

## On-Line Geometric Modeling Notes

### REFINEMENT

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

#### Overview

Bézier curves, B-spline curves and subdivision curves are all based upon the input of a control polygon and the specification of an algorithmic method that constructs a curve from this sequence of points. Fundamental to these methods is the concept of a *refinement*. These refinement methods, as defined mathematically, can be quite complex. However, in practice they are quite simple and usually easy to implement.

In these notes, we discuss the mathematical notion of refinement.

---

#### What is a Refinement Scheme

A *refinement* process is a scheme which defines a sequence of control polygons

$$\begin{aligned} & \mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n \\ & \mathbf{P}_0^1, \mathbf{P}_1^1, \dots, \mathbf{P}_{n_1}^1 \\ & \mathbf{P}_0^2, \mathbf{P}_1^2, \dots, \mathbf{P}_{n_2}^2 \\ & \vdots \\ & \mathbf{P}_0^k, \mathbf{P}_1^k, \dots, \mathbf{P}_{n_k}^k \\ & \vdots \end{aligned}$$

where for any  $k > 0$ , each  $\mathbf{P}_j^k$  can be written as

$$\mathbf{P}_j^k = \sum_{i=0}^{n_{k-1}} \alpha_{i,j,k} \mathbf{P}_i^{k-1}$$

That is, any element  $\mathbf{P}_j^k$  can be written as a linear combination of the control points  $\{\mathbf{P}_0^{k-1}, \mathbf{P}_1^{k-1}, \dots, \mathbf{P}_{n_{k-1}}^{k-1}\}$

from the control polygon generated in the prior step. For each fixed  $j$  and  $k$  the sequence  $\alpha_{i,j,k}$  is frequently called a *mask*.

This is a very general scheme, and quite complex to manage and analyze. It covers the cases where the number of control points in each successive polygon is allowed to increase (Chaikin's Curves and Doo-Sabin's subdivision surfaces are examples of this), or must decrease (de Casteljau's algorithm for generating Bézier Curves is an example of this).

It would be incredibly rare to use the entire set of control points from the  $k - 1$ st sequence to calculate each new control point in the  $k$ th sequence as there may be thousands of points to consider – and so in general we assume that most of the  $\alpha_{i,j,k}$ s are zero. To simplify things further, we frequently limit this to a *uniform scheme*, where the  $\alpha$ s are independent of the level of refinement ( $k$ ). This implies that the scheme is basically the same at each iteration of the refinement process. A further simplification, where the mask is the same for every point of a control polygon, is called a *stationary scheme*.

If all points that result from a refinement process lie on the lines joining the points of a control polygon, the process is typically called a “corner cutting scheme”. An example of such a scheme is the Chaikin's Curve.

---

## A Matrix Method for Refinement

The equation

$$\mathbf{P}_j^k = \sum_{i=0}^{n_{k-1}} \alpha_{i,j,k} \mathbf{P}_i^{k-1}$$

can be written in matrix form as

$$\mathbf{P}_j^k = \begin{bmatrix} \alpha_{0,j,k} & \alpha_{1,j,k} & \cdots & \alpha_{n_{k-1},j,k} \end{bmatrix} \begin{bmatrix} \mathbf{P}_0^{k-1} \\ \mathbf{P}_1^{k-1} \\ \vdots \\ \mathbf{P}_{n_k}^{k-1} \end{bmatrix}$$

and overall

$$\begin{bmatrix} \mathbf{P}_0^k \\ \mathbf{P}_1^k \\ \vdots \\ \mathbf{P}_{n_k}^k \end{bmatrix} = S_k \begin{bmatrix} \mathbf{P}_0^{k-1} \\ \mathbf{P}_1^{k-1} \\ \vdots \\ \mathbf{P}_{n_{k-1}}^{k-1} \end{bmatrix}$$

where  $S_k$  is the refinement matrix

$$S_k = \begin{bmatrix} \alpha_{0,0,k} & \alpha_{1,0,k} & \cdots & \alpha_{n_{k-1},0,k} \\ \alpha_{0,1,k} & \alpha_{1,1,k} & \cdots & \alpha_{n_{k-1},1,k} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{0,n_k,k} & \alpha_{1,n_k,k} & \cdots & \alpha_{n_{k-1},n_k,k} \end{bmatrix}$$

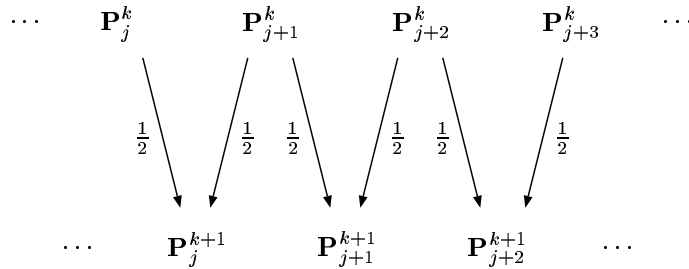
and is an  $(n_k + 1) \times (n_{k-1} + 1)$  matrix. In general it is best to think of this matrix as being sparse (i.e. most of the entries being zero) with non-zero entries clustered along the diagonal.

### Example – A Stationary Uniform Refinement Scheme

Suppose we are given the control polygon  $\{\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n\}$ . Define the refinement scheme by the following equation

$$\mathbf{P}_j^k = \frac{1}{2} (\mathbf{P}_j^{k-1} + \mathbf{P}_{j+1}^{k-1})$$

where  $0 \leq j \leq n - k$  and  $k = 0, 1, 2, \dots, n - 1$ . In other words, each successive point in the refinement is taken to be the midpoint of the line segment joining the two corresponding points in the previous control polygon.



Note here that two of the  $\alpha$ s are  $\frac{1}{2}$  and the remainder are zero.

In this case, the refinement process stops after  $n - 1$  steps – as the control polygon for each step of the refinement has one fewer points than does the control polygon in the previous step – the final control polygon having one point.

To represent this refinement process via matrices, the refinement matrix  $S_k$  is

$$S_k = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & \cdots & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

where the matrix is  $k \times (k + 1)$ .

If this refinement is taken to completion, we have just calculated a point on the  $n$ th degree Bézier curve defined by this control polygon.

---

### Example – A Non-Stationary Uniform Subdivision Scheme

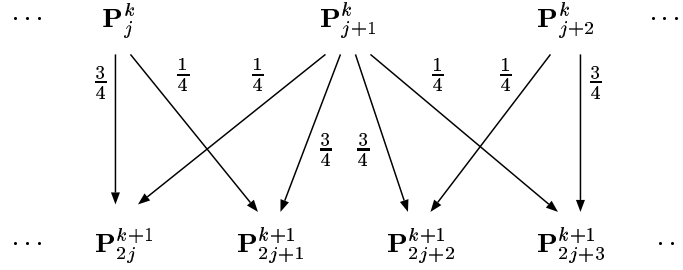
Suppose we are given the control polygon  $\{\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n\}$ . Define a refinement scheme by the following

$$\mathbf{P}_{2j}^k = \frac{3}{4}\mathbf{P}_j^k + \frac{1}{4}\mathbf{P}_{j+1}^k$$

and

$$\mathbf{P}_{2j+1}^k = \frac{1}{4}\mathbf{P}_j^k + \frac{3}{4}\mathbf{P}_{j+1}^k$$

for  $j = 0, 1, 2, 3, \dots$



Notice that this gives us a new control polygon

$$\begin{aligned}
 \mathbf{P}_0^1 &= \frac{3}{4}\mathbf{P}_0 + \frac{1}{4}\mathbf{P}_1 \\
 \mathbf{P}_1^1 &= \frac{1}{4}\mathbf{P}_0 + \frac{3}{4}\mathbf{P}_1 \\
 \mathbf{P}_2^1 &= \frac{3}{4}\mathbf{P}_1 + \frac{1}{4}\mathbf{P}_2 \\
 \mathbf{P}_3^1 &= \frac{1}{4}\mathbf{P}_1 + \frac{3}{4}\mathbf{P}_2 \\
 \mathbf{P}_4^1 &= \frac{3}{4}\mathbf{P}_2 + \frac{1}{4}\mathbf{P}_3 \\
 \mathbf{P}_5^1 &= \frac{1}{4}\mathbf{P}_2 + \frac{3}{4}\mathbf{P}_3 \\
 &\vdots
 \end{aligned}$$

Applying this refinement process to a control polygon of length  $n + 1$  gives a new control polygon of length  $2n$ .

This is just Chaikin's Algorithm for curve generation. As the algorithm proceeds the number of control points gets arbitrarily large, but converges to a unique curve.

---

### Refinement Schemes for Meshes

Similar methods (with much more notationally complex mathematics) exist for control meshes that result in surface generation algorithms. In general, the idea is the same – the refinement operation generates new control points from the control points of the previous mesh.

---

### Summary

Refinement schemes generate an important class of curve and surface drawing algorithms that are useful in geometric modeling. The schemes generate a sequence of control polygons in the two-dimensional case, or control meshes in the three dimensional case that can be used for curve generation. The methods are useful in the case of Bézier curves and Bézier patches as well as in the generation of subdivision curves and surfaces.

---

## References

- [1] DYN, N., GREGORY, J., AND LEVIN, D. Analysis of uniform binary subdivision schemes for curve design. *Constructive Approximation* 7, 2 (1991), 127–148.
  - [2] DYN, N., AND LEVIN, D. The subdivision experience. In *Curves and Surfaces II* (1991), A. L. H. P.J. Laurent and L. Schumaker, Eds., pp. 1–17.
  - [3] MICCHELLI, C. A., AND PRAUTZSCH, H. Uniform refinement of curves. *Linear Algebra and Its Applications* 114/115 (1989), 841–870.
- 

All contents copyright (c) 1996, 1997, 1998, 1999, 2000  
Computer Science Department, University of California, Davis  
All rights reserved.

## On-Line Geometric Modeling Notes

### SUBDIVISION CURVES

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

Curve generation methods are an important topic in computer graphics and geometric modeling. A new set of methods is now becoming popular which utilize a control polygon, as in the Bézier or B-spline case, but instead of using analytic methods to directly calculate points on the curve, these methods successively refine the control polygons into an sequence of control polygons that, in the limit, converge to a curve. By doing this, freedom from a closed-form mathematical expression is achieved, and a wide variety of curve types can be expressed. The curves are commonly called *subdivision* curves as the methods are based upon the binary subdivision of the uniform B-spline curves.

As it turns out, curves are fairly straightforward, and the interesting cases are surfaces and solids where the topology of the underlying control mesh can be quite complex. However, the curve cases are easier to present and therefore an understanding of these sections is important before proceeding to surfaces and solids. Most of the techniques presented here are similar to the techniques that are utilized in the surface cases.

To understand the basis of this method, the reader should first consult the notes on Chaikin's curve – which can be pointed to as the first of these algorithms.

- 
- What is a Subdivision/Refinement Process?
  - Chaikin's Curves - Curve generation by cutting corners.
  - Subdivision Curves based upon the Quadratic Uniform B-Spline Curve
  - Subdivision Curves based upon the Cubic Uniform B-Spline Curve
    - Classifying Points on the Subdivision as Vertex and Edge Points

- Direct Calculation of Points on Cubic Subdivision Curves
    - Rewriting the Refinement Procedure in Terms of a Matrix Equation
    - Calculating the Tangent Vectors at a Point
- 

**All contents copyright (c) 1996, 1997, 1998, 1999, 2000  
Computer Science Department, University of California, Davis  
All rights reserved.**



## On-Line Geometric Modeling Notes

# CHAIKIN'S ALGORITHMS FOR CURVES

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

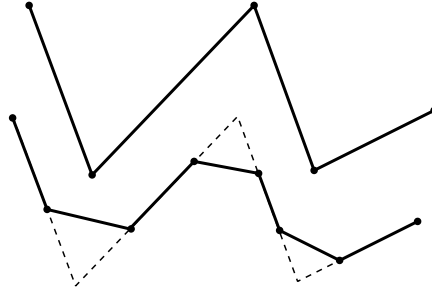
### Overview

In 1974, George Chaikin [1] gave a lecture at the University of Utah in which he specified a novel procedure for generating curves from a limited number of points. This algorithm is interesting as it was one of the first corner cutting or refinement algorithms specified to generate a curve from a set of control points, or control polygon. The algorithm had been largely forgotten by the graphics/geometric-modeling community in the flurry of activity studying B-spline curves and surfaces, as it has been shown to generate a quadratic uniform B-spline curve. However, researchers have now drifted away from the rigidity of curve and surface models based upon an underlying analytical form, and the basic paradigm of Chaikin – his curves were generated by successive refinement of a control polygon – is now utilized to generate a wide variety of curve and surface types.

---

### The Corner-Cutting Paradigm

Chaikin had a different idea. Researchers since Bézier had been working with curves generated by control polygons but had focused their analysis on the underlying analytical representation, frequently based upon Bernstein polynomials. Chaikin decided to develop algorithms that worked with the control polygon directly – so-called geometric algorithms. His curve generation scheme is based upon “corner cutting” where the algorithm generates a new control polygon by cutting the corners off the original one. The figure below illustrates this idea, where an initial control polygon has been refined into a second polygon (slightly offset) by cutting off the corners of the first sequence.



Clearly we could then take this second control polygon and cut the corners off it, producing a third sequence, etc. In the limit, hopefully we would have a curve. This was Chaikin's idea.

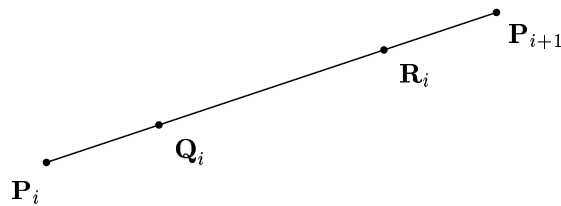
---

### Chaikin's Method

Chaikin utilized fixed ratios on cutting off his corners, so that they were all cut the same. When written down mathematically, Chaikin's method proceeds as follows: Given a control polygon  $\{\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n\}$ , we refine this control polygon by generating a new sequence of control points

$$\{\mathbf{Q}_0, \mathbf{R}_0, \mathbf{Q}_1, \mathbf{R}_1, \dots, \mathbf{Q}_{n-1}, \mathbf{R}_{n-1}\}$$

where each new pair of points  $\mathbf{Q}_i, \mathbf{R}_i$  are to be taken taken to be at a ratio of  $\frac{1}{4}$  and  $\frac{3}{4}$  between the endpoints of the line segment  $\overline{\mathbf{P}_i \mathbf{P}_{i+1}}$ .



That is

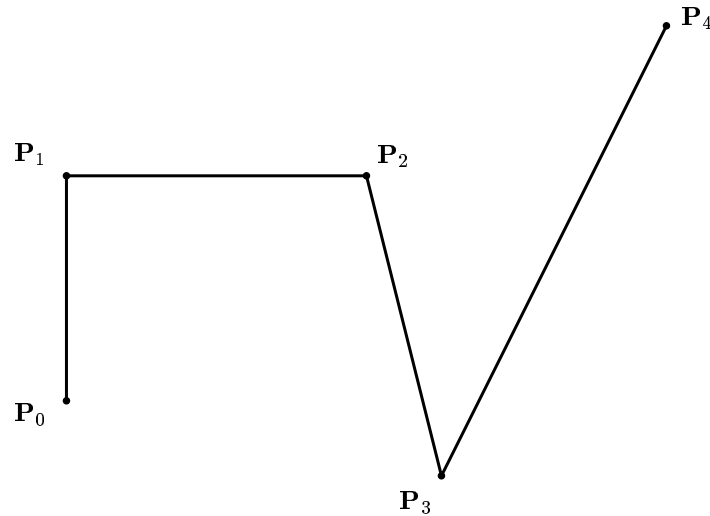
$$\begin{aligned}\mathbf{Q}_i &= \frac{3}{4}\mathbf{P}_i + \frac{1}{4}\mathbf{P}_{i+1} \\ \mathbf{R}_i &= \frac{1}{4}\mathbf{P}_i + \frac{3}{4}\mathbf{P}_{i+1}\end{aligned}$$

These  $2n$  new points can be considered a new control polygon – a *refinement* of the original control polygon.

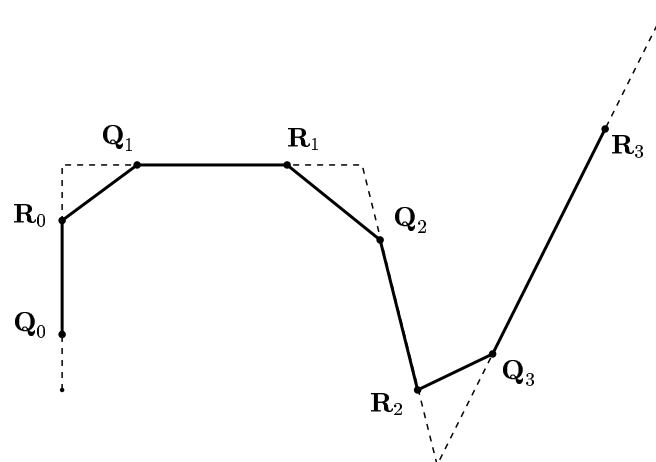
---

### Example – How Chaikin's Algorithm Works

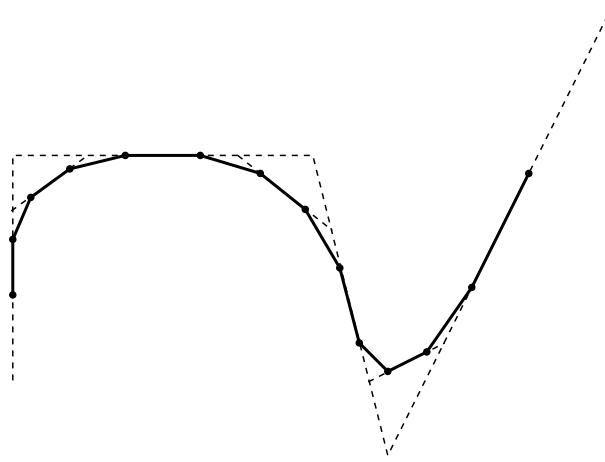
To give an example of Chaikin's Algorithm, consider the following control polygon:



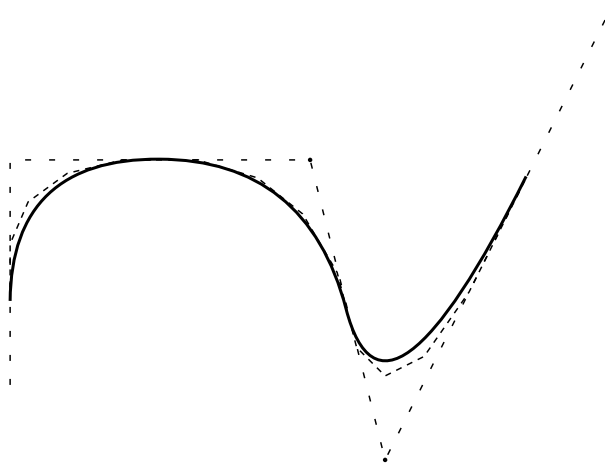
Chaikin's algorithm generates the points  $Q_i$  and  $R_i$  and uses these points to refine the curve and obtain the control polygon shown in the figure below



These points are in turn utilized to generate a new refinement,



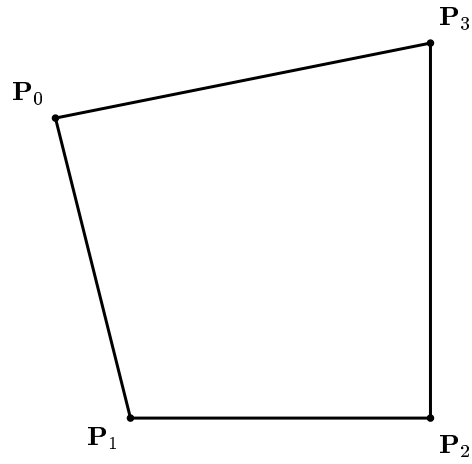
and again, these points are utilized to obtain another refinement, etc. The following illustration shows the initial control polygon, the third control polygon in the sequence, and the final curve.



