Direct computation of a control vertex position on any subdivision level

Loic Barthe, Leif Kobbelt

Direct computation of a control vertex position on any subdivision level

Loïc Barthe and Leif Kobbelt

Computer Graphics Group, RWTH Aachen, Ahornstrasse 55, 52074 Aachen, Germany
{barthe,kobbelt}@cs.rwth-aachen.de,
WWW home page: http://www.rwth-graphics.de/

Abstract. In this paper, we present general closed form equations for directly computing the position of a vertex at different subdivision levels for both triangular and quadrilateral meshes. These results are obtained using simple computations and they lead to very useful applications, especially for adaptive subdivision. We illustrate our method on Loop's and Catmull-Clark's subdivision schemes.

1 Introduction

Since their introduction to Computer Graphics [1–3], subdivision surfaces have become a widely used surface representation in both animation and freeform shape modeling. A subdivision operator globally subdivides a coarse mesh into a finer one and after several subdivision iterations the meshes quickly converge to a smooth surface. Since the subdivision operator refines all faces of the mesh, the number of faces increases exponentially: For a classical diadic subdivision (like Loop's or Catmull-Clark's subdivision [4, 1]) the number of polygons is multiplied by 4 after *one* subdivision step, and hence by 4^k after k steps. A few steps are enough to generate a massive mesh that even the fastest hardware cannot handle interactively.

In order to overcome this disadvantage, adaptive subdivision [5, 6, 3] allows us to subdivide the mesh locally, only where some geometric detail is present. An accurate approximation of the surface is then provided while avoiding the generation of huge meshes (Fig. 1). Nowadays, adaptive subdivision is a standard technique used in applications like mesh editing, mesh simplification, and view-dependent rendering [7–12].

Since more steps of subdivision are performed in some parts of the mesh, vertices in the direct neighborhood of a central vertex can lie at different subdivision levels. The drawback is that when approximation schemes are used, vertices are displaced through subdivision, i.e. the same vertex has different positions when it is used to compute new vertices at different steps of subdivision. This requires an easy access to the position of a vertex at different subdivision levels and the limit position of every vertex has to be known in order to evaluate the current error. The application of *one* step of subdivision on a vertex requires

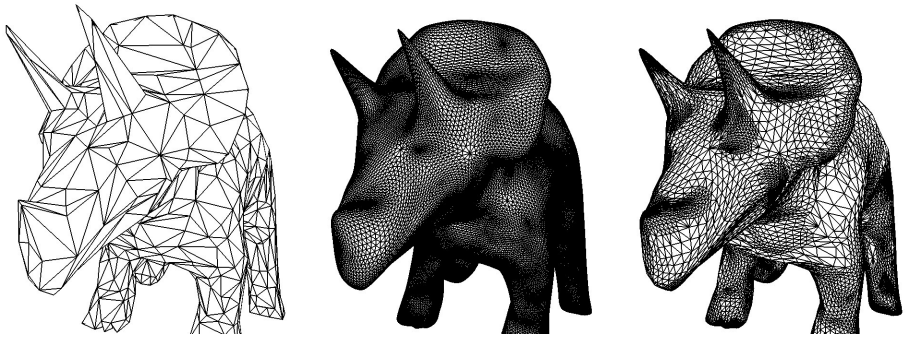

Fig. 1. *Three* steps of uniform Loop's subdivision performed on a $1.4K$ triangles triceratops model (left) yields a mesh composed of $90K$ triangles (center). An equivalent approximation of the limit surface is obtained with only $29K$ triangles when we use adaptive subdivision (right).

the knowledge of the position of its neighbors, and since vertices have always to be evaluated at different subdivision level, either expensive computations or large memory consumption is to be expected.

