jp

we can progressively obtain a finer fit by performing more iterations.

- **Generality:** Since our algorithm is based on simple geometric procedures, it can be easily extended to curves and surfaces defined by control vertices.

2. Related work

Research on fitting subdivision surfaces to point clouds has been conducted extensively [HKD93, HDD*94, STKK99, LLS01, MMT*04, MK05, ZC06, MMN07, CWQ*07]. A method for generating fair Catmull-Clark surfaces is proposed by [HKD93]. First, a set of given mesh vertices is interpolated and then the surface is faired by minimizing a quadratic norm that combines thin plate and membrane energies. The method solves a linear system using sparse LU decomposition. A least squares fitting algorithm to a scattered data in terms of Loop subdivision surface is studied by [HDD*94, MK05]. The parametrization is computed by projecting each vertex point onto a piecewise linear approximation of the subdivision surface in [HDD*94], while the parametrization is evaluated on the true subdivision surface [MK05] using the exact evaluation procedure [Sta98]. These methods [HDD*94, MK05] also rely on nonlinear minimization methods.

A fast Loop subdivision surface fitting method which interactively defines an initial mesh, and refines it through an iterative local approximation method, is presented by [STKK99]. The method is useful when one needs to quickly generate a surface that captures the overall shape of the scanned geometry, however the method may not be suitable for generating a surface that precisely interpolates the data points, since only a subset of the dense mesh is used in the fitting process.

An algorithm for fitting Catmaull-Clark subdivision surfaces to a given shape through a fast local adaptation procedure based on quasi-interpolation is introduced by [LLS01]. The method employs an adaptive approximation algorithm that approaches the target surface gradually by generating a sequence of surfaces. Since quasi-interpolation does not require the solution of a linear system, it is simpler than the commonly used interpolation and least squares methods, but may result in larger errors in high curvature regions as indicated by [MK05].

A squared distance minimization (SDM), which is introduced by [PL03], is applied by [CWQ*07] for fitting a Loop subdivision surface to unorganized points. An initial surface is iteratively deformed towards the target surface by minimizing an objective function defined by local quadratic approximants of the squared distance function to the target surface. They express the second order Taylor approximant of the squared distance function in the principal frame at the normal footpoint. The method is an *active contour method* [KWT88] in the sense that an initial shape defined by a Loop subdivision surface is iteratively deformed towards the model shape by changing the control vertices of the initial shape. In each iteration a linear system is solved for the unknown displacement vectors. Since SDM is based on local optimization, its approximation results can be sensitive to the initial surface, and requires more research on the initial selection of the control vertices.

The algorithm developed by [ZC06] consists of a two-phase subdivision process, a topological modification of the control mesh and a subsequent modified Catmull-Clark subdivision to construct a smooth surface that interpolates some or all of the vertices of a mesh with arbitrary topology. The computational complexity of their algorithm is $O(n)$ where n is the number of the vertices, however the paper does not address the problem of global optimal fairness, and the resulting shape may exhibit some unwanted undulations.

Our algorithm, which does not require the solution of linear system, employs an iterative method introduced by [LWD03] for interpolating a set of ordered points by B-spline curves/surfaces, and [MMN07] for interpolating a triangular or a quadrilateral mesh of arbitrary topological type.

The method by [LWD03] first establishes a one-to-one relationship between the given data points and the knots, and computes the displacement vectors between the given points and their corresponding points on the B-spline curve/surface at knots, and displace the corresponding control vertices by the displacement vectors iteratively until the magnitude of the difference vectors become zero. They prove the convergence of the algorithm based on the eigenvalue analysis of the matrix which relates the displacement vectors of the k-*th* step and the $k+1$-*th* step.

Iterative geometric algorithm developed by [MMN07] constructs a smooth surface that interpolates a triangular or a quadrilateral mesh of arbitrary topological type. They start their algorithm by assuming that the given triangular/quadrilateral mesh is a control net of a Loop/Catmull-Clark subdivision surface. The control vertices are iteratively updated globally by a closest point computation and an displacing procedure without solving a linear system. If the convergence is slow in some vertices, one can terminate the global iterations and conduct local iterations for the vertices with slow convergence to save the computational time.

