Submission to GMP 2015

One-step Bicubic Interpolation

Submission number: 77

Abstract

In this paper, a new interpolation scheme for Catmull-Clark subdivision (CCS) surfaces is introduced. With this new scheme, one can generate a bicubic interpolating surface to interpolate a set of millions of data points in just one step, instead of an iterative process usually required for large data sets. Furthermore, the computed interpolating surface has the same local property as CCS surface, i.e., changing a data point will only change the shape of the interpolating surface locally, the first time ever to have such a property for a bicubic CCS interpolating scheme. The construction process is based on two techniques: surface offsetting and mesh decomposition . The surface offsetting technique ensures the shape of the data set is faithfully resembled, so the method has the power of a global method; the mesh decomposition technique enables us to solve the problem using a one-step, local approach, instead of solving a global linear system using an iterative approach. Test results show that interpolating surfaces can be efficiently generated by the new method for large data sets and the generated interpolating surfaces have very high surface quality. Hence, the new scheme is especially suitable for applications in reverse engineering and 3D printing.

1 Introduction

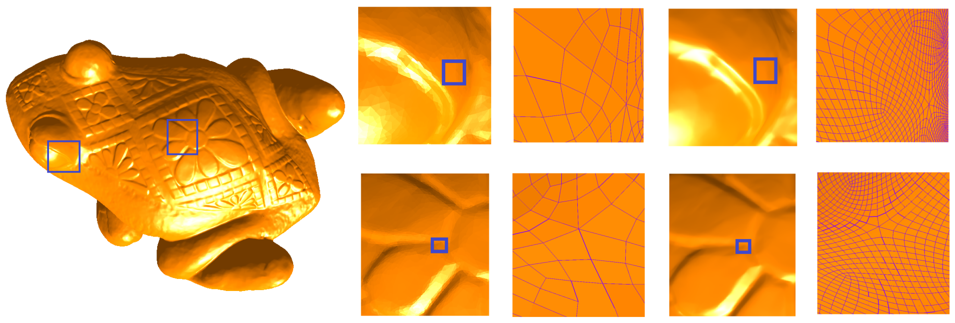

Figure 1: Frog example by our new interpolation scheme. Left is the interpolating surface. Right two rows are enlarged view of frog eye and back pattern, from left to right: a) original surface; b) enlarged control mesh of blue box in a); c) our new interpolating surface; d)enlarged data mesh of blue box in b).

Freeform surfaces are widely used in computer graphics. Traditionally NURBS handles freeform surfaces in CAD/CAM [5]. NURBS have a rigid rectangular control grid, therefore surfaces are represented by a collection of trimmed patches, and their continuity across the patch boundaries had to be manually enforced.

In the past decade, subdivision surfaces have become popular in surface representation. They are simpler than traditional spline methods and are able to handle arbitrary topology. Subdivision schemes use mainly three types of mesh structure: quadrilateral, triangular and hexagonal. Quad faces and Triangular faces are commonly used for practical applications. Catmull-Clark subdivision (CCS) scheme [2] and Loop scheme [9] are the most widely used schemes in quad and triangular mesh structures, respectively. In particular, Catmull-Clark subdivision surfaces (CCSSs) have become a standard modeling/representation scheme in computer animation and gaming.

A Catmull-Clark subdivision surface (CCSS) is the limit surface of a sequence of subdivision steps performed on a given control mesh. At each step new vertices are introduced and old vertices are updated. The CCS scheme is an approximating scheme, i.e., a CCSS smoothly approximates, but does not interpolate the given control mesh. However, construction of smooth interpolating surfaces is important in many applications, including computer aided design, statistical data modeling and face recognition. This means, given a "data mesh", one needs to construct a control mesh so that the CCS limit surface of this control mesh would interpolate the given data mesh.

This interpolation problem can be solved directly or iteratively. A direct method such as the earlier work of Halstead [6] can be used if the data mesh is relatively small or the corresponding linear system is non-singular. For data mesh with hundreds of data points, or the corresponding linear system is singular, a progressive subdivision scheme [3] [4] can be used. This method iteratively generates a new control mesh by adding to the current control mesh the difference between the current control mesh and its corresponding data points on the CCS limit surface. The resulting linear system is positive definite and improves the convergence speed of the CCS control mesh generation process.

Besides the convergence speed issue, the interpolating surface obtained sometimes could possess excessive undulations [6]. The Fairing techniques proposed in [10] [14] smooth an interpolating surface by including more constraints but that also increases the size of the control mesh. Some alternative methods [7] [15] improve shapes by choosing good initial control mesh or adding more control points to control the shape locally.

A recent iterative approach has the advantages of both a local method and a global method [8], i.e., it can handle meshes of thousands of data points and complex topology while capable of faithfully reproducing the shape of any given mesh. Besides, this approach provides a way to expand a mesh into an infinite series of meshes (surfaces) which allows classical applications such as texture mapping and morphing to be solved differently.

But the above iterative interpolating methods are all approximating and they all have an efficiency problem and computation errors when the number of data points is millions. We will present a solution to this problem in this paper, i.e., we will present a precise interpolating scheme for CCS that can efficiently handle data sets with millions of data points. The new method can generate a bicubic surface to interpolate a set of millions of data points in just one step. Furthermore, the computed interpolating surface has the same local property as a CCS surface, i.e., changing a set of data points will only change the shape of the interpolating surface locally around these data points, the first time ever to have such a property for a CCS interpolating scheme.

The construction process is based on two techniques: mesh decomposition and surface offsetting. The mesh decomposition technique enables us to solve the problem using a one-step, local approach; the surface offsetting technique ensures the shape of the data set is faithfully resembled. Hence the method has the advantages of both a local method and a global method, and yet it does not require an iterative approach. According to our knowledge, this is the first time a combination of mesh decomposition and surface offsetting is used in subdivision surface interpolation process. Test results show that the new method produce very good results for large data sets. Fig. 1 shows a frog with 1,200,002 interpolation points, running time of our new scheme to compute all data points on the limit surface is 49.912 seconds, only slightly higher than that of CCS scheme (38.530 seconds). From the enlarged view of frog eye and back pattern, we see that even though the frog has million of control points, without subdivision it is only C^0 continuous after zoom-in, while after our interpolation, the limit surface is C^2 everywhere except at extraordinary points where it is C^1 continuous.