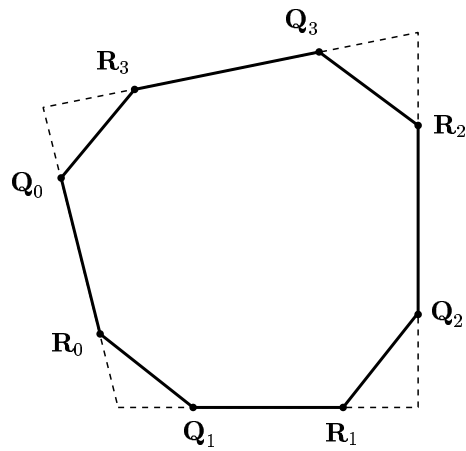

---

### Example – A Closed Curve

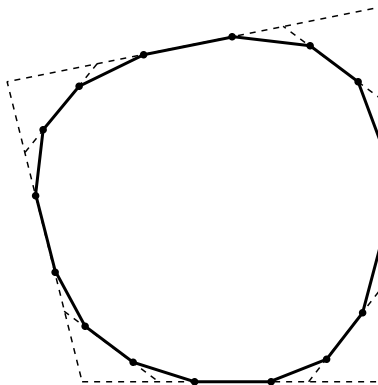
To illustrate Chaikin's curve on a closed control polygon, consider the following figure.



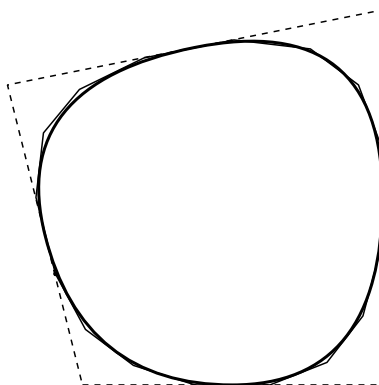
In this case, the control point indices are taken modulo  $n + 1$  (or 4, in the case of the figure). We can still apply Chaikin's method to this figure, obtaining



This new control polygon can then be utilized to obtain a second refinement as in the following figure



and it is clear that we could continue this process indefinitely. For graphics purposes, we will stop after a number of refinements and approximate the curve by connecting the points of the resulting control polygon by straight lines. The initial control polygon and the second refinement are shown in the following figure along with the resulting curve.



---

## Summary

Chaikin specified a simple scheme by which curves could be generated from a given control polygon. The idea is unique in that the underlying mathematical description is ignored in favor of a geometric algorithm that just selects new control points along the line segments of the original control polygon.

It should be noted that Chaikin's curve has been shown to be equivalent to a quadratic B-spline curve [2] (a piecewise quadratic Bézier curve). However, Chaikin's method avoids the analytical definition of B-splines and provides a simple, elegant curve drawing mechanism.

---

## References

- [1] CHAIKIN, G. An algorithm for high speed curve generation. *Computer Graphics and Image Processing* 3 (1974), 346–349.
- [2] RIESENFELD, R. On Chaikin's algorithm. *IEEE Computer Graphics and Applications* 4, 3 (1975), 304–310.

**All contents copyright (c) 1996, 1997, 1998, 1999**  
**Computer Science Department, University of California, Davis**  
**All rights reserved.**

## On-Line Geometric Modeling Notes

# QUADRATIC UNIFORM B-SPLINE CURVE REFINEMENT

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

### Overview

Subdivision methods for curve generation are based upon a procedure which successively refines a control polygon into a sequence of control polygons that, in the limit, converges to a curve. The curves are commonly called *subdivision* curves as the refinement methods are based upon the binary subdivision of uniform B-spline curves.

The uniform B-spline curves, surfaces and solids have been extensively studied in the literature and subdivision methods for these objects are well known. We develop here the refinement method for a quadratic uniform B-spline curve and show that the refinement is exactly that specified by Chaikin's Algorithm [1].

---

### The Matrix Equation for the Quadratic Uniform B-Spline Curve

Given a set of control points  $\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n$  the quadratic uniform B-spline curve  $\mathbf{P}(t)$  defined by these control points can be defined in segments by the  $n - 1$  equations

$$\mathbf{P}(t) = \begin{bmatrix} 1 & t & t^2 \end{bmatrix} M \begin{bmatrix} \mathbf{P}_k \\ \mathbf{P}_{k+1} \\ \mathbf{P}_{k+2} \end{bmatrix}$$

for  $k = 0, 1, \dots, n - 2$ , and  $0 \leq t \leq 1$ , and where

$$M = \begin{bmatrix} 1 & 1 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix}$$

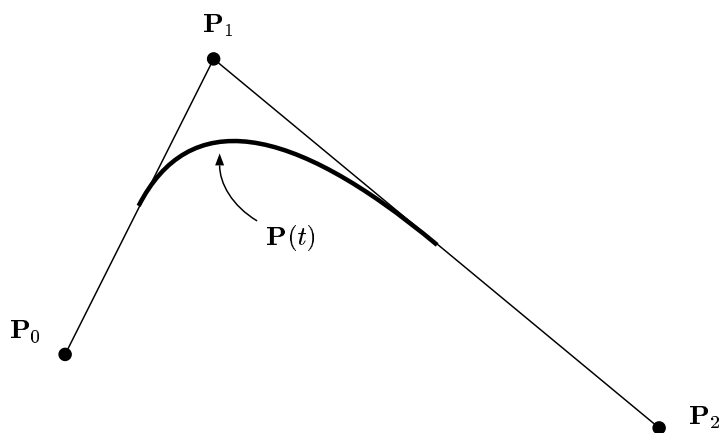
The matrix  $M$ , when multiplied by  $\begin{bmatrix} 1 & t & t^2 \end{bmatrix}$  defines the quadratic uniform B-spline blending functions.



---

## Splitting and Refinement

We will begin by studying the binary subdivision of a quadratic uniform B-spline curve  $\mathbf{P}(t)$  defined by the control polygon  $\{\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2\}$ , containing only three points, and then extend this to control polygons containing larger numbers of points.



We can perform a binary subdivision of the curve, by applying one of two splitting matrices

$$S^L = \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \end{bmatrix}$$
$$S^R = \frac{1}{4} \begin{bmatrix} 1 & 3 & 0 \\ 0 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix}$$

to the control polygon. (When applied to the control polygon  $S^L$  gives the first half of the curve, and  $S^R$  gives the second half.)

As it turns out, several of the control points for the two subdivided components are the same. Thus, we

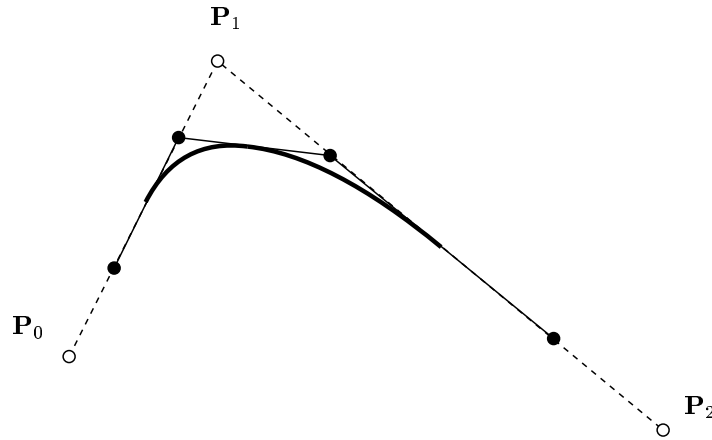
can combine these matrices, creating a  $4 \times 3$  matrix

$$R = \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix}$$

and apply it to a control polygon as follows:

$$\begin{bmatrix} \mathbf{P}_0^1 \\ \mathbf{P}_1^1 \\ \mathbf{P}_2^1 \\ \mathbf{P}_3^1 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix} = \begin{bmatrix} \frac{3}{4}\mathbf{P}_0 + \frac{1}{4}\mathbf{P}_1 \\ \frac{1}{4}\mathbf{P}_0 + \frac{3}{4}\mathbf{P}_1 \\ \frac{3}{4}\mathbf{P}_1 + \frac{1}{4}\mathbf{P}_2 \\ \frac{1}{4}\mathbf{P}_1 + \frac{3}{4}\mathbf{P}_2 \end{bmatrix}$$

and so the refined curve has control points that are positioned as in the following illustration:



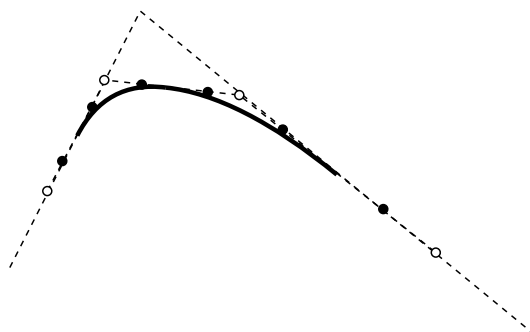
i.e. at  $\frac{1}{4}$  and  $\frac{3}{4}$  along each of the lines of the control polygon. These are the same points as are developed in Chaikin's method

---

### The General Refinement Procedure

The general procedure is – given a control polygon, we can generate a refinement of this set of points by constructing new points along each edge of the original polygon at a distance of  $\frac{1}{4}$  and  $\frac{3}{4}$  between the endpoints of the edge. The general idea behind *subdivision curves* is to assemble these points into a new

control polygon which can then be used as input to another refinement operation, generating a new set of points and another control polygon – and then continue this process until a refinement is reached that accurately represents the curve to a desired resolution. The following illustration shown the second refinement in the case above.



Since this general refinement procedure is developed from the binary subdivision of a uniform B-spline curve, and the control points of the refined polygon are unions of those of the subdivided curves, we then have that the control points of the refined polygons must converge to the curve. That is, in the limit, the sequence of control polygons generated by the refinement procedure converges to a quadratic uniform B-spline curve. This shows that Chaikin's curve is really a quadratic uniform B-spline curve.

---

## Summary

Given a control polygon we can specify a simple process that can be used to generate successive refinements of the control polygon and, in the limit, converges to the uniform B-spline curve defined by the original control polygon. This scheme is the basis for the development of the Doo-Sabin subdivision scheme which generalizes Chaikin's algorithm to surfaces.

---

## References

- [1] CHAIKIN, G. An algorithm for high speed curve generation. *Computer Graphics and Image Processing* 3 (1974), 346–349.

- [2] RIESENFELD, R. On Chaikin's algorithm. *IEEE Computer Graphics and Applications* 4, 3 (1975), 304–310.

---

All contents copyright (c) 1996, 1997, 1998, 1999, 2000  
Computer Science Department, University of California, Davis  
All rights reserved.

## On-Line Geometric Modeling Notes

# CUBIC UNIFORM B-SPLINE CURVE REFINEMENT

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

### Overview

The binary subdivision of the uniform B-spline curves and surfaces motivate much of the work on subdivision curves and surfaces – where the word “subdivision” here is taken from the process that is used to subdivide a curve or surface into multiple components. In the uniform B-spline case a subdivided component shares many of its individual control points with other components, allowing us to define the totality of control points generated through subdivision as a refinement of the original control polygon. These new control points can be assembled into a new control polygon and refined again by the same methods. Successive refinements produce a sequence of control polygons that in the limit converge to a curve.

The uniform B-spline curves, surfaces and solids have been extensively studied in the literature and subdivision methods for these objects are well known. We develop here the refinement method for a cubic uniform B-spline curve. The analysis is similar to that presented in the quadratic case however, the refinement algorithm can be specified in a different manner which eventually allows us to use eigenanalysis and directly calculate points on the curve.

---

### The Matrix Equation for the Cubic Uniform B-Spline Curve

Given a set control polygon  $\{\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n\}$  the cubic uniform B-spline curve  $\mathbf{P}(t)$  defined by these control points can be defined in segments by the  $n - 2$  equations

$$\mathbf{P}(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} M \begin{bmatrix} \mathbf{P}_k \\ \mathbf{P}_{k+1} \\ \mathbf{P}_{k+2} \\ \mathbf{P}_{k+3} \end{bmatrix}$$

for  $k = 0, 1, \dots, n - 3$ , and  $0 \leq t \leq 1$ , and where

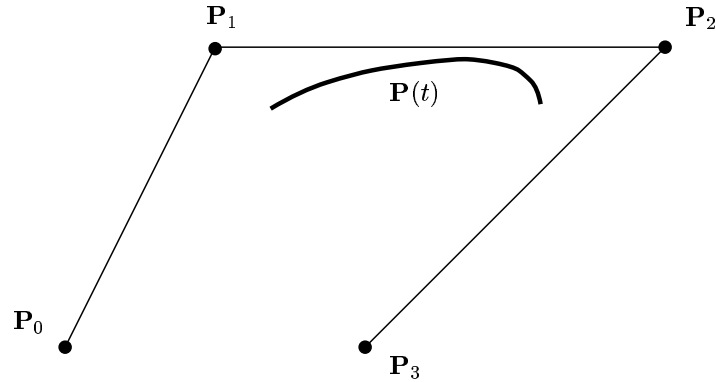
$$M = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

The matrix  $M$ , when multiplied by  $\begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix}$  defines the cubic uniform B-spline blending functions.

---

### Splitting and Refinement

We will begin by studying the binary subdivision of a cubic uniform B-spline curve  $\mathbf{P}(t)$  defined by the four control points  $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$  and  $\mathbf{P}_3$ . Such a curve is shown in the following illustration.



We can perform a binary subdivision by applying one of the two splitting matrices

$$S^L = \frac{1}{8} \begin{bmatrix} 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \end{bmatrix}$$

$$S^R = \frac{1}{8} \begin{bmatrix} 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \end{bmatrix}$$

to the control polygon. (When applied to the control polygon  $S^L$  gives the first half of the curve, and  $S^R$  gives the second half.)

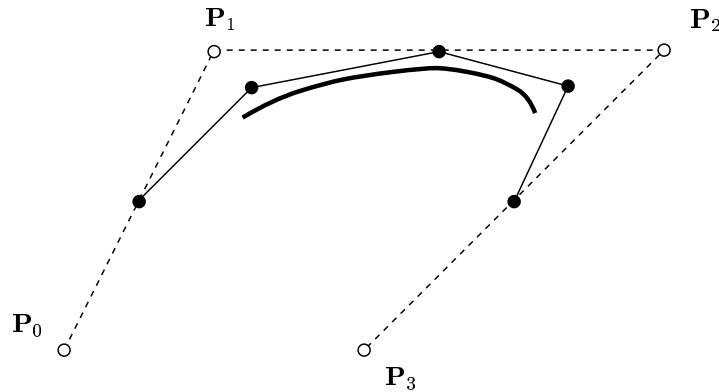
As it turns out, several of the control points for the two subdivided components are the same. Thus, we can combine these matrices, creating a  $5 \times 4$  matrix

$$\frac{1}{8} \begin{bmatrix} 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \end{bmatrix}$$

and apply it to a control polygon as follows

$$\begin{bmatrix} \mathbf{P}_0^1 \\ \mathbf{P}_1^1 \\ \mathbf{P}_2^1 \\ \mathbf{P}_3^1 \\ \mathbf{P}_4^1 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}$$

generating a new control polygon which serves as the refinement of the original. The five control points of this new control polygon specify the two subdivided halves of the curve – and therefore specify the curve itself.




---

### The General Refinement Procedure

In the case of the quadratic curve we were able to state exactly a single procedure for the points of the refinement. In this case, it is not so easy. However, if we examine the rows of  $5 \times 4$  matrix used in the refinement, we see that they have two distinct forms. This motivates us to classify the points of the refinement as vertex and edge points, which is exhibited in another section. This classification makes the description of the refinement process quite straightforward.

---

## Summary

In the case of a uniform cubic B-spline curve we can define process that takes the defining control polygon and creates a sequence of control polygons by refinement. This sequence converges to the curve defined by the original control polygon. The procedure is similar to that given in the quadratic case as it is generated through the matrices for binary subdivision of the curve.

---

## References

- [1] CATMULL, E., AND CLARK, J. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design 10* (Sept. 1978), 350–355.
- 

All contents copyright (c) 1996, 1997, 1998, 1999, 2000  
Computer Science Department, University of California, Davis  
All rights reserved.



## On-Line Geometric Modeling Notes

### VERTEX POINTS AND EDGE POINTS

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

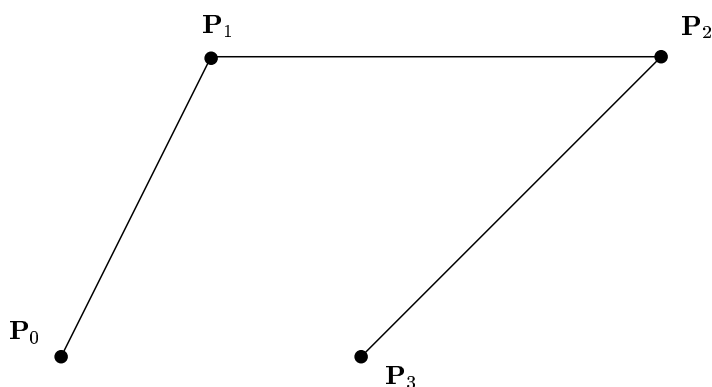
#### Overview

The refinement process defined by the cubic uniform B-spline curve generates a sequence of control polygons that converges to the curve. This process can be specified by examining the binary subdivision methods for the curve to obtain a unique set of control points that generates the subdivision. However, in the case of the cubic curve, there is another way to look at the generation of the points of a refined control polygon. Each of these points can be classified as either a “vertex point” or an “edge point,” and methods can be specified to calculate each point. This procedure turns out to be quite useful as they will allow us to directly calculate points on the limit curve without going through the refinement.

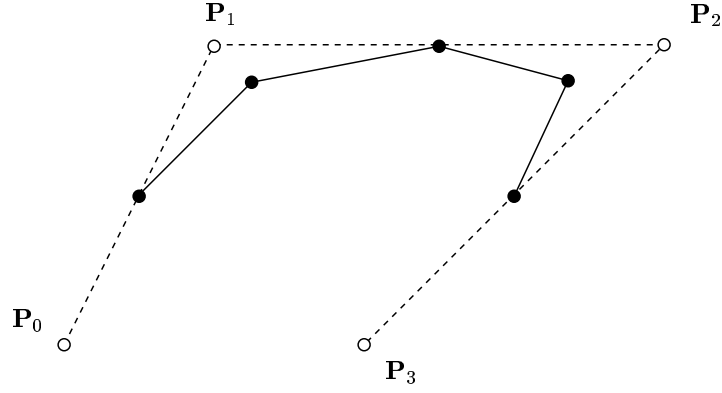
---

#### Classifying the Refinement Points

Suppose we are given a control polygon  $\{P_0, P_1, P_2, P_3\}$ .



If we use the refinement method motivated by the binary subdivision of the curve, we generate a new control polygon shown below



We note that three of the new control points appear to lie at the midpoints of the three respective line segments. These points will be classified as “edge points”. The other points all lie close to on of the vertices of the original control polygon, and will be called “vertex points”.

