Title:
3D Shape Generation by Iterative Interpolation of Meshes with Arbitrary Topology Using General Catmull-Clark Subdivision Surfaces

Authors:
Shuhua Lai, slai@ggc.edu, Georgia Gwinnett College
Jason Lai, jlai@gsmst.edu, Gwinnett School of Mathematics, Science, and Technology
Anastasia Kazadi, ansm226@g.uky.edu, University of Kentucky
Seifalla A. Moustafa, seifalla.moustafa@uky.edu, University of Kentucky
Fuhua (Frank) Cheng, cheng@cs.uky.edu, University of Kentucky

Keywords:
Subdivision Surface, Interpolation, 3D Mesh, 3D Shape Generation, General Catmull-Clark Subdivision Surface

DOI: 10.14733/cadconfP.2022.88-92

Introduction:
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 for the 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 control points, which converges to a mesh whose subdivision surface interpolates the given input mesh M. Most importantly, this infinite iterative series of control points can be calculated locally based on the chosen subdivision scheme. Hence, the matrices A and A^{-1} do not have to be actually 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 until some given error tolerance is reached.

The concept of iterative interpolation has been presented in the literature before [5, 6]. The main differences between our algorithm and existing approaches are fivefold: 1. Our algorithm is the first one that derives the iterative equation from the perspective of computing A^{-1}. 2. The existence, convergence and uniqueness of A^{-1} are guaranteed (the proof will be provided in the complete paper). 3. Our iterative interpolation algorithm, converging 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.

Fast Iterative Interpolation:
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 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 M, i.e., if P is the control mesh of $\mathbf{I}(M)$ then we must have $M = A^* P$. We also assume the subdivision

scheme considered here is the Catmull-Clark scheme. Hence, **I**(*M*) and **S**(*M*) are Catmull-Clark subdivision surfaces. However, the techniques presented here work for other subdivision schemes as well.

To find **I**(*M*), we just need to find *P* because **I**(*M*) = **S**(*P*). As mentioned above, $P = A^{-1} * M$. For any given mesh (A is only dependent on the given mesh and the subdivision scheme), 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. However, we should not directly compute the A^{-1} because the matrix A could be huge (the size of A is *N*×*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, which is very similar to the famous Newton iterative approach in obtaining equation roots.

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

$$L = \sum_{i=0}^{\infty} (E - A)^i = (E - A)^0 + (E - A)^1 + (E - A)^2 + (E - A)^3 + \cdots \tag{2.1}$$

where E is an identity matrix that has the same dimension of A. The convergence of the above infinite series can be proven with the eigen analysis of A. By multiplying (E–A) to both sides of Eq. (2.1), we have

$$(E - A)L = (E - A)^1 + (E - A)^2 + (E - A)^3 + (E - A)^4 + \cdots \tag{2.2}$$

By subtracting the left (right) side of Eq. (2.2) from the left (right) side of Eq. (2.1), respectively, we get

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

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

$$A^{-1} = L = \sum_{i=0}^{\infty} (E - A)^i \tag{2.3}$$

Eq. (2.3) 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, we have

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

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

$$M_i = (E - A)^i M.$$

By eigen analysis of matrix A, it can be proven that $\lim_{i \to \infty} M_i = 0$. As a result, **I**($M_i$) approaches to **0** when i tends to ∞. In addition, because subdivision is a linear process, we have $\sum S(M_i) = S(\sum M_i)$. Therefore, *P* is the control mesh of the interpolating surface **I**(*M*), i.e., **I**(*M*) = **S**(*P*). Based on the definition of M_i, we have

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

Based on the above equation, for any $n \geq 0$, we can have the following

$$\begin{cases} M_{n+1} = M_n - A * M_n \\ M_n = M_{n-1} - A * M_{n-1} \\ M_{n-1} = M_{n-2} - A * M_{n-2} \\ M_{n-2} = M_{n-3} - A * M_{n-3} \\ \cdots \\ M_1 = M_0 - A * M_0 \end{cases} \tag{2.4}$$

Adding up all the values on the left side and adding up all the values on the right side of all equations in Eq. (2.4), respectively, also letting

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

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

$$P_{n+1} = P_n + M - A * P_n \tag{2.5}$$

Eq. (2.5) 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. (2.3) should not be used to construct the interpolating surface directly, because it requires costly matrix multiplications. Actually, there is no need to compute the matrix A at all, let alone computing the matrix A^{-1}, in the construction of the interpolating surface. We can use Eq. (2.5) to find the control mesh $P = P_\infty$ of the interpolating surface iteratively and this is because $A * P_n$ can be calculated locally 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, even though they are used in the derivation process of Eq. (2.5). On the other hand, because A is invertible, it is easy to see that $A^{-1} * M = P$. 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 $A^{-1} * M$ 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. (2.5) until some given error tolerance is reached (see Section 4). Hence there is no problem to deal with meshes with large number of vertices.

It can be proven that Eq. (2.5) converges at an exponential rate. Hence good interpolation results can be obtained in just a few iterations. Nevertheless, error can be explicitly calculated as $\|M - A * P_n\|$ 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 (1) 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.