The remaining part of the paper is organized as follows. Section 2 covers earlier CCS interpolation schemes. Section 3 introduces the concept of the new one-step scheme. Section 4 is the mathematical setup of the new scheme. Section 5 discusses the behavior of the new scheme. Section 6 concludes.

2 Related Work

2.1 CCS and mesh structure

A mesh of arbitrary topology into a CCS mesh with only quadrilateral faces and each face has at most one extraordinary vertex in at most two recursive subdivision steps [2]. The CCS scheme divides the control vertices of a given/converted CCS mesh into three categories: *vertex points*, *edge points*, and *face points*.

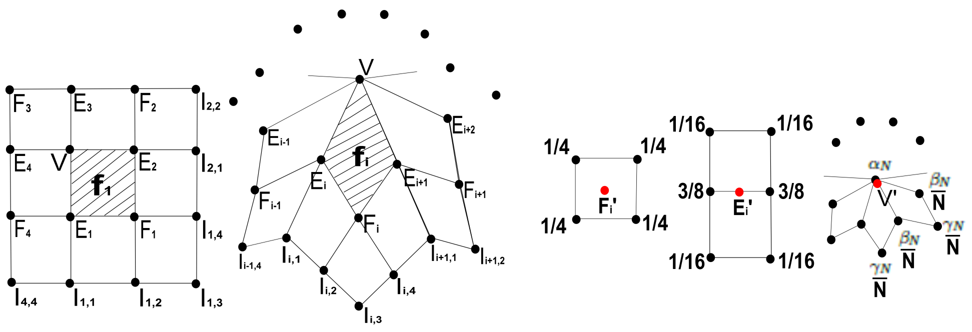

Figure 2: (a),(b): CCS regular and extraordinary faces;(c),(d),(e): CCS subdivision masks for new face, edge and vertex points.

A popular way to index the control vertices of a subdivision face is shown on the Fig. 2(a) for a regular face and the Fig. 2(b) for an extraordinary face, where V is a vertex point, E_i's are edge points, F_i's are face points and $I_{i,j}$'s are inner ring control vertices. New vertices within each subdivision step are generated as follows:

$$\begin{aligned} V' &= \alpha_N V + \beta_N \sum_{i=1}^{N} E_i/N + \gamma_N \sum_{i=1}^{N} F_i/N \\ E_i' &= \frac{3}{8}(V + E_i) + \frac{1}{16}(E_{i+1} + E_{i-1} + F_i + F_{i-1}) \\ F_i' &= \frac{1}{4}(V + E_i + E_{i+1} + F_i) \end{aligned} \tag{1}$$

where N is the valence of vertex V, with $\alpha_N = 1 - \frac{7}{4N}$, $\beta_N = \frac{3}{2N}$, *and* $\gamma_N = \frac{1}{4N}$. These subdivision rules work for the inner ring control vertices as well since these control vertices and the subsequently generated new control vertices are also vertex, edge or face points. For control vertices generated after n^{th} subdivisions [11], we have

$$\mathbf{C_n = A^n C_0}, \qquad \mathbf{\bar{C}_n = \bar{A} A^{n-1} C_0}, \quad \mathbf{n \geq 1}, \tag{2}$$

where $\mathbf{C_n}$ is $2N + 8$ control vertices of $\mathbf{f_i}$ (Fig 2) after n^{th} subdivision , $\mathbf{\bar{C}_n}$ is $2N + 17$ control vertices after one subdivision on $\mathbf{C_{n-1}}$, N is valence of V, $\mathbf{A}$ and $\mathbf{\bar{A}}$ are their corresponding extended subdivision matrices with size $(2N + 8) \times (2N + 8)$ and $(2N + 17) \times (2N + 8)$ respectively, and $\mathbf{C_0}$ is the original $(2N + 8)$ control vertices of $\mathbf{f_i}$.

2.2 CCS interpolation schemes

Note that , a CCS limit surface will not interpolate the control vertices of its control mesh, but approximate them. To interpolate a given data mesh with CCS, traditionally, it is achieved by solving a global linear system [6] [1] [12] [3] [4],

$$\widetilde{\mathbf{A}}x = b \tag{3}$$

where $\widetilde{\mathbf{A}}$ is a square matrix determined by subdivision rules and mesh topology, x is a column vector of control points to be determined, b is a column vector of data points in the given data mesh. If $\widetilde{\mathbf{A}}$ is small and nonsingular, we can directly obtain the control mesh by calculating its inverse $\widetilde{\mathbf{A}}^{-1}$ directly. However, a direct method will not work or not work well if $\widetilde{\mathbf{A}}$ is singular or of large size. In such a case, an iterative method needs to be applied. Traditionally, stationary iterative methods like Jacobi, Gauss-Seidel or Successive Over-relaxation can be used to solve a large linear system. The issue with these methods is the convergence rate - they are slow when data set is large. When $\widetilde{\mathbf{A}}$ is singular, the least-squares method can be applied. There are faster iterative methods to solve large scale data set [1] [12], however since (3) is a global system, convergence rate will still be not satisfactory when we are dealing with thousands of data points.

To avoid dealing with singular linear systems and to improve iteration speed, a progressive subdivision scheme [3] [4] was developed. This method iteratively generates a new control mesh by adding to current control mesh the difference between this control mesh and its corresponding data points on the CCS limit surface. Its linear system developed is positive definite and can improve the convergence speed of CCS control mesh generation process which satisfies (3). Recently in [8] a fast iterative scheme is presented and it is shown that their iterative process converges to a unique solution.

