www.cadanda.com
Computer-Aided Design AND APPLICATIONS

Iterative Interpolation based 3D Shape Generation Using General Catmull-Clark Subdivision Surfaces

Shuhua Lai[1], Jason Lai[2], Anastasia Kazadi[3], Seifalla A. Moustafa[3], Fuhua (Frank) Cheng[3]

[1]Georgia Gwinnett College, slai@ggc.edu
[2]Gwinnett School of Mathematics, Science, and Technology, jlai@gsmst.edu
[3]University of Kentucky, ansm226@g.uky.edu, seifalla.moustafa@uky.edu, cheng@cs.uky.edu

Corresponding author: Shuhua Lai, slai@ggc.edu

Abstract. An algorithm for fast construction of a smooth subdivision surface that interpolates the vertices of an arbitrary input mesh M is presented. The central idea of the proposed algorithm is to find the inverse A^{-1} of the matrix A that calculates the limit points of M with respect to a chosen subdivision scheme. However, instead of a costly matrix computation process, a technique to calculate A^{-1} indirectly by representing it as an infinite series of matrices is developed. With this infinite series of matrices, one can construct an infinite iterative series of meshes, which converges to a mesh whose subdivision surface interpolates the given input mesh M. Most importantly, control points of these infinite iterative series of meshes can be calculated locally based on the chosen subdivision scheme. Hence, the matrices A and A^{-1} do not have to be really constructed. They are simply used in a theoretical derivation to obtain the iteration formula. The construction of the interpolation surface is done basically by iteratively adjusting vertices of the given mesh locally until some given error tolerance is reached. The concept of iterative interpolation has been presented in the literature before. The main differences between our algorithm and existing approaches are five-fold: 1). Our algorithm is the first one that derives the iterative equation from the perspective of computing A^{-1}. 2). Rigorous mathematical proof is provided that guarantees the existence, convergence and uniqueness of A^{-1}. 3). Our iterative interpolation algorithm, as proven in the paper, converges at an exponential rate, is a local process and does not involve costly matrix computation. Hence the new method is very fast and can handle meshes with large number of vertices. 4). Our algorithm does not require fairing in the construction process because solution to the above interpolation process is unique. 5). Although only the general Catmull-Clark subdivision surface is used here for deriving the iterative algorithm, the idea of the proposed algorithm works for other popular subdivision schemes as well.

Keywords: subdivision surface, 3D shape modeling, interpolation
DOI: https://doi.org/10.14733/cadaps.2023.479-488

1 Introduction

The study of smooth surface construction from data points produced by sampling or a laser range scanning system has become increasingly important in computer graphics, reverse engineering, geometric modeling, interactive design and animation [1, 5, 4]. Interpolation is the most widely used technique in this smooth surface construction process. There are many subdivision surface based interpolation algorithms available in the literature. These algorithms can be roughly divided into three types. The first, called *interpolating subdivision* [6, 8], generates an interpolation surface by applying well designed subdivision rules. The second one, called *global optimization* [7, 10], does the work by building a global linear system with some fairness constraints to avoid undesired undulations. And the last one, called *progressive interpolation* [2, 3, 18], uses some local adjustment of mesh vertices to achieve an new mesh so that the subdivision surface of the new mesh interpolates the original mesh. In this paper, we propose an algorithm that is a type of *progressive interpolation* methods. Our new method has the advantages of *global property*, which means it generates smooth interpolating subdivision surfaces that resemble the shape of the given meshes well. Meanwhile our new method has the advantages of *local property*, which means it iteratively generates new meshes by moving vertices in the input mesh through performing affine transformation on nearby vertices locally. Because it is done locally, our new algorithm is very fast and very easy to implement.

Subdivision surfaces [1, 4, 5, 12] are efficient smooth surface representation for meshes of arbitrary topology. Recently with parametrization of subdivision surfaces becoming available [9, 17], they have become popular in computer graphics, geometric modeling and animation because of their capability in modeling complex shape of arbitrary topology, their relatively high visual quality and their stability and efficiency in numerical computation. Subdivision surfaces cover both *parametric forms* [15, 17] and *discrete forms*. Parametric forms are good for design and representation and discrete forms are good for machining and tessellation (including FE mesh generation). Therefore we have a representation scheme that is good for almost all applications. Powerful interpolation techniques using subdivision surfaces as a representation scheme certainly are needed.