In this light, if we denote the new control polygon generated by the binary subdivision method as  $\{\mathbf{E}_0, \mathbf{V}_0, \mathbf{E}_1, \mathbf{V}_1, \mathbf{E}_2\}$  then by combining and applying the splitting matrices that generate the binary subdivision of the curve we obtain

$$\begin{bmatrix} \mathbf{E}_0 \\ \mathbf{V}_0 \\ \mathbf{E}_1 \\ \mathbf{V}_1 \\ \mathbf{E}_2 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}$$

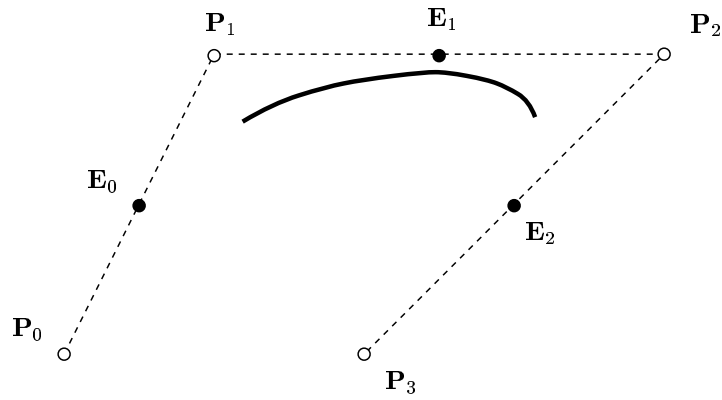
and so the edge points  $\mathbf{E}_0$ ,  $\mathbf{E}_1$  and  $\mathbf{E}_2$  are calculated by

$$\begin{aligned}\mathbf{E}_0 &= \frac{1}{8}(4\mathbf{P}_0 + 4\mathbf{P}_1) \\ &= \frac{\mathbf{P}_0 + \mathbf{P}_1}{2}\end{aligned}$$

$$\begin{aligned}\mathbf{E}_1 &= \frac{1}{8}(4\mathbf{P}_1 + 4\mathbf{P}_2) \\ &= \frac{\mathbf{P}_1 + \mathbf{P}_2}{2}\end{aligned}$$

$$\begin{aligned}\mathbf{E}_2 &= \frac{1}{8}(4\mathbf{P}_2 + 4\mathbf{P}_3) \\ &= \frac{\mathbf{P}_2 + \mathbf{P}_3}{2}\end{aligned}$$

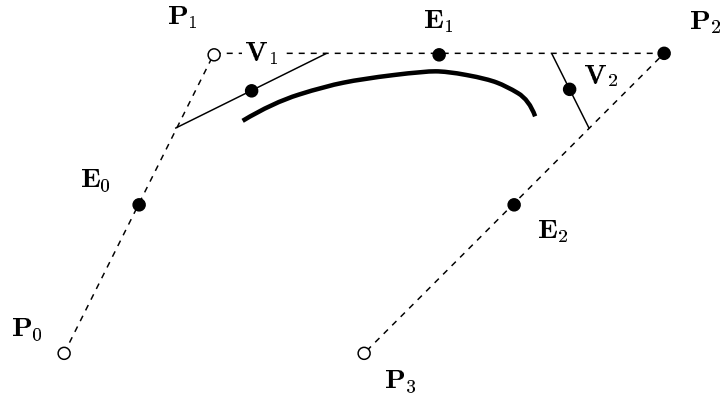
and indeed are the midpoints of the line segments connecting the original control points. These points are illustrated in the following figure:



The  $\mathbf{V}_0$  and  $\mathbf{V}_1$  are calculated by

$$\begin{aligned}
\mathbf{V}_0 &= \frac{1}{8}(\mathbf{P}_0 + 6\mathbf{P}_1 + \mathbf{P}_2) \\
&= \frac{1}{8}((\mathbf{P}_0 + \mathbf{P}_1) + 4\mathbf{P}_1 + (\mathbf{P}_1 + \mathbf{P}_2)) \\
&= \frac{1}{4}\left(\frac{\mathbf{P}_0 + \mathbf{P}_1}{2} + 2\mathbf{P}_1 + \frac{\mathbf{P}_1 + \mathbf{P}_2}{2}\right) \\
&= \frac{1}{4}(\mathbf{E}_0 + 2\mathbf{P}_1 + \mathbf{E}_1) \\
&= \frac{\frac{\mathbf{E}_0 + \mathbf{P}_1}{2} + \frac{\mathbf{P}_1 + \mathbf{E}_1}{2}}{2}
\end{aligned}$$

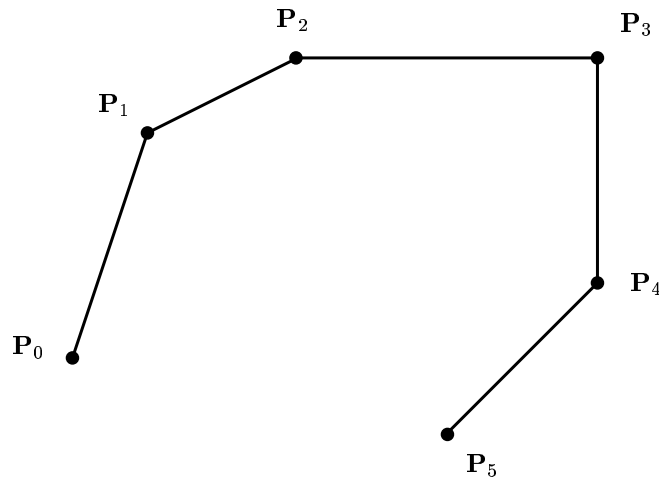
$$\begin{aligned}
\mathbf{V}_1 &= \frac{1}{8}(\mathbf{P}_1 + 6\mathbf{P}_2 + \mathbf{P}_3) \\
&= \frac{1}{8}((\mathbf{P}_1 + \mathbf{P}_2) + 4\mathbf{P}_2 + (\mathbf{P}_2 + \mathbf{P}_3)) \\
&= \frac{1}{4}\left(\frac{\mathbf{P}_1 + \mathbf{P}_2}{2} + 2\mathbf{P}_2 + \frac{\mathbf{P}_2 + \mathbf{P}_3}{2}\right) \\
&= \frac{1}{4}(\mathbf{E}_1 + 2\mathbf{P}_2 + \mathbf{E}_2) \\
&= \frac{\frac{\mathbf{E}_1 + \mathbf{P}_2}{2} + \frac{\mathbf{P}_2 + \mathbf{E}_2}{2}}{2}
\end{aligned}$$



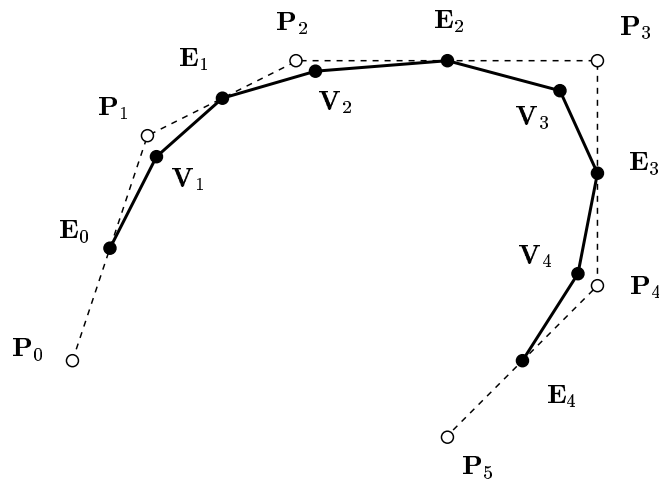
and it is seen that these are the midpoint of the line segment that joins the respective midpoints of the two line segments from the vertex point to the respective edge points. Therefore, we only need to be able to specify midpoints of line segments to be able to specify the refined control polygon.

### An Example of the Refinement Algorithm

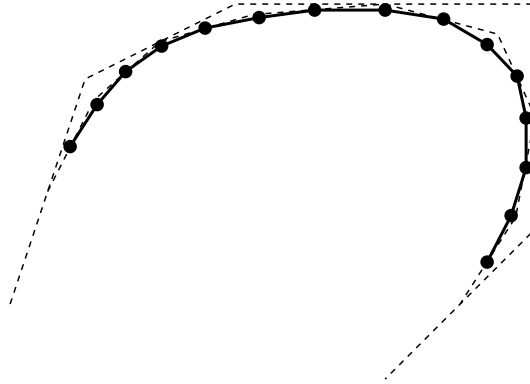
We illustrate the workings of this refinement algorithm via the following three figures. First consider a control polygon as is shown below.



Calculate the edge and vertex points producing a refinement of the original control polygon.



Assemble these points into a new control polygon and utilize it as input again to the refinement process – producing new edge and vertex points from this set of points.



The general idea behind *subdivision curves* is to utilize the control points generated through refinement as input to another refinement operation, and then continue this process until a refinement is reached that accurately represents the curve to a desired resolution.

It is well known in the case of uniform B-spline curves that these refinement points converge in the limit to the curve itself.

---

## Summary

In the case of cubic uniform B-spline curves we can consider the refinement procedure to consist of the generation of edge points and vertex points from a given set of control points – each of which can be generated by taking only the midpoints of line segments. This classification of the control points will become very useful in the direct calculation of points on the limit curve.

---

## References

- [1] CATMULL, E., AND CLARK, J. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design* 10 (Sept. 1978), 350–355.

## On-Line Geometric Modeling Notes

### DEVELOPING A MATRIX EQUATION FOR REFINEMENT

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

#### Overview

The basic operation in the development of subdivision curves is the refinement procedure. In the cubic case, this can be written as

$$\begin{bmatrix} \mathbf{P}_0^1 \\ \mathbf{P}_1^1 \\ \mathbf{P}_2^1 \\ \mathbf{P}_3^1 \\ \mathbf{P}_4^1 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}$$

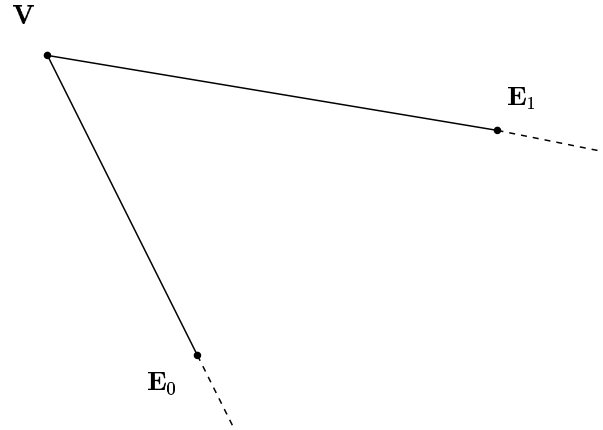
where  $\{\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3\}$  is the original control polygon and  $\{\mathbf{P}_0^1, \mathbf{P}_1^1, \mathbf{P}_2^1, \mathbf{P}_3^1, \mathbf{P}_4^1\}$  is the result of the refinement.

In this case, we can also classify points of the refinement as vertex points and edge points, and this enables us to look at the problem from another point of view. Here we examine what happens to a particular control point (vertex point) under this refinement procedure. Examining this closely, we can develop an alternate matrix equation that can also be used to represent the refinement procedure.

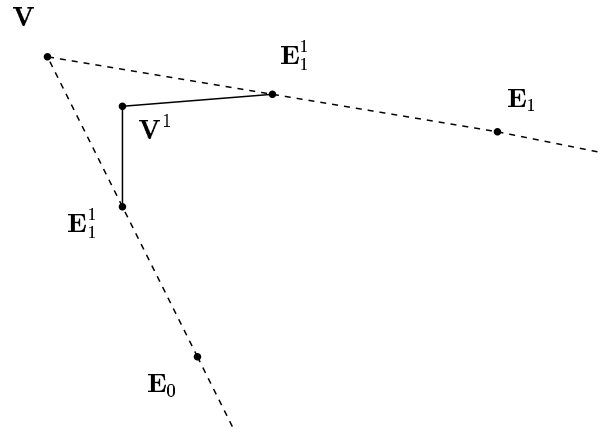
---

#### Refinement at a Vertex Point

The general idea in this development is to focus on a single vertex point of the control polygon and the two points immediately adjacent to this vertex. Call this vertex  $\mathbf{V}$  and the two points  $\mathbf{E}_0$  and  $\mathbf{E}_1$  – as in the following figure.



In the refinement procedure, we note that vertex points and edge points alternate in the refined control polygon. We will use these three points to create a new vertex point  $V^1$  and two new edge points  $E_0^1$  and  $E_1^1$ .



We can write this in vector form (we note that the matrices are applied to vectors of points, and therefore the operations must be affine). Writing this in vector form

$$\begin{bmatrix} V \\ E_0 \\ E_1 \end{bmatrix} \text{ is refined into } \begin{bmatrix} V^1 \\ E_0^1 \\ E_1^1 \end{bmatrix}$$

and we can calculate this refinement by using the equations to calculate the vertex and edge points – obtain-



ing

$$\begin{aligned}
\begin{bmatrix} \mathbf{V}^1 \\ \mathbf{E}_0^1 \\ \mathbf{E}_1^1 \end{bmatrix} &= \begin{bmatrix} \frac{1}{4} \left( \frac{\mathbf{V} + \mathbf{E}_0}{2} + 2\mathbf{V} + \frac{\mathbf{V} + \mathbf{E}_1}{2} \right) \\ \frac{\mathbf{V} + \mathbf{E}_0}{2} \\ \frac{\mathbf{V} + \mathbf{E}_1}{2} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{8} (\mathbf{E}_0 + 6\mathbf{V} + \mathbf{E}_1) \\ \frac{\mathbf{V} + \mathbf{E}_0}{2} \\ \frac{\mathbf{V} + \mathbf{E}_1}{2} \end{bmatrix} \\
&= \frac{1}{8} \begin{bmatrix} 6 & 1 & 1 \\ 4 & 4 & 0 \\ 4 & 0 & 4 \end{bmatrix} \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix}
\end{aligned}$$

and so this refinement process can be implemented as a matrix multiplication. We usually call the matrix

$$A = \frac{1}{8} \begin{bmatrix} 6 & 1 & 1 \\ 4 & 4 & 0 \\ 4 & 0 & 4 \end{bmatrix}$$

the *refinement matrix*.

---

### So How Do You Do Refinement In General?

Given a control polygon  $\{\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n\}$ , we utilize the vectors

$$\begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_0 \\ \mathbf{P}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{P}_2 \\ \mathbf{P}_1 \\ \mathbf{P}_3 \end{bmatrix}, \dots, \begin{bmatrix} \mathbf{P}_{n-1} \\ \mathbf{P}_{n-2} \\ \mathbf{P}_n \end{bmatrix}$$

and apply this refinement matrix to each vector, giving new vertex and edge points.

$$\begin{bmatrix} \mathbf{V}_0 \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix}, \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{E}_1 \\ \mathbf{E}_2 \end{bmatrix}, \dots, \begin{bmatrix} \mathbf{V}_{n-2} \\ \mathbf{E}_{n-2} \\ \mathbf{E}_{n-1} \end{bmatrix}$$

These new points are then assembled into new control polygon

$$\{\mathbf{E}_0, \mathbf{V}_0, \mathbf{E}_1, \mathbf{V}_1, \dots, \mathbf{V}_{n-3}, \mathbf{E}_{n-2}, \mathbf{V}_{n-2}, \mathbf{E}_{n-1}\}$$

which forms the same refinement as with the subdivision method.

---

## Summary

It is possible to write the refinement process for cubic subdivision curves in a matrix form which focuses on the action of the refinement on a single control point. In this case, we can calculate new vertex points and edges points just by applying the refinement matrix to the vertex-edge-point vector.

We note that this procedure does take more processor time than the refinement based upon binary subdivision. However, it will enable us to analyze the refinement operation further and generate a procedure to directly calculate a point on the curve without resulting to refinement.

---

## References

- [1] HALSTEAD, M., KASS, M., AND DEROSE, T. Efficient, fair interpolation using Catmull-Clark surfaces. In *Computer Graphics (SIGGRAPH '93 Proceedings)* (Aug. 1993), J. T. Kajiya, Ed., vol. 27, pp. 35–44.

---

All contents copyright (c) 1996, 1997, 1998, 1999, 2000  
Computer Science Department, University of California, Davis  
All rights reserved.

## On-Line Geometric Modeling Notes

# EIGENVALUES AND EIGENVECTORS

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

In engineering applications, eigenvalue problems are among the most important problems connected with matrices. In this section we give the basic definitions of eigenvalues and eigenvectors and some of the basic results of their use.

---

### What are Eigenvalues and Eigenvectors?

Let  $A$  be an  $n \times n$  matrix and consider the vector equation

$$A\vec{v} = \lambda\vec{v}$$

where  $\lambda$  is a scalar value.

It is clear that if  $\vec{v} = \vec{0}$ , we have a solution for any value of  $\lambda$ . A value of  $\lambda$  for which the equation has a solution with  $\vec{v} \neq \vec{0}$  is called an *eigenvalue* or *characteristic value* of the matrix  $A$ . The corresponding solutions  $\vec{v} \neq \vec{0}$  are called *eigenvectors* or *characteristic vectors* of  $A$ . In the problem above, we are looking for vectors that when multiplied by the matrix  $A$ , give a scalar multiple of itself.

The set of eigenvalues of  $A$  is commonly called the *spectrum* of  $A$  and the largest of the absolute values of the eigenvalues is called the *spectral radius* of  $A$ .

---

### How Do We Calculate the Eigenvalues?

It is easy to see that the equation

$$A\vec{v} = \lambda\vec{v}$$

can be rewritten as

$$(A - \lambda I)\vec{v} = 0$$

where  $I$  is the identity matrix. A matrix equation of this form can only be solved if the determinant of the matrix is nonzero (see Cramer's Rule) – that is, if

$$\det(A - \lambda I) = 0$$

Since this equation is a polynomial in  $\lambda$ , commonly called the *characteristic polynomial*, we only need to find the roots of this polynomial to find the eigenvalues.

We note that to get a complete set of eigenvalues, one may have to extend the scope of this discussion into the field of complex numbers.

---

### How Do We Calculate the Eigenvectors?

The eigenvalues must be determined first. Once these are known, the corresponding eigenvectors can be calculated directly from the linear system

$$(A - \lambda I)\vec{v} = 0$$

It should be noted that if  $\vec{v}$  is an eigenvector, then so is  $k\vec{v}$  for any scalar  $k$ .

---

### Right Eigenvectors

Given an eigenvalue  $\lambda$ , The eigenvector  $\vec{r}$  that satisfies

$$A\vec{r} = \lambda\vec{r}$$

is sometimes called a (right) *eigenvector* for the matrix  $A$  corresponding to the eigenvalue  $\lambda$ . If  $\lambda_1, \lambda_2, \dots, \lambda_r$  are the eigenvalues and  $\vec{r}_1, \vec{r}_2, \dots, \vec{r}_r$  are the corresponding right eigenvectors, then is easy to see that the set of right eigenvectors form a basis of a vector space. If this vector space is of dimension  $n$ , then we can construct an  $n \times n$  matrix  $R$  whose columns are the components of the right eigenvectors, which has the

property that

$$AR = R\Lambda$$

where  $\Lambda$  is the diagonal matrix

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \lambda_n \end{bmatrix}$$

whose diagonal elements are the eigenvalues. By appropriate numbering of the eigenvalues and eigenvectors, it is possible to arrange the columns of the matrix  $R$  so that  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ .

### Left Eigenvectors

A vector  $\vec{l}$  so that

$$\vec{l}^T A = \lambda \vec{l}^T$$

is called a left eigenvector for  $A$  corresponding to the eigenvalue  $\lambda$ . If  $\lambda_1, \lambda_2, \dots, \lambda_r$  are the eigenvalues and  $\vec{l}_1, \vec{l}_2, \dots, \vec{l}_r$  are the corresponding left eigenvectors, then it is easy to see that the set of left eigenvectors form a basis of a vector space. If this vector space is of dimension  $n$ , then we can construct an  $n \times n$  matrix  $L$  whose rows are the components of the left eigenvectors, which has the property that

$$LA = \Lambda L$$

It is possible to choose the left eigenvectors  $\vec{l}_1, \vec{l}_2, \dots$  and right eigenvectors  $\vec{r}_1, \vec{r}_2, \dots$  so that

$$\vec{l}_i \cdot \vec{r}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

This is easily done if we define  $L = R^{-1}$  and define the components of the left eigenvectors to be the

elements of the respective rows of  $L$ . Beginning with  $AR = R\Lambda$  and multiplying both sides on the left by  $R^{-1}$ , we obtain

$$R^{-1}AR = \Lambda$$

and multiplying on the right by  $R^{-1}$ , we have

$$R^{-1}A = \Lambda R^{-1}$$

which implies that any row of  $R^{-1}$  satisfies the properties of a left eigenvector.

