Interpolating Nets Of Curves By Smooth Subdivision Surfaces

Adi Levin*

Tel Aviv University

Abstract

A subdivision algorithm is presented for the computation and representation of a smooth surface of arbitrary topological type interpolating a given net of smooth curves. The algorithm belongs to a new class of subdivision schemes called *combined subdivision schemes*. These schemes can exactly interpolate a net of curves given in any parametric representation. The surfaces generated by our algorithm are G^2 except at a finite number of points, where the surface is G^1 and has bounded curvature. The algorithm is simple and easy to implement, and is based on a variant of the famous Catmull-Clark subdivision scheme.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve,surface, solid and object modeling.

Additional Keywords: Subdivision, Interpolation, Combined Subdivision schemes, Net of curves.

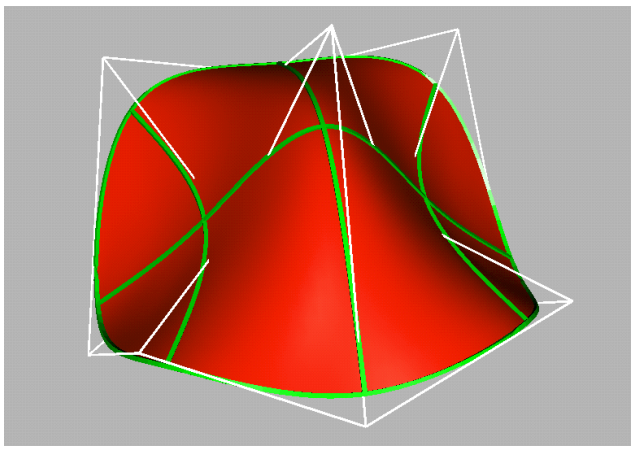

Figure 1: Interpolation of a net of curves

1 INTRODUCTION

Subdivision schemes provide efficient algorithms for the design, representation and processing of smooth surfaces of arbitrary topological type. Their simplicity and their multiresolution structure make them attractive for applications in 3D surface modeling, and in computer graphics [2, 4, 5, 6, 11, 18].

A common task in surface modeling is that of interpolating a given net of smooth curves by a smooth surface. A typical solution, using either subdivision surfaces or NURBS surfaces (or other kinds of spline surfaces), is based on establishing the connection between parts of the control net which defines the surface, and certain curves on the surface. For example, the boundary curves of NURBS surfaces are NURBS curves whose control polygon is the boundary polygon of the NURBS surface control net. Hence, curve interpolation conditions are translated into conditions on the control net. Fairing techniques [5, 15, 17] can be used to calculate a control net satisfying those conditions. Using subdivision surfaces, this can be carried out, in general, for given nets of arbitrary topology (see [12, 13]).

However, the curves that can be interpolated using that approach are restricted by the representation chosen for the surface. NURBS surfaces are suitable for interpolating NURBS curves; Doo-Sabin surfaces can interpolate quadratic B-spline curves [12, 13]; Other kinds of subdivision surfaces can be shown to interpolate specific kinds of subdivision curves. Furthermore, interpolation of curves that have small features requires a large control net, making the fairing process slower and more complicated.

This paper presents a new subdivision scheme specially designed for the task of interpolating nets of curves. This scheme falls into the category of *combined subdivision schemes* [7, 8, 10], where the underlying surface is represented not only by a control net, but also by given parametric curves (or in general, given interpolation conditions or boundary conditions). The scheme repeatedly applies a subdivision operator to the control net, which becomes more and more dense. In the limit, the vertices of the control net converge to a smooth surface. Point-wise evaluations of the given curves participate in every iteration of the subdivision, and the limit surface interpolates the given curves, regardless of their representation.

Figure 1 illustrates a surface generated by our algorithm. The surface is defined by an initial control net that consists of 11 vertices, and by a net of intersecting curves, shown in green. The edges of the control net are shown as white lines.

The combined subdivision scheme presented in this paper is based on the famous Catmull-Clark subdivision scheme. Our algorithm applies Catmull-Clark's scheme almost everywhere on the control net. The given curves affect the control net only locally, at parts of the control net that are near the given curves.

The motivation behind the specific subdivision rules, and the smoothness analysis of the scheme are presented in [9]. In the following sections, we describe Catmull-Clark's scheme, and we present the details of our scheme.

2 CATMULL-CLARK'S SCHEME

Camull Clark's subdivision scheme is defined over closed nets of arbitrary topology, as an extension of the tensor product bi-cubic B-spline subdivision scheme (see [1, 3]). Variants of the original scheme were analyzed by Ball and Storry [16]. Our algorithm em-

*adilev@math.tau.ac.il, http://www.math.tau.ac.il/~adilev

ploys a variant of Catmull-Clark's scheme due to Sabin [14], which generates limit surfaces that are G^2 everywhere except at a finite number of irregular points. In the neighborhood of those points the surface curvature is bounded. The irregular points come from vertices of the original control net that have valency other than 4, and from faces of the original control net that are not quadrilateral.