In this paper we present a novel approximation method that is based on the iterative geometric algorithm [MMN07].

3. Overview of the algorithm

As an input we are given a triangular mesh M represented by a tuple (K, P) where K is a simplicial complex specifying the topological type of the mesh, and $\mathrm{P} = \{\mathrm{p}_1, \mathrm{p}_2, \ldots, \mathrm{p}_n\} \in \mathrm{R}^3$ is a set of vertex positions. Unlike interpolation, we usually do not know in advance how many control vertices are needed to keep the deviation errors within the prescribed

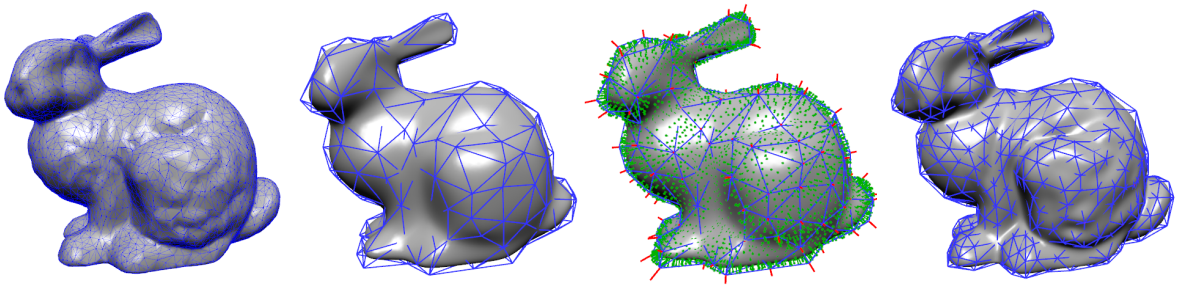


Figure 1: *Illustration of main steps of our geometric algorithm. (a) Input mesh overlaid on the shaded image. (b) QEM-simplified mesh and its corresponding limit Loop subdivision surface. (c) Computation of error vectors (green) and displacement vectors magnified by the scale of 2(red). (d) Final approximated Loop subdivision surface with its control vertices.*

tolerance. Accordingly the approximation algorithm usually starts with an initial mesh, which has a much lower number of triangulations than the input mesh, and increases the number of control vertices iteratively until it satisfies the prescribed tolerance. We first simplify the given input triangular mesh M=$M^{(O)}$ using QEM [GH97], which maintains high fidelity to the original model. We denote the QEM-simplified mesh as M_Q=$M^{(1)}$. The number of vertices of M_Q depends on the accuracy required and model complexity. We use about 1∼5% of the number of vertices of M for M_Q which generally leads to very good results, as suggested by [MK05].

For each vertex $\mathrm{p}_i \in M$, we compute the closest point on the approximating surface S using Stam's exact evaluation procedure [Sta98]. Following [MK05] the parameter value of the closest point of $\mathrm{p}_i \in M$ on S is denoted by $t_i =< f_i, (v_i, w_i) >$ where $f_i \in M_Q$ indicates the patch to which p_i is mapped, and $(1-v_i-w_i, v_i, w_i)$ are the barycentric coordinates.

We iteratively improve the approximation by relocating the positions of the control vertices, inserting control vertices in regions where the deviations are large. Whenever the control vertices are inserted, we apply connectivity regularization followed by edge-flips to remove elongated triangles. We indicate k-th iteration by attaching a superscript (k).

We define two errors:

1. Average error E_{ave}:

$$E_{ave} = \frac{\sum_{i=1}^{n} \| \mathbf{p}_i - \mathbf{S}(t_i^{(k)}) \|_2}{n} \tag{1}$$

2. Maximum error E_{max}:

$$E_{max} = \max_{1 \le i \le n} \| \mathbf{p}_i - \mathbf{S}(t_i^{(k)}) \|_2 \tag{2}$$

With $M^{(1)}$ as an initial control of the approximating Loop subdivision surface, a sequence of solutions $M^{(2)}, M^{(3)}, \ldots$ is generated and $M^{(k)}$ gradually converges to a solution whose maximum error and average error divided by the bounding box diagonal of the input mesh are within the tolerances ε_{max} and ε_{ave}, respectively.