### Diagonalization of a Matrix

Given an  $n \times n$  matrix  $A$ , we say that  $A$  is diagonalizable if there is a matrix  $X$  so that

$$X^{-1}AX = \Lambda$$

where

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \lambda_n \end{bmatrix}$$

It is clear from the above discussions that if all the eigenvalues are real and distinct, then we can use the matrix of right eigenvectors  $R$  as  $X$ .

### References

**All contents copyright (c) 1996, 1997, 1998, 1999, 2000  
Computer Science Department, University of California, Davis  
All rights reserved.**

## On-Line Geometric Modeling Notes

# EIGENVALUES AND EIGENVECTORS OF THE REFINEMENT MATRIX

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

### Overview

The basic operation in the development of subdivision curves is the refinement procedure. In the cubic case, the points of the refinement can be specified as vertex and edge points and the refinement procedure can be specified by a matrix operation. In this case the refinement matrix  $A$  is defined to be

$$A = \frac{1}{8} \begin{bmatrix} 6 & 1 & 1 \\ 4 & 4 & 0 \\ 4 & 0 & 4 \end{bmatrix}$$

In these notes, we develop the eigenvalues and eigenvectors of the refinement matrix which play an important role in the analysis of subdivision curves. This matrix is also diagonalizable and we use the eigenvalues and eigenvectors to calculate this diagonal decomposition. Finally, the diagonal decomposition allows an easy calculation of the inverse of the matrix.

---

### The Eigenvalues of the Refinement Matrix

One of the eigenvalues is easy to calculate. Since the rows of  $A$  all sum to one, we have

$$A \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$



and so 1 is an eigenvalue of  $A$ , with corresponding eigenvector  $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ . The other eigenvalues can be calculated from the characteristic polynomial and turn out to be  $\frac{1}{2}$  and  $\frac{1}{4}$ .

The (right) eigenvectors for the eigenvalues  $1, \frac{1}{2}, \frac{1}{4}$  can be calculated to be

$$\vec{r}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \vec{r}_2 = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, \vec{r}_3 = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}$$

respectively, and in a similar fashion the left eigenvectors turn out to be

$$\begin{aligned} \vec{l}_1 &= \begin{bmatrix} \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \end{bmatrix} \\ \vec{l}_2 &= \begin{bmatrix} 0 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \\ \vec{l}_3 &= \begin{bmatrix} -\frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{bmatrix} \end{aligned}$$

---

### Diagonalization of the Matrix

We will let  $L$  be the  $3 \times 3$  matrix whose rows are the left eigenvectors of  $A$ ,

$$L = \begin{bmatrix} \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \\ 0 & \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}$$

and  $R$  be the  $3 \times 3$  matrix whose columns are the right eigenvectors of  $A$ ,

$$R = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 1 & 2 \\ 1 & -1 & 2 \end{bmatrix}$$

noting that  $R = L^{-1}$ . Then since the  $3 \times 3$  matrix  $A$  has 3 distinct eigenvalues, it is diagonalizable [1] and can be written as

$$A = R\Lambda L$$

where  $\Lambda$  is the diagonal matrix whose diagonal elements are the eigenvalues of  $A$ .

$$\Lambda = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix}$$


---

### The Inverse of the Refinement Matrix

Since the refinement matrix can be written as  $A = R\Lambda L$ , it is easy to generate the inverse of  $A$ .

$$\begin{aligned} A^{-1} &= L^{-1}\Lambda^{-1}R^{-1} \\ &= R\Lambda^{-1}L \end{aligned}$$

since  $R = L^{-1}$ . The matrix  $\Lambda^{-1}$  is just

$$\Lambda^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

The inverse of  $A$  then works out to be

$$A^{-1} = \frac{1}{2} \begin{bmatrix} 4 & -1 & -1 \\ -4 & 5 & 1 \\ -4 & 1 & 5 \end{bmatrix}$$


---

### Summary

The eigenvalues and eigenvectors of the refinement matrix are straightforward to calculate. Since this matrix is well conditioned it can be diagonalized and written in form  $R\Lambda L$  which will be very useful when analyzing subdivision curves.

---

## References

- [1] GOLUB, G. H., AND VAN LOAN, C. F. *Matrix Computations*, 2nd ed. Johns Hopkins University Press, 1989.
- [2] HALSTEAD, M., KASS, M., AND DEROSE, T. Efficient, fair interpolation using Catmull-Clark surfaces. In *Computer Graphics (SIGGRAPH '93 Proceedings)* (Aug. 1993), J. T. Kajiya, Ed., vol. 27, pp. 35–44.

---

**All contents copyright (c) 1996, 1997, 1998, 1999, 2000**  
**Computer Science Department, University of California, Davis**  
**All rights reserved.**

## On-Line Geometric Modeling Notes

# **DIRECT CALCULATION OF POINTS ON CUBIC SUBDIVISION CURVES**

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

Given an initial control polygon we can define a refinement process that generates a sequence of control polygons from the original. In the limit, this sequence converges to the uniform B-spline curve specified by the original control polygon. By examining this refinement process from a different angle, we can specify a procedure that allows us to directly calculate points on the curve.

---

### **Direct Calculation of Points on the Curve**

In the cubic case, we define the general refinement procedure by classifying the points of a refinement as either “vertex” or “edge” points. Using this classification we can cleverly write this refinement process in a matrix form. This form takes a control point and its two adjacent control points of a refinement and applies a refinement matrix as follows

$$\begin{bmatrix} \mathbf{V}^1 \\ \mathbf{E}_0^1 \\ \mathbf{E}_1^1 \end{bmatrix} = A \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix}$$

where  $A$  is called the refinement matrix and is given by

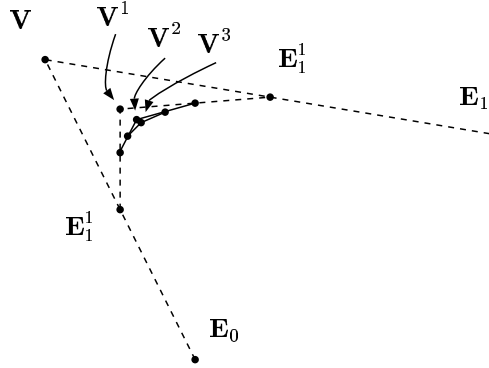
$$A = \frac{1}{8} \begin{bmatrix} 6 & 1 & 1 \\ 4 & 4 & 0 \\ 4 & 0 & 4 \end{bmatrix}$$

Here we have denoted the control points as vertex and edge points. This procedure creates two new edge points and a vertex point which are part of a new control polygon that represents the curve. We can apply  $A$

again and obtain

$$\begin{bmatrix} \mathbf{V}^2 \\ \mathbf{E}_0^2 \\ \mathbf{E}_1^2 \end{bmatrix} = A \begin{bmatrix} \mathbf{V}^1 \\ \mathbf{E}_0^1 \\ \mathbf{E}_1^1 \end{bmatrix} = A^2 \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix}$$

where  $A^2 = AA$ . In general, we can generate points on the  $k$ th refinement by applying  $A$   $k$ -times. In the limit, as  $k$  approaches infinity, we obtain a point on the curve.



We call this point  $\mathbf{V}^\infty$  and note that it is equal to  $\lim_{k \rightarrow \infty} \mathbf{V}^k$ .

---

### Calculating the Limit Point

The surprising thing is that we can actually calculate this limit. We utilize the fact that the refinement matrix can be diagonalized, which defines the matrix  $A$  by

$$A = R\Lambda L$$

where

$$R = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 1 & 2 \\ 1 & -1 & 2 \end{bmatrix}$$

is the matrix whose columns are the right eigenvectors of  $A$ ,

$$L = \begin{bmatrix} \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \\ 0 & \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}$$

is the matrix whose rows are the left eigenvectors of  $A$ , and  $\Lambda$  is the matrix of eigenvalues

$$\Lambda = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix}$$

Since  $R = L^{-1}$ , this allows us to write

$$A^2 = (R\Lambda L)^2 = R\Lambda L R\Lambda L = R\Lambda\Lambda L = R\Lambda^2 L$$

(Noting that  $R = L^{-1}$ ) or, in general

$$A^k = R\Lambda^k L$$

To calculate the limit, first consider applying  $A$   $k$ -times

$$\begin{aligned}
\begin{bmatrix} \mathbf{V}^k \\ \mathbf{E}_0^k \\ \mathbf{E}_1^k \end{bmatrix} &= A^k \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix} \\
&= (R\Lambda L)^k \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix} \\
&= R\Lambda^k L \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix} \\
&= R \begin{bmatrix} (1)^k & 0 & 0 \\ 0 & (\frac{1}{2})^k & 0 \\ 0 & 0 & (\frac{1}{4})^k \end{bmatrix} L \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix}
\end{aligned}$$

Now as  $k \rightarrow \infty$ , this approaches

$$R \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} L \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix}$$

and by substituting for  $R$  and  $L$ , this becomes

$$\begin{aligned}
R \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} L \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & -1 \\ 1 & 1 & 2 \\ 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \\ 0 & \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{bmatrix} \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \\ 0 & \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{bmatrix} \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix} \\
&= \begin{bmatrix} \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \\ \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \\ \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \end{bmatrix} \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix} \\
&= \frac{1}{6} \begin{bmatrix} 4 & 1 & 1 \\ 4 & 1 & 1 \\ 4 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix}
\end{aligned}$$

That is, the vertex and edge points all converge to the same value, which is just the dot product of the left eigenvector of  $A$  that corresponds to the eigenvalue one, and the original vertex-edge vector of the refinement – obtaining the point  $\frac{1}{6}(4\mathbf{V} + \mathbf{E}_0 + \mathbf{E}_1)$ .

In short, given any vertex  $\mathbf{V}_i$  with corresponding edge points  $\mathbf{E}_i$  and  $\mathbf{E}_{i+1}$ , we can directly calculate points on the curve by dotting the left eigenvector that corresponds to the eigenvalue 1 by the vertex-edge vector

$$\frac{1}{6} \begin{bmatrix} 4 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{V}_i \\ \mathbf{E}_i \\ \mathbf{E}_{i+1} \end{bmatrix}$$

This says that any vertex point on any refinement directly corresponds to a point on the curve.

---

### Examples

Consider a cubic uniform B-spline curve with control polygon  $\{\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3\}$ . This curve can be



written as

$$\mathbf{P}(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}$$

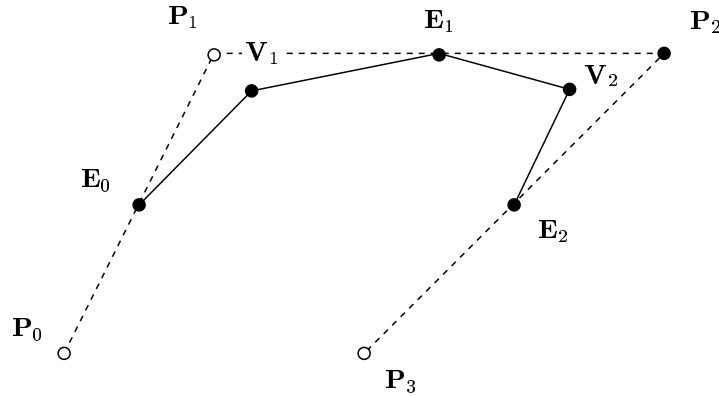
Consider the first step of the refinement using  $\mathbf{P}_1$  as the vertex with  $\mathbf{P}_0$  and  $\mathbf{P}_2$  as the two respective edge points. The corresponding point on the curve can be calculated directly as

$$\frac{1}{6} \begin{bmatrix} 4 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_0 \\ \mathbf{P}_2 \end{bmatrix} = \frac{1}{6} (\mathbf{P}_0 + 4\mathbf{P}_1 + \mathbf{P}_2)$$

which is exactly the point  $\mathbf{P}(0)$  on the curve.

Similarly we can calculate a point on the curve using the vertex  $\mathbf{P}_2$  with  $\mathbf{P}_1$  and  $\mathbf{P}_3$  as the two respective edge points. In this case, we find that this is exactly the point  $\mathbf{P}(1)$  on the curve.

Carrying this one step further, suppose we generate one full refinement of the curve, generating new edge and vertex points.



These new points become the input to the refinement process, and if we consider  $\mathbf{E}_1$  as the vertex point,

with  $\mathbf{V}_1$  and  $\mathbf{V}_2$  as the respective edge points, we obtain the point on the curve

$$\begin{aligned} \begin{bmatrix} \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \end{bmatrix} \begin{bmatrix} \mathbf{E}_1 \\ \mathbf{V}_1 \\ \mathbf{V}_2 \end{bmatrix} &= \begin{bmatrix} \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \end{bmatrix} \begin{bmatrix} \frac{1}{2} (\mathbf{P}_1 + \mathbf{P}_2) \\ \frac{1}{8} (\mathbf{P}_0 + 6\mathbf{P}_1 + \mathbf{P}_2) \\ \frac{1}{8} (\mathbf{P}_1 + 6\mathbf{P}_2 + \mathbf{P}_3) \end{bmatrix} \\ &= \frac{1}{48} (\mathbf{P}_0 + 23\mathbf{P}_1 + 23\mathbf{P}_2 + \mathbf{P}_3) \end{aligned}$$

which is exactly  $\mathbf{P}(\frac{1}{2})$ .

---

## Summary

It is possible, using eigenanalysis, to formulate simple procedures that calculate directly a point on the limit curve. This, in many cases, can be used to replace the overall refinement process.

The tangent vector on the curve at this limit point can also be directly calculated, by much the same procedure.

---

## References

- [1] CATMULL, E., AND CLARK, J. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design* 10 (Sept. 1978), 350–355.
  - [2] HALSTEAD, M., KASS, M., AND DEROSE, T. Efficient, fair interpolation using Catmull-Clark surfaces. In *Computer Graphics (SIGGRAPH '93 Proceedings)* (Aug. 1993), J. T. Kajiya, Ed., vol. 27, pp. 35–44.
- 

All contents copyright (c) 1996, 1997, 1998, 1999, 2000  
Computer Science Department, University of California, Davis  
All rights reserved.

## On-Line Geometric Modeling Notes

# DIRECT CALCULATION OF THE TANGENT VECTOR ON CUBIC SUBDIVISION CURVES

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

### Overview

Given an initial control polygon we can define a refinement process that generates a sequence of control polygons from the original. In the limit, this sequence converges to a uniform B-spline curve. We can specify a procedure, based upon eigenanalysis of the refinement matrix, that allows us to exactly calculate a point on the limit curve.

We show in these notes, that by applying the eigenanalysis further we can also calculate directly the tangent vectors on the limit curve. In this case it is the eigenvector corresponding to the eigenvalue  $\frac{1}{2}$  that plays the primary role.

---

### Direct Calculation of Points on the Curve

Select a control point  $\mathbf{V}_i$  and let  $\mathbf{E}_i$  and  $\mathbf{E}_{i+1}$  be the adjacent control points (thought of as edge points of the refinement). Given the refinement matrix

$$A = \frac{1}{8} \begin{bmatrix} 6 & 1 & 1 \\ 4 & 4 & 0 \\ 4 & 0 & 4 \end{bmatrix}$$

we can show that repeatedly applying  $A$  to the vector

$$\begin{bmatrix} \mathbf{V}_i \\ \mathbf{E}_i \\ \mathbf{E}_{i+1} \end{bmatrix}$$

gives us a point  $V^\infty$  on the curve. This point can be directly calculated as the dot product of the left eigenvector of  $A$  that corresponds to the eigenvalue one, and the original vertex-edge vector of the refinement – obtaining the point  $\frac{1}{6}(4\mathbf{V} + \mathbf{E}_0 + \mathbf{E}_1)$ .

---

### Tangent Vectors to the Curve

We can also use this analysis to address the tangent vectors on the curve. Given a vertex point  $\mathbf{V}$  and the two adjacent edge points  $\mathbf{E}_0$  and  $\mathbf{E}_1$  respectively, we wish to look at the unit vectors

$$\lim_{k \leftarrow \infty} \frac{\mathbf{E}_0^k - \mathbf{V}^\infty}{\|\mathbf{E}_0^k - \mathbf{V}^\infty\|}$$

which is the limit of unit vectors formed from the limit point on the curve to the converging edge points – this should converge to the unit tangent vector at the point  $\mathbf{V}^\infty$ .

To calculate this quantity, we will use the diagonalization of the matrix  $A$  which states that

$$\begin{bmatrix} \mathbf{V}^k \\ \mathbf{E}_0^k \\ \mathbf{E}_1^k \end{bmatrix} = A^k \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix} = R\Lambda^k L \begin{bmatrix} \mathbf{V} \\ \mathbf{E}_0 \\ \mathbf{E}_1 \end{bmatrix}$$

Now, let  $\vec{l}_1, \vec{l}_2$  and  $\vec{l}_3$  be the left eigenvectors of the refinement matrix  $A$ , and  $\vec{r}_1, \vec{r}_2$  and  $\vec{r}_3$  be the row vectors of the matrix of right eigenvalues of the refinement matrix  $A$ . Then to calculate  $\mathbf{E}_0^k$  we have

$$\begin{aligned} \mathbf{E}_0^k &= \vec{r}_2 \cdot [\Lambda^k L \vec{\mathbf{P}}] \\ &= \vec{r}_2 \cdot \left( \Lambda^k \begin{bmatrix} \vec{l}_1 \cdot \vec{\mathbf{P}} \\ \vec{l}_2 \cdot \vec{\mathbf{P}} \\ \vec{l}_3 \cdot \vec{\mathbf{P}} \end{bmatrix} \right) \\ &= \vec{r}_2 \cdot \begin{bmatrix} \vec{l}_1 \cdot \vec{\mathbf{P}} \\ (\frac{1}{2})^k \vec{l}_2 \cdot \vec{\mathbf{P}} \\ (\frac{1}{4})^k \vec{l}_3 \cdot \vec{\mathbf{P}} \end{bmatrix} \\ &= \vec{l}_1 \cdot \vec{\mathbf{P}} + \left(\frac{1}{2}\right)^k \vec{l}_2 \cdot \vec{\mathbf{P}} + 2 \left(\frac{1}{4}\right)^k \vec{l}_3 \cdot \vec{\mathbf{P}} \end{aligned}$$

where we have used  $\vec{r}_2 = (1, 1, 2)$ . Since

$$\mathbf{V}^\infty = \vec{l}_1 \cdot \vec{\mathbf{P}}$$

we have

$$\begin{aligned} \lim_{k \rightarrow \infty} \frac{\mathbf{E}_0^k - \mathbf{V}^\infty}{\|\mathbf{E}_0^k - \mathbf{V}^\infty\|} &= \lim_{k \rightarrow \infty} \left( \frac{\left(\frac{1}{2}\right)^k \vec{l}_2 \cdot \vec{\mathbf{P}} + 2 \left(\frac{1}{4}\right)^k \vec{l}_3 \cdot \vec{\mathbf{P}}}{\left\| \left(\frac{1}{2}\right)^k \vec{l}_2 \cdot \vec{\mathbf{P}} + 2 \left(\frac{1}{4}\right)^k \vec{l}_3 \cdot \vec{\mathbf{P}} \right\|} \right) \\ &= \lim_{k \rightarrow \infty} \left( \frac{\vec{l}_2 \cdot \vec{\mathbf{P}} + 2 \left(\frac{1}{2}\right)^k \vec{l}_3 \cdot \vec{\mathbf{P}}}{\left\| \vec{l}_2 \cdot \vec{\mathbf{P}} + 2 \left(\frac{1}{2}\right)^k \vec{l}_3 \cdot \vec{\mathbf{P}} \right\|} \right) \\ &= \frac{\vec{l}_2 \cdot \vec{\mathbf{P}}}{\|\vec{l}_2 \cdot \vec{\mathbf{P}}\|} \\ &= \frac{\mathbf{E}_0 - \mathbf{E}_1}{\|\mathbf{E}_0 - \mathbf{E}_1\|} \end{aligned}$$

(note the division by  $\left(\frac{1}{2}\right)^k$  in the second step).