A net $N = (V, E)$ consists of a set of vertices V and the topological information of the net E, in terms of edges and faces. A net is closed when each edge is shared by exactly two faces.

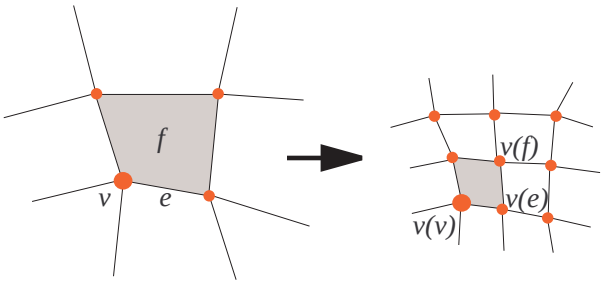

Figure 2: Catmull-Clark's scheme.

The vertices V' of the new net $N' = (V', E')$ are calculated by applying the following rules on N (see figure 2):

1. For each old face f, make a new face-vertex $v(f)$ as the weighted average of the old vertices of f, with weights W_n that depend on the valency n of each vertex.

2. For each old edge e, make a new edge-vertex $v(e)$ as the weighted average of the old vertices of e and the new face vertices associated with the two faces originally sharing e. The weights W_n (which are the same as the weights used in rule 1) depend on the valency n of each vertex.

3. For each old vertex v, make a new vertex-vertex $v(v)$ at the point given by the following linear combination, whose coefficients $\alpha_n, \beta_n, \gamma_n$ depend on the valency n of v:

   $\alpha_n \cdot$ (the centroid of the new edge vertices of the edges meeting at v) + $\beta_n \cdot$ (the centroid of the new face vertices of the faces sharing those edges) + $\gamma_n \cdot v$.

The topology E' of the new net is calculated by the following rule:

> For each old face f and for each vertex v of f, make a new quadrilateral face whose edges join $v(f)$ and $v(v)$ to the edge vertices of the edges of f sharing v (see figure 2).

The formulas for the weights $\alpha_n, \beta_n, \gamma_n$ and W_n are given in the appendix.

3 THE CONTROL NET

Our subdivision algorithm is defined both on closed nets and on open nets. In the case of open nets, we make a distinction between *boundary vertices* and *internal vertices* (and between *boundary edges* and *internal edges*). The control net that is given as input to our scheme consists of vertices, edges, faces and given smooth curves. We assume that these are C^2 parametric curves. An edge which is associated with a segment of a curve, is called a *c-edge*. Both of its vertices are called *c-vertices*. All the other edges and vertices are *ordinary vertices* and *ordinary edges*.

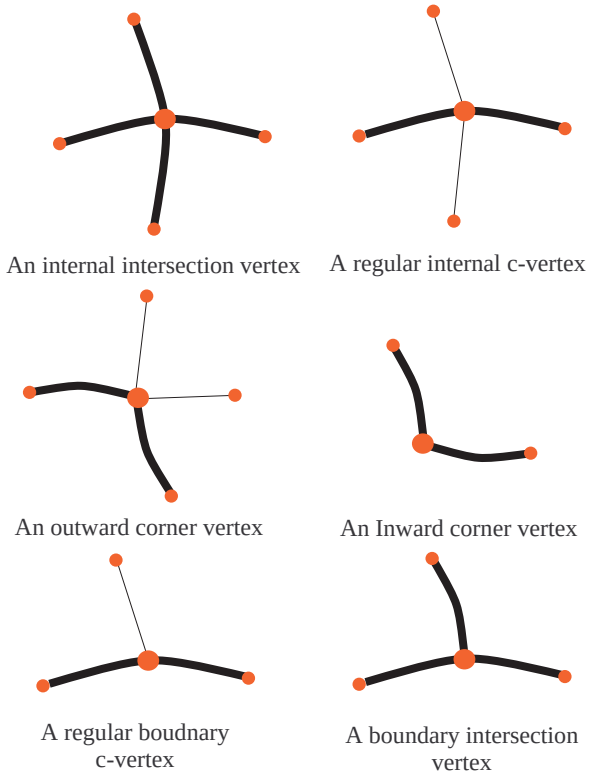

Figure 3: The different kinds of c-vertices. C-edges are marked by bold curved lines. Usual edges are shown as thin lines.

In case two c-edges that share a c-vertex are associated with two different curves, the c-vertex is associated with two curves, and we call it an *intersection vertex*. Every *c-vertex* is thus associated with a parameter value on a curve, while *intersection vertices* are associated with two curves and two different parameter values. In case of intersection vertices, we require that the two curves intersect at those parameter values.

Every *c-edge* contains a pointer to a curve c, and to a segment on that curve designated by a parameter interval $[u_0, u_1]$. The vertices of that edge are associated with the points $c(u_0)$ and $c(u_1)$ respectively. We require that in the original control net, the parameter intervals be all of constant length for all the c-edges associated with a single curve c, namely $|u_1 - u_0| = \text{const}$. In order to fulfill this requirement, the c-vertices along a curve c can be chosen to be evenly spaced with respect to the parameterization of the curve c, or the curve c can be reparameterized appropriately such that the c-vertices of c are evenly spaced with respect to the new parameterization.