Besides convergence speed, the interpolating surface obtained by solving (3) sometimes is unsatisfactory because of excessive undulations [6]. Halstead [6] notes that the the undulations appear because they are not indicated by the shape of original mesh.

The Fairing techniques proposed in [10] [14] smooth an interpolating surface by including more constraints but increasing size of control mesh. Some alternative methods [7] [15] improve shapes by choosing good initial control mesh or adding more control points to control shape locally.

The above iterative methods focus on improving convergence speed of solving (3) or introducing additional constraints to handle surface artifact, they are all approximating, not exactly interpolating. Two questions remain unsolved,

1. by solving a global linear system, the obtained interpolation control mesh depends on all control vertices of original data mesh, such that the scheme lacks local support.
2. convergence speed is not satisfactory when handle large data-set.

It is natural to ask the following question:

"Is it possible to have a precise interpolating scheme other than approximating ones, without solving a global linear system, and not iterative, while preserving the easy implementation and local support features of CCS?"

Most recently, a direct scheme is developed in [13], this scheme generates an interpolation surface by directly applying a bi-quintic Bezier crust on CCS limit surface. Although its surface quality is similar to that of CCS limit surface, we would like to explore a solution of using lower degree polynomials instead of attaching a bi-degree 5 spline surface.

In this paper, we present a bicubic one-step CCS interpolation scheme by reducing the CCS global interpolation problem to finding a local offsetting surface. Fig. 3 shows a hollowed cube example implemented both by our new interpolation scheme and traditional scheme. We see that with traditional scheme, the converted control mesh will be away from the shape of original mesh, such that its interpolation surface has unwanted undulation, while the interpolation surface generated by our new scheme is similar to original mesh and does not show significant undulation.

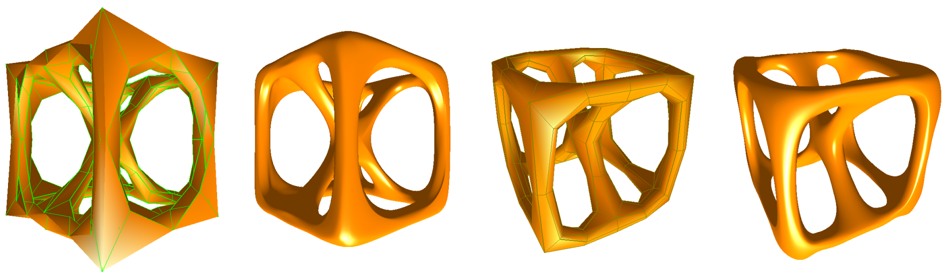

Figure 3: one example of new interpolation scheme. (a): converted control mesh by solving (3); (b): CCS limit surface of converted mesh; (c): given data mesh; (d): limit surface of the new interpolation scheme;

3 One Step Bi-cubic Interpolation

As stated in previous sections, current CCS interpolation schemes with iterative approaches suffer from the slow convergence when data set size is large, especially, when data set is of size millions, these iterative approaches could not handle. In this paper, we introduce a global/local hybrid method. The idea is to separate interpolation surface into two parts, which we define as base and offsetting surfaces. Given a data mesh to interpolate, the base surface is its

CCS limit surface, and the offsetting surface is the surface interpolating difference vectors between interpolating points and their CCS limit points. The new interpolation surface is obtained by adding these two surfaces parametrically (as shown in Fig. 4). With proper selection of the offsetting surface, we can generate interpolating surface directly without iteration, such that the new scheme can handle extreme large data sets efficiently.

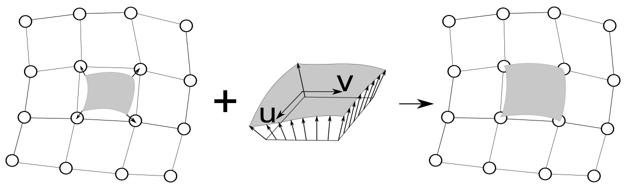

Figure 4: Left: given data mesh with its CCS limit surface (gray shaded), middle: offsetting surface (gray shaded) interpolating difference vectors; right: interpolating surface (gray shaded)

The base surface of our new scheme is obtained by CCS, so we restrict the subdivision scheme of the offsetting surface to CCS. Also, to make the generated interpolation surface C^1 at extraordinary data points and C^2 everywhere else, we expect the mesh of offsetting surface shall have the same number of extraordinary points as that of the base surface. We propose that the mesh structure of offsetting surface corresponding to one base face could be 1x1, 2x2, 3x3,...(Fig. 5).

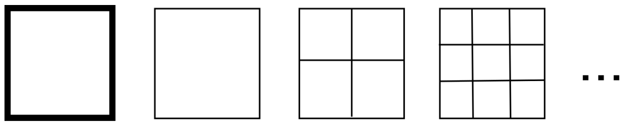

Figure 5: (a):base face; (b),(c),(d): possible corresponding offsetting mesh faces of 1x1, 2x2, 3x3, ...

We expect the offsetting surface to be computed directly, and the shape of final interpolation surface to be as close to that of base surface as possible. Additional constraints could to be applied that the surface normals of interpolated data points on interpolation surface shall be the same as those of base surface.

Theorem 1. *For mesh decomposition of offsetting mesh corresponding to one face of base mesh, 3x3 is the minimum mesh decomposition to compute interpolation of each difference vector locally without involving other difference vectors, such that the surface normals on the interpolation surface are the same as those of the base surface.*

Proof. If mesh decomposition is 1x1, then the offsetting mesh has the same mesh structure of base mesh. By (3), the offsetting mesh has to be computed globally and iteratively.

If mesh decomposition is 2x2, then computing the offsetting mesh is exactly the partial interpolation schemes of adding one layer of control vertices. Although partial interpolation can change the shape locally, by (3), the offsetting mesh still has to be computed globally and iteratively as illustrated in [7] [15].