4. Displacements of control vertices

We assume that the QEM-simplified mesh M_Q is the initial control mesh of the Loop subdivision surface. We begin the geometric algorithm by computing the closest point on $\mathrm{S}(t_i^{(k)})$ for each data point $\mathrm{p}_i^{(k)}$ by applying Newton's method. At each Newton iteration step we linearize the Loop subdivision surface at the solution $(v,w)_{i,j}^{(k)}$ [MK05], where subscript j represents the *j-th* Newton iteration, and we orthogonally project p_i onto the tangent plane, yielding a 2×2 linear system for the correction $(\Delta v, \Delta w)_{i,j}^{(k)}$:

$$(\mathrm{p} - (\mathrm{S}(t) + \mathrm{S}_v(t)\Delta v + \mathrm{S}_w(t)\Delta w) \cdot \mathrm{S}_v(t) \mid_{i,j}^{(k)} = 0$$
$$(\mathrm{p} - (\mathrm{S}(t) + \mathrm{S}_v(t)\Delta v + \mathrm{S}_w(t)\Delta w) \cdot \mathrm{S}_w(t) \mid_{i,j}^{(k)} = 0 \,.$$

The initial values for Newton's iteration can be generated by orthogonally projecting $\mathrm{p}_i^{(k)}$ onto the triangle whose vertices are obtained by computing the limit points of the vertices of the current mesh $M^{(k)}$. Let us denote a 1-ring neighbors of $\mathrm{p}_i^{(k)}$ by $\mathrm{v}_1, \ldots, \mathrm{v}_N$ where we drop subscript i,and superscript (k) for the sake of simplicity. Then the limit position of $\mathrm{p}_i^{(k)}$ is given by

$$(\mathrm{p}_i^{(k)})^{\infty} = \beta_N \mathrm{p}_i^{(k)} + (1-\beta_N)\frac{\mathrm{v}_1 + \ldots \mathrm{v}_N}{N} \,,$$

where

$$\alpha_N = \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{N}\right)^2 + \frac{3}{8}, \qquad \beta_N = \frac{3}{11 - 8\alpha_N}$$

As the iteration proceeds, given data points become closer to the approximating Loop subdivision surface, and the parameter values of the closest points will not change from iteration to iteration. Therefore, in such cases we may terminate the computation of finding the closest points to save the computational time. However, in the final step we compute

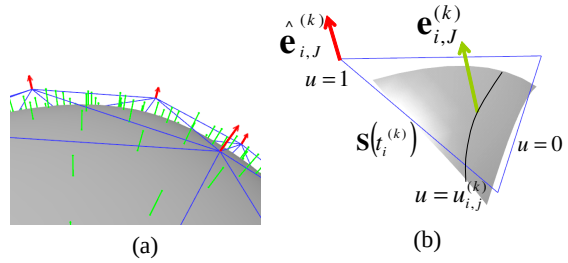


Figure 2: *Approximation by geometric algorithm. (a) QEM-simplified control mesh (blue) and its corresponding limit surface evaluated by Stam's algorithm. The green lines represent the lines connecting the given data points P and their corresponding closest points on the limit surface. The red lines with arrows show the displacement vectors. (b) The relation between the weighted error vector $\hat{e}_{i,J}^{(k)}$ and the error vector $e_{i,J}^{(k)}$.*

the closest points for all the data points to guarantee that the errors are within the prescribed tolerance.

After the first iteration, we can start Newton's method from the patch and the parameter value assigned in the previous iteration. Implementation issues for the Newton's method are fully discussed in [MK05, MMN07].