The restrictions on the control net are that every boundary edge is a c-edge (i.e. the given net of curves contains all the boundary curves of the surface), and that we allow only the following types of *c-vertices* to exist in the net (see figure 3):

A regular internal c-vertex A c-vertex with four edges emanating from it: Two c-edges that are associated with the same curve, and two ordinary edges from opposite sides of the curve.

A regular boundary c-vertex A c-vertex with 3 edges emanating from it: Two boundary edges that are associated with the same curve, and one other ordinary internal edge.

An internal intersection vertex A c-vertex with 4 edges emanating from it: Two c-edges that are associated with the same curve, and two other c-edges that are associated with a second curve, from opposite sides of the first curve.

A boundary intersection vertex A c-vertex with 3 edges emanating from it: Two c-edges that are associated with the same curve, and another c-edge associated with a different curve.

An inward corner vertex A c-vertex with 2 c-edges emanating from it, each associated with a different curve.

An outward corner vertex A c-vertex with 4 edges emanating from it: Two consequent c-edges that are associated with two different curves and two ordinary edges.

In particular, we do not handle more than two curves intersecting at one point.

In our algorithm, there is an essential difference between c-vertices and ordinary vertices: While the location $p(v)$ of ordinary vertices of the original control net is determined by the designer, the location of c-vertices is calculated in a preprocessing stage of the algorithm (the exact procedure is described in §4).

Every c-vertex v which is associated with a parameter value u on the curve c, has associated with it a three-dimensional vector $d(v)$, which determines the second partial derivative of the limit surface at the point $c(u)$ in the cross-curve direction (The differentiation is made with respect to a local parameterization that is induced by the subdivision process. The cross-curve direction at a c-vertex v is the limit direction of the ordinary edge emanating for v). We call the value $d(v)$ the *cross-curve second derivative* associated with the vertex v.

Every intersection vertex v has associated with it two three-dimensional vectors $d_1(v), d_2(v)$ that correspond to the two curves c_1, c_2 that are associated with v. At the intersection between two curves, the surface second derivatives in the two curve directions are determined by the curves, therefore the user does not have control over the cross-curve second derivatives there. Their initialization procedure is described below.

For c-vertices that are not intersection vertices, the vectors $d(v)$ in the initial control net are determined by the designer and they affect the shape of the limit surface. Several ways of initializing the values $d(v)$ are discussed in §5.

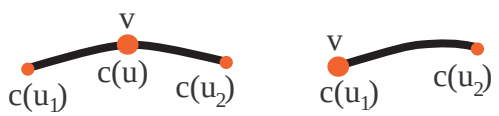

Figure 4: For each c-vertex v that is associated with a curve c we define the second difference $\Delta^2 c(v)$.

Throughout the scheme we apply second difference operators to the given curves. Let v denote a c-vertex associated with a curve c. We define the *second difference of c at v*, denoted by $\Delta^2 c(v)$ as follows (see figure 4): If v is associated with the end of the curve c, then there is a single c-edge emanating from v that is associated with the parameter interval $[u_1, u_2]$ on c. In this case

$$\Delta^2 c(v) = 4c(u_1) - 8c\left(\frac{u_1+u_2}{2}\right) + 4c(u_2).$$

In case there are two c-edges emanating from v that are associated with the parameter intervals $[u_1, u], [u, u_2]$ on c, we define

$$\Delta^2 c(v) = c(u_1) - 2c(u) + c(u_2).$$

The values $d_1(v), d_2(v)$ at the intersection vertex v which is associated with two curves c_1 and c_2, are initialized by

$$\begin{aligned} d_1(v) &= \Delta^2 c_1(v) \\ d_2(v) &= \Delta^2 c_2(v). \end{aligned} \quad (1)$$

We say that $d_1(v)$ is the cross-curve second derivative associated with v with respect to the curve c_2. Similarly, $d_2(v)$ is the cross-curve second derivative associated with v with respect to the curve c_1.

4 THE COMBINED SCHEME

In the preprocessing stage of our algorithm, we calculate $p(v)$ for every c-vertex of the original control net, according to the following rules: In case v is an intersection vertex which is associated with the point $c(u)$, its location is given by

$$p(v) = c(u) - \frac{d_1(v) + d_2(v)}{6}. \quad (2)$$

In case v is not an intersection vertex, its location is given by

$$p(v) = c(u) - \frac{\Delta^2 c(v) + d(v)}{6}. \quad (3)$$

From (2) and (3) it is clear why the c-vertices do not necessarily lie on the given curves. Notice, for example, in figure 8 how the boundary vertices of the original control net are 'pushed away' from the given boundary curve, due to the term $\Delta^2 c(v)$ in (3).