A simple method to avoid this phenomenon is rather to interpolate the control mesh so that the position of the vertices remains unchanged through subdivision. However standard stationary interpolatory subdivision schemes (like butterfly subdivision [13, 14]) are known to provide limit surfaces with artifacts and oscillations, and small support approximation schemes providing better quality limit surfaces are still preferable. Therefore, the fundamental requirements that we have to address are the computation of the position of the mesh vertices at the limit and at an arbitrary subdivision step.

The computation of the limit position of a vertex has already been studied [15–17]. Closed form solutions are derived using the eigendecomposition of the subdivision matrix S, where S is the operator which maps the vicinity of a central vertex into the same vicinity on the next refinement level. The limit position is then expressed as an affine combination of vertices of the control mesh, having its coefficients given by the dominant left eigenvector. Hence, our main contribution is the inexpensive computation of the vertex position at any intermediate step of subdivision. Indeed, if this evaluation is too expensive, it becomes preferable to simply store the different positions of the vertices after each subdivision step.

The size of the subdivision matrix S grows linearly with the valency of the central vertex, and different valencies yield different eigendecompositions. Therefore, complications are to be expected if we try to derive closed form equations for the different vertex positions from the standard formulation of the subdivision matrix (as done in [16]). To overcome this disadvantage, a known solution is to use the Fourier transform of the matrix S [15, 17]. However as shown by Loop [4] for triangular meshes when he studied the convergence of his scheme,

for all valencies, the subdivision matrix S can be represented by a 2×2 matrix by exploiting its block circulant structure. This procedure avoids the computation of the Fourier transform and it has been later exploited by Kobbelt [18] to provide his $\sqrt{3}$ scheme with simple closed form equations to evaluate the positions of a vertex at both the limit and after m subdivision steps.

In this paper, we present more general closed form equations for computing the position of a vertex at any intermediate subdivision level. After initialization, the evaluation of different vertex positions do not require any access to the neighbors, which allows us to avoid memory consumption while providing computationally inexpensive solutions.

The first part is the extension of this approach to its general form for triangular meshes. It illustrates how this simple computation process leads us to elegant closed form equations where the position of a vertex after m subdivision steps is defined as the interpolation between its initial position and its position in the limit. The general solution is then applied to Loop's scheme [4] for illustration purposes.

The second part represents a more important contribution. We apply the same procedure on quadrilateral meshes and we show how the different structure of the subdivision matrix leads us to slightly more complicated equations that have to be handled with a mathematical software. Also in this case, after initialization, no neighbor information is required and we illustrate on *two* versions of Catmull-Clark's subdivision [1, 2, 19] that it still provides practical useful solutions.

2 Triangular lattice

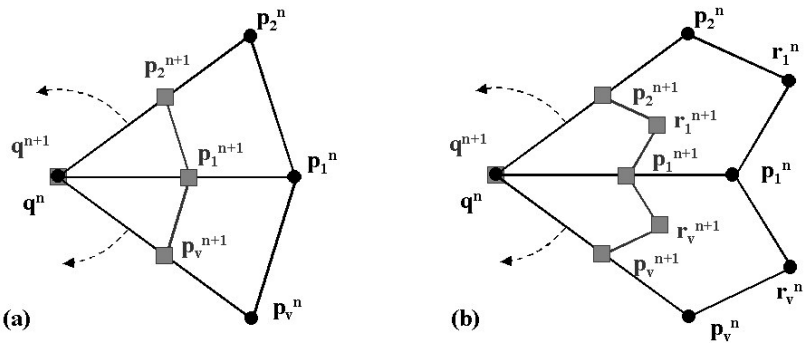

Fig. 2. One step of standard diadic subdivision performed on the central vertex q^n and its 1-ring neighborhood: (a) on a triangular mesh, and (b) on a rectangular mesh. The superscript index indicates the subdivision level while the lowerscript index is the vertex number.