Once the closest point is computed, the error vector can be expressed as $\mathrm{e}_i^{(k)} = \mathrm{p}_i^{(k)} - \mathrm{S}(t_i^{(k)})$. In interpolation, we simply displace the control vertices by the error vectors

$$\mathrm{p}_i^{(k+1)} = \mathrm{p}_i^{(k)} + \mathrm{e}_i^{(k)} ,$$

and hence we have one-to-one correspondence between the data points and the control vertices of the interpolating Loop subdivision surface [MMN07], whereas in approximation we do not have such one-to-one correspondence. Therefore it is not trivial how to distribute the error vector $\mathrm{e}_i^{(k)}$ to the most appropriate control points of the approximating Loop subdivision surface.

Let us assume that there are m closest points of P on $\mathrm{S}(t)$ which lie on the patches of the one-ring neighbor of $\mathrm{p}_i^{(k)}$. For each closest point of P we compute the error vector $\mathrm{e}_{i,J}^{(k)}$ (see Figure 2(a)), where the subscript J denotes the index for the closest points, and weigh it according to the barycentric coordinates $u_{i,J}^{(k)} = 1.0 - v_{i,J}^{(k)} - w_{i,J}^{(k)}$ as follows:

$$\hat{\mathbf{e}}_{i,J}^{(k)} = u_j^{(k)} \mathbf{e}_{i,J}^{(k)}, \quad J = 1, \ldots, m . \tag{3}$$

Note that $u_{i,J}^{(k)}$=1 at $(\mathrm{p}_i^{(k)})^{\infty}$, and $u_{i,J}^{(k)}$=0 along the edge opposite to $\mathrm{p}_i^{(k)}$ as illustrated in Figure 2 (b). The amount of displacement of the control vertices $\mathrm{d}_i^{(k)}$ is obtained by adding all the weighted error vectors $\hat{\mathbf{e}}_{i,j}^{(k)}$ and dividing by the sum of the weights $u_{i,J}^{(k)}$, yielding:

$$\mathrm{d}_i^{(k)} = \frac{\sum_J \hat{\mathbf{e}}_{i,J}^{(k)}}{\sum_J u_{i,J}^{(k)}} . \tag{4}$$

Finally the control vertices for the $k+1$-th iteration will become

$$\mathrm{p}_i^{(k+1)} = \mathrm{p}_i^{(k)} + \mathrm{d}_i^{(k)} . \tag{5}$$

We repeat the above steps (3) $\sim$ (5) until Eq.(1) and Eq.(2) divided by the bounding box diagonal of the input mesh become smaller than the prescribed tolerances ε_{max} and ε_{ave} for all the input data points.

The proof of convergence for the non-uniform cubic B-spline curve interpolation and non-uniform bi-cubic B-spline surface interpolation is studied by [LWD03]. In their interpolation scheme the displacement vector is computed by taking the difference between the given data points and its corresponding points at knots on the B-spline curve/surface. It is also shown experimentally by [MMN07] that when displacement vector is evaluated between the given data point and the corresponding closest point on an interpolating Loop/Catmull-Clark surface, eventually the surface interpolates given data points. While a complete analysis of the convergence of the approximation scheme is beyond the scope of this paper, the intuition behind the approximation algorithm is not difficult to understand.

5. Topological adjustments

When the error norm does not become small enough in some regions, we insert new control vertices by applying 1-to-4 split with crack-fixing 1-to-2 split [MK05] as shown in Figure 3 (a). We call this one face split a *basic case split*. In a basic case, the central shaded triangle is split into four triangles at the mid points of the edges (red points). To maintain the mesh compatibility, 1-to-2 split is conducted on each neighboring face. The central white triangle of Figure 3 (b) shares two of its edges with 1-to-4 split triangles (shaded). We call this two face split a " special case split". In special cases 1-to-4 split is conducted instead of 1-to-2 split to maintain the valid mesh connectivity. Since the vertices affected by the adaptive insertion (black and red points) have a one-to-one correspondence with the vertices that have been subdivided once, we assign the corresponding subdivided vertices to the black and red vertices to minimize the modification of $\mathrm{S}(t)$ [MK05].