Therefore, it is the left eigenvector of  $A$  corresponding to the eigenvalue  $\frac{1}{2}$  which determines the tangent vector to the curve, and given any vertex  $\mathbf{V}$  with corresponding edge points  $\mathbf{E}_0$  and  $\mathbf{E}_1$ , we can directly calculate the tangent vector at the limit point  $\mathbf{V}^\infty$  dotting the left eigenvector that corresponds to the eigenvalue  $\frac{1}{2}$  by the vertex-edge vector

$$\frac{1}{2} \begin{bmatrix} 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} \mathbf{V}_i \\ \mathbf{E}_i \\ \mathbf{E}_{i+1} \end{bmatrix}$$

i.e. by subtracting  $\mathbf{E}_{i+1} - \mathbf{E}_i$

---

## Summary

It is possible, using eigenanalysis, to formulate simple procedures that calculate directly a point on the limit curve, and the tangent vector on the curve at this point. This makes this refinement procedure quite simple to use.

---

## References

- [1] CATMULL, E., AND CLARK, J. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design* 10 (Sept. 1978), 350–355.
- [2] HALSTEAD, M., KASS, M., AND DEROSE, T. Efficient, fair interpolation using Catmull-Clark surfaces. In *Computer Graphics (SIGGRAPH '93 Proceedings)* (Aug. 1993), J. T. Kajiya, Ed., vol. 27, pp. 35–44.

---

All contents copyright (c) 1996, 1997, 1998, 1999, 2000  
Computer Science Department, University of California, Davis  
All rights reserved.

## On-Line Geometric Modeling Notes

### SUBDIVISION SURFACES

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

Surface generation methods are an important topic in computer graphics and computer-aided geometric design research. Much of the important work to date has concentrated on surfaces based upon closed-form mathematical expressions (Bézier Curves, Spline Methods), but these methods are limited as to the number of surfaces and surface types that can be generated.

A new set of methods is now becoming popular which utilize a mesh of polygonal shapes, or a sequence of meshes, to describe a surface. By doing this, freedom from the closed-form mathematical expression is achieved, and a wide variety of surface types can be expressed. The surfaces are commonly called *subdivision* surfaces as they are based upon the binary subdivision of the uniform B-spline curve/surface. In general, they are defined by a initial polygonal mesh, along with a subdivision (or *refinement*) operation which, given a polygonal mesh, will generate a new mesh that has a greater number of polygonal elements, and is hopefully “closer” to some resulting surface. By repetitively applying the subdivision procedure to the initial mesh, we generate a sequence of meshes that (hopefully) converges to a resulting surface.

As it turns out, this is a well known process when the mesh has a “rectangular” structure and the subdivision procedure is an extension of binary subdivision for uniform B-spline surfaces. Therefore, we first present a somewhat extensive study of the uniform B-spline case, and then show how these results can be generalized to treat the case when the mesh is not based on a rectangular structure.

These algorithms are from a class called “corner cutting” algorithms – that is, their action can be described by cutting the corners off of polygonal meshes. To understand the basis of this method, the reader should first review the section on subdivision curves which includes Chaikin’s Algorithm – which can be pointed to as the first of these subdivision algorithms. The study of quadratic and cubic B-spline surfaces lead respectively to the two better known surface subdivision schemes, the Doo-Sabin and Catmull-Clark methods. We also outline a method by Charles Loop that is based upon a mesh with a triangular based structure.

- 
- What is a Subdivision/Refinement Process?
  - Subdivision methods for biquadratic uniform B-spline surfaces
  - Doo-Sabin Surfaces
  - Subdivision methods for bicubic uniform B-spline surfaces
  - Catmull-Clark Surfaces
  - Loop Surfaces
- 

**All contents copyright (c) 1996, 1997, 1998, 1999, 2000  
Computer Science Department, University of California, Davis  
All rights reserved.**



## On-Line Geometric Modeling Notes

### REFINEMENT

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

#### Overview

Bézier curves, B-spline curves and subdivision curves are all based upon the input of a control polygon and the specification of an algorithmic method that constructs a curve from this sequence of points. Fundamental to these methods is the concept of a *refinement*. These refinement methods, as defined mathematically, can be quite complex. However, in practice they are quite simple and usually easy to implement.

In these notes, we discuss the mathematical notion of refinement.

---

#### What is a Refinement Scheme

A *refinement* process is a scheme which defines a sequence of control polygons

$$\begin{aligned} & \mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n \\ & \mathbf{P}_0^1, \mathbf{P}_1^1, \dots, \mathbf{P}_{n_1}^1 \\ & \mathbf{P}_0^2, \mathbf{P}_1^2, \dots, \mathbf{P}_{n_2}^2 \\ & \vdots \\ & \mathbf{P}_0^k, \mathbf{P}_1^k, \dots, \mathbf{P}_{n_k}^k \\ & \vdots \end{aligned}$$

where for any  $k > 0$ , each  $\mathbf{P}_j^k$  can be written as

$$\mathbf{P}_j^k = \sum_{i=0}^{n_{k-1}} \alpha_{i,j,k} \mathbf{P}_i^{k-1}$$

That is, any element  $\mathbf{P}_j^k$  can be written as a linear combination of the control points  $\{\mathbf{P}_0^{k-1}, \mathbf{P}_1^{k-1}, \dots, \mathbf{P}_{n_{k-1}}^{k-1}\}$

from the control polygon generated in the prior step. For each fixed  $j$  and  $k$  the sequence  $\alpha_{i,j,k}$  is frequently called a *mask*.

This is a very general scheme, and quite complex to manage and analyze. It covers the cases where the number of control points in each successive polygon is allowed to increase (Chaikin's Curves and Doo-Sabin's subdivision surfaces are examples of this), or must decrease (de Casteljau's algorithm for generating Bézier Curves is an example of this).

It would be incredibly rare to use the entire set of control points from the  $k - 1$ st sequence to calculate each new control point in the  $k$ th sequence as there may be thousands of points to consider – and so in general we assume that most of the  $\alpha_{i,j,k}$ s are zero. To simplify things further, we frequently limit this to a *uniform scheme*, where the  $\alpha$ s are independent of the level of refinement ( $k$ ). This implies that the scheme is basically the same at each iteration of the refinement process. A further simplification, where the mask is the same for every point of a control polygon, is called a *stationary scheme*.

If all points that result from a refinement process lie on the lines joining the points of a control polygon, the process is typically called a “corner cutting scheme”. An example of such a scheme is the Chaikin's Curve.

---

## A Matrix Method for Refinement

The equation

$$\mathbf{P}_j^k = \sum_{i=0}^{n_{k-1}} \alpha_{i,j,k} \mathbf{P}_i^{k-1}$$

can be written in matrix form as

$$\mathbf{P}_j^k = \begin{bmatrix} \alpha_{0,j,k} & \alpha_{1,j,k} & \cdots & \alpha_{n_{k-1},j,k} \end{bmatrix} \begin{bmatrix} \mathbf{P}_0^{k-1} \\ \mathbf{P}_1^{k-1} \\ \vdots \\ \mathbf{P}_{n_k}^{k-1} \end{bmatrix}$$

and overall

$$\begin{bmatrix} \mathbf{P}_0^k \\ \mathbf{P}_1^k \\ \vdots \\ \mathbf{P}_{n_k}^k \end{bmatrix} = S_k \begin{bmatrix} \mathbf{P}_0^{k-1} \\ \mathbf{P}_1^{k-1} \\ \vdots \\ \mathbf{P}_{n_{k-1}}^{k-1} \end{bmatrix}$$

where  $S_k$  is the refinement matrix

$$S_k = \begin{bmatrix} \alpha_{0,0,k} & \alpha_{1,0,k} & \cdots & \alpha_{n_{k-1},0,k} \\ \alpha_{0,1,k} & \alpha_{1,1,k} & \cdots & \alpha_{n_{k-1},1,k} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{0,n_k,k} & \alpha_{1,n_k,k} & \cdots & \alpha_{n_{k-1},n_k,k} \end{bmatrix}$$

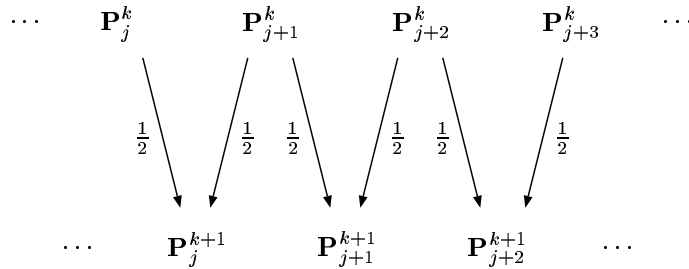
and is an  $(n_k + 1) \times (n_{k-1} + 1)$  matrix. In general it is best to think of this matrix as being sparse (i.e. most of the entries being zero) with non-zero entries clustered along the diagonal.

### Example – A Stationary Uniform Refinement Scheme

Suppose we are given the control polygon  $\{\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n\}$ . Define the refinement scheme by the following equation

$$\mathbf{P}_j^k = \frac{1}{2} (\mathbf{P}_j^{k-1} + \mathbf{P}_{j+1}^{k-1})$$

where  $0 \leq j \leq n - k$  and  $k = 0, 1, 2, \dots, n - 1$ . In other words, each successive point in the refinement is taken to be the midpoint of the line segment joining the two corresponding points in the previous control polygon.



Note here that two of the  $\alpha$ s are  $\frac{1}{2}$  and the remainder are zero.

In this case, the refinement process stops after  $n - 1$  steps – as the control polygon for each step of the refinement has one fewer points than does the control polygon in the previous step – the final control polygon having one point.

To represent this refinement process via matrices, the refinement matrix  $S_k$  is

$$S_k = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & \cdots & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

where the matrix is  $k \times (k + 1)$ .

If this refinement is taken to completion, we have just calculated a point on the  $n$ th degree Bézier curve defined by this control polygon.

---

### Example – A Non-Stationary Uniform Subdivision Scheme

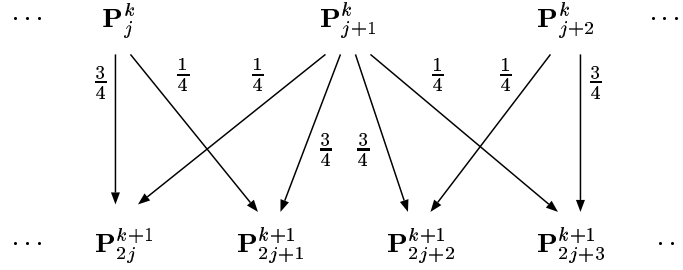
Suppose we are given the control polygon  $\{\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n\}$ . Define a refinement scheme by the following

$$\mathbf{P}_{2j}^k = \frac{3}{4}\mathbf{P}_j^k + \frac{1}{4}\mathbf{P}_{j+1}^k$$

and

$$\mathbf{P}_{2j+1}^k = \frac{1}{4}\mathbf{P}_j^k + \frac{3}{4}\mathbf{P}_{j+1}^k$$

for  $j = 0, 1, 2, 3, \dots$



Notice that this gives us a new control polygon

$$\begin{aligned}
 \mathbf{P}_0^1 &= \frac{3}{4}\mathbf{P}_0 + \frac{1}{4}\mathbf{P}_1 \\
 \mathbf{P}_1^1 &= \frac{1}{4}\mathbf{P}_0 + \frac{3}{4}\mathbf{P}_1 \\
 \mathbf{P}_2^1 &= \frac{3}{4}\mathbf{P}_1 + \frac{1}{4}\mathbf{P}_2 \\
 \mathbf{P}_3^1 &= \frac{1}{4}\mathbf{P}_1 + \frac{3}{4}\mathbf{P}_2 \\
 \mathbf{P}_4^1 &= \frac{3}{4}\mathbf{P}_2 + \frac{1}{4}\mathbf{P}_3 \\
 \mathbf{P}_5^1 &= \frac{1}{4}\mathbf{P}_2 + \frac{3}{4}\mathbf{P}_3 \\
 &\vdots
 \end{aligned}$$

Applying this refinement process to a control polygon of length  $n + 1$  gives a new control polygon of length  $2n$ .

This is just Chaikin's Algorithm for curve generation. As the algorithm proceeds the number of control points gets arbitrarily large, but converges to a unique curve.

---

### Refinement Schemes for Meshes

Similar methods (with much more notationally complex mathematics) exist for control meshes that result in surface generation algorithms. In general, the idea is the same – the refinement operation generates new control points from the control points of the previous mesh.

---

### Summary

Refinement schemes generate an important class of curve and surface drawing algorithms that are useful in geometric modeling. The schemes generate a sequence of control polygons in the two-dimensional case, or control meshes in the three dimensional case that can be used for curve generation. The methods are useful in the case of Bézier curves and Bézier patches as well as in the generation of subdivision curves and surfaces.

---

## References

- [1] DYN, N., GREGORY, J., AND LEVIN, D. Analysis of uniform binary subdivision schemes for curve design. *Constructive Approximation* 7, 2 (1991), 127–148.
  - [2] DYN, N., AND LEVIN, D. The subdivision experience. In *Curves and Surfaces II* (1991), A. L. H. P.J. Laurent and L. Schumaker, Eds., pp. 1–17.
  - [3] MICCHELLI, C. A., AND PRAUTZSCH, H. Uniform refinement of curves. *Linear Algebra and Its Applications* 114/115 (1989), 841–870.
- 

All contents copyright (c) 1996, 1997, 1998, 1999, 2000  
Computer Science Department, University of California, Davis  
All rights reserved.

## On-Line Geometric Modeling Notes

# BIQUADRATIC UNIFORM B-SPLINE SURFACE REFINEMENT

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

### Overview

Subdivision surfaces are based upon the binary subdivision of the uniform B-spline surface. In general, they are defined by a initial polygonal mesh, along with a subdivision (or *refinement*) operation which, given a polygonal mesh, will generate a new mesh that has a greater number of polygonal elements, and is hopefully “closer” to some resulting surface. By repetitively applying the subdivision procedure to the initial mesh, we generate a sequence of meshes that (hopefully) converges to a resulting surface.

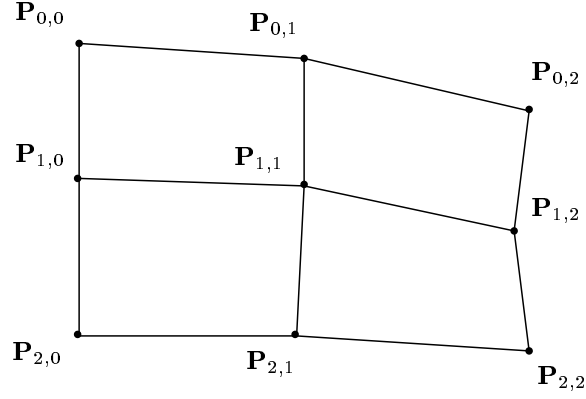
As it turns out, this is a well known process when the mesh has a “rectangular” structure and the subdivision procedure is an extension of binary subdivision for uniform B-spline surfaces. In these notes we present a study of the quadratic uniform B-spline case, and develop the refinement rules that can be generalized into the Doo-Sabin subdivision method.

---

### The Matrix Equation for a Uniform Biquadratic Spline Surface

Consider the biquadratic uniform B-spline surface  $\mathbf{P}(u, v)$  defined by the  $3 \times 3$  array of control points

$$P = \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} \end{bmatrix}$$



and the following equation (in matrix form)

$$\mathbf{P}(u, v) = \begin{bmatrix} 1 & u & u^2 \end{bmatrix} M P M^T \begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix}$$

where  $M$  is the  $3 \times 3$  matrix

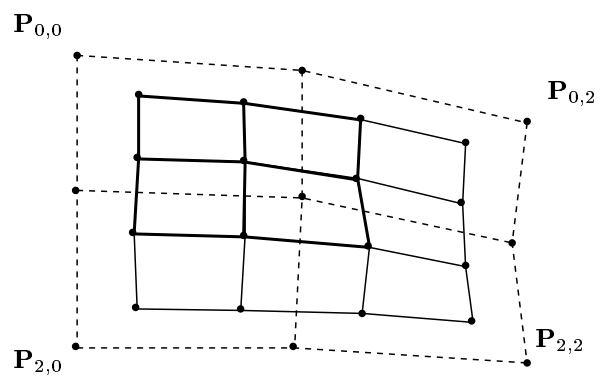
$$M = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix}$$

The matrix  $M$  defines the quadratic uniform B-spline blending functions when multiplied by  $\begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix}$ .

### Subdividing the Surface

This patch can be subdivided into four subpatches, which can be generated from 16 unique sub-control points. We focus on the subdivision scheme for only one of the four (the subpatch corresponding to  $0 \leq u, v \leq \frac{1}{2}$ ), as the others will follow by symmetry. The following figure illustrates the 16 points produced by subdividing into four subpatches. We have outlined the initial subpatch that we consider below. It should be noted that the four “interior” control points are utilized by each of the four subpatch components of the subdivision.