Subdivision surfaces are used as the surface representation in our new interpolation technique. Without loss of generality, we shall assume the subdivision scheme considered in this paper is general Catmull-Clark subdivision scheme [9]. But the idea behind our approach works for other subdivision schemes as well, for instance, Loop subdivision[12] or Doo-Sabin [5] subdivision scheme.

2 Previous Work

Interpolation is an extensively studied classic problem in computer graphics and geometric modeling. When it comes to interpolating arbitrary topology meshes, subdivision surfaces are the most frequently used representation scheme. There are three major ways to interpolate a given mesh with a subdivision surface: *interpolating subdivision* [6, 11, 8, 16, 19], *global optimization* [7, 13] and *progressive interpolation* [3, 2, 18].

In the first approach, *interpolating subdivision*, a subdivision scheme that interpolates the control vertices, such as the Butterfly scheme[6], Zorin et al's improved version [19] or Kobbelt's scheme [8], is used to generate the interpolating surface. New vertices are defined as local affine combinations of nearby vertices. This approach is simple and easy to implement. It can handle meshes with large number of vertices. The resulting surfaces are C^1 continuous. However, since no vertex is ever moved once it is computed, any distortion in the early stage of the subdivision will persist. This makes interpolating subdivision very sensitive to the irregularity in the given mesh. When the mesh vertices are dense enough, the undesired artifacts would not be so clear to see. But when the mesh vertices are not so dense, the effect of undesired artifacts becomes obvious on the resulting interpolating surfaces.

The second approach, *global optimization*, usually needs to build a global linear system with some constraints [14]. The solution to the global linear system is an interpolating mesh whose limit surface interpolates the vertices of the given mesh. This approach usually requires some fairness constraints in the interpolation process, such as the energy functions presented in [7], to avoid undesired undulations. Although this approach

seems more complicated, it results in a traditional subdivision surface. For example, the method in [7] results in a Catmull-Clark subdivision surface (CCSS), which is C^2 continuous almost everywhere and whose properties are well studied and understood. The shape of the resulting interpolating surface resembles that of the given mesh more faithfully because of the global property. The problem with this approach is that it is difficult to handle meshes with large number of vertices.

The third approach, *progressive interpolation*, uses some local adjustment of mesh vertices to achieve an new mesh so that the subdivision surface of the new mesh interpolates the original mesh. It also results in a traditional subdivision surface. Most progressive interpolation algorithms generate smooth interpolating subdivision surfaces that resemble the shape of the given meshes well hence a fairing process is not needed because they do not need to solve a global linear system, instead they generate new meshes progressively by performing local linear transformation on nearby vertices.

3 Basic Idea

Given a mesh M and a subdivision scheme, our task is to find a smooth subdivision surface to interpolate M. We use the following notations in the paper: A refers to the matrix [9] that calculates all the limit points of M with respect to the given subdivision scheme; $\mathbf{S}(M)$ refers to the limit surface of M; $\mathbf{I}(M)$ refers to the subdivision surface that interpolates M and satisfies the property that limit points of its control mesh equal to M, i.e., if P is the control mesh of $\mathbf{I}(M)$ then we must have $M = \mathsf{A} * P$. We also assume the subdivision scheme considered here is the Catmull-Clark scheme. Hence, $\mathbf{I}(M)$ and $\mathbf{S}(M)$ are Catmull-Clark subdivision surfaces. However, the techniques presented here work for other subdivision schemes as well.

To find $\mathbf{I}(M)$, we just need to find P such that $\mathbf{I}(M) = \mathbf{S}(P)$, which means $M = \mathsf{A} * P$, or $P = \mathsf{A}^{-1} * M$. For any given mesh (A is only dependent on the topology of the given mesh M and the chosen subdivision scheme [9]), if we can find A^{-1}, then the interpolation problem is solved. So we just need to prove that for any given mesh, A^{-1} exists and, for any given mesh, we are able to compute A^{-1} directly or indirectly. We will prove the existence of A^{-1} in Section 6. However, we should not directly compute the A^{-1} because the matrix A could be huge (the size of A is $N \times N$, where N is the number of vertices of the input mesh M). Hence for meshes with thousands or even millions of vertices, it is not cost effective to directly find A^{-1} and also the result could be numerically unstable. Our basic idea in this paper is to compute A^{-1} indirectly using an iterative approach.