Since these edge split operations induce extraordinary vertices, we need to improve the mesh quality by regularizing its connectivity. This can be done by minimizing the following function [SG03, MK05]:

$$R(M) = \sum_{\mathrm{v} \in M} (d\mathrm{v}) - d_{opt}(\mathrm{v}))^2 ,$$

where $d(\mathrm{v})$ is the valence of vertex v and $d_{opt}(\mathrm{v})$ is its optimal degree. The optimal value for boundary vertices is

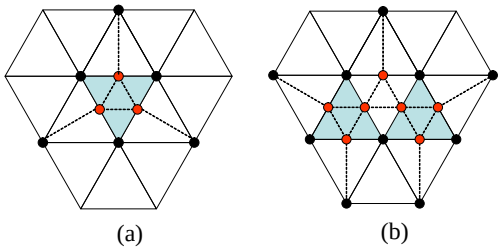


Figure 3: *Face-split operation of control vertices for (a) a basic case split, (b) a special case split.*

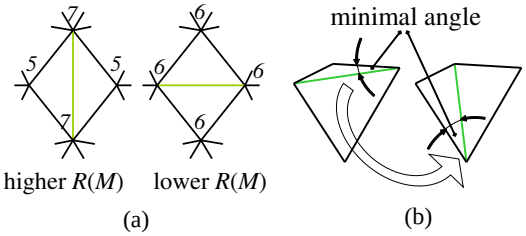


Figure 4: *Topological adjustments: (a) Edge-flip operations are conducted if the regularity measure $R(M)$ decreases. (b) Edge-flip operations will also be conducted to maximize any minimal angles.*

$d_{opt}(\mathrm{v}) = 4$, and for the rest of the vertices it is $d_{opt}(\mathrm{v}) = 6$. As illustrated in Figure 4 (a), we evaluate the value of $R(M)$ for before and after the edge flip and choose the one that has lower $R(M)$. Another problem that often arises is the appearance of elongated edges. The elongated edges often cause unwanted undulations, thus we need to remove them. We apply edge-flips if the minimal angle between all the angles of the triangles adjacent to the edge is increased as shown in Figure 3 (b).

The overall algorithm for topological adjustments can be summarized as follows:

1. Insertion of control vertices once in five iterations of the main approximation loop to the regions where error norm does not become small enough.
2. Connectivity regularization by edge flipping.
3. Angle improving edge-flips.
4. Necessity to re-compute the initial values of Newton's iteration for the data points whose closest points lie on the patch that are affected by the topological adjustments.

6. Experimental results

We have implemented the above algorithm on a Pentium IV 3.0GHz PC with 2GB RAM. The effectiveness of our approximation algorithm can also be seen from the companion video clips captured using our approximation algorithm.

Figure 5 illustrates a plot of the average as well as maximum deviations of the approximation with respect to the

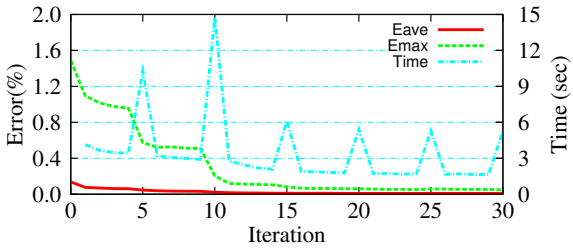


Figure 5: *The plot shows the convergence of our algorithm applied to armadillo model.*

number of iterations of the main approximation loop for the armadillo model. The figure also shows the time spent for each iteration. Both error measures are scaled by a bounding box diagonal of the model. The average error E_{ave}=0.01% and the maximum error E_{max}=0.05% are reached after 16 iterations, and 30 iterations, respectively. The spikes in the graph which appear in every five iterations are due to the time for the topological adjustments of the control vertices.