We begin with the study of triangular lattices. Since we only consider small support subdivision schemes, the vicinity of the central vertex is restricted to its "1-ring" neighborhood (as illustrated in Figure 2a). Each row of the subdivision matrix is a smoothing (or subdivision) rule which computes a new vertex as an affine combination of vertices of the original mesh, and once all the rotational symmetries are exploited, the subdivision of this set of vertices is written in the well known form:

$$\begin{bmatrix} q^{n+1} \\ \hline p_1^{n+1} \\ p_2^{n+1} \\ \vdots \\ p_v^{n+1} \end{bmatrix} = \left[\begin{array}{c|cccc} a_0 & a_1 & \cdots & \cdots & a_v \\ \hline b_0 & b_1 & b_2 & \cdots & b_v \\ b_0 & b_v & b_1 & b_2 & \cdots \\ \vdots & & & \ddots & \\ b_0 & b_2 & \cdots & b_v & b_1 \end{array}\right] \begin{bmatrix} q^n \\ \hline p_1^n \\ p_2^n \\ \vdots \\ p_v^n \end{bmatrix}, \tag{1}$$

with

$$\sum_{i=0}^{v} a_i = 1, \sum_{i=0}^{v} b_i = 1, \text{ and hence } \sum_{i=1}^{v} a_i = 1 - a_0, \sum_{i=1}^{v} b_i = 1 - b_0. \tag{2}$$

In equation (1), q^n is the central vertex at the n^{th} step of subdivision, v is its valency and vertices p_j^n ($j = \{1, \ldots, v\}$) are its direct neighbors. Vertex q^{n+1} is the new position of vertex q^n after *one* subdivision step, vertices p_j^{n+1} are new vertices which are the direct neighbors of vertex q^{n+1}. The subdivision matrix S is the square matrix in Equation (1).
Let us denote P^n as:

$$P^n = \frac{1}{v} \sum_{j=1}^{v} p_i^n.$$

Equations (1) and (2) allow us to express the sum of the new 1-ring vertices in terms of the old vertices as:

$$\begin{aligned} P^{n+1} &= \frac{1}{v} \sum_{j=1}^{v} p_j^{n+1} = \frac{1}{v} \sum_{j=1}^{v} \left[b_0 q^n + \sum_{i=1}^{v} b_{i-j+1} p_i^n \right] \\ &= b_0 q^n + (1 - b_0) P^n. \end{aligned}$$

Hence, relation (1) can be simplified in a form which simply involves a 2×2 matrix in the computation (independent from valencies v):

$$\begin{bmatrix} q^{n+1} \\ P^{n+1} \end{bmatrix} = \begin{bmatrix} a_0 & 1 - a_0 \\ b_0 & 1 - b_0 \end{bmatrix} \begin{bmatrix} q^n \\ P^n \end{bmatrix}.$$

When m steps of subdivision are performed, the subdivision matrix S is applied to the power m. This multi-step rule can also be written as an affine combination of the vertex q^n and the average of its neighbors P^n, and once the 2×2 matrix is expressed in terms of its eigendecomposition we rewrite the subdivision rules as:

$$\begin{bmatrix} q^{n+m} \\ P^{n+m} \end{bmatrix} = \frac{b_0}{1-a_0+b_0} \begin{bmatrix} 1 & \frac{1-a_0}{b_0} \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & (a_0-b_0)^m \end{bmatrix} \begin{bmatrix} 1 & \frac{1-a_0}{b_0} \\ 1 & -1 \end{bmatrix} \begin{bmatrix} q^n \\ P^n \end{bmatrix}. \quad (3)$$

In Equation (3), when m tends to infinity, $(a_0 - b_0)^m$ tends to 0, because $0 < b_0 < a_0 < 1$ (variational diminishing property), hence, the limit position q^∞ of vertex q^n is directly computed using the following closed form equation:

$$q^\infty = \beta_\infty q^n + (1-\beta_\infty)P^n \quad \text{with } \beta_\infty = \frac{b_0}{1-a_0+b_0} \quad (4)$$