4 Fast Iterative Interpolation

In this section, we present our approach for obtaining A^{-1} by iterations. Let

$$\mathsf{L} = \sum_{i=0}^{\infty} (\mathsf{E} - \mathsf{A})^i = (\mathsf{E} - \mathsf{A})^0 + (\mathsf{E} - \mathsf{A})^1 + (\mathsf{E} - \mathsf{A})^2 + (\mathsf{E} - \mathsf{A})^3 + \cdots \tag{1}$$

where E is an identity matrix that has the same dimension of A. The convergence of the above infinite series will be proven in Section 6. By multiplying $(\mathsf{E} - \mathsf{A})$ to both sides of Eq. (1), we have

$$(\mathsf{E} - \mathsf{A})\mathsf{L} = (\mathsf{E} - \mathsf{A})^1 + (\mathsf{E} - \mathsf{A})^2 + (\mathsf{E} - \mathsf{A})^3 + (\mathsf{E} - \mathsf{A})^4 + \cdots \tag{2}$$

By subtracting the left (right) side of Eq. (2) from the left (right) side of Eq. (1), respectively, we get

$$(\mathsf{E} - (\mathsf{E} - \mathsf{A}))\mathsf{L} = (\mathsf{E} - \mathsf{A})^0$$

Simplifying the above equation, and noting that $(\mathsf{E} - \mathsf{A})^0 = \mathsf{E}$, we get $\mathsf{AL} = \mathsf{E}$, which leads to

$$\mathsf{A}^{-1} = \mathsf{L} = \sum_{i=0}^{\infty} (\mathsf{E} - \mathsf{A})^i \tag{3}$$

Eq. (3) provides an indirect way to compute the inverse of matrix A. But it still has too much costly matrix computation and is not efficient for obtaining A^{-1}. Furthermore, based on the definition of P and M in the previous section, we have

$$P = \mathsf{A}^{-1}M = \sum_{i=0}^{\infty}(\mathsf{E}-\mathsf{A})^i M.$$

Let $M_0 = M$ be the given mesh and for any $i \geq 0$, define

$$M_i = (\mathsf{E}-\mathsf{A})^i M.$$

It can be proven (see Section 6) that

$$\lim_{i\to\infty} M_i = \mathbf{0}.$$

As a result, $\mathbf{I}(M_i)$ approaches to $\mathbf{0}$ when i tends to ∞. This is due to the convex hull property of subdivision surfaces. In addition, because subdivision is a linear process, we have $\sum \mathbf{S}(M_i) = \mathbf{S}(\sum M_i)$. Therefore, P is the control mesh of the interpolating surface $\mathbf{I}(M)$, i.e., $\mathbf{I}(M) = \mathbf{S}(P)$. From the above definition of M_i, we have

$$M_{i+1} = (\mathsf{E}-\mathsf{A})^{i+1}M = (\mathsf{E}-\mathsf{A})(\mathsf{E}-\mathsf{A})^i M = (\mathsf{E}-\mathsf{A})M_i = M_i - \mathsf{A}M_i.$$

Hence, for any $n > 0$, we can have the following.

$$\begin{cases} M_{n+1} = M_n - \mathsf{A} * M_n \\ M_n \quad = M_{n-1} - \mathsf{A} * M_{n-1} \\ M_{n-1} = M_{n-2} - \mathsf{A} * M_{n-2} \\ M_{n-2} = M_{n-3} - \mathsf{A} * M_{n-3} \\ \quad \cdots\cdots \\ M_1 \quad = M_0 - \mathsf{A} * M_0 \end{cases} \tag{4}$$

Adding up all the values on the left side and adding up all the values on the right side of all the equations in Eq. (4), respectively, also letting

$$P_n = \sum_{i=0}^{n} M_i$$

and noting $M_0 = M$, we get $P_{n+1} - M = P_n - \mathsf{A} * P_n$. Simplifying it we have

$$P_{n+1} = P_n + M - \mathsf{A} * P_n. \tag{5}$$

Eq. (5) is the most important result of this paper because it provides an iterative way to calculate P, which basically is P_∞.

Just like we mentioned before, Eq. (3) should not be used to construct the interpolating surface directly, because it requires costly matrix multiplications. Actually, in the construction of the interpolating surface, there is no need to compute the matrix A at all, let alone computing the matrix A^{-1}. We can use Eq. (5) to find the control mesh $P = P_\infty$ of the interpolating surface iteratively and this is because $\mathsf{A} * P_n$ can be calculated locally [9] according to the chosen subdivision scheme without the need to construct the matrix A. As a result, matrices A and A^{-1} are not needed in the iterative process at all, even though they are used in the derivation process of Eq. (5).