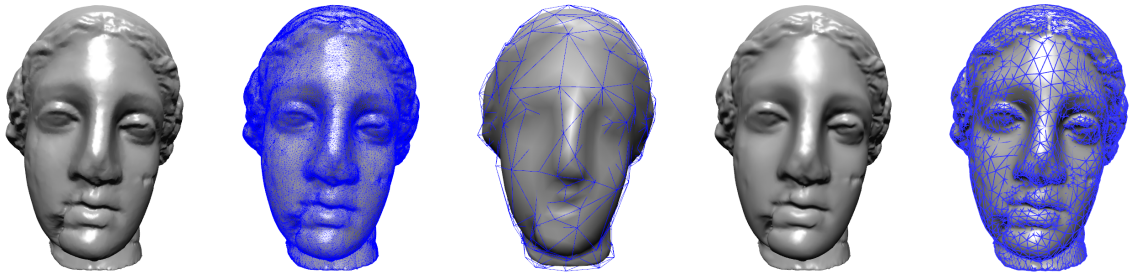
The models that we use for experiments are a Stanford bunny, Igea, a buddha, and an armadillo which are shown in Figures 1, 6, 7, 8. Here we note that the number of vertices used in Figure 1 is much less than the one used in the numerical experiments. We also note that the Igea, Buddha, and armadillo models in Figures 6, 7, 8 are all computed with $\varepsilon_{avg} = 0.01\%$ and $\varepsilon_{max} = 0.05\%$, which are different from the ones in Table 1.

Figure 6 shows the shaded image of the input mesh, input mesh overlaid on the shade image, QEM-simplified mesh and its corresponding limit surface, final mesh overlaid on the final surface , and the final surface of the Igea model. Figure 7 shows the shaded image of the input mesh, the final surface, and their close-up views of the buddha model. Figure 8 shows the front and rear views of the armadillo model of the shaded image of the input mesh as well as the final mesh.

We compare our results with those of the approximation method developed by [MK05], as well as [CWQ*07] and tabulate them in Table 1. We include times for reading input files, QEM mesh simplification and initializing dynamic parameters as a pre-computation. First we compare with [MK05]. The PC on which they ran the experiments is a 2.8 GHz Pentium IV with 2 GB RAM. We start the computation with the same number of initial control vertices as in [MK05], and set the same termination tolerances ε_{rms} and ε_{max}. Here we note that [MK05] use the root mean square errors ε_{rms} instead of ε_{ave}. Although there are some differences in the resulting number of vertices of the final mesh, our method is at least six times faster than the methods of [MK05].

Model	Original n	Initial n	Final n	E_{max} (%)	E_{rms} (%)	Precomputation	Total time
Our Results							
Bunny	37K	612	7098	0.049	0.0108	4sec	22sec
Igea	134K	336	1572	0.238	0.0701	14sec	1min:8sec
Armadillo	173K	433	3267	0.248	0.0618	19sec	1min:49sec
Buddha	543K	4091	26838	0.049	0.0101	1min	7min:6sec
Results in [MK05]							
Armadillo	173K	433	2820	0.248	0.0598	N/A	12min:26sec
Buddha	543K	4093	17995	0.049	N/A	N/A	64min
Bunny	37K	612	8440	0.048	N/A	N/A	5min
Igea	134K	336	1553	0.247	0.0575	N/A	8min:29
Results in [CWQ*07]							
Buddha	543K	4662	18715	0.43*	0.03*	83min:58sec	114min:6sec
Bunny	35K	919	996	0.82*	0.09*	6min:1sec	7min:18sec
Igea	134K	526	2385	0.36*	0.05*	4min:11sec	6min:43sec

Table 1: *Comparisons with other methods. (* The errors are based on uniformly scaled data when the longest side of the bounding box has the length equal to 1.0.)*

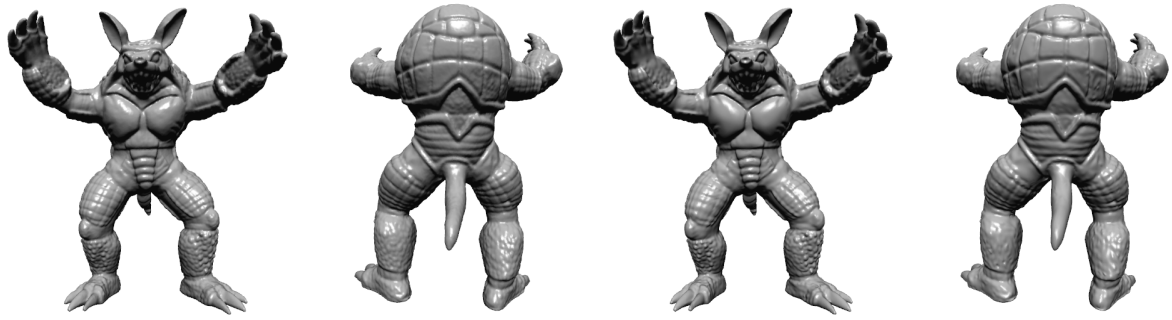


(a) (b) (c) (d) (e)