If mesh decomposition is 3x3 or larger, then by (1), the computation of offsetting mesh to interpolate a difference vector and its normal can be computed locally without impacts from other difference vectors. QED □

With Theorem 1, we choose mesh decomposition of offsetting mesh to be 3x3, such that the free variables can be minimized while interpolation of difference vectors can be computed locally. With this selection, the coefficient matrix of $\widetilde{\mathbf{A}}$ for offsetting mesh has the following form,

$$\widetilde{\mathbf{A}} = \begin{bmatrix} a_1 & & & & \\ & a_2 & & & \\ & & ... & & \\ & & & ... & \\ & & & & a_m \end{bmatrix} \tag{4}$$

where a_k are row vector of the k^{th} interpolation data point b_k, and all other entries in $\widetilde{\mathbf{A}}$ are zero. Such that (3) can be rewritten as

$$a_1 x_1 = b_1,\ a_2 x_2 = b_2,\ ...,\ a_m x_m = b_m \tag{5}$$

With mesh structure shown in Fig. 5(d) and equation (5), we separate a global linear system of (3) into a group of local linear system for each interpolation data point. With this, we are possible to construct a direct interpolation scheme on CCS data mesh. Since this interpolating control mesh of offsetting surface divides a face of base mesh into three equal parametric segments in u and v direction, we name it 1/3 scheme.

With selection of 1/3 scheme on offsetting mesh, our algorithm works as follows. Given a CCS data mesh M , we can derive all difference vectors (between data point and its CCS limit point if we implement CCS on M) on each data points. Then we construct offsetting mesh using 1/3 scheme shown in Fig 5(d). If we parametrically add the limit surfaces of M and ΔM (Fig. 6), we obtain a limit surface which interpolates M. This interpolation surface has the same surface continuity as base surface.

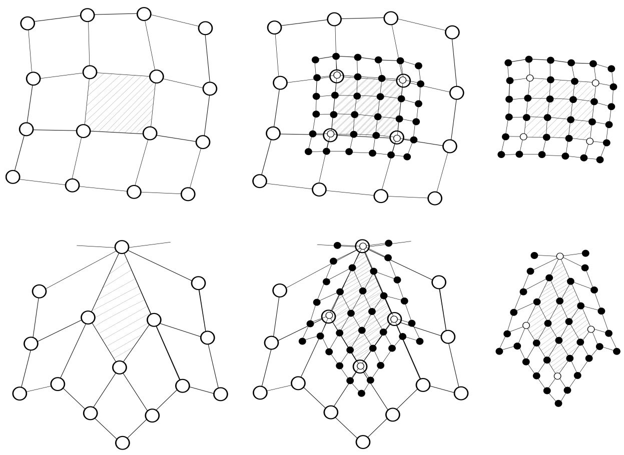
Figure 6: 1/3 scheme on regular and extraordinary faces. (a) base mesh (b) offsetting mesh together with the base mesh, (c) offsetting mesh

With our new scheme on offsetting mesh, we separate the global linear system into a set of local linear sub-systems, the computed offsetting mesh is less fluctuated. And the resulting interpolation surface has the same local support on each data point and the quality of CCS limit surface on M can be preserved.

4 Mathematical Setup

In this section, we put our new scheme into rigorous mathematical setting, and show the properties of the resulting interpolating surface.

Our new interpolating surface is the sum of two parametric surfaces, one is the base surface, just CCS limit surface of given data mesh, another is the CCS offsetting surface, which interpolates the difference vectors between given data points and their corresponding points on its CCS limit surface.

The base surface, the CCS limit surface of given data mesh, can be parameterized, its parameterization is as follows,

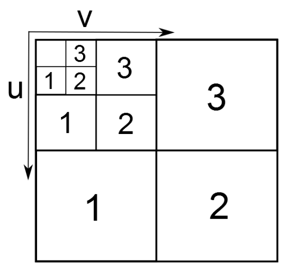

Figure 7: Ω-Partition of CCS

First we define the limit surface of a CCS face f_i as $S(u, v)$, the three regular bicubic B-Spline patches after the n-th CCS as $S_{n,b}$, $n \geq 1$, $b = 1, 2, 3$. The Ω-partition (Fig. 7)is defined by:

$\Omega_{n,b}$, $n \geq 1$, $b = 1, 2, 3$, with

$$\begin{aligned}\Omega_{n,1} &= (\frac{1}{2^n}, \frac{1}{2^{n-1}}] \times [0, \frac{1}{2^n}] \\ \Omega_{n,2} &= (\frac{1}{2^n}, \frac{1}{2^{n-1}}] \times (\frac{1}{2^n}, \frac{1}{2^{n-1}}] \\ \Omega_{n,3} &= [0, \frac{1}{2^n}] \times (\frac{1}{2^n}, \frac{1}{2^{n-1}}]\end{aligned}$$

For any $(u, v) \in [0, 1] \times [0, 1]$, $(u, v) \neq (0, 0)$, there is an $\Omega_{n,b}$ containing (u, v). We can find the value of $S(u, v)$ by mapping $\Omega_{n,b}$ to the unit square $[0, 1] \times [0, 1]$ and finding the corresponding $(\overline{u}, \overline{v})$.

After the mapping, we compute $S_{n,b}$ at $(\overline{u}, \overline{v})$. The value of $S(0, 0)$ is the limit at $(0, 0)$.

In the above process, n and b can be computed by:

$$\begin{aligned}n(u, v) &= min\{\lceil \log_{\frac{1}{2}} u \rceil, \lceil \log_{\frac{1}{2}} v \rceil\} + 1 \\ b(u, v) &= k,\ if\ (u, v) \in \Omega_{n,k},\ k = 1, 2, 3\end{aligned}$$

The $S(u, v)$ can be expressed as follows

$$S(u, v) = W^T(u, v) K^n D_b M \mathbf{C_n^b} \tag{6}$$

where $W(u, v)$ is the 16-power-basis vector with$[1, u, v,\ u^2, uv, v^2, u^3, u^2v, uv^2, v^3, u^3v, u^2v^2, uv^3, u^3v^2, u^2v^3, u^3v^3]$, M is the B-spline coefficient matrix, K is a diagonal matrix with $Diag(1, 2, 2, 4, 4, 4, 8, 8, 8, 8, 16, 16, 16, 32, 32, 64)$, and D_b is an upper triangular marix depending on b only.