On the other hand, because A is invertible, it is easy to see that $\mathsf{A}^{-1}M = P$. Hence P is indeed the ONLY mesh (having the same topology as M) whose limit surface interpolates the given M using the given subdivision

scheme whose subdivision matrix is A. Consequently a fairing process is not needed in the construction of the interpolation surface because for the given subdivision scheme (say Catmull-Clark subdivision) there exists only one such surface that interpolates the given mesh. Traditional interpolation techniques need a fairing process because extra vertices are added in the interpolation process to smooth out the resulting surface. These extra vertices, with possibly improperly assigned positions, lead to undulations in the interpolating surface because they need to be interpolated as well. Our approach avoids the fairing process, which leads to less undesired undulation artifacts.

Also traditionally, people tried to directly find $\mathrm{A}^{-1}M$ by solving a linear system. It is difficult to deal with meshes that have large number of vertices that way. With our new algorithm, P can be constructed not by solving a linear system, but by iteratively applying eq. (5) until some given error tolerance is reached (see Section 5). Hence there is no problem to deal with meshes with large number of vertices.

It can be proven that Eq. (5) converges at an exponential rate (see Section 6). Hence good interpolation results can be obtained in just a few iterations. Nevertheless, error can be explicitly calculated as $||M - \mathrm{A} * P_n||$ and the iteration stop criteria can be determined based on some given error tolerance. Because it converges very fast, the new interpolation technique is suitable for interactive shape design. Figure (2) shows some test results. We can see from these examples that smooth and visually pleasant interpolation shapes can be obtained for dense and complicated meshes. All the test cases are done in less than one second.

5 Pseudo Code of the Iterative Interpolation Algorithm

In this section, we provide the pseudo code using the above iterative approach for finding the interpolation surface of an input mesh of arbitrary topology.

Iterative Interpolation Algorithm:

Input: a mesh M, a error tolerance ϵ, and a choice of subdivision scheme.
Output: a mesh P, whose subdivision surface interpolates M. Let $P = M$;

Do {
    Locally calculate $Q = \mathrm{A} * P$ using the given subdivision scheme.
    /*Note that in the above step, A does not need to be constructed.*/
    $d = ||M - Q||$;
    $P = P + M - Q$;
} while($d > \epsilon$);

Perform subdivision on P to obtain the interpolation surface of M.

6 Mathematical Proof

In this section we provide rigorous mathematical proof for the proposed iterative interpolation method. First of all, let's construct the matrix A.

6.1 Construction of Matrix A

Even though A is not needed in the implementation of the iteration process, for our proof, it is better to show what A is. Here we construct A using generalized Catmull-Clark subdivision scheme. Other schemes can be done similarity. For generalized Catmull-Clark subdivision scheme, new face points and new edge points are calculated the same way as they are in a standard Catmull-Clark subdivision scheme, but the new vertex points

after one subdivision are calculated differently using the following formula.

$$V' = \frac{n-2}{n}V + \frac{1}{n^2}\sum_{j=1}^{n}(\alpha V + (1-\alpha)E_j) + \frac{1}{n^2}\sum_{j=1}^{n} f_j,$$

where $0 \leq \alpha \leq 1$ and f_j are the new face points after one subdivision. When $\alpha = 0$, it becomes the standard Catmull-Clark subdivision scheme. The limit point of the vertex V_i of degree n_i can be calculated as follows [9].

$$V_i^{\infty} = \frac{1}{n_i(n_i+5)}(b_{ii}V_i + \sum_j b_{ij}E_j + \sum_j b_{ij}F_j),$$

where

$$\begin{aligned}
b_{ii} &= (n_i-1)n_i + n_i\alpha + \textstyle\sum \frac{4}{d_{ij}} \\
b_{ij} &= (2-\alpha)(1 + \tfrac{2}{d_{ij}} + \tfrac{2}{d_{ji}}), \text{ if } (\mathrm{V_i}, \mathrm{V_j}) \text{ is an edge} \\
b_{ij} &= 4/d_{ij}, \text{ if } (\mathrm{V_i}, \mathrm{V_j}) \text{ is a diagonal line of a face} \\
b_{ij} &= 0, \text{ if } (\mathrm{V_i}, \mathrm{V_j}) \text{ is not in a common face}
\end{aligned}$$