Figure 6: *Igea: (a) Shaded image of the input mesh (134K vertices). (b)Input mesh overlaid on the shade image. (c) QEM-simplified mesh (1343 vertices) and its corresponding limit surface. (d) Final mesh overlaid on the final surface (12445 vertices). (e) Final surface.*

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

Figure 7: *Buddha:(a) Shaded image of the input mesh (543K vertices). (b) Final Loop subdivision surface (26838 vertices). (c) and (d) are the close-up views of the shaded image of the input mesh. (e) and (f) are the close-up views of approximated surface.*



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

Figure 8: *Armadillo: (a) and (b) are the shaded images of the input mesh (173K vertices). (c) and (d) are the final Loop subdivision surfaces (26110 vertices).*

Model	Iteration	Pre-computation		Iterative geometric algorithm		
		QEM	Initialize	Newton's Method	Remesh	Other
Bunny	21	7.72%	8.14%	62.29%	8.08%	13.77%
Igea	32	3.86%	9.37%	68.18%	7.22%	11.37%
Armadillo	27	4.00%	6.79%	75.19%	6.71%	7.31%
Buddha	32	3.68%	5.26%	48.53%	36.14%	6.39%

Table 2: *Calculation time for each function.*

The method of [CWQ*07] consist of two main phases, pre-computation and minimization. Chen et al. ran the experiments on a PC with an Intel Xeon 2.8 GHz and a 2 GB RAM. The models are scaled into a rectangular box with the longest side equal to 1.0. For setting up the squared distance error functions, they pre-compute the distance field, as well as the tangential and curvature information of the target surface. As indicated in Table 1, the time for pre-computation is much longer than the minimization time. Although there are some differences in the termination tolerances, if we compare the total time, our method is also at least six times faster than [CWQ*07] and, even if we compare only the minimization part, our method is still much faster.

Finally we list in Table.2 a detailed breakdown of the ratios of the computational time took for constructing the models in Figure 1, 6, 7, 8, which are all computed with $\varepsilon_{ave} = 0.01\%$ and $\varepsilon_{max} = 0.05\%$. The second column lists the number of iterations, the third column indicates the ratio of computational time for reading the input data and the QEM simplification. The fourth column is the time ratio for initializing the dynamic parameters, and the fifth column is the percentage of time took in the Newton iterations. The sixth column represents the time ratio for remeshing, and the seventh column shows time ratio for the rest of the computation including updating the control points. We observe that the time for computing the closest points using Newton's method takes two thirds of the entire computational time.

7. Conclusion and Future Work

We have developed an efficient algorithm for constructing Loop subdivision surfaces that approximates the vertices of a triangular mesh. The method is based on geometric algorithms, which iteratively update the control vertices without solving a linear system. Comparing with conventional fitting methods, which rely on solving linear systems, our algorithm is faster, easier to implement, and more general. Although our algorithm does not employ explicit fairing tools in the formulation, the resulting surfaces are visually pleasing.

As a topic of future research, we plan to further investigate the remeshing scheme, and explore other applications of the geometric algorithms in geometric modeling and computer graphics.

Acknowledgements

A portion of this work was supported by the Japan Society for the Promotion of Science, Grants-in-Aid for Scientific Research under the grant number 16560116. We wish to thank Daisuke Hirano for his help. The Igea and the Armadillo models are courtesy of Cyberware. The bunny and the laughing buddha models are courtesy of the Stanford University Computer Graphics Laboratory.