From equations (3) and (4), we remove the neighbor information and we derive the expression of the position of vertex q^n after m subdivision steps in terms of its initial position q^n and its limit position q^∞. This leads us to the expected computation based only on the own vertex information:

$$\boxed{q^{n+m} = \mu(m)q^n + (1-\mu(m))\, q^\infty \quad \text{with } \mu(m) = (a_0-b_0)^m} \quad (5)$$

The closed form equations (4) and (5) applied to Kobbelt's $\sqrt{3}$ subdivision scheme are already given in [18] hence we present as an example their application in the case of **Loop's subdivision** [4]:

$$a_0 = 1-\alpha_v \quad \text{with } \alpha_v = \tfrac{5}{8} - \left(\tfrac{3}{8} + \tfrac{1}{4}\cos\left(\tfrac{2\pi}{v}\right)\right)^2 \quad \text{and} \quad b_0 = \tfrac{3}{8}$$

$$\boxed{\beta_\infty = \frac{1}{1+\frac{8}{3}\alpha_v} \quad \text{and} \quad \mu(m) = \left(\frac{5}{8} - \alpha_v\right)^m}$$

3 Quadrilateral lattice

The case of a quadrilateral lattice is more complicated because the 1-ring neighborhood is composed of *two* different rotationally symmetric sets of vertices ($\{p_j^n\}$ and $\{r_j^n\}$, $j = \{1, \ldots, v\}$)(see Figure 2b). Therefore, the subdivision of a vertex q^n and its 1-ring neigbors is defined by the following equation:

$$\begin{bmatrix} q^{n+1} \\ \hline p_1^{n+1} \\ \vdots \\ p_v^{n+1} \\ \hline r_1^{n+1} \\ \vdots \\ r_v^{n+1} \end{bmatrix} = \left[\begin{array}{c|ccc|ccc} a_0 & a_1 & \cdots & a_v & a_{v+1} & \cdots & a_{2v} \\ \hline b_0 & b_1 & \cdots & b_v & b_{v+1} & \cdots & b_{2v} \\ \vdots & & \ddots & & & \ddots & \\ b_0 & b_2 & \cdots & b_1 & b_{v+2} & \cdots & b_{v+1} \\ \hline c_0 & c_1 & \cdots & c_v & c_{v+1} & \cdots & c_{2v} \\ \vdots & & \ddots & & & \ddots & \\ c_0 & c_2 & \cdots & c_1 & c_{v+2} & \cdots & c_{v+1} \end{array}\right] \begin{bmatrix} q^n \\ \hline p_1^n \\ \vdots \\ p_v^n \\ \hline r_1^n \\ \vdots \\ r_v^n \end{bmatrix}.$$

The subdivision matrix S has *four* circulant blocks and in a similar manner than in Section 2 we express S as a 3×3 matrix (independant from valencies v):

$$\begin{bmatrix} q^{n+1} \\ P^{n+1} \\ R^{n+1} \end{bmatrix} = \begin{bmatrix} a_0 & a_s & 1-a_0-a_s \\ b_0 & b_s & 1-b_0-b_s \\ c_0 & c_s & 1-c_0-c_s \end{bmatrix} \begin{bmatrix} q^n \\ P^n \\ R^n \end{bmatrix}, \tag{6}$$

$$\text{where } R^n = \frac{1}{v}\sum_{j=1}^{v} r_j^n \text{ and } x_s = \sum_{i=1}^{v} x_i \text{ for } x \in \{a, b, c\}.$$

Following the procedure presented in Section 2 we use Matlab code to compute the different closed form equations for β_∞, γ_∞, $\mu(m)$ and $\nu(m)$ in order to evaluate the different positions of a vertex q^{n+m}.