$\mathbf{C_n^b}$ is the control points vector of $S_{n,b}$, with

$$\mathbf{C_n^b} = P_b \bar{A} A^{n-1} \mathbf{C_0} \tag{7}$$

where P_b is the selection matrix of $b = 1, 2, 3$. $\bar{A}$, A and $\mathbf{C_0}$ are extended subdivision matrices and original control vertices as shown in (2).

(6) and (7) illustrates the parametric form of base surface, here we introduce the offsetting surfaces. The offsetting surfaces are defined as the CCS surface working on difference vectors between interpolation points and their corresponding data points with 9 times faces of original mesh (Fig. 6). The offsetting surfaces on f_i have 9 CCS sub-faces (Fig. 8 right), we define them as $f_{i,1}$, $f_{i,2}$, ..., $f_{i,9}$, and they can be parameterized by using (6) and (7) with parametric values (u_1, v_1), (u_2, v_2), ..., (u_9, v_9).

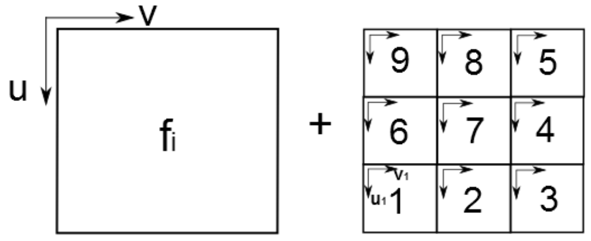

Figure 8: left is base surface, right is the offsetting surfaces

We define $\Delta S_m(u_m, v_m)$ as the parametric offsetting surface for $f_{i,m}$, $m = 1, ..., 9$.

$$\Delta S_m(u_m, v_m) = W^T(u_m, v_m) K^n D_b M \mathbf{C^b_{m,n}} \tag{8}$$
$$n(u_m, v_m) = min\{\lceil \log_{\frac{1}{2}} u_m \rceil, \lceil \log_{\frac{1}{2}} v_m \rceil\} + 1$$
$$b(u_m, v_m) = k, \ if \ (u_m, v_m) \in \Omega_{n,k}, \ k = 1, 2, 3$$

with

$$\mathbf{C^b_{m,n}} = P_b \bar{A} A^{n-1} \mathbf{C_{m,0}} \tag{9}$$

where $\mathbf{C_{m,0}}$ is the initial offsetting control mesh for $f_{i,m}$.

Such that, the offsetting surface $\Delta S(u, v)$ on f_i is the union of all 9 sub-surfaces with the same $\Omega - Partition$ as in (6), with

$$\Delta S(u, v) = \Delta S_1(u_1, v_1) \cup \Delta S_2(u_2, v_2) \cup ... \cup \Delta S_9(u_9, v_9)$$

Since the resulting limit surface is the sum of base surface and offsetting surfaces (illustrated in Fig. 8), we can define the resulting surface $\bar{S}(u, v)$ as,

$$\bar{S}(u, v) = S(u, v) + \Delta S(u, v) \tag{10}$$

In order to define (10), a reparametrization need to be done on $\Delta S_m(u_m, v_m)$. The mapping from parametric values of sub-faces $f_{i,m}$ to the f_i is defined by

$$(u_m, v_m) = (\phi(u), \phi(v)) \text{ , with}$$

$$m(u,v) = \begin{cases} 1, & \text{if } 3u \in (2,3] \text{ and } 3v \in [0,1] \\ 2, & \text{if } 3u \in (2,3] \text{ and } 3v \in (1,2] \\ 3, & \text{if } 3u \in (2,3] \text{ and } 3v \in (2,3] \\ 4, & \text{if } 3u \in (1,2] \text{ and } 3v \in (2,3] \\ 5, & \text{if } 3u \in [0,1] \text{ and } 3v \in (2,3] \\ 6, & \text{if } 3u \in (1,2] \text{ and } 3v \in [0,1] \\ 7, & \text{if } 3u \in (1,2] \text{ and } 3v \in (1,2] \\ 8, & \text{if } 3u \in [0,1] \text{ and } 3v \in (1,2] \\ 9, & \text{if } 3u \in [0,1] \text{ and } 3v \in [0,1] \end{cases}$$

and

$$\phi(t) = \begin{cases} 3t, & \text{if } 3t \in [0,1] \\ 3t-1, & \text{if } 3t \in (1,2] \\ 3t-2, & \text{if } 3t \in (2,3] \end{cases}$$

Since functions b, n and k in (6) and (8) take different parametric values as input, to combine (6) and (8) into (10), we define $\widetilde{b}$, $\widetilde{n}$ as mapping from b and n, we get

$$\Delta S(u, v) = W^T(\phi(u), \phi(v)) K^{\widetilde{n}} D_{\widetilde{b}} M \mathbf{C^{\widetilde{b}}_{m,\widetilde{n}}} \tag{11}$$

where

$$\widetilde{n}(u,v) = n(\phi(u), \phi(v))$$
$$\widetilde{b}(u,v) = b(\phi(u), \phi(v))$$

Given a data mesh M, in (10), the new interpolating surface $\bar{S}(u,v)$ for an arbitrary face f_i is calculated by adding an offsetting surface $\Delta S(u,v)$ to the base CCS surface $S(u,v)$. $\Delta S(u,v)$ is given in (11), while $S(u,v)$ is represented in (6). For an arbitrary (u,v), one can calculate the limit point of $\bar{S}(u,v)$ directly with (10), (6) and (11).