Each iteration of the subdivision algorithm consists of the following steps: First, Catmull-Clark's scheme as described in §2 is used to calculate the new ordinary vertices. Next, the new c-vertices are calculated (this includes all the boundary vertices). Finally, we perform local 'corrections' on new ordinary vertices that are neighbors of c-vertices.

4.1 Calculation Of Ordinary Vertices

Step 1 of the combined scheme creates the new control net topology, and calculates all the new ordinary vertices, by applying Catmull-Clark's scheme. Since Catmull-Clark's scheme was designed for closed nets, we adapt it a little bit near the surface boundaries, by considering the boundary vertices to have valency 4 when calculating new ordinary vertices that are affected by the boundary vertices.

4.2 Calculation Of C-Vertices

In step 2, the data associated with the new c-vertices is calculated, by the following procedure:

Let e denote a c-edge on the old control net, which corresponds to the parameter interval $[u_0, u_1]$ of the curve c. Let v_0, v_1 denote the vertices of e. We associate the vertex $v(e)$ with $c\left(\frac{u_0+u_1}{2}\right)$, and we calculate the new cross-curve second derivative for $v(e)$ by the following simple rule:

$$d(v(e)) = \frac{d(v_0) + d(v_1)}{8}. \quad (4)$$

In case v_0 or v_1 are intersection vertices (and therefore, contain two cross-curve second derivative vectors d_1 and d_2), the one taken in (4) should be the cross-curve second derivative with respect to the curve c.

Let v denote a c-vertex on the old control net. We associate $v(v)$ with the same curve and the same parameter value on that curve,

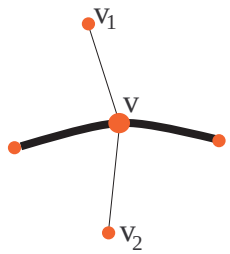

Figure 5: Local corrections near a regular internal c-vertex

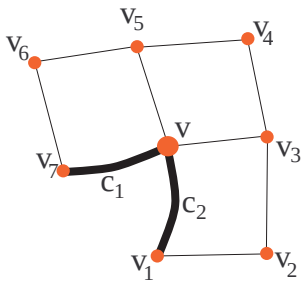

Figure 6: Local corrections near an outward corner.

as v had. In case v is an intersection vertex, we set $d_1(v(v))$ and $d_2(v(v))$ by (1). Otherwise, the new cross-curve second derivative at $v(v)$ is inherited from v by the following rule:

$$d(v(v)) = \frac{d(v)}{4}. \tag{5}$$

Step 2 is completed by calculating the location of every c-vertex using (2) and (3).

As the subdivision iterations proceed, the values $d(v)$ and $\Delta^2 C(v)$ decay at a rate of 4^{-k}, where k is the level of subdivision. Therefore the c-vertices converge to points on the curves, which provides the interpolation property (see figure 8).

4.3 Local Corrections Near C-Vertices

Step 3 performs local modifications to the resulting control net near regular internal c-vertices, and near outward corners. Ordinary vertices that are neighbors of regular internal c-vertices are recalculated by the following rule: Let v denote a regular internal c-vertex, and let v_1 and v_2 denote its two neighboring ordinary vertices (see figure 5). Let $p(v_1), p(v_2)$ denote the locations of v_1 and v_2 that resulted from step 1 of the algorithm. Let $p(v)$ denote the location of v that resulted from step 2 of the algorithm. We calculate the *corrected* locations $\hat{p}(v_1), \hat{p}(v_2)$ by

$$\begin{aligned} \hat{p}(v_1) &= p(v) + \frac{d(v)}{2} + \frac{p(v_1) - p(v_2)}{2}, \\ \hat{p}(v_2) &= p(v) + \frac{d(v)}{2} + \frac{p(v_2) - p(v_1)}{2}. \end{aligned} \tag{6}$$

A different correction rule is applied near outward corner vertices. Let v denote an outward corner vertex, and let $v_1, \ldots, v_7$ denote its neighboring vertices (see figure 6). The vertex v corresponds to the curve c_1 at the parameter value u_1, and to the curve c_2 at the parameter value u_2. In particular, $c_1(u_1) = c_2(u_2)$.

Let $p(v), p(v_1), \ldots, p(v_7)$ denote the locations of $v, v_1, \ldots, v_7$ that resulted from steps 1 and 2 of the algorithm. Let a be the vector $a = \frac{1}{4}(1, -1, -1, 2, -1, -1, 1)$. We calculate the corrected locations for $v_2, \ldots, v_6$ by the following rules:

$$\begin{aligned} t &= \sum_{i=1}^{7} a_i p(v_i), \\ \hat{p}(v_3) &= \frac{1}{3}p(v_3) + \frac{2}{3}\left(2p(v) - p(v_7) + \Delta^2 c_1(v)\right), \\ \hat{p}(v_5) &= \frac{1}{3}p(v_5) + \frac{2}{3}\left(2p(v) - p(v_1) + \Delta^2 c_2(v)\right). \\ \hat{p}(v_2) &= \frac{1}{3}p(v_2) + \frac{2}{3}\left(\hat{p}(v_3) + p(v_1) - p(v) - t\right) \\ \hat{p}(v_6) &= \frac{1}{3}p(v_6) + \frac{2}{3}\left(\hat{p}(v_5) + p(v_7) - p(v) - t\right) \\ \hat{p}(v_4) &= \frac{1}{3}p(v_4) + \frac{2}{3}\left(\hat{p}(v_5) + \hat{p}(v_3) - p(v) + t\right) \end{aligned} \tag{7}$$