Note that the above formula is used for a vertex whose surrounding faces might not be four-sided. Hence in the above formula, E_j are the edge points, but F_j are all the generalized faces points. d_{ij} are the number of sides of the face, of which (V_i, V_j) is an edge or a diagonal line (see figure (1)). Note that d_{ij} might not equal to d_{ji} because the two faces adjacent to the edge (V_i, V_j) could be of different side. But if (V_i, V_j) is a diagonal line of a face, $d_{ij} = d_{ji}$. According to the above definition, it is easy to see that

$$\mathrm{A}_{ij} = \frac{1}{n_i(n_i+5)}b_{ij}$$

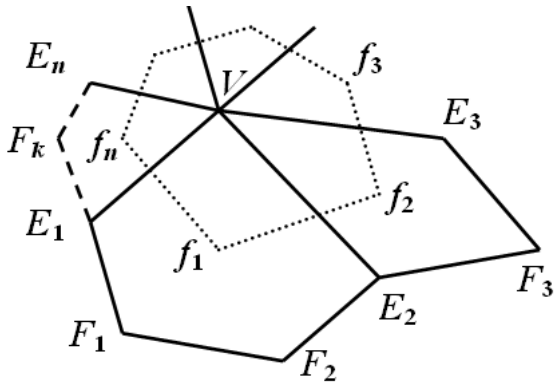


Figure 1: Vertex V with valence n and its neighboring face points and edge points. Note that faces might not be 4-sided.

6.2 Eigen Analysis of Matrix A

To prove A is invertible, and to prove $(\mathrm{E}-\mathrm{A})^i$ approaches $\mathbf{0}$ when i approaches ∞, we just need to prove that any A's eigen value $\lambda_i \in (0, 1]$. It is easy to check that $\mathrm{A}_{ij} \geq 0$ and for each row, $\sum_j \mathrm{A}_{ij} = 1$, hence $\lambda_i \leq 1$; Therefore we just need to show all the eigen values λ_i of A are $\lambda_i > 0$. A common coefficient $1/n_i/(n_i+5)$

can be factored out for each row of A. If we define a matrix B as $B_{ij} = b_{ij}$, A can be represented as $A = diag(1/n_i/(n_i+5)) * B$. It is easy to verify that B is symmetric. Hence to prove all B's eigen values are bigger than 0 is equivalent to prove B is positive definite.

For any vector $X \neq 0$, if $X^T BX > 0$ then B is positive definite. It is easy to see this if we expand $X^T BX$ as follows.

$$X^T BX = \sum_{all\ E} 2b_{ij}x_i x_j + \sum_{all\ D} 2b_{ij}x_i x_j + \sum_i b_{ii}x_i^2$$

$$= \sum_{all\ E}(b_{ij} - \frac{4}{d_{ij}} - \frac{4}{d_{ji}})(x_i + x_j)^2 + \sum_{all\ F}\frac{4}{d_{ij}}(x_i + x_j + \cdots + x_p)^2$$

$$+ \sum_i (b_{ii} - \sum_{(V_i,V_j)\ \in E}(b_{ij} - \frac{4}{d_{ij}} - \frac{4}{d_{ji}}) - \sum_{(V_i,V_j)\in E}\frac{4}{d_{ij}})x_i^2$$

where E denotes all edges, D denotes diagonal lines of faces and F denotes faces of the given mesh. Let

$$\rho_i = b_{ii} - \sum_{(V_i,V_j)\ \in E}(b_{ij} - \frac{4}{d_{ij}} - \frac{4}{d_{ji}}) - \sum_{(V_i,V_j)\in E}\frac{4}{d_{ij}}.$$

For now, we assume $\alpha = 0$, which gives us the standard Catmull-Clark subdivision surface. Because $b_{ij} > 0$, $\frac{4}{d_{ij}} > 0$ and $b_{ij} - \frac{4}{d_{ij}} - \frac{4}{d_{ji}} = 2 > 0$, to prove $X^T BX$, we just need to show $\rho_i > 0$. By plugging in b_{ii} and b_{ij}, we have $\rho_i = n_i^2 - 3 * n_i$.