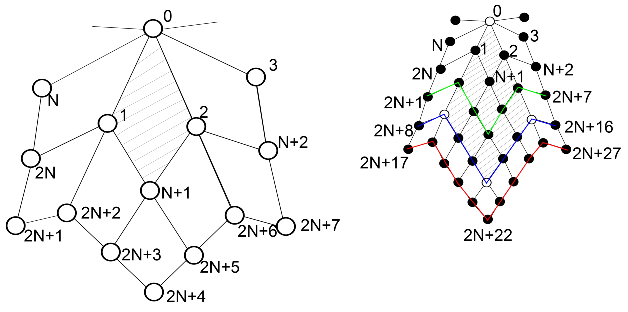

Figure 9: (a) base mesh (b) offsetting mesh

Since $\mathbf{C_0}$ in (7) is the original $2N+8$ control vertices of CCS on f_i, $S(u,v)$ can be explicitly computed. The value of offsetting surface $\Delta S(u,v)$ depends on the offsetting control meshes $\mathbf{C_{m,0}}$'s defined in (9). Given a f_i with valence N, we define the control points in base mesh M as $V_0, V_1, ..., V_{2N+7}$ (where V_0 is vertex point, $V_1, ..., V_N$ edge points, and $V_{N+1}, ..., V2N$ face points), and control points in offsetting mesh as $\Delta V_0, ..., \Delta V_{2N+27}$. The orderings are shown in Fig. 9, the additional vertices in offsetting mesh $\Delta V_{2N+8}, .., \Delta V_{2N+16}$ and $\Delta V_{2N+17}, .., \Delta V_{2N+27}$ (blue and red line in Fig. 9(b)) are defined with counter-clock ordering. With Ω-Partition (Fig. 8), we obtain the control meshes of 9 sub-faces $\mathbf{C_{m,0}}$, ..., $\mathbf{C_{m,9}}$, where $\mathbf{C_{m,9}}$ has $2N+8$ control vertices, all others have 16 control vertices.

Since each CCS limit point d_V of a control point V is the affine combination of this vertex point, edge points E_i's and face points F_i's (Fig 2),

$$d_V = \frac{N}{N+5}V + \frac{4}{N+5}\sum_{i=1}^{N} E_i/N + \frac{1}{N+5}\sum_{i=1}^{N} F_i/N, \tag{12}$$

in order to interpolate all data points in M, as shown in section 3, $\Delta S(u,v)$'s must interpolate the difference vectors ΔM between these data points and corresponding CCS limit points, thus these control meshes must satisfy

$$\begin{aligned} d_{\Delta V_0} &= V_0 - d_{V_0}, \quad & d_{\Delta V_{2N+9}} &= V_1 - d_{V_1} \\ d_{\Delta V_{2N+12}} &= V_{N+1} - d_{V_{N+1}}, \quad & d_{\Delta V_{2N+15}} &= V_2 - d_{V_2} \end{aligned} \tag{13}$$

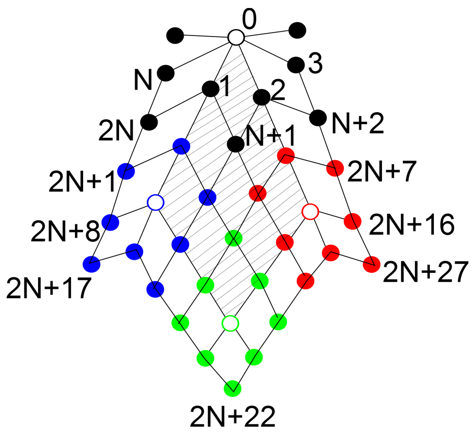

Figure 10: linear independence of interpolation offsetting data points

With (12), for data point $d_{\Delta V_0}$ at ΔV_0 (black circle in Fig. 9), we have

$$a_0 x_0 = d_{\Delta V_0}, \tag{14}$$

where a_0 is a row vector of size $2N+1$ with $[\frac{N}{N+5}, \frac{4}{(N+5)N}, ..., \frac{1}{(N+5)N}, ...]$, x_0 is a vector of of size $2N+1$ with $[\Delta V_0, \Delta V_1, ..., \Delta V_{N+1}, ...]$ (black circle/dots in Fig. 9). Similarly we can obtain the linear equations for $d_{\Delta V_{2N+9}}$, $d_{\Delta V_{2N+12}}$ and $d_{\Delta V_{2N+12}}$.

In Fig. 10, the relevant control points of each interpolation offsetting data points (circles) are marked with different colors, we see that the computation of each interpolation offsetting data point is independent from the computation of other interpolation offsetting data points in the mesh.

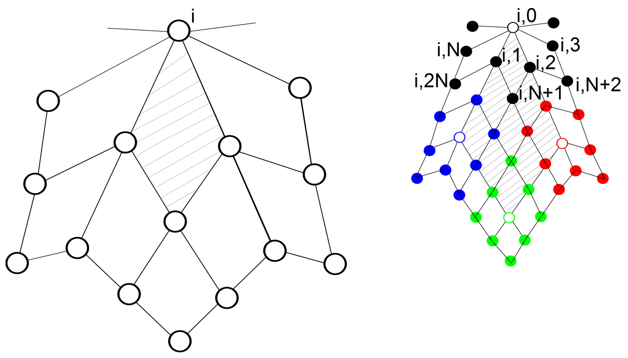

Figure 11: (a) base mesh (b) offsetting mesh

With (13) and (14), we can obtain a local linear system for each data point in M as shown in (4) and (5). Since earlier naming of vertices are based on face, here we rename the vertices for entire mesh. As shown in Fig. 11, for each non-boundary data point P_i of valence N in M we define $2N+1$ control points in offsetting control mesh ΔM as $P_{i,0}, P_{i,1}, ...P_{i,2N}$, we then obtain k (number of non-boundary points in M) linear equations, where each non-boundary data points are computed locally with (14),

$$a_i \bar{x}_i = \bar{d}_i, i \in [0, k-1] \tag{15}$$

where a_i is the coefficient row vector of size 2N+1 defined in (14), $\bar{x}_i$ is the vector of corresponding control points $P_{i,0}, ..., P_{i,2N}$ in ΔM, and $\bar{d}_i$ is the difference vector of P_i and its CCS limit point. To interpolate a data mesh M with k non-boundary data points, we need to construct an offsetting mesh ΔM of size $9k$, which satisfy the interpolation requirement set in (15), k local linear equations.