There are cases when a single vertex has more than one corrected location, for example an ordinary vertex which is a neighbor of several c-vertices. In these cases we calculate all the corrected locations for such a vertex, using (6) or (7) and define the new location of that vertex to be the arithmetic mean of all the corrected locations. Situations like these occur frequently at the first level of subdivision. The only possibility for a vertex to have more than one corrected location after the first subdivision iteration, is near intersection vertices; The vertex always has two corrected locations, and its new location is taken to be their arithmetic mean.

5 DISCUSSION

The cross-curve second derivatives $d(v)$ of the original control net as determined by the designer, play an important role in determining the shape of the limit surface. As part of constructing the initial control net, a 3D vector $d(v)$ should be initialized by the designer, for every *regular internal c-vertex* and for every *regular boundary c-vertex*.

In case the initial control net contains only intersection vertices (such as the control net in figure 1), (1) determines all the cross-curve second derivatives. Otherwise they can be initialized by any kind of heuristic method.

We suggest the following heuristic approach to initialize $d(v)$ in case v is a regular internal c-vertex: Let v be associated with the curve c at the parameter value u, and let v_1, v_2 denote the two ordinary vertices that are neighbors of v (see figure 5). It seems reasonable to calculate $d(v)$ such that

$$p(v_1) + p(v_2) - 2p(v) = d(v),$$

because we know that this relation holds in the limit. Since $p(v)$ itself depends on $d(v)$ according to (3), we get the following formula for $d(v)$:

$$d(v) = \frac{3}{2}\left(p(v_1) + p(v_2)\right) - 3c(u) + \frac{1}{2}\Delta^2 c(v). \tag{8}$$

In case v is a regular boundary c-vertex, which lies between two boundary intersection vertices v_1, v_2 (see figure 7), one should probably consider the second derivatives at v_1, v_2 when determining $d(v)$. The following heuristic rule can be used:

$$d(v) = \frac{\Delta^2 c_1(v) + \Delta^2 c_2(v)}{2}, \tag{9}$$

where v_1, v_2 are associated with $c_1(u_1)$ and $c_2(u_2)$ respectively.

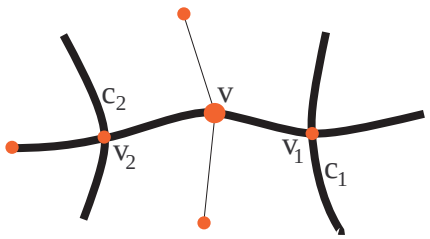

Figure 7: A regular boundary c-vertex between two boundary intersection vertices

The are many cases when the choice $d(v) = 0$ generates the nicest shapes when v is a regular boundary c-vertex. Recall that the natural interpolating cubic spline has zero second derivative at its ends.

Other ways of determining $d(v)$ may employ variational principles. One can choose $d(v)$ such as to minimize a certain fairness measure of the entire surface.

6 CONCLUSIONS

With *combined subdivision schemes* that extend the notion of the known subdivision schemes, it is simple to generate surfaces of arbitrary topological type that interpolate nets of curves given in any parametric representation. The scheme presented in this paper is easy to implement and generates nice looking and almost G^2 surfaces, provided that the given curves are C^2. These surfaces are suitable for machining purposes since they have bounded curvature.

The current algorithm is restricted to nets of curves where no more than two curves intersect at one point, which is a considerable restriction for many applications. However, we believe that the basic idea of applying subdivision rules that explicitly involve the given curve data, and the general theory of combined subdivision schemes can be extended to handle nets where three or more curves intersect at one point, as well as nets with irregular c-vertices.

The proposed scheme can work even if the given curves are not C^2, since it only uses point-wise evaluations. In case the curves are C^1, for example, the limit surface will be only G^1. Moreover, in case a given curve has a local 'fault', and otherwise it is C^2, the local 'fault' will have only a local effect on the limit surface.

Creases in the limit surface can be introduced along a given curve by avoiding the corrections made to vertices near that curve in step 3 of the subdivision. This causes the curve to act as a boundary curve to the surface on both sides of the curve.

Concerning the computation time, notice that most of the computational work in each iteration is spent in the first step of the subdivision iteration, namely, in applying Catmull-Clark's scheme. The local corrections are very simple, and apply only near c-vertices (whose number, after a few iterations, is much lower than that of the ordinary vertices).