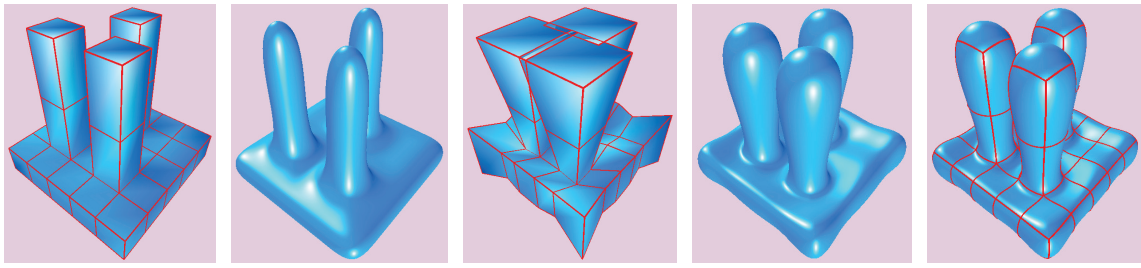
Obviously, when $n_i \geq 4$, we have $\rho_i > 0$. Hence B is positive definite. Also we can conclude that B is positive definite if there exists at least one vertex V_k in the given mesh such that $n_k \geq 4$, i.e., $\rho_k > 0$, then $X^T BX > 0$. This can be proven by contradiction. Suppose this is not the case, then there exists an $X \neq 0$ such that $X^T BX = 0$. It is easy to see that $x_k = 0$ otherwise $X^T BX \geq \rho_k x_k^2 > 0$. In addition, all x_j where (V_k, V_j) is an edge or a diagonal line must be 0 as well otherwise $X^T BX \geq (b_{kj} - \frac{4}{d_{kj}} - \frac{4}{d_{jk}})(x_k + x_j)^2 = (b_{kj} - \frac{4}{d_{kj}} - \frac{4}{d_{jk}})x_j^2 > 0$. Similarly, all vertices directly or indirectly connecting to V_k are all equal to 0. Because M is a connected mesh, all x_i are 0, which contradicts $X \neq 0$. Hence if there exists at least one vertex whose valance is bigger than 3, then B is positive definite as well.

When all n_i are 3 in a mesh and $\alpha = 0$, then B is not positive definite. However, we can change α to modify the standard Catmull-Clark subdivision scheme into a general Catmull-Clark subdivision scheme, such that B is positive definite. It is easy to verify when $\alpha > 0$, and $n_i \geq 3$, we have $\rho_i > 0$. Therefore to make B positive definite, we just need to find α, such that $b_{ij} - \frac{4}{d_{ij}} - \frac{4}{d_{ji}} >= 0$. By solving this inequality, we have $\alpha \in (0, \frac{6}{7}]$. Let α be any value in this range, say $\alpha = 0.5$, B becomes positive definite.

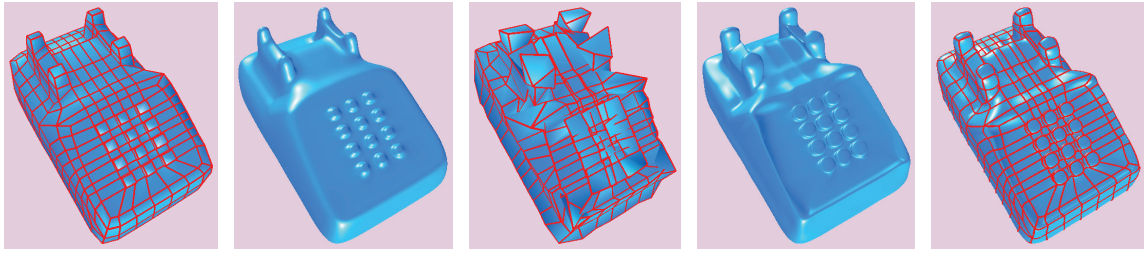
Because B is positive definite, A is positive definite as well. Therefore A is invertible and all its eigenvalues are $\in (0, 1]$. We call a mesh, whose A is invertible *uniquely interpolatable* using some subdivision scheme. According to the above proof, we can make any given mesh uniquely interpolatable using general Catmull-Clark subdivision scheme. For other subdivision scheme, similar proof can be done. For example, it can be proven for Loop and Doo-Sabin subdivision surfaces, it is always uniquely interpolatable for any given mesh.

6.3 Proof of Convergence in an Exponential Rate

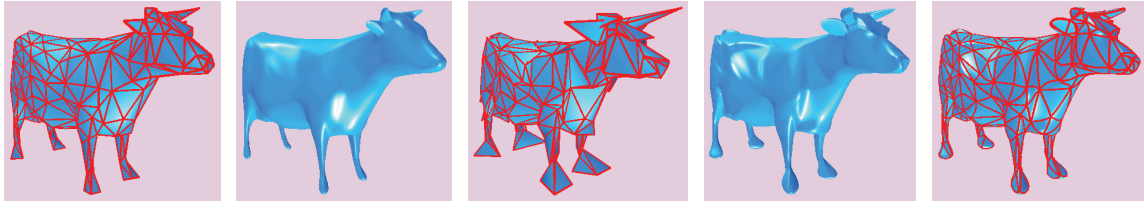
Because all A's eigenvalues are $\in (0, 1]$, all the eigenvalues of $(E - A)$ are $\in [0, 1)$. Hence $(E - A)^i$ converges to **0** when i approaches to infinity. As a result, $\sum_{i=0}^{n}(E - A)^i$ converges to A^{-1} when n approaches to infinity.