To solve the local linear system set by (14), we can have too many freedom for ΔM. Additional constraints need to be introduced. In this paper, we choose a simple solution to solve this local linear system, for each P_i in M, we choose corresponding control points $P_{i,,m}$ in ΔM as

$$P_{i,m} = \bar{d}_i, \ m = 0, ..., 2N. \tag{16}$$

With choice of control points of ΔM shown in (16), the new interpolating surface is obtained by stretching the original CCS limit surface on M, such that a smooth interpolation limit surface is obtained.

Above, we introduced the mathematical setup of our new interpolation scheme. (6) and (9) define original CCS limit surface, (11) and (16) define the offsetting surface, then (10) defines our new interpolation surface by adding original CCS limit surface with offsetting surface.

5 Behavior of New Interpolation Scheme

In this section, we discuss behavior of our new one-step interpolation scheme.

Our new scheme will generate 2 CCS meshes, one is the given base mesh M to interpolate, another is the offsetting mesh ΔM. If M has k non-boundary data points, then ΔM has 9k control points. In order to interpolate data mesh M, our one-step interpolation surface is obtained by adding limit surface of ΔM to the limit surface of M.

Since both M and ΔM are CCS control meshes, we can derive

Theorem 2. *Our new one-step interpolation surface is C^2 continuous everywhere except at extraordinary point of M, where it is C^1 continuous.*

Proof. With mesh structure defined in Fig. 9, the offsetting mesh ΔM has exactly the same number of extraordinary faces as M. Since M and ΔM are both CCS meshes, their CCS limit surfaces are C^2 continuous everywhere except at extraordinary points. At an arbitrary limit point (not extraordinary point) on the new interpolation surface, it is trivial to prove with (10) (by computing 1st and 2nd order derivatives) that it is C^2 continuous. At arbitrary extraordinary point of M, since both limit surfaces of M and ΔM are C^1 at extraordinary points, at $\bar{S}(0,0)$, the resulting surface must be also C^1 continuous. □

We notice from (10),(6) and(11) that our new scheme has the local support. This is in contrast with traditional interpolation schemes where local support is lost. In traditional interpolation schemes, if a data point is changed, then by solving a global linear system (3), all control points in x might be changed, such that the limit surface will change globally. This artifact prevents us from certain applications requiring matching surfaces between 2 3D objects in CAD/Computer Graphics, such as mold manufacturing, parts assembling. Our new scheme maintains the local

Data mesh		Running Time (seconds)	
Example	Vertices	CCS	1/3 scheme
Fig. 1	1,200,002	38.53	49.212
Fig. 11	596,423	17.812	23.618
Fig. 12	1,500,354	46.275	61.501

Table 1: Comparing running time of CCS and new 1/3 interpolation scheme for large data sets.

support of CCS, such that change one data point will not change the interpolation limit surface globally, instead it will change only 2 rings of surfaces surrounding that data point, the same local support as CCS.

Theorem 3. *The new interpolation surface has the same local support as its CCS base surface.*

Proof. Our new interpolation surface is obtained by parametrically adding two CCS surfaces, 1st part is original CCS base surface, 2nd part is the offsetting surface. The 1st part has local support. The control points of 2nd part is derived from difference vectors between control points and data points of 1st part shown in (15) and (16), such that the resulting new interpolating surface has the same local support as the original CCS limit surface. □

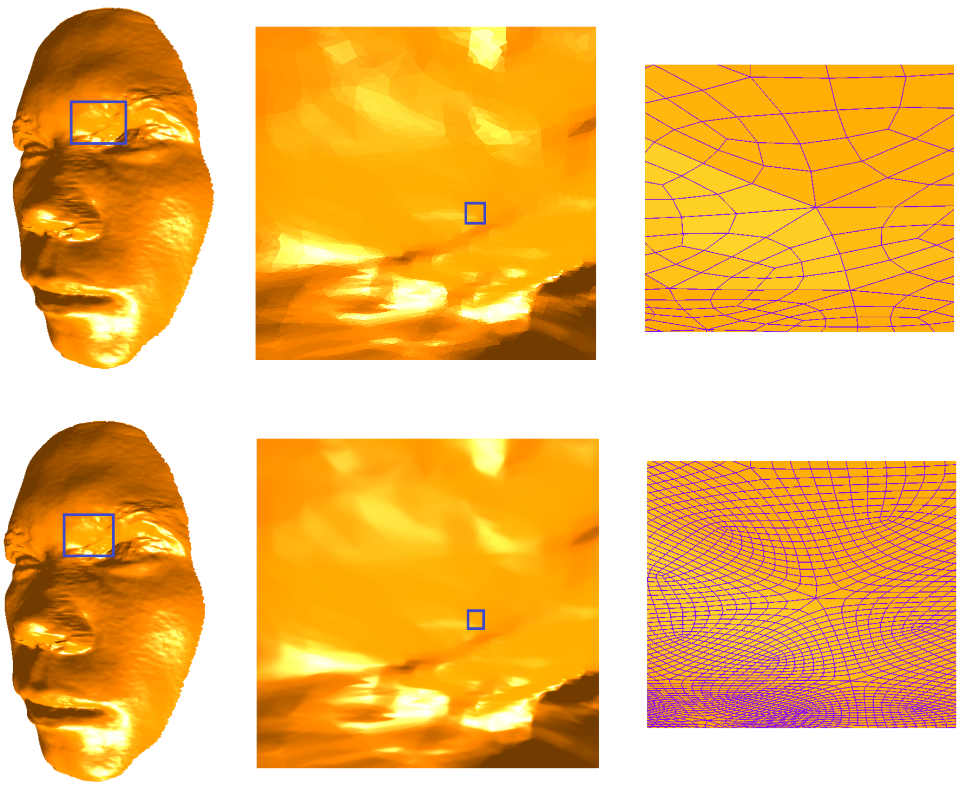

Figure 12: A face mask. Top row shows the views of original data mesh, from left to right: a). original surface; b). enlarged view of blue box in a); c) enlarged mesh view of blue box in b). Bottom row shows the generated interpolation limit surface with one-step scheme, and has the same sequence of enlarged views as in top row