From Equation (6) and its eigendecomposition we obtain the expressions:

$$q^{n+1} = a_0 q^n + a_s P^n + (1 - a_0 - a_s) R^n \tag{7}$$

$$q^{m+n} = \beta(m) q^n + \gamma(m) P^n + (1 - \beta(m) - \gamma(m)) R^n \tag{8}$$

$$q^{\infty} = \beta_\infty q^n + \gamma_\infty P^n + (1 - \beta_\infty - \gamma_\infty) R^n \tag{9}$$

Using Equations (7) and (9), we remove the neighborhood information from Equation (8) as follows:
Let

$$\begin{bmatrix} q^n \\ q^{n+1} \\ q^{\infty} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ a_0 & a_s & 1-a_0-a_s \\ \beta_\infty & \gamma_\infty & 1-\beta_\infty-\gamma_\infty \end{bmatrix} \begin{bmatrix} q^n \\ P^n \\ R^n \end{bmatrix} = T \begin{bmatrix} q^n \\ P^n \\ R^n \end{bmatrix},$$

then q^{n+m} is expressed as

$$q^{m+n} = \begin{bmatrix} \beta(m) & \gamma(m) & 1-\beta(m)-\gamma(m) \end{bmatrix} T^{-1} \begin{bmatrix} q^n \\ q^{n+1} \\ q^{\infty} \end{bmatrix},$$

which leads us to the final closed form equation:

$$\boxed{q^{n+m} = \mu(m)q^n + \nu(m)q^{n+1} + (1 - \mu(m) - \nu(m))\, q^\infty}$$

In actual applications, the equations are greatly simplified and the formulas provided by the code lead us to practical solutions. These is illustrated for *two* versions of Catmull-Clark's subdivision.

Standard Catmull-Clark's subdivision [19]: (Table 1)

$$a_0 = 1 - \frac{7}{4v}, \quad a_s = \frac{6}{4v}, \quad b_0 = \frac{3}{8}, \quad b_s = \frac{1}{2}, \quad c_0 = \frac{1}{4}, \quad c_s = \frac{1}{2}$$

Table 1. Coefficients to compute the positions q^∞ and q^{n+m} of a vertex q^n of the mesh when it is subdivided using the standard Catmull-Clark's subdivision.

Valency	β_∞	γ_∞	$\mu(m)$	$\nu(m)$
3	$\frac{3}{8}$	$\frac{1}{2}$	0	$6\left(\frac{1}{6}\right)^m$
4	$\frac{4}{9}$	$\frac{4}{9}$	$\frac{-1}{3}\left(\frac{1}{4}\right)^m + \frac{4}{3}\left(\frac{1}{16}\right)^m$	$\frac{16}{3}\left(\left(\frac{1}{4}\right)^m - \left(\frac{1}{16}\right)^m\right)$
5	$\frac{1}{2}$	$\frac{2}{5}$	$-0.3165(0.3225)^m + 1.3165(0.0775)^m$	$4.0825((0.3225)^m - (0.0775)^m)$
6	$\frac{6}{11}$	$\frac{4}{11}$	$\frac{-2}{7}\left(\frac{3}{8}\right)^m + \frac{9}{7}\left(\frac{1}{12}\right)^m$	$\frac{24}{7}\left(\left(\frac{3}{8}\right)^m - \left(\frac{1}{12}\right)^m\right)$

Bilinear subdivision plus averaging Catmull-Clark's subdivision [19]: In this case, the subdivision rules are independant of the vertex valency.

$$a_0 = \frac{9}{16}, \quad a_s = \frac{3}{8}, \quad b_0 = \frac{3}{8}, \quad b_s = \frac{1}{2}, \quad c_0 = \frac{1}{4}, \quad c_s = \frac{1}{2}$$

$$\boxed{\begin{array}{c} \forall v: \quad \beta_\infty = \gamma_\infty = \dfrac{4}{9} \\ \mu(m) = \dfrac{-1}{3}\left(\dfrac{1}{4}\right)^m + \dfrac{4}{3}\left(\dfrac{1}{16}\right)^m, \quad \nu(m) = \dfrac{16}{3}\left(\left(\dfrac{1}{4}\right)^m - \left(\dfrac{1}{16}\right)^m\right). \end{array}}$$

4 Conclusion

We have presented a general method for directly computing the position of a vertex on any subdivision level without accessing neighbor information, and this, in the case of triangular and quadrilateral lattices. These results can be

directly applied to any subdivision scheme whose mask does not exceed the 1-ring neighborhood, as illustrated on Loop's and Catmull-Clark's subdivision. If the scheme's mask exceed *one* ring, the procedure for quad-meshes can be applied to derive closed form equations with more terms.