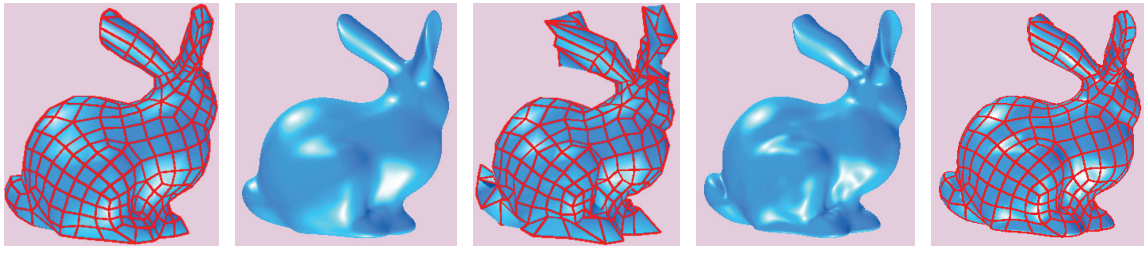
(a) Input mesh (b) Subdivision surface of mesh (a) (c) Interpolation mesh with the same topology as mesh (a) (d) Subdivision surface of mesh (c) (e) Surface (d) interpolating input mesh (a)



(f) Input mesh (g) Subdivision surface of mesh (f) (h) Interpolation mesh with the same topology as mesh (f) (i) Subdivision surface of mesh (h) (j) Surface (i) interpolating input mesh (f)



(k) Input mesh (l) Subdivision surface of mesh (k) (m) Interpolation mesh with the same topology as mesh (k) (n) Subdivision surface of mesh (m) (o) Surface (n) interpolating input mesh (k)



(p) Input mesh (q) Subdivision surface of mesh (p) (r) Interpolation mesh with the same topology as mesh (p) (s) Subdivision surface of mesh (r) (t) Surface (s) interpolating input mesh (p)

Fig. 2: Examples of iterative mesh interpolation.

Also it can be proven that eq. (3) or eq. (5) converges in an exponential rate. To prove this it is sufficient to prove $||P_\infty - P_n||$ converges to $\mathbf{0}$ in an exponential rate.

$$\begin{aligned}
&||P_\infty - P_n|| \\
&= ||\sum_{i=n+1}^{\infty}(\mathsf{E}-\mathsf{A})^i M|| \\
&= ||(\mathsf{E}-\mathsf{A})^{n+1} * \mathsf{A}^{-1} * M|| \\
&= ||\mathsf{X}\Lambda^{n+1}\mathsf{X}^{-1} * \mathsf{A}^{-1} * M|| \\
&<= ||\Lambda^{n+1}|| * ||\mathsf{X}|| * ||\mathsf{X}^{-1}|| * ||\mathsf{A}^{-1}|| * ||M|| \\
&= ||\Lambda^{n+1}|| * k,
\end{aligned}$$

where k is a constant and Λ are the diagonal matrix whose Λ_{ii} is the ith eigenvalue of matrix $(\mathsf{E}-\mathsf{A})$. Suppose the biggest eigenvalue of matrix $(\mathsf{E}-\mathsf{A})$ is λ, then $||\Lambda^{n+1}|| \leq \lambda^{n+1}$. As proved above, $0 \leq \lambda < 1$, hence we have

$$||P_\infty - P_n|| \leq k * \lambda^{n+1}$$

According to the definition of convergence rate, it can be seen that P_n converges to P_∞ exponentially.

7 Test Results

The proposed techniques have been implemented in *C++* using *OpenGL* as the supporting graphics system on the Windows platform. The implementation is simple and straightforward. Quite a few examples have been tested with the techniques described here. All the examples have extra-ordinary vertices. Some of the tested results are shown in Figure (2). The intention of our implementation is to demonstrate the versatility and capability of the new representation for surfaces and meshes given in eq. (5), hence we did not compare our results with other similar methods. Nevertheless, our method is easy to understand, easy to implement, and yet visually pleasant results still are achievable. One main advantage of our new method is that it is very fast (as proved theoretically in the above section). All the test examples shown in this paper are done in less than a second in a normal laptop.