To subdivide the surface, we consider the reparameterization of the surface by  $u' = \frac{u}{2}$  and  $v' = \frac{v}{2}$  and

define this new surface as  $\mathbf{P}'(u, v)$ . Substituting these into the equation, we obtain

$$\begin{aligned}
\mathbf{P}'(u, v) &= \mathbf{P}\left(\frac{u}{2}, \frac{v}{2}\right) \\
&= \begin{bmatrix} 1 & \frac{u}{2} & \left(\frac{u}{2}\right)^2 \end{bmatrix} M P M^T \begin{bmatrix} 1 \\ \frac{v}{2} \\ \left(\frac{v}{2}\right)^2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & u & u^2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix} M P M^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix}^T \begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & u & u^2 \end{bmatrix} M M^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix} M P M^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix}^T (M^{-1})^T M^T \begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & u & u^2 \end{bmatrix} M \left( M^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix} M \right) P \left( M^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix} (M^{-1})^T \right) M^T \begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & u & u^2 \end{bmatrix} M \left( M^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix} M \right) P \left( M^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix} M \right)^T M^T \begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & u & u^2 \end{bmatrix} M P' M^T \begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix}
\end{aligned}$$

where  $P' = S P S^T$  and

$$S = M^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix} M$$

Through this process, we have written the surface  $\mathbf{P}'(u, v)$  as

$$\mathbf{P}'(u, v) = \begin{bmatrix} 1 & u & u^2 \end{bmatrix} M S M^T \begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix}$$

for some  $3 \times 3$  control point array  $S$ . This implies that  $\mathbf{P}'(u, v)$  is a uniform biquadratic B-spline patch. The matrix  $S$  is typically called the “splitting matrix”, and is given by

$$\begin{aligned} S &= \frac{1}{2} \begin{bmatrix} 2 & -1 & 0 \\ 2 & 1 & 0 \\ 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix} \\ &= \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \end{bmatrix} \end{aligned}$$

and so the control point mesh  $P'$  corresponding to the subdivided patch is related to the original control points mesh by

$$P' = S P S^T$$

By carrying out the algebra, we have that

$$\begin{aligned} P' &= \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} \end{bmatrix} \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \end{bmatrix}^T \\ &= \frac{1}{16} \begin{bmatrix} 3\mathbf{P}_{0,0} + \mathbf{P}_{1,0} & 3\mathbf{P}_{0,1} + \mathbf{P}_{1,1} & 3\mathbf{P}_{0,2} + \mathbf{P}_{1,2} \\ \mathbf{P}_{0,0} + 3\mathbf{P}_{1,0} & \mathbf{P}_{0,1} + 3\mathbf{P}_{1,1} & \mathbf{P}_{0,2} + 3\mathbf{P}_{1,2} \\ 3\mathbf{P}_{1,0} + \mathbf{P}_{2,0} & 3\mathbf{P}_{1,1} + \mathbf{P}_{2,1} & 3\mathbf{P}_{1,2} + \mathbf{P}_{2,2} \end{bmatrix} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 3 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \frac{1}{16} \begin{bmatrix} \mathbf{P}'_{0,0} & \mathbf{P}'_{0,1} & \mathbf{P}'_{0,2} \\ \mathbf{P}'_{1,0} & \mathbf{P}'_{1,1} & \mathbf{P}'_{1,2} \\ \mathbf{P}'_{2,0} & \mathbf{P}'_{2,1} & \mathbf{P}'_{2,2} \end{bmatrix} \end{aligned}$$

where the  $\mathbf{P}'_{i,j}$  can be written as

$$\begin{aligned}
\mathbf{P}'_{0,0} &= \frac{1}{16} (3(3\mathbf{P}_{0,0} + \mathbf{P}_{1,0}) + (3\mathbf{P}_{0,1} + \mathbf{P}_{1,1})) \\
\mathbf{P}'_{0,1} &= \frac{1}{16} ((3\mathbf{P}_{0,0} + \mathbf{P}_{1,0}) + 3(3\mathbf{P}_{0,1} + \mathbf{P}_{1,1})) \\
\mathbf{P}'_{0,2} &= \frac{1}{16} (3(3\mathbf{P}_{0,1} + \mathbf{P}_{1,1}) + (3\mathbf{P}_{0,2} + \mathbf{P}_{1,2})) \\
\mathbf{P}'_{1,0} &= \frac{1}{16} (3(\mathbf{P}_{0,0} + 3\mathbf{P}_{1,0}) + (\mathbf{P}_{0,1} + 3\mathbf{P}_{1,1})) \\
\mathbf{P}'_{1,1} &= \frac{1}{16} ((\mathbf{P}_{0,0} + 3\mathbf{P}_{1,0}) + 3(\mathbf{P}_{0,1} + 3\mathbf{P}_{1,1})) \\
\mathbf{P}'_{1,2} &= \frac{1}{16} (3(\mathbf{P}_{0,1} + 3\mathbf{P}_{1,1}) + (\mathbf{P}_{0,2} + 3\mathbf{P}_{1,2})) \\
\mathbf{P}'_{2,0} &= \frac{1}{16} (3(3\mathbf{P}_{1,0} + \mathbf{P}_{2,0}) + (3\mathbf{P}_{1,1} + \mathbf{P}_{2,1})) \\
\mathbf{P}'_{2,1} &= \frac{1}{16} ((3\mathbf{P}_{1,0} + \mathbf{P}_{2,0}) + 3(3\mathbf{P}_{1,1} + \mathbf{P}_{2,1})) \\
\mathbf{P}'_{2,2} &= \frac{1}{16} (3(3\mathbf{P}_{1,1} + \mathbf{P}_{2,1}) + (3\mathbf{P}_{1,2} + \mathbf{P}_{2,2}))
\end{aligned}$$

These equations can be looked at in two ways:

1. Each of these points  $\mathbf{P}_{i,j}$  utilizes the four points on a certain face of the rectangular mesh, and calculates a new point by weighing the four points in the ratio of 9-3-3-1. Thus, this algorithm can be specified by using *subdivision masks*, which specify the ratios of the points on a face to generate the new points. In this case, the subdivision masks are as follows

$$\begin{array}{cccc}
\begin{array}{ccc} 9 & \text{---} & 3 \\ | & & | \\ 3 & \text{---} & 1 \end{array} & 
\begin{array}{ccc} 3 & \text{---} & 1 \\ | & & | \\ 9 & \text{---} & 3 \end{array} & 
\begin{array}{ccc} 1 & \text{---} & 3 \\ | & & | \\ 3 & \text{---} & 9 \end{array} & 
\begin{array}{ccc} 3 & \text{---} & 9 \\ | & & | \\ 1 & \text{---} & 3 \end{array}
\end{array}$$

2. Each of these equations is built from weighing the points on an edge in the ratio of 3-1. and then weighing the resulting points in the ratio 3-1. These are exactly the ratios of Chaikin's curve and so this method can be looked upon as a extension of Chaikin's curve to surfaces.

---

## Generating the Refinement Procedure

To generate the subdivision surface, we consider all 16 of the possible points generated through the binary subdivision of the quadratic patch. It is easily seen that each of these points can be generated through considering other subdivisions of the patch  $P(u, v)$  and can be defined by the same subdivision masks

$$\begin{array}{cc} 9 & \text{---} & 3 \\ | & & | \\ 3 & \text{---} & 1 \end{array} \quad \begin{array}{cc} 3 & \text{---} & 1 \\ | & & | \\ 9 & \text{---} & 3 \end{array} \quad \begin{array}{cc} 1 & \text{---} & 3 \\ | & & | \\ 3 & \text{---} & 9 \end{array} \quad \begin{array}{cc} 3 & \text{---} & 9 \\ | & & | \\ 1 & \text{---} & 3 \end{array}$$

---

## Summary

The extension of Chaikin's Curve to surfaces is quite straightforward. The analysis has produced a simple mask that can be utilized to define the new points on each face (one per vertex). Connecting up these vertices into a new mesh is straightforward.

The interesting extension of this algorithm is when the control point mesh does not have a rectangular topological structure. In this case, we can still utilize the same paradigm and this was accomplished by Donald Doo and Malcolm Sabin in their Doo-Sabin surfaces

---

## References

- [1] CHAIKIN, G. An algorithm for high speed curve generation. *Computer Graphics and Image Processing* 3 (1974), 346–349.
- [2] DOO, D. A subdivision algorithm for smoothing down irregularly shaped polyhedrons. In *Proced. Int'l Conf. Interactive Techniques in Computer Aided Design* (1978), pp. 157–165. Bologna, Italy, IEEE Computer Soc.

## On-Line Geometric Modeling Notes

### DOO-SABIN SURFACES

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

Subdivision Surfaces utilize a mesh of polygonal shapes, or a sequence of meshes, to describe a surface. The surfaces are commonly called *subdivision* surfaces as they are based upon the binary subdivision of the uniform B-spline curve/surface. In general, they are defined by a initial polygonal mesh, along with a subdivision (or *refinement*) operation which, given a polygonal mesh, will generate a new mesh that has a greater number of polygonal elements, and is hopefully “closer” to some resulting surface. By repetitively applying the subdivision procedure to the initial mesh, we generate a sequence of meshes that (hopefully) converges to a resulting surface.

Donald Doo and Malcolm Sabin took the refinement paradigm developed by George Chaikin and, by adapting the refinement techniques for the bi-quadratic uniform B-spline surface were able to develop a new surface definition procedure based upon refinement. This study led to a simple procedure by which a surface could be developed via a polyhedral mesh of points.

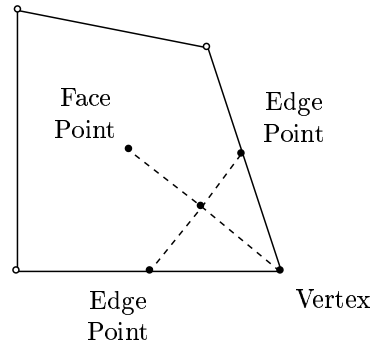
---

#### The Procedure for the Biquadratic Uniform B-Spline Patch

Given the subdivision masks generated for subdivision of biquadratic uniform B-spline patches

$$\begin{array}{cccc} \begin{array}{ccc} 9 & \text{---} & 3 \\ | & & | \\ 3 & \text{---} & 1 \end{array} & \begin{array}{ccc} 3 & \text{---} & 1 \\ | & & | \\ 9 & \text{---} & 3 \end{array} & \begin{array}{ccc} 1 & \text{---} & 3 \\ | & & | \\ 3 & \text{---} & 9 \end{array} & \begin{array}{ccc} 3 & \text{---} & 9 \\ | & & | \\ 1 & \text{---} & 3 \end{array} \end{array}$$

Doo and Sabin observed that the points are simply the average of four particular points taken in a polygon – the vertex for which the new point is being defined, the two *edge points* (the midpoints of the edges that are adjacent to this vertex in the polygon), and the *face point* (average of the vertices of the polygon). This can be seen in the following illustration:



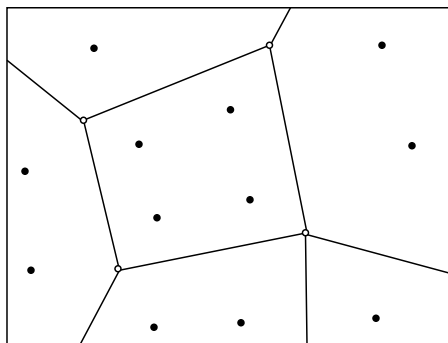
Thus the uniform B-spline case is easy, just generate these new points on each face. Each new face generated has four vertices, and the new points form a rectangular grid, from which we can again use this procedure to obtain new points on the faces, etc.

---

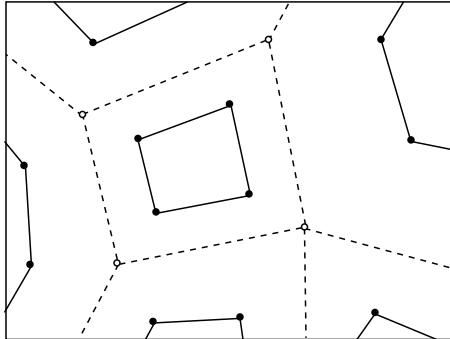
### The Procedure for Meshes of Arbitrary Topology

This procedure can be utilized to specify a refinement procedure for surfaces of arbitrary topologies. The rules of the refinement are as follows:

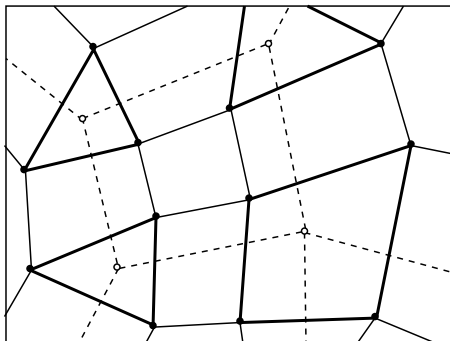
- For each vertex  $P_i$  of each face of the object, generate a new point  $P'_i$  as average of the vertex, the two edge points and the face point of the face.



- For each face, connect the new points that have been generated for each vertex of the face.

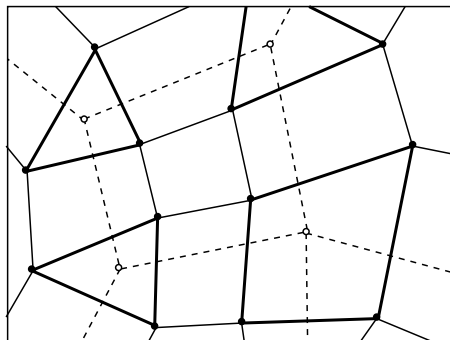


- For each vertex, connect the new points that have been generated for the faces that are adjacent to this vertex.



- For each edge, connect the new points that have been generated for the faces that are adjacent to this edge.

The polygons generated through this refinement step become the set of polygons for the next step.

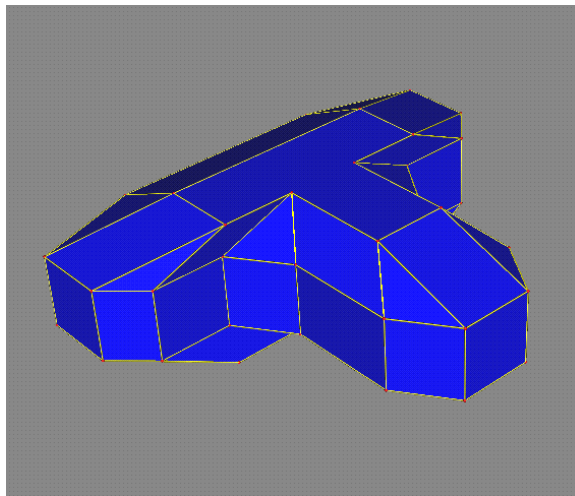
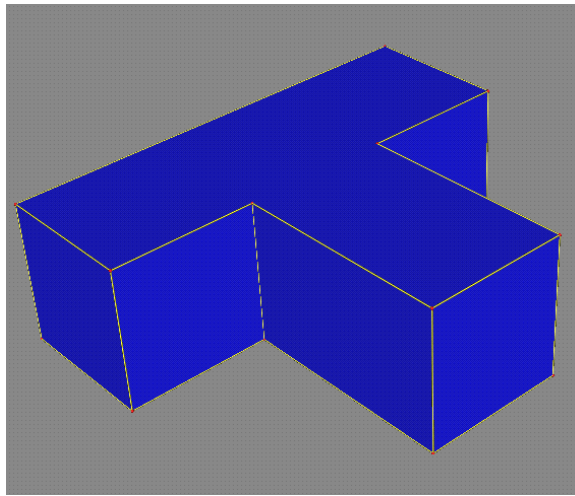


We note the appearance of the “cutting off the corners” of the polygonal mesh – from which these class of algorithms derives their name.

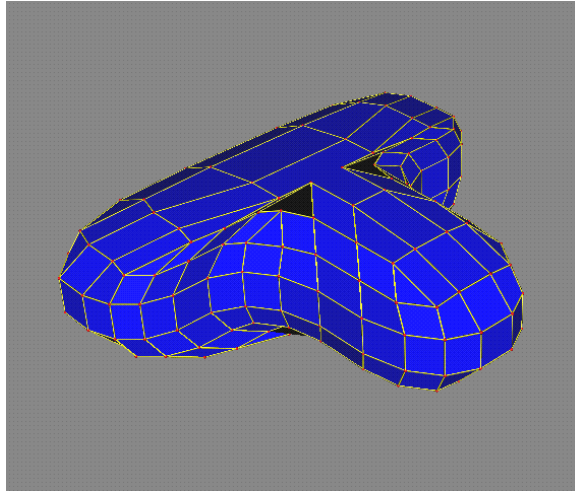


---

The following illustrations show three iterations of a Doo-Sabin bicycle seat. We note that locally these are equivalent to uniform quadratic B-spline surfaces, except at the corner points of the initial control point set (usually called *extraordinary points*).



We note that after the first subdivision, all vertices have *valence* four – which is a characteristic of the Doo-Sabin method. Note the triangular facets that arise in the corners. This develops because the original corner vertices have *valence* three (i.e. three edges are adjacent to the point). These triangular facets continue throughout the algorithm and converge to the “extraordinary points”.



We note that the last illustration is shaded poorly as the dark polygons are nonplanar and the rendering algorithm is assuming polygonal objects.

---

### Summary

Since the algorithm for refinement of bi-quadratic B-spline patches solves into a procedure that defines new points on each face in the refinement process, and these new points only depend on the face point, the vertex on a face and two edge midpoints for edges adjacent to the vertex, it is easily extended into an algorithm that works on a mesh with an arbitrary topology.

This Doo-Sabin surface is locally a bi-quadratic B-spline surface, except at a finite number of control points.

---

### References

- [1] DOO, D. A subdivision algorithm for smoothing down irregularly shaped polyhedrons. In *Proced. Int'l Conf. Interactive Techniques in Computer Aided Design* (1978), pp. 157–165. Bologna, Italy, IEEE Computer Soc.
- [2] DOO, D., AND SABIN, M. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design* 10 (Sept. 1978), 356–360.

---

**All contents copyright (c) 1996, 1997, 1998, 1999, 2000  
Computer Science Department, University of California, Davis  
All rights reserved.**

## On-Line Geometric Modeling Notes

# BICUBIC UNIFORM B-SPLINE SURFACE REFINEMENT

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

### Overview

Subdivision surfaces are based upon the binary subdivision of the uniform B-spline surface. In general, they are defined by a initial polygonal mesh, along with a subdivision (or *refinement*) operation which, given a polygonal mesh, will generate a new mesh that has a greater number of polygonal elements, and is hopefully “closer” to some resulting surface. By repetitively applying the subdivision procedure to the initial mesh, we generate a sequence of meshes that (hopefully) converges to a resulting surface. As it turns out, this is a well known process when the mesh has a “rectangular” structure and the subdivision procedure is an extension of binary subdivision for uniform B-spline surfaces.

Ed Catmull and Jim Clark, while still graduate students at the University of Utah, sought to extend Doo and Sabin’s algorithm to bicubic surfaces. Since the methods of Doo-Sabin is based upon the binary subdivision of the uniform bi-quadratic B-spline patches, Catmull and Clark believed that study of the cubic case would lead to a better subdivision surface generation scheme.

---

### A Matrix Equation for the Bicubic Uniform Spline Surfaces

Consider the bicubic uniform B-spline surface  $\mathbf{P}(u, v)$  defined by the  $4 \times 4$  array of control points

$$P = \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \mathbf{P}_{0,3} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \mathbf{P}_{1,3} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \mathbf{P}_{2,3} \\ \mathbf{P}_{3,0} & \mathbf{P}_{3,1} & \mathbf{P}_{3,2} & \mathbf{P}_{3,3} \end{bmatrix}$$

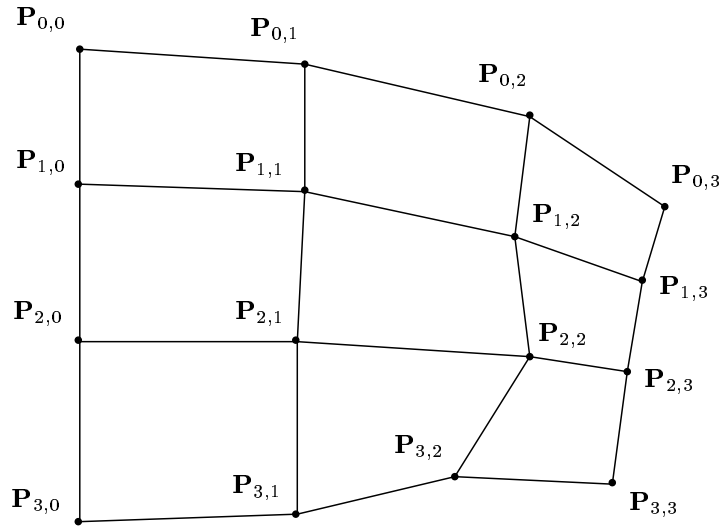
where

$$\mathbf{P}(u, v) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} M P M^T \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}$$

where  $M$  is the  $4 \times 4$  matrix

$$M = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

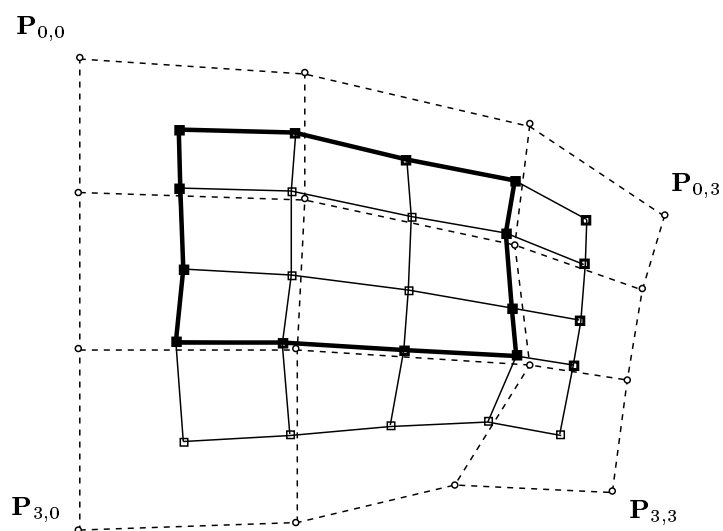
The control point mesh for such a patch is shown in the figure below.



### Subdividing the Bicubic Patch

The bicubic uniform B-spline patch can be subdivided into four subpatches, which can be generated from 25 unique sub-control points. We focus on the subdivision scheme for only one of the four (the subpatch corresponding to  $0 \leq u, v \leq \frac{1}{2}$ ), as the others will follow by symmetry. The following figure illustrates the 25 points produced by subdividing into four subpatches. We have outlined the initial subpatch that we

consider below. It should be noted that the nine “interior” control points are utilized by each of the four subpatch components of the subdivision.