As explained in Section 1, the direct application to adaptive subdivision makes these closed form equations very useful for practical applications and they can be used to provide different approximation schemes with adaptive subdivision as a standard operator for meshes [20].

Acknowledgments: The first author has partially been funded by the E.U. through the MINGLE project. Contract HPRN-CT-1999-00117

References

1. Catmull, E, Clark, J.: Recursively generated B-spline surfaces on arbitrary topological meshes. Computer Aided Design **10**,6, (1978) 350–355
2. Doo, D., Sabin, M.A.: Analysis of the behaviour of recursive subdivision surfaces near extraordinary points. Computer Aided Design **10**,6, (1978) 356–360
3. Zorin, D., Schröder, P.: Subdivision for modeling and animation. SIGGRAPH 2000 course notes, (2000)
4. Loop, C.: Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, (1987)
5. Vasilescu, M., Terzopoulos, D.: Adaptive meshes and shells: Irregular triangulation, discontinuities and hierarchical subdivision. Proceedings of Computer Vision and Pattern Recognition, (1992) 829–832
6. Verfürth, R.: A review of a posteriori error estimation and adaptive mesh refinement techniques. Wiley-Teubner, (1996)
7. Zorin, D., Schröder, P., Sweldens, W.: Interactive multiresolution mesh editing. Proceedings of SIGGRAPH 1997, ACM, (1997) 259–268
8. Xu, Z., Kondo, K.: Local subdivision process with Doo-Sabin subdivision surfaces. Proceedings of Shape Modeling International, (2002) 7–12
9. Lee, A., Moreton, H., Hoppe, H.: Displaced subdivision surfaces. Proceedings of SIGGRAPH 2000, ACM, (2000) 85–94
10. Hoppe, H.: View-dependent refinement of progressive meshes. Proceedings of SIGGRAPH 1997, ACM, (1997) 189–198
11. Kamen, Y., Shirman, L.: Triangle Rendering Using Adaptive Subdivision. IEEE Computer Graphics and Applications **18**,2, (1998) 356–360
12. Hoppe, H.: Smooth view-dependent level-of-detail control and its application in terrain rendering. IEEE Visualization, (1998) 35–42
13. Dyn, N., Levin, D., Gregory, J.: A butterfly subdivision scheme for surface interpolation with tension control. ACM Transaction on Graphics, **9**,2, (1990) 160–169
14. Zorin, D., Schröder, P., Sweldens, W.: Interpolating subdivision for meshes with arbitrary topology. Proceedings of SIGGRAPH 1997, ACM, (1996) 189–192
15. Halstead, M., Kass, M., DeRose, T.: Efficient, fair interpolation using Catmull-Clark surfaces. Proceedings of SIGGRAPH 1993, ACM, (1993) 35–43
16. Hoppe, H., DeRose, T., Duchamp, T., Halstead, M.: Piecewise smooth surfaces reconstruction. Proceedings of SIGGRAPH 1994, ACM, (1994) 295–302
17. Stam, J.: Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. Proceedings of SIGGRAPH 1998, ACM, (1998) 395–404
18. Kobbelt, L.: $\sqrt{3}$-subdivision. Proceedings of SIGGRAPH 2000, ACM, (2000) 103–112
19. Warren, J., Weimer, H.: Subdivision methods for geometric design: a constructive approach. San Fransisco: Morgan Kaufman, (2002) 209–212
20. OpenMesh subdivision tool: http://www.openmesh.org/