8 Summary

We present a fast iterative algorithm for constructing a smooth subdivision surface that interpolates the vertices of a mesh with arbitrary topology. The interpolating surface is obtained by iteratively adjusting the vertex positions locally. Hence there is no need for solving linear systems, and no need for fairing in the interpolation surface construction process. Because of this, the new method can handle meshes with large number of vertices and yet good interpolation results can still be achieved. It is proved that for any mesh, the iterative process is convergent at an exponential rate, hence the method is very fast and effective. It is also proved that of all the meshes that have the same topology of a given mesh, there is only ONE mesh whose limit surface interpolates the given mesh. Our iterative interpolation method converges to this mesh.

ACKNOWLEDGEMENTS

This research is supported in part by NNSFC (Grant No. 61572020)

Shuhua Lai, http://orcid.org/0000-0001-5678-6848
Fuhua (Frank) Cheng, http://orcid.org/0000-0003-2098-3027

REFERENCES

[1] Catmull, E.; Clark, J.: Recursively generated b-spline surfaces on arbitrary topological meshes. Computer-Aided Design, 1978.

[2] Chen, Z.; Luo, X.; Tan, L.; Ye, B.; Chen, J.: Progressive interpolation based on catmull-clark subdivision surfaces. In Advances in Geometric Modeling and Processing, 2009. http://doi.org/10.1111/j.1467-8659.2008.01328.x.

[3] Cheng, F.; Fan, F.; Lai, S.; etc: Progressive interpolation using loop subdivision surfaces. In Advances in Geometric Modeling and Processing, 2008. http://doi.org/10.1007/978-3-540-79246-8_43.

[4] DeRose, T.; Kass, M.; Truong, T.: Subdivision surfaces in character animation. In Proceedings of SIGGRAPH, 1998. http://doi.org/10.1145/280814.280826.

[5] Doo, D.; Sabin, M.: Behavior of recursive division surfaces near extraordinary points. Computer-Aided Design, 1978.

[6] Dyn, N.; Levin, D.; Gregory, J.A.: A butterfly subdivision scheme for surface interpolation with tension control. ACM Transactions on Graphics, 1990.

[7] Halstead, M.; Kass, M.; DeRose, T.: Efficient, fair interpolation using catmull-clark surfaces. In Proceedings of SIGGRAPH, 1993. http://doi.org/10.1145/166117.166121.

[8] Kobbelt, L.: Interpolatory subdivision on open quadrilateral nets with arbitrary topology. Computer Graphics Forum, 1996.

[9] Lai, S.; Cheng, F.: Parametrization of general catmull clark subdivision surfaces and its application. Computer Aided Design & Applications, 2006. http://doi.org/10.1080/16864360.2006.10738436.

[10] Lai, S.; Cheng, F.: Similarity based interpolation of meshes with arbitrary topology using catmull-clark subdivision surfaces. The Visual Computer, 2006. http://doi.org/10.1007/s00371-006-0072-9.

[11] Levin, A.: Interpolating nets of curves by smooth subdivision surfaces. In Proceedings of SIGGRAPH, 1999. http://doi.org/10.1145/311535.311541.

[12] Loop, C.T.: Smooth subdivision surface based on triangle. Ph.D. thesis, Department of Mathematics, University of Utah, 1987.

[13] Nasri, A.H.: Surface interpolation on irregular networks with normal conditions. Computer Aided Geometric Design, 1991.

[14] Nasri, A.H.; Sabin, M.A.: Taxonomy of interpolation constraints on recursive subdivision curves. The Visual Computer, 2002.

[15] Reif, U.: A unified approach to subdivision algorithms near extraordinary points. Computer Aided Geometric Design, 1995.

[16] Schaefer, S.; Warren, J.: A factored interpolatory subdivision scheme for quadrilateral surfaces. Curves and Surface Fitting, 2002.

[17] Stam, J.: Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In Proceedings of SIGGRAPH, 1998. http://doi.org/10.1145/280814.280945.

[18] Zhang, L.; She, X.; Ge, X.; Tan, J.: Adaptive fitting algorithm of progressive interpolation for loop subdivision surface. International Journal of Distributed Sensor Networks, 2018.

[19] Zorin, D.; Schröder, P.; Sweldens, W.: Interpolating subdivision for meshes with arbitrary topology. In Proceedings of SIGGRAPH, 1996. http://doi.org/10.1145/237170.237254.