Using the analysis tools we have developed in [7, 8], other combined subdivision schemes can be constructed to perform other tasks, such as the generation of surfaces that satisfy certain boundary conditions, including tangent plane conditions [10], and even curvature continuity conditions.

Figures 8-19 show several surfaces created by our algorithm.

Acknowledgement

This work is sponsored by the Israeli Ministry of Science. I thank Nira Dyn for her guidance and many helpful comments, and Peter Schröder for his constant encouragement and advice.

References

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

[2] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 85–94. ACM SIGGRAPH, 1998.

[3] D. Doo and M. Sabin. Behaviour of recursive division surface near extraordinary points. *Computer Aided Design*, 10:356–360, 1978.

[4] N. Dyn, J. A. Greogory, and D. Levin. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9:160–169, 1990.

[5] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using catmull-clark sutfaces. In *SIGGRAPH 93 Conference Proceedings*, Annual Conference Series, pages 35–44. ACM SIGGRAPH, 1993.

[6] L. Kobbelt, T. Hesse, H. Prautzsch, and K. Schweizerhof. Interpolatory subdivision on open quadrilateral nets with arbitrary topology. *Computer Graphics Forum*, 15:409–420, 1996. Eurographics '96 issue.

[7] A. Levin. Analysis of combined subdivision schemes 1. in preparation, available on the web at http://www.math.tau.ac.il/~adilev, 1999.

[8] A. Levin. Analysis of combined subdivision schemes 2. in preparation, available on the web at http://www.math.tau.ac.il/~adilev, 1999.

[9] A. Levin. Analysis of combined subdivision schemes for the interpoation of curves. SIGGRAPH'99 CDROM Proceedings, 1999.

[10] A. Levin. Combined subdivision schemes for the design of surfaces satisfying boundary conditions. To appear in CAGD, 1999.

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

[12] A. H. Nasri. Curve interpolation in recursively generated b-spline surfaces over arbitrary topology. *Computer Aided Geometric Design*, 14:No 1, 1997.

[13] A. H. Nasri. Interpolation of open curves by recursive subdivision surface. In T. Goodman and R. Martin, editors, *The Mathematics of Surfaces VII*, pages 173–188. Information Geometers, 1997.

[14] M. Sabin. Cubic recursive division with bounded curvature. In P. J. Laurent, A. le Mehaute, and L. L. Schumaker, editors, *Curves and Surfaces*, pages 411–414. Academic Press, 1991.

[15] J. Schweitzer. *Analysis and Applications of Subdivision Surfaces*. PhD thesis, University of Washington, Seattle, 1996.

[16] D. J. T. Storry and A. A. Ball. Design of an n-sided surface patch. *Computer Aided Geometric Design*, 6:111–120, 1989.

[17] G. Taubin. A signal processing approach to fair surface design. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 351–358. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[18] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. *Computer Graphics Proceedings (SIGGRAPH 96)*, pages 189–192, 1996.

Appendix

We present the procedure for calculating the weights mentioned in §2, as formulated by Sabin in [14].

Let $n > 2$ denote a vertex valency. Let $k := \cos(\pi/n)$. Let x be the unique real root of

$$x^3 + (4k^2 - 3)x - 2k = 0,$$

satisfying $x > 1$. Then

$$\begin{aligned} W_n &= x^2 + 2kx - 3, \\ \alpha_n &= 1, \\ \gamma_n &= \frac{kx + 2k^2 - 1}{x^2(kx+1)}, \\ \beta_n &= -\gamma_n. \end{aligned} \tag{10}$$

n	W_n	γ_n
3	1.23606797749979…	0.06524758424985…
4	1	0.25
5	0.71850240323974…	0.40198344690335…
6	0.52233339335931…	0.52342327689253…
7	0.39184256502794…	0.61703187134796…

Table 1: The weights used in Sabin's variant of Catmull-Clark's subdivision scheme

The original paper by Sabin [14] contains a mistake: the formulas for the parameters α, β and γ that appear in §4 there, are $\beta := 1,\ \gamma := -\alpha$.

The weights W_n and γ_n for $n = 3, \ldots, 7$ are given in table 1.

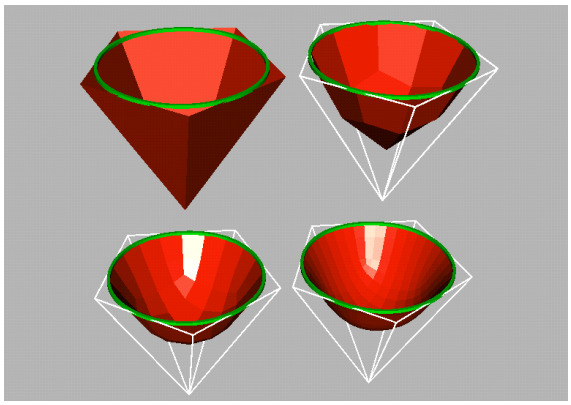

Figure 8: Three iterations of the algorithm. We have chosen $d(v) = 0$ for every c-vertex v, which results in parabolic points on the surface boundary.