In our new scheme, the CCS base surface part is global, but the offsetting surface part is local. By parametrically adding an offsetting surface which locally interpolating all difference vectors between data points and their CCS limit points to its CCS base surface on the given data mesh, the generated interpolating surface follows the global shape of original CCS base surface while local offsetting surfaces enforce the interpolation directly.

Implementation results in Fig. 3 show that the resulting interpolation surfaces generated by our new scheme is smooth and of high quality. No fairing is generally needed to resolve the undulation caused by solving global linear system of traditional schemes. Furthermore the new scheme works well for large data sets. Table 1 shows the comparison of running time for Fig. 1, Fig. 12 and Fig. 13 between CCS and our new interpolation scheme (machine spec: CPU intel i5-2430M, RAM 4GB). From table 1, one can conclude that our new scheme can handle millions of interpolating data points efficiently.

6 Conclusion

Traditional CCS interpolation schemes obtain an interpolation surface by solving a global linear system in (3), this will bring difficulties for the iterative scheme to handle large data set. Besides, the resulting interpolating surface does not have local property.

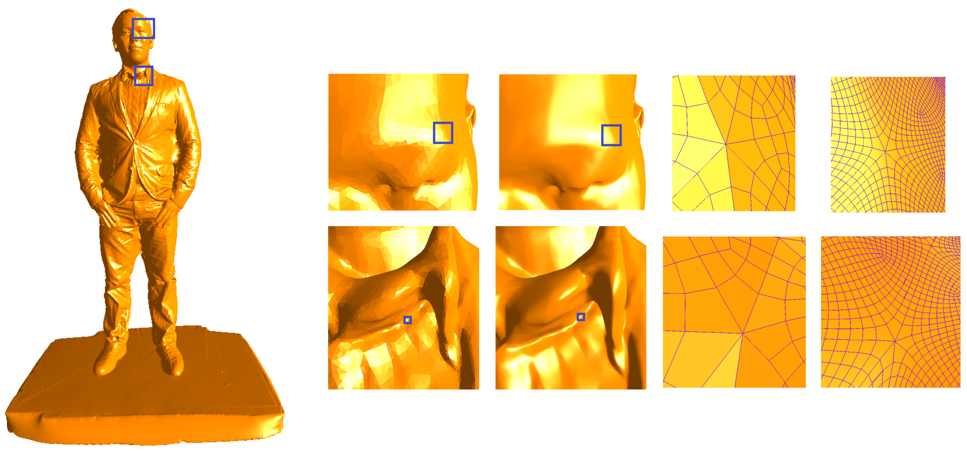

Figure 13: A statue. Top is our new interpolation limit surface. Bottom two rows shows enlarged views of two blue boxes in top image, from left to right. a). enlarged view of blue box in original surface with given data mesh; b) interpolation limit surface of a); c) enlarged mesh view of blue box in a); d) enlarged mesh view of interpolation limit surface of c).

In this paper, by combining two techniques: mesh decomposition and surface offsetting, we present a new interpolation scheme for Carmull-Clark subdivision surfaces that would not only be able to handle very large data sets (with millions of data points), but also allow the generated interpolating surface to have local property. Implementation result shows that a smooth and high quality interpolation surface can be generated by applying this new scheme, this is an important technique for applications with large data sets such as reverse engineering of scanned data sets and 3D printing.

Our next step is to do further research on offsetting surfaces, explore various control points selection on offsetting mesh and verify the impact of different selections.

References

[1] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[2] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355, 1978.

[3] Z. Chen, X. Luo, L. Tan, B. Ye, and J. Chen. Progressive interpolation based on catmull-clark subdivision surfaces. In *Computer Graphics Forum*, volume 27, pages 1823–1827. Wiley Online Library, 2008.

[4] F.-H. F. Cheng, F.-T. Fan, S.-H. Lai, C.-L. Huang, J.-X. Wang, and J.-H. Yong. Loop subdivision surface based progressive interpolation. *Journal of Computer Science and Technology*, 24(1):39–46, 2009.

[5] G. E. Farin. *Curves and surfaces for CAGD [electronic resource]: a practical guide.* Morgan Kaufmann, 2002.

[6] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, page 44. ACM, 1993.

[7] S. Lai and F. Cheng. Similarity based interpolation using Catmull–Clark subdivision surfaces. *The Visual Computer*, 22(9):865–873, 2006.

[8] S. Lai and F. Cheng. Fast mesh interpolation and mesh decomposition with applications. *International Journal of Image and Graphics*, 12(01), 2012.

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

[10] N. J. Lott and D. Pullin. Method for fairing b-spline surfaces. *COMP. AIDED DES.*, 20(10):597–604, 1988.

[11] J. Stam. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, page 404. ACM, 1998.

[12] R. Szeliski. Fast surface interpolation using hierarchical basis functions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(6):513–528, 1990.

[13] J. Wang and F. Cheng. Bezier Crust on Quad Subdivision Surface. In *Pacific Graphics Short Papers*, pages 29–34. The Eurographics Association, 2013.

[14] C. Zhang, P. Zhang, and F. F. Cheng. Fairing spline curves and surfaces by minimizing energy. *Computer-Aided Design*, 33(13):913–923, 2001.

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