References

[CWQ*07] CHENG K.-S. D., WANG W., QIN H., WONG K.-Y., YANG H., LIU Y.: Design and analysis of optimization methods for subdivision surface fitting. *IEEE Transactions on Visualization and Computer Graphics 13*, 5 (2007), 878–890.

[DKT98] DEROSE T., KASS M., TRUONG T.: Subdivision surfaces in character animation. In *Proceedings of ACM SIGGRAPH 98* (1998), pp. 85–94.

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings of ACM SIGGRAPH 97* (1997), pp. 209–216.

[HDD*94] HOPPE H., DEROSE T., DUCHAMP T., HALSTEAD M., JIN H., MCDONALD J., J.SCHWEITZER, STUETZLE W.: Piecewise smooth surface reconstruction. In *Proceedings of ACM SIGGRAPH 94* (1994), pp. 295–302.

[HJ99] HOSCHEK J., JÜTTLER. B.: Techniques for fair and shape preserving surface fitting with tensor-product b-splines. In *Shape Preserving Representations in Computer-Aided Geometric Design*. Nova Science Pub, 1999, pp. 163–85.

[HKD93] HALSTEAD M., KASS M., DEROSE T.: Efficient, fair interpolation using catmull-clark surfaces. In *Proceedings of ACM SIGGRAPH 93* (1993), pp. 35–44.

[HL93] HOSCHEK J., LASSER D.: *Fundamentals of Computer Aided Geometric Design*. A. K. Peters, Wellesley, MA, 1993.

[KWT88] KASS M., WITKIN A., TERZOPOULOS. D.: Snakes: Active contour models. *International Journal of Computer Vision 1*, 4 (1988), 321–332.

[LLS01] LITKE N., LEVIN A., SCHRÖDER P.: Fitting subdivision surfaces. In *Proceedings of the conference on Visualization '01* (2001), pp. 319–324.

[Loo87] LOOP C. T.: *Smooth subdivision surface based on triangle*. Master's thesis, University of Utah, 1987.

[LWD03] LIN H., WANG G., DONG C.: Constructing iterative non-uniform B-spline curve and surface to fit data points. *Science in China 33* (2003), 912–923.

[MK05] MARINOV M., KOBBELT L.: Optimization methods for scattered data approximation with subdivision surfaces. *Graphical Models 67*, 5 (2005), 452–473.

[MMN07] MAEKAWA T., MATSUMOTO Y., NAMIKI K.: Interpolation by geometric algorithm. *Computer-Aided Design 39*, 4 (2007), 313–323.

[MMT*04] MA W., MA X., TSO S., , PAN Z.: A direct approach for subdivision surface fitting from a dense triangle mesh. *Computer-Aided Design 36*, 6 (2004), 525–536.

[PL03] POTTMANN H., LEOPOLDSEDER S.: A concept for parametric surface fitting which avoids the parametrization problem. *Computer Aided Geometric Design 20*, 6 (2003), 343–362.

[PT97] PIEGL L., TILLER W.: *The NURBS book (2nd ed.)*. Springer-Verlag New York, Inc., 1997.

[SG03] SURAZHSKY V., GOTSMAN C.: Explicit surface remeshing. In *Proceedings of the 2003 Eurographics* (2003), pp. 20–30.

[Sta98] STAM J.: Evaluation of Loop subdivision surfaces. In *SIGGRAPH 98 CDROM Proceedings, 1998.* (1998).

[STKK99] SUZUKI H., TAKEUCHI S., KIMURA F., KANAI T.: Subdivision surface fitting to a range of points. In *Proceedings of the 7th Pacific Conference on Computer Graphics and Applications* (1999), p. 158.

[WARV02] WEISS V., ANDOR L., RENNER G., VÁRADY T.: Advanced surface fitting techniques. *Computer Aided Geometric Design 19*, 1 (2002), 19–42.

[ZC06] ZHENG J., CAI Y.: Interpolation over arbitrary topology meshes using a two-phase subdivision scheme. *IEEE Transactions on Visualization and Computer Graphics 12*, 3 (2006), 301–310.