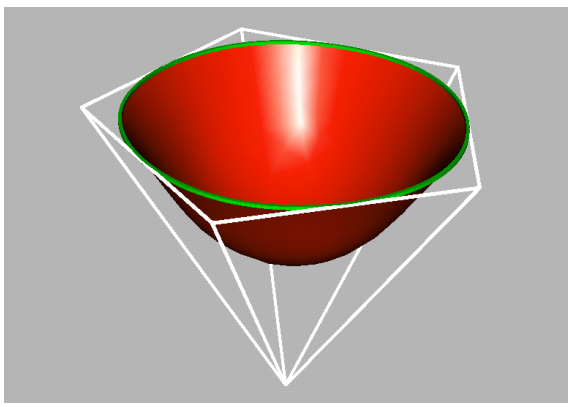

Figure 9: The limit surface of the iterations shown in figure 8

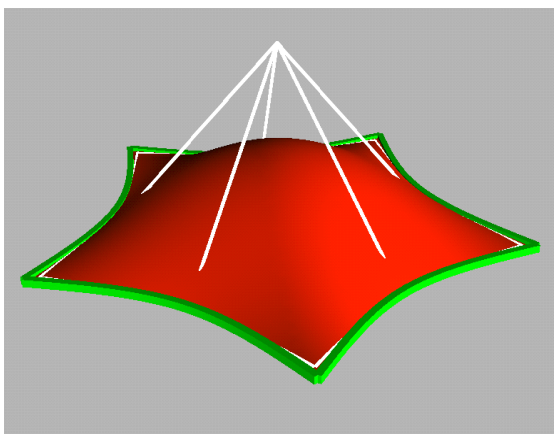

Figure 10: A 5-sided surface generated from a simple control net, with zero $d(v)$ for all c-vertices v. Our algorithm easily fills arbitrary N-sided patches.

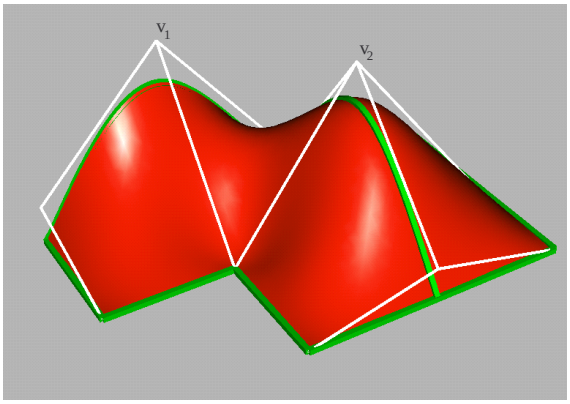

Figure 11: A surface with an outward corner. We used (8) to calculate $d(v_2)$, and set $d(v_1) = 0$.

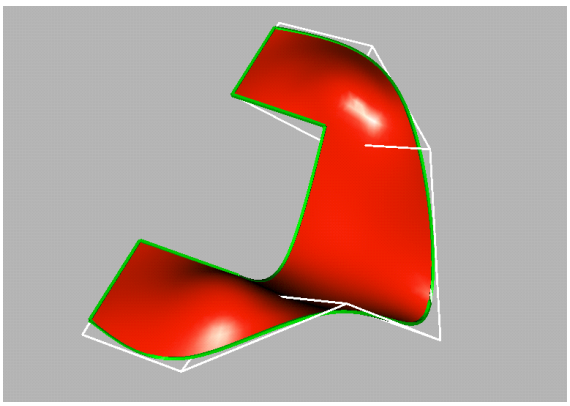

Figure 12: A surface with non smooth boundary curves, and zero cross-curve second derivatives

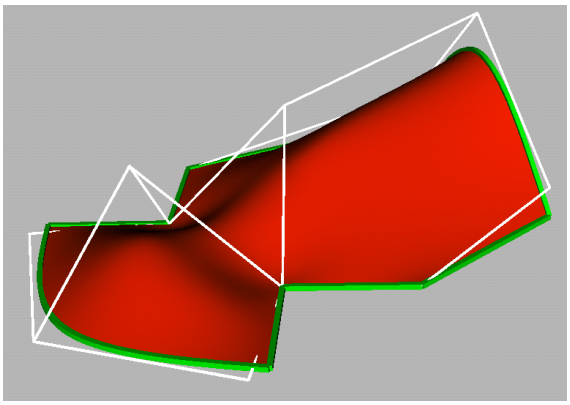

Figure 13: A surface with non smooth boundary curves, and zero cross-curve second derivatives

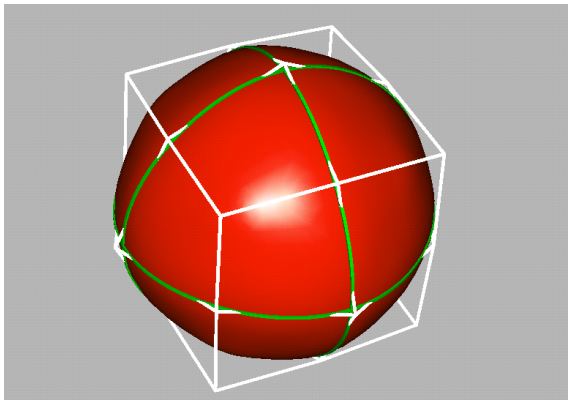

Figure 14: A closed surface. The cross curve second derivatives for regular internal c-vertices were calculated using (9).