This subpatch can be generated by reparameterizing the surface by the variables  $u'$  and  $v'$ , where  $u' = \frac{u}{2}$

and  $v' = \frac{v}{2}$ . Substituting these into the equation, we obtain

$$\begin{aligned}
\mathbf{P}(u', v') &= \mathbf{P}\left(\frac{u}{2}, \frac{v}{2}\right) \\
&= \begin{bmatrix} 1 & \frac{u}{2} & \left(\frac{u}{2}\right)^2 & \left(\frac{u}{2}\right)^3 \end{bmatrix} M P M^T \begin{bmatrix} 1 \\ \frac{v}{2} \\ \left(\frac{v}{2}\right)^2 \\ \left(\frac{v}{2}\right)^3 \end{bmatrix} \\
&= \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{8} \end{bmatrix} M P M^T \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{8} \end{bmatrix}^T \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix} \\
&= \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} M M^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{8} \end{bmatrix} M P M^T \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{8} \end{bmatrix}^T (M^{-1})^T M^T \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix} \\
&= \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} M S P S^T M^T \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix} \\
&= \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} M P' M^T \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}
\end{aligned}$$

where  $P' = S P S^T$  and

$$S = M^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{8} \end{bmatrix} M$$

Through this process, we have written the surface  $\mathbf{P}'(u, v)$  as

$$\mathbf{P}'(u, v) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} M P' M^T \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}$$

for some  $4 \times 4$  control point array  $P'$ . This implies that  $\mathbf{P}'(u, v)$  is a uniform bicubic B-spline patch. The matrix  $S$  is typically called the “splitting matrix”, and is straightforward to calculate. It is given by

$$H_1 = \frac{1}{8} \begin{bmatrix} 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \end{bmatrix}$$

By carrying out the algebra, we can calculate the control point array  $P'$  by

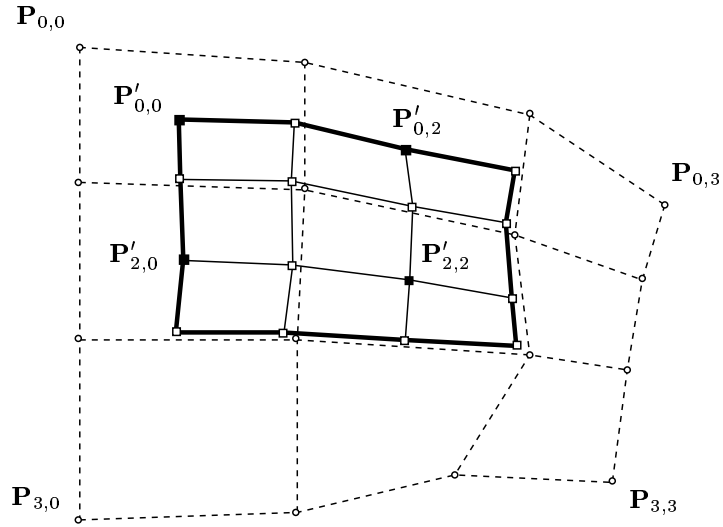
$$\begin{aligned} P' &= \frac{1}{8} \begin{bmatrix} 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \mathbf{P}_{0,3} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \mathbf{P}_{1,3} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \mathbf{P}_{2,3} \\ \mathbf{P}_{3,0} & \mathbf{P}_{3,1} & \mathbf{P}_{3,2} & \mathbf{P}_{3,3} \end{bmatrix} \frac{1}{8} \begin{bmatrix} 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \end{bmatrix}^T \\ &= \frac{1}{8} \begin{bmatrix} 4\mathbf{P}_{0,0} + 4\mathbf{P}_{1,0} & 4\mathbf{P}_{0,1} + 4\mathbf{P}_{1,1} & 4\mathbf{P}_{0,2} + 4\mathbf{P}_{1,2} & 4\mathbf{P}_{0,3} + 4\mathbf{P}_{1,3} \\ \mathbf{P}_{0,0} + 6\mathbf{P}_{1,0} + \mathbf{P}_{2,0} & \mathbf{P}_{0,1} + 6\mathbf{P}_{1,1} + \mathbf{P}_{2,1} & \mathbf{P}_{0,2} + 6\mathbf{P}_{1,2} + \mathbf{P}_{2,2} & \mathbf{P}_{0,3} + 6\mathbf{P}_{1,3} + \mathbf{P}_{2,3} \\ 4\mathbf{P}_{1,0} + 4\mathbf{P}_{2,0} & 4\mathbf{P}_{1,1} + 4\mathbf{P}_{2,1} & 4\mathbf{P}_{1,2} + 4\mathbf{P}_{2,2} & 4\mathbf{P}_{1,3} + 4\mathbf{P}_{2,3} \\ \mathbf{P}_{1,0} + 6\mathbf{P}_{2,0} + \mathbf{P}_{3,0} & \mathbf{P}_{1,1} + 6\mathbf{P}_{2,1} + \mathbf{P}_{3,1} & \mathbf{P}_{1,2} + 6\mathbf{P}_{2,2} + \mathbf{P}_{3,2} & \mathbf{P}_{1,3} + 6\mathbf{P}_{2,3} + \mathbf{P}_{3,3} \end{bmatrix} \frac{1}{8} \begin{bmatrix} 4 & 1 & 0 & 0 \\ 4 & 6 & 4 & 1 \\ 0 & 1 & 4 & 6 \\ 0 & 0 & 6 & 1 \end{bmatrix} \end{aligned}$$



and we obtain

$$\begin{aligned}
\mathbf{P}'_{0,0} &= \frac{\mathbf{P}_{0,0} + \mathbf{P}_{1,0} + \mathbf{P}_{0,1} + \mathbf{P}_{1,1}}{4} \\
\mathbf{P}'_{0,1} &= \frac{\mathbf{P}_{0,0} + \mathbf{P}_{1,0} + 6(\mathbf{P}_{0,1} + \mathbf{P}_{1,1}) + \mathbf{P}_{0,2} + \mathbf{P}_{1,2}}{16} \\
\mathbf{P}'_{0,2} &= \frac{\mathbf{P}_{0,1} + \mathbf{P}_{1,1} + \mathbf{P}_{0,2} + \mathbf{P}_{1,2}}{4} \\
\mathbf{P}'_{0,3} &= \frac{\mathbf{P}_{0,1} + \mathbf{P}_{1,1} + 6(\mathbf{P}_{0,2} + \mathbf{P}_{1,2}) + \mathbf{P}_{0,3} + \mathbf{P}_{1,3}}{16} \\
\mathbf{P}'_{1,0} &= \frac{\mathbf{P}_{0,0} + \mathbf{P}_{0,1} + 6(\mathbf{P}_{1,0} + \mathbf{P}_{1,1}) + \mathbf{P}_{2,0} + \mathbf{P}_{2,1}}{16} \\
\mathbf{P}'_{1,1} &= \frac{\mathbf{P}_{0,0} + 6\mathbf{P}_{1,0} + \mathbf{P}_{2,0} + 6(\mathbf{P}_{0,1} + 6\mathbf{P}_{1,1} + \mathbf{P}_{2,1}) + \mathbf{P}_{0,2} + 6\mathbf{P}_{1,2} + \mathbf{P}_{2,2}}{64} \\
\mathbf{P}'_{1,2} &= \frac{\mathbf{P}_{0,1} + \mathbf{P}_{0,2} + 6(\mathbf{P}_{1,1} + \mathbf{P}_{1,2}) + \mathbf{P}_{2,1} + \mathbf{P}_{2,2}}{16} \\
\mathbf{P}'_{1,3} &= \frac{\mathbf{P}_{0,1} + 6\mathbf{P}_{1,1} + \mathbf{P}_{2,1} + 6(\mathbf{P}_{0,2} + 6\mathbf{P}_{1,2} + \mathbf{P}_{2,2}) + \mathbf{P}_{0,3} + 6\mathbf{P}_{1,3} + \mathbf{P}_{2,3}}{64} \\
\mathbf{P}'_{2,0} &= \frac{\mathbf{P}_{1,0} + \mathbf{P}_{2,0} + \mathbf{P}_{1,1} + \mathbf{P}_{2,1}}{4} \\
\mathbf{P}'_{2,1} &= \frac{\mathbf{P}_{1,0} + \mathbf{P}_{2,0} + 6(\mathbf{P}_{1,1} + \mathbf{P}_{2,1}) + \mathbf{P}_{1,2} + \mathbf{P}_{2,2}}{16} \\
\mathbf{P}'_{2,2} &= \frac{\mathbf{P}_{1,1} + \mathbf{P}_{2,1} + \mathbf{P}_{1,2} + \mathbf{P}_{2,2}}{4} \\
\mathbf{P}'_{2,3} &= \frac{\mathbf{P}_{1,1} + \mathbf{P}_{2,1} + 6(\mathbf{P}_{1,2} + \mathbf{P}_{2,2}) + \mathbf{P}_{1,3} + \mathbf{P}_{2,3}}{16} \\
\mathbf{P}'_{3,0} &= \frac{\mathbf{P}_{1,0} + \mathbf{P}_{1,1} + 6(\mathbf{P}_{2,0} + \mathbf{P}_{2,1}) + \mathbf{P}_{3,0} + \mathbf{P}_{3,1}}{16} \\
\mathbf{P}'_{3,1} &= \frac{\mathbf{P}_{1,0} + 6\mathbf{P}_{2,0} + \mathbf{P}_{3,0} + 6(\mathbf{P}_{1,1} + 6\mathbf{P}_{2,1} + \mathbf{P}_{3,1}) + \mathbf{P}_{1,2} + 6\mathbf{P}_{2,2} + \mathbf{P}_{3,2}}{64} \\
\mathbf{P}'_{3,2} &= \frac{\mathbf{P}_{1,1} + \mathbf{P}_{1,2} + 6(\mathbf{P}_{2,1} + \mathbf{P}_{2,2}) + \mathbf{P}_{3,1} + \mathbf{P}_{3,2}}{16} \\
\mathbf{P}'_{3,3} &= \frac{\mathbf{P}_{1,1} + 6\mathbf{P}_{2,1} + \mathbf{P}_{3,1} + 6(\mathbf{P}_{1,2} + 6\mathbf{P}_{2,2} + \mathbf{P}_{3,2}) + \mathbf{P}_{1,3} + 6\mathbf{P}_{2,3} + \mathbf{P}_{3,3}}{64}
\end{aligned}$$

Each of these points can be classified into three categories – face points, edge points and vertex points – depending on each points relationship to the original control point mesh. The points  $\mathbf{P}'_{0,0}$ ,  $\mathbf{P}'_{0,2}$ ,  $\mathbf{P}'_{2,0}$  and  $\mathbf{P}'_{2,2}$ , which are shown in the figure below



are called “face” points, and are calculated as the average of the four points that bound the respective face. If we define the face point  $F_{i,j}$  to be the average of the points  $P_{i,j}$ ,  $P_{i+1,j}$ ,  $P_{i,j+1}$  and  $P_{i+1,j+1}$ , then we can

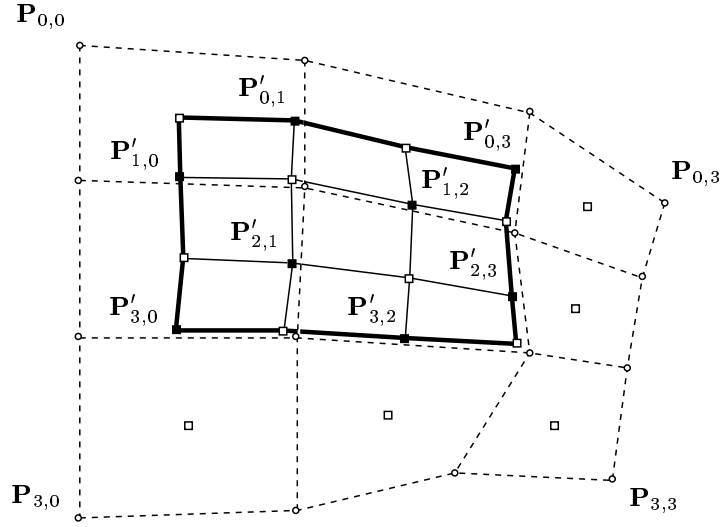
rewrite the above equations with these face points substituted on the right-hand side, and obtain

$$\begin{aligned}
\mathbf{P}'_{0,0} &= \mathbf{F}_{0,0} \\
\mathbf{P}'_{0,1} &= \frac{4\mathbf{F}_{0,0} + 4\mathbf{F}_{0,1} + 4\mathbf{P}_{0,1} + 4\mathbf{P}_{1,1}}{16} \\
\mathbf{P}'_{0,2} &= \mathbf{F}_{0,1} \\
\mathbf{P}'_{0,3} &= \frac{4\mathbf{F}_{0,1} + 4\mathbf{F}_{0,2} + 4\mathbf{P}_{0,2} + 4\mathbf{P}_{1,2}}{16} \\
\mathbf{P}'_{1,0} &= \frac{4\mathbf{F}_{0,0} + 4\mathbf{F}_{1,0} + 4\mathbf{P}_{1,0} + 4\mathbf{P}_{1,1}}{16} \\
\mathbf{P}'_{1,1} &= \frac{4\mathbf{F}_{0,0} + 4\mathbf{F}_{0,1} + 4\mathbf{F}_{1,0} + 4\mathbf{F}_{1,1} + 4\mathbf{P}_{1,0} + 4\mathbf{P}_{0,1} + 32\mathbf{P}_{1,1} + 4\mathbf{P}_{2,1} + 4\mathbf{P}_{1,2}}{64} \\
\mathbf{P}'_{1,2} &= \frac{4\mathbf{F}_{0,1} + 4\mathbf{F}_{1,1} + 4\mathbf{P}_{1,1} + 4\mathbf{P}_{1,2}}{16} \\
\mathbf{P}'_{1,3} &= \frac{4\mathbf{F}_{0,1} + 4\mathbf{F}_{0,2} + 4\mathbf{F}_{1,1} + 4\mathbf{F}_{1,2} + 4\mathbf{P}_{1,1} + 4\mathbf{P}_{0,2} + 32\mathbf{P}_{1,2} + 4\mathbf{P}_{2,2} + 4\mathbf{P}_{1,3}}{64} \\
\mathbf{P}'_{2,0} &= \mathbf{F}_{1,0} \\
\mathbf{P}'_{2,1} &= \frac{4\mathbf{F}_{1,0} + 4\mathbf{F}_{1,1} + 4\mathbf{P}_{1,1} + 4\mathbf{P}_{2,1}}{16} \\
\mathbf{P}'_{2,2} &= \mathbf{F}_{1,1} \\
\mathbf{P}'_{2,3} &= \frac{4\mathbf{F}_{1,1} + 4\mathbf{F}_{1,2} + 4\mathbf{P}_{1,2} + 4\mathbf{P}_{2,2}}{16} \\
\mathbf{P}'_{3,0} &= \frac{4\mathbf{F}_{1,0} + 4\mathbf{F}_{2,0} + 4\mathbf{P}_{2,0} + 4\mathbf{P}_{2,1}}{16} \\
\mathbf{P}'_{3,1} &= \frac{4\mathbf{F}_{1,0} + 4\mathbf{F}_{2,0} + 4\mathbf{F}_{1,1} + 4\mathbf{F}_{2,1} + 4\mathbf{P}_{2,0} + 4\mathbf{P}_{1,1} + 32\mathbf{P}_{2,1} + 4\mathbf{P}_{3,1} + 4\mathbf{P}_{2,2}}{64} \\
\mathbf{P}'_{3,2} &= \frac{4\mathbf{F}_{1,1} + 4\mathbf{F}_{2,1} + 4\mathbf{P}_{2,1} + 4\mathbf{P}_{2,2}}{16} \\
\mathbf{P}'_{3,3} &= \frac{4\mathbf{F}_{1,1} + 4\mathbf{F}_{2,1} + 4\mathbf{F}_{1,2} + 4\mathbf{F}_{2,2} + 4\mathbf{P}_{2,1} + 4\mathbf{P}_{1,2} + 32\mathbf{P}_{2,2} + 4\mathbf{P}_{3,2} + 4\mathbf{P}_{2,3}}{64}
\end{aligned}$$

Simplifying these equations, we obtain

$$\begin{aligned}
\mathbf{P}'_{0,0} &= \mathbf{F}_{0,0} \\
\mathbf{P}'_{0,1} &= \frac{\mathbf{F}_{0,0} + \mathbf{F}_{0,1} + \mathbf{P}_{0,1} + \mathbf{P}_{1,1}}{4} \\
\mathbf{P}'_{0,2} &= \mathbf{F}_{0,1} \\
\mathbf{P}'_{0,3} &= \frac{\mathbf{F}_{0,1} + \mathbf{F}_{0,2} + \mathbf{P}_{0,2} + \mathbf{P}_{1,2}}{4} \\
\mathbf{P}'_{1,0} &= \frac{\mathbf{F}_{0,0} + \mathbf{F}_{1,0} + \mathbf{P}_{1,0} + \mathbf{P}_{1,1}}{4} \\
\mathbf{P}'_{1,1} &= \frac{\mathbf{F}_{0,0} + \mathbf{F}_{0,1} + \mathbf{F}_{1,0} + \mathbf{F}_{1,1} + \mathbf{P}_{1,0} + \mathbf{P}_{0,1} + 8\mathbf{P}_{1,1} + \mathbf{P}_{2,1} + \mathbf{P}_{1,2}}{16} \\
\mathbf{P}'_{1,2} &= \frac{\mathbf{F}_{0,1} + \mathbf{F}_{1,1} + \mathbf{P}_{1,1} + \mathbf{P}_{1,2}}{4} \\
\mathbf{P}'_{1,3} &= \frac{\mathbf{F}_{0,1} + \mathbf{F}_{0,2} + \mathbf{F}_{1,1} + \mathbf{F}_{1,2} + \mathbf{P}_{1,1} + \mathbf{P}_{0,2} + 8\mathbf{P}_{1,2} + \mathbf{P}_{2,2} + \mathbf{P}_{1,3}}{16} \\
\mathbf{P}'_{2,0} &= \mathbf{F}_{1,0} \\
\mathbf{P}'_{2,1} &= \frac{\mathbf{F}_{1,0} + \mathbf{F}_{1,1} + \mathbf{P}_{1,1} + \mathbf{P}_{2,1}}{4} \\
\mathbf{P}'_{2,2} &= \mathbf{F}_{1,1} \\
\mathbf{P}'_{2,3} &= \frac{\mathbf{F}_{1,1} + \mathbf{F}_{1,2} + \mathbf{P}_{1,2} + \mathbf{P}_{2,2}}{4} \\
\mathbf{P}'_{3,0} &= \frac{\mathbf{F}_{1,0} + \mathbf{F}_{2,0} + \mathbf{P}_{2,0} + \mathbf{P}_{2,1}}{4} \\
\mathbf{P}'_{3,1} &= \frac{\mathbf{F}_{1,0} + \mathbf{F}_{2,0} + \mathbf{F}_{1,1} + \mathbf{F}_{2,1} + \mathbf{P}_{2,0} + \mathbf{P}_{1,1} + 8\mathbf{P}_{2,1} + \mathbf{P}_{3,1} + \mathbf{P}_{2,2}}{16} \\
\mathbf{P}'_{3,2} &= \frac{\mathbf{F}_{1,1} + \mathbf{F}_{2,1} + \mathbf{P}_{2,1} + \mathbf{P}_{2,2}}{4} \\
\mathbf{P}'_{3,3} &= \frac{\mathbf{F}_{1,1} + \mathbf{F}_{2,1} + \mathbf{F}_{1,2} + \mathbf{F}_{2,2} + \mathbf{P}_{2,1} + \mathbf{P}_{1,2} + 8\mathbf{P}_{2,2} + \mathbf{P}_{3,2} + \mathbf{P}_{2,3}}{16}
\end{aligned}$$

In examining these equations, we see that the points  $\mathbf{P}'_{0,1}$ ,  $\mathbf{P}'_{0,3}$ ,  $\mathbf{P}'_{1,0}$ ,  $\mathbf{P}'_{1,2}$ ,  $\mathbf{P}'_{2,1}$ ,  $\mathbf{P}'_{2,3}$ ,  $\mathbf{P}'_{3,0}$  and  $\mathbf{P}'_{3,2}$ , which are called “edge” points, are given as the average of four points – the two points that define the original edge and the two new face points of the faces sharing the edge. This is shown in the following figure.