Algorithm for Iterative Interpolation:

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

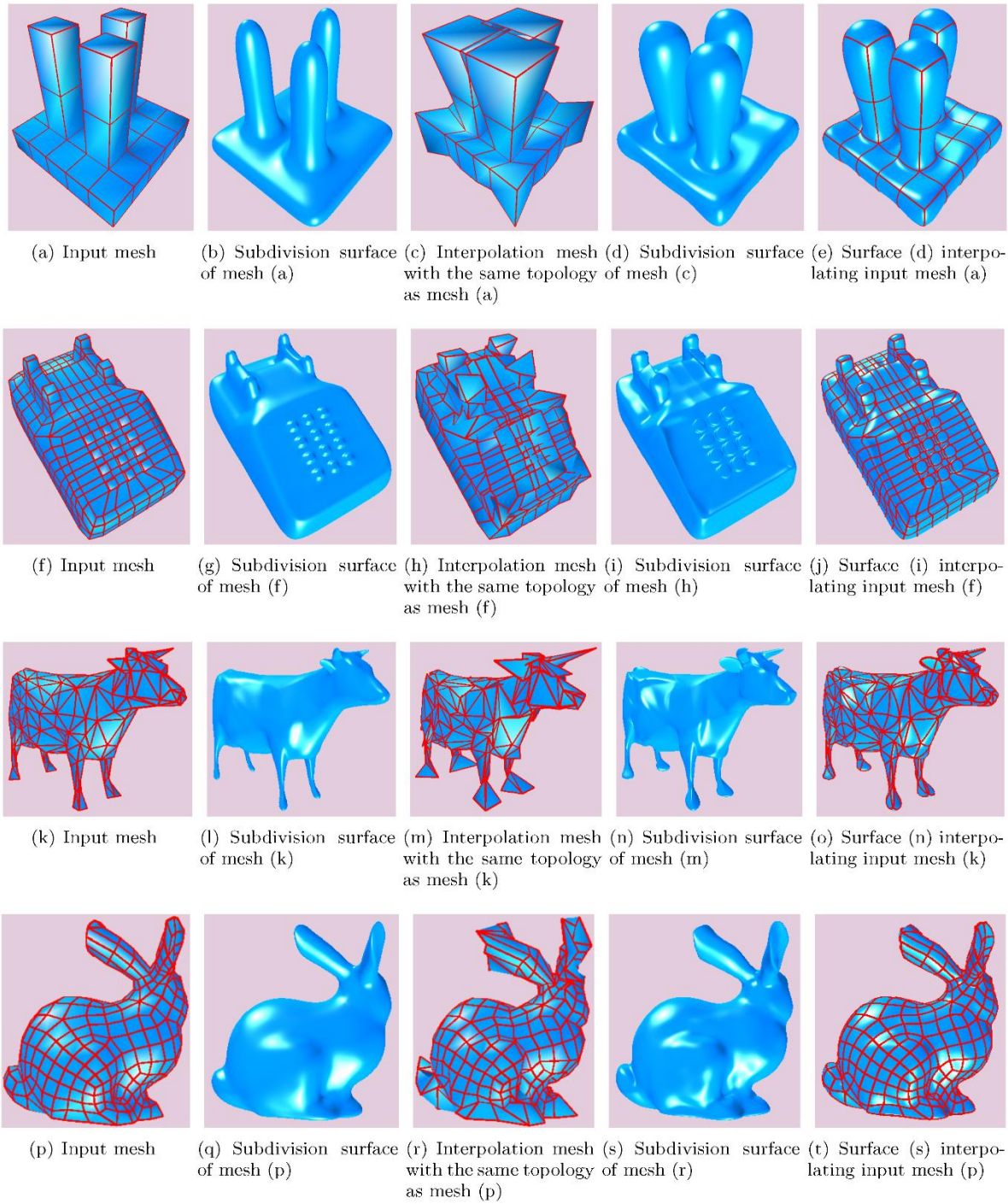

Fig. 1: Examples of iterative mesh interpolation.

Iterative Interpolation Algorithm:

Input: a mesh M, an error tolerance ϵ, and a choice of subdivision scheme.

Output: a mesh P, whose subdivision surface interpolates M.

Let $P = M$;
Do {
    Locally calculate $Q = A * P$ 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.

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 1. The intention of our implementation is to demonstrate the versatility and capability of the new representation for surfaces and meshes given in eq. (2.5), 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 few seconds in a normal laptop.

Concluding Remarks:

A fast iterative algorithm for constructing a smooth subdivision surface that interpolates the vertices of a mesh with arbitrary topology is presented. The interpolating surface is obtained by iteratively adjusting the vertex positions locally. Hence there is no need for solving large 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 generated. It is proved that for any mesh, the iterative process is convergent in 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.

References:

[1] Catmull, E.; Clark, J.: Recursively generated B-spline surfaces on arbitrary topological meshes, Computer-Aided Design, 1978, 10(6):350-355. https://doi.org/10.1016/0010-4485(78)90110-0

[2] Lai, S.; Cheng, F.: Parametrization of General Catmull-Clark Subdivision Surfaces and its Application, Computer Aided Design & Applications, 3, 1-4 (2006), 513-522. https://doi.org/10.1080/16864360.2006.10738490

[3] Zorin, D.; Schröder, P.; Sweldens, W.: Interpolating Subdivision for meshes with arbitrary topology, ACM SIGGRAPH, 1996, 189-192. https://doi.org/10.1145/237170.237254

[4] Kobbelt, L.: Interpolatory subdivision on open quadrilateral nets with arbitrary topology, Computer Graphics Forum, Eurographics, V.15, 1996. https://doi.org/10.1111/1467-8659.1530409

[5] Chen, Z.; Luo, X.; Tan, L.; Ye, B.; Chen, J.: Progressive Interpolation based on Catmull-Clark Subdivision Surfaces, Proceedings of Advances in Geometric Modeling and Processing 2009. https://doi.org/10.1111/j.1467-8659.2008.01328.x

[6] 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. https://doi.org/10.1177/1550147718812355