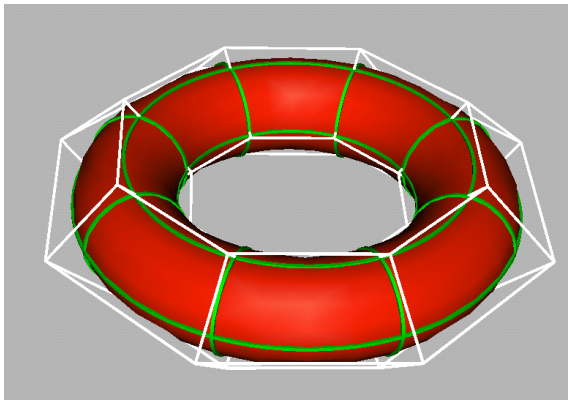

Figure 15: A Torus-like surface, from a net of circles.

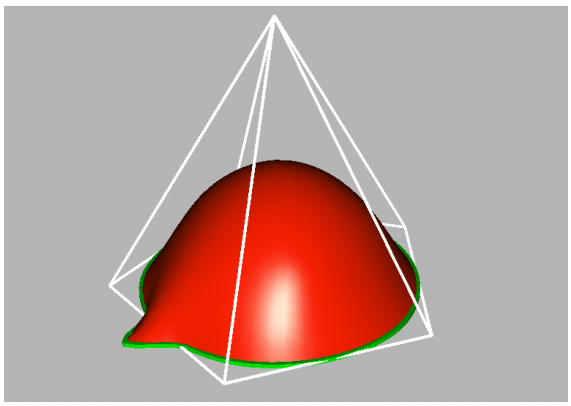

Figure 16: Introducing small perturbations to the given curves results in small and local perturbations of the limit surface. Notice that the original control net does not contain the information of the small perturbations. These come directly from the data of the curves.

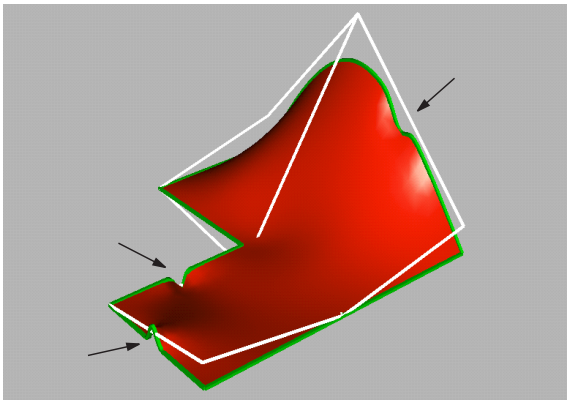

Figure 17: Small perturbations to the given curves result in small and local deformation of the limit surface.

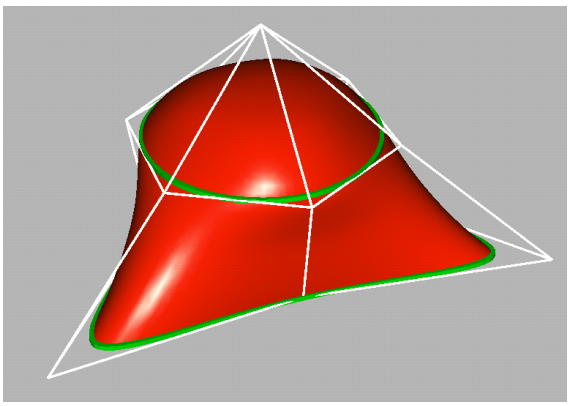

Figure 18: A surface constructed from two given sections. The cross curve second derivatives for the regular internal c-vertices were calculated using (8). For boundary vertices, we took $d(v) = 0$.

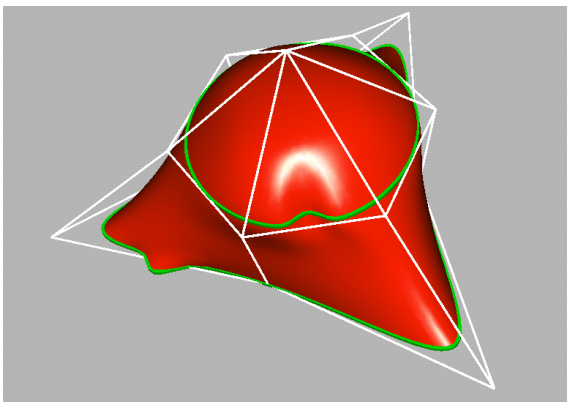

Figure 19: the same surface as in figure 18 after introducing small perturbations in the section curves.