These edge points  $\mathbf{E}_{i,j}$  can be calculated either as

$$\mathbf{E}_{i,j} = \frac{\mathbf{F}_{i,j-1} + \mathbf{F}_{i,j} + \mathbf{P}_{i,j} + \mathbf{P}_{i+1,j}}{4}$$

or

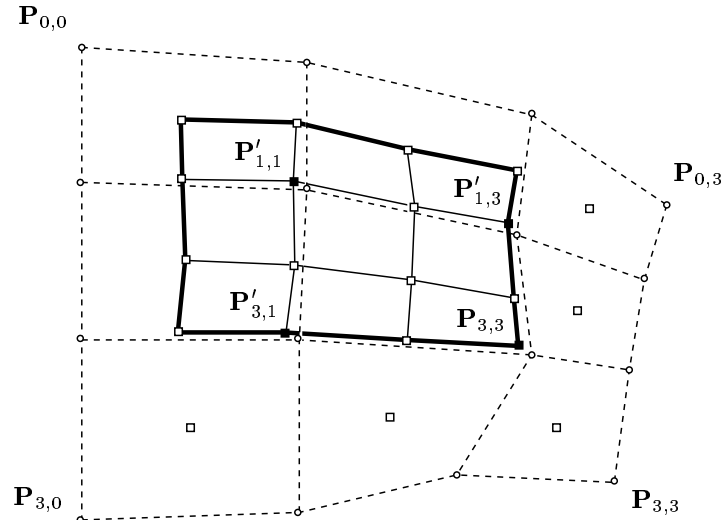
$$\mathbf{E}_{i,j} = \frac{\mathbf{F}_{i-1,j} + \mathbf{F}_{i,j} + \mathbf{P}_{i,j} + \mathbf{P}_{i,j+1}}{4}$$

depending on which side of the edge point the two faces lie. If we replace these edge points on the right-hand

side of the equations above, we obtain

$$\begin{aligned}
\mathbf{P}'_{0,0} &= \mathbf{F}_{0,0} \\
\mathbf{P}'_{0,1} &= \mathbf{E}_{0,1} \\
\mathbf{P}'_{0,2} &= \mathbf{F}_{0,1} \\
\mathbf{P}'_{0,3} &= \mathbf{E}_{0,2} \\
\mathbf{P}'_{1,0} &= \mathbf{E}_{1,0} \\
\mathbf{P}'_{1,1} &= \frac{\mathbf{F}_{0,0} + \mathbf{F}_{0,1} + \mathbf{F}_{1,0} + \mathbf{F}_{1,1} + \mathbf{P}_{1,0} + \mathbf{P}_{0,1} + 8\mathbf{P}_{1,1} + \mathbf{P}_{2,1} + \mathbf{P}_{1,2}}{16} \\
\mathbf{P}'_{1,2} &= \mathbf{E}_{1,2} \\
\mathbf{P}'_{1,3} &= \frac{\mathbf{F}_{0,1} + \mathbf{F}_{0,2} + \mathbf{F}_{1,1} + \mathbf{F}_{1,2} + \mathbf{P}_{1,1} + \mathbf{P}_{0,2} + 8\mathbf{P}_{1,2} + \mathbf{P}_{2,2} + \mathbf{P}_{1,3}}{16} \\
\mathbf{P}'_{2,0} &= \mathbf{F}_{1,0} \\
\mathbf{P}'_{2,1} &= \mathbf{E}_{2,1} \\
\mathbf{P}'_{2,2} &= \mathbf{F}_{1,1} \\
\mathbf{P}'_{2,3} &= \mathbf{E}_{2,2} \\
\mathbf{P}'_{3,0} &= \mathbf{E}_{3,0} \\
\mathbf{P}'_{3,1} &= \frac{\mathbf{F}_{1,0} + \mathbf{F}_{2,0} + \mathbf{F}_{1,1} + \mathbf{F}_{2,1} + \mathbf{P}_{2,0} + \mathbf{P}_{1,1} + 8\mathbf{P}_{2,1} + \mathbf{P}_{3,1} + \mathbf{P}_{2,2}}{16} \\
\mathbf{P}'_{3,2} &= \mathbf{E}_{3,2} \\
\mathbf{P}'_{3,3} &= \frac{\mathbf{F}_{1,1} + \mathbf{F}_{2,1} + \mathbf{F}_{1,2} + \mathbf{F}_{2,2} + \mathbf{P}_{2,1} + \mathbf{P}_{1,2} + 8\mathbf{P}_{2,2} + \mathbf{P}_{3,2} + \mathbf{P}_{2,3}}{16}
\end{aligned}$$

The remaining four points,  $\mathbf{P}'_{1,1}$ ,  $\mathbf{P}'_{1,3}$ ,  $\mathbf{P}'_{3,1}$  and  $\mathbf{P}'_{3,3}$ , as shown in the figure below



are called “vertex points”. These points, as can be seen above, are somewhat complex, but after some reduction it can be seen that

$$\mathbf{P}'_{i,j} = \frac{\mathbf{Q} + 2\mathbf{R} + \mathbf{S}}{4}$$

where  $\mathbf{Q}$  is the average of the face points of the faces adjacent to the vertex point,  $\mathbf{R}$  is the average of the midpoints of the edges adjacent to the vertex point and  $\mathbf{S}$  is the corresponding vertex from the original mesh. For example, if we consider the point  $\mathbf{P}'_{1,3}$ , then

$$\begin{aligned}\mathbf{Q} &= \frac{\mathbf{F}_{0,1} + \mathbf{F}_{0,2} + \mathbf{F}_{1,1} + \mathbf{F}_{1,2}}{4} \\ \mathbf{R} &= \frac{\frac{\mathbf{P}_{0,2} + \mathbf{P}_{1,2}}{2} + \frac{\mathbf{P}_{1,1} + \mathbf{P}_{1,2}}{2} + \frac{\mathbf{P}_{1,3} + \mathbf{P}_{1,2}}{2} + \frac{\mathbf{P}_{2,2} + \mathbf{P}_{1,2}}{2}}{4} \\ \mathbf{S} &= \mathbf{P}_{1,2}\end{aligned}$$

All sixteen points of the subdivision have now been characterized in terms of face points, edge points and vertex points, and a geometric method has been developed to calculate these points.

---

### Extending this Subdivision Procedure to the Entire Patch

We note that all 25 of the points can actually be calculated in this manner, as for example  $\mathbf{P}'_{4,4}$  is a face point and can be calculated as the average of the four points bounding the face. In general, we call the mesh generated by the 25 points as a refinement of the original mesh. In this case, we can state the following rules to generate the points for the refinement of the surface:

- For each face in the original mesh, generate the new face points – which are the average of all the original points defining the face.
- For each internal edge of the original mesh (i.e. an edge not on the boundary), generate the new edge points – which are calculated as the average of four points: the two points which define the edge, and the two new face points for the faces that are adjacent to the edge.
- For each internal vertex of the original mesh (i.e. a vertex not on the boundary of the mesh), generate the new vertex points – which are calculated as the average of  $\mathbf{Q}$ ,  $2\mathbf{R}$  and  $\mathbf{S}$ , where  $\mathbf{Q}$  is the average of the new face points of all faces adjacent to the original vertex point,  $\mathbf{R}$  is the average of the midpoints of all original edges incident on the original vertex point, and  $\mathbf{S}$  is the original vertex point.

The process generated from these rules actually extends to arbitrary rectangular meshes, so we can perform this process again on our refined mesh of 25 elements, producing a second refinement of the original mesh. In this case, we know that this represents yet another subdivision and that eventually, if we keep refining, this “limit mesh” will converge to the original uniform B-spline surface.

Thus, this process gives us a sequence of meshes, each of which is a refinement of the mesh directly above, and which converges to the surface in the limit. For the purposes of rendering such a surface we can simply let the refinement process go until we have a mesh that is “sufficiently close” to the actual surface and then utilize the mesh for rendering purposes.

---

## Summary

The subdivision of the bicubic uniform B-spline surface produces a simple procedure based upon face points, edge points and vertex points, and can be extended to be a refinement procedure for an extended mesh based upon a rectangular topology. Catmull and Clark were able to take this procedure and produce a refinement strategy that works on a mesh of arbitrary topology.

---

## References

- [1] CATMULL, E., AND CLARK, J. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design* 10 (Sept. 1978), 350–355.

---

All contents copyright (c) 1996, 1997, 1998, 1999, 2000  
Computer Science Department, University of California, Davis  
All rights reserved.



## On-Line Geometric Modeling Notes

# CATMULL-CLARK SURFACES

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

### Overview

Utilizing the subdivision for bicubic uniform B-spline surfaces, Ed Catmull and Jim Clark, following the methodology of Doo and Sabin noted that the subdivision rules expressed for the cubic B-spline surface not only work for arbitrary rectangular meshes, but can also be extended to meshes of an arbitrary topology. This extension was accomplished by generalizing the definition of a face point, by modifying the method for calculating vertex points (which extends that of the uniform B-spline case), and by specifying a method for reconnecting the points into a mesh.

---

### Specifying the Refinement Procedure

Given a mesh of control points with an arbitrary topology, we can generalize the face point, edge point, vertex point specification from the uniform B-spline surface calculations to obtain

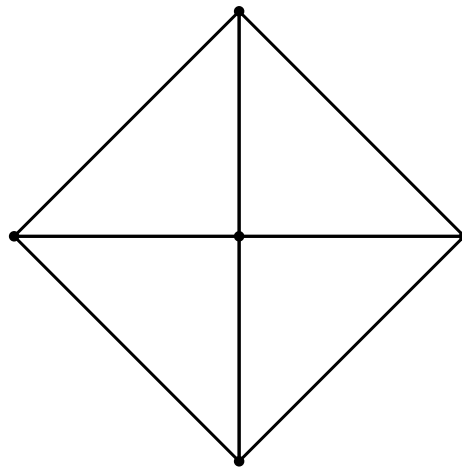
- For each face of the mesh, generate the new face points – which are the average of all the original points defining the face (We note that faces may have 3, 4, 5, or many points now defining them).
- Generate the new edge points – which are calculated as the average of the midpoints of the original edge with the two new face points of the faces adjacent to the edge.
- Calculate the new vertex points – which are calculated as the average of  $\mathbf{Q}$ ,  $2\mathbf{R}$  and  $\frac{(n-3)\mathbf{S}}{n}$ , where  $\mathbf{Q}$  is the average of the new face points of all faces adjacent to the original face point,  $\mathbf{R}$  is the average of the midpoints of all original edges incident on the original vertex point, and  $\mathbf{S}$  is the original vertex point.

The mesh is reconnected by the following method.

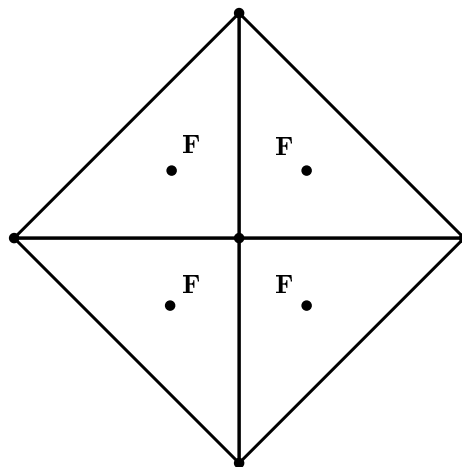
- Each new face point is connected to the new edge points of the edges defining the original face.
  - Each new vertex point is connected to the new edge points of all original edges incident on the original vertex point.
- 

### Example

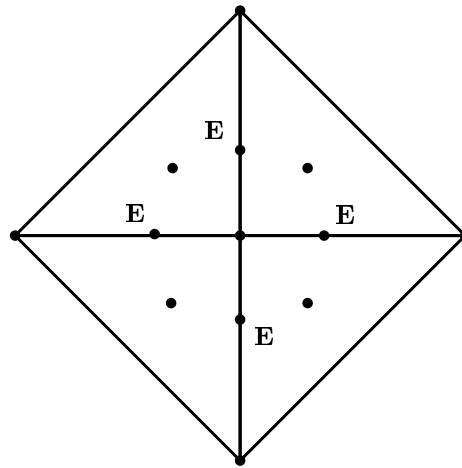
For an example of this process consider the mesh consisting of four triangles in a diamond pattern, as is illustrated below.



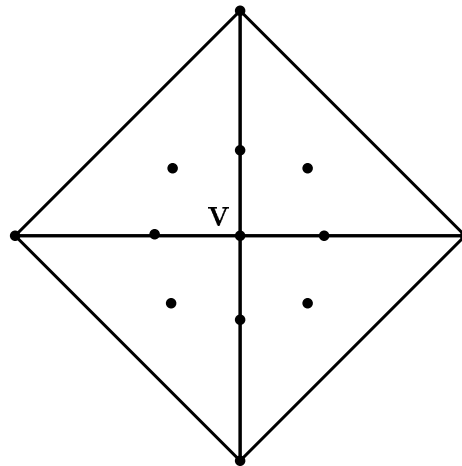
First, we construct the face points, which are calculated as the average of the points that make up each of the original faces. These points are shown in the figure below, labeled with an **F**.



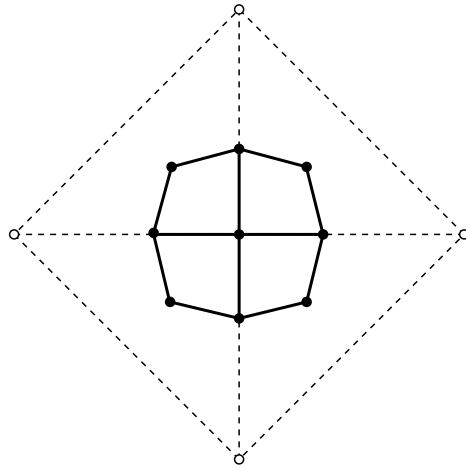
Now, we construct the edge points, which are calculated as the average of the four points – the two original points which define the edge, and the two new face points for the faces that are adjacent to the edge. These points are shown in the figure below and are labeled with an **E**.



We now construct the single vertex point. This point, at least in two dimensions, is identical with the center of the diamond. This point is shown in the figure below.



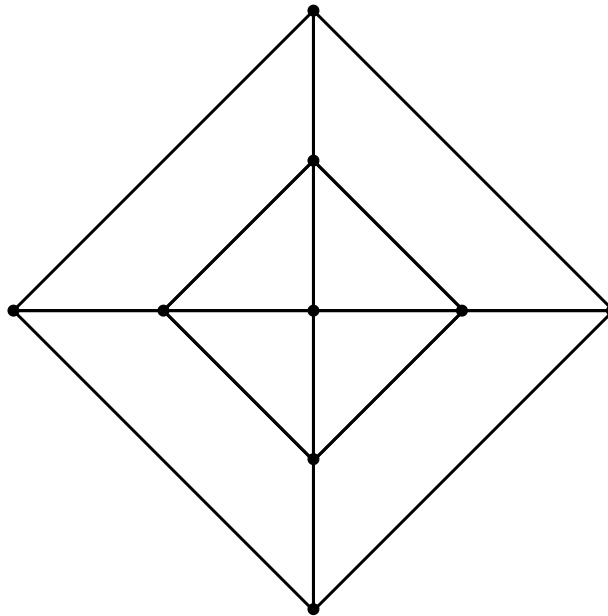
Finally, we connect the edges to the points we have generated: First connecting the face points to the edge points that correspond to edges on the face, and then connecting the vertex point to the edge points.



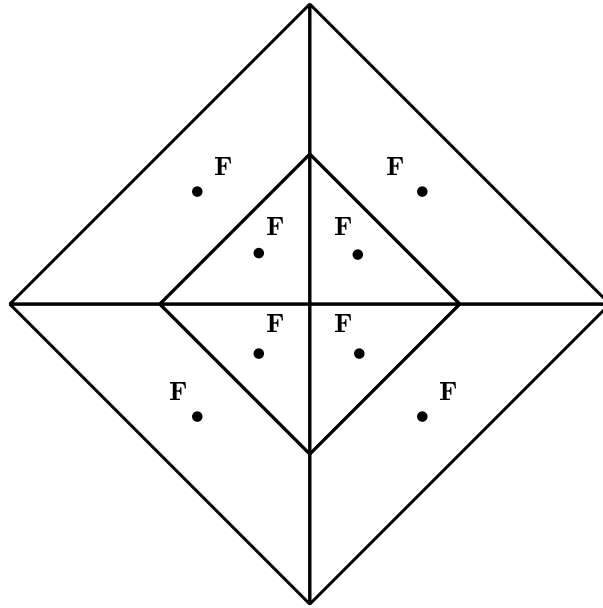
We note now that the each face of the refined mesh has four edges, and our above algorithm with four edges can now be used.

---

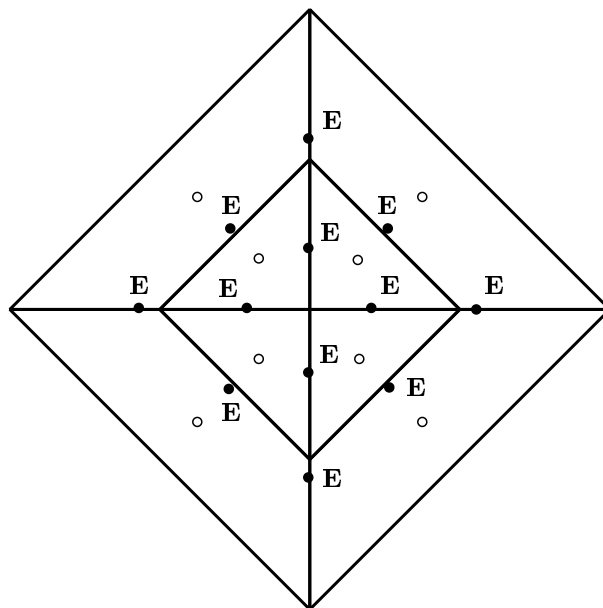
For an expanded example of this process consider the mesh illustrated below.



First, we construct the face points, which are calculated as the average of the points that make up each of the original faces. These points are shown in the figure below, labeled with an  $\mathbf{F}$ .

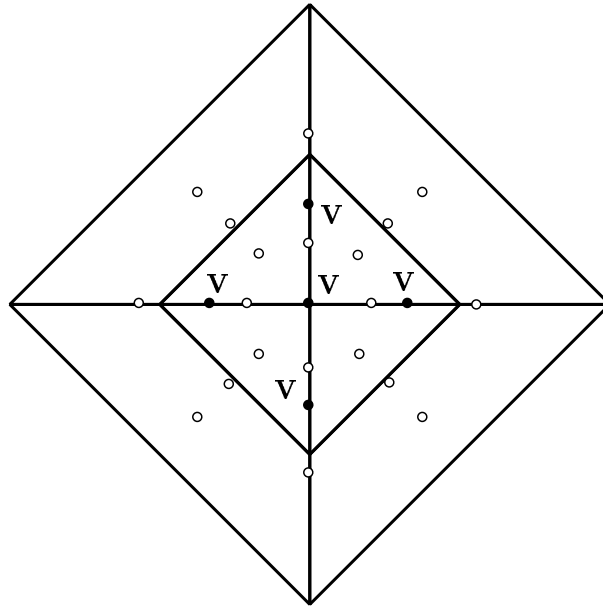


Now, we construct the edge points, which are calculated as the average of the four points – the two original points which define the edge, and the two new face points for the faces that are adjacent to the edge. These points are shown in the figure below and are labeled with an E (The face points calculated in the previous step are indicated as points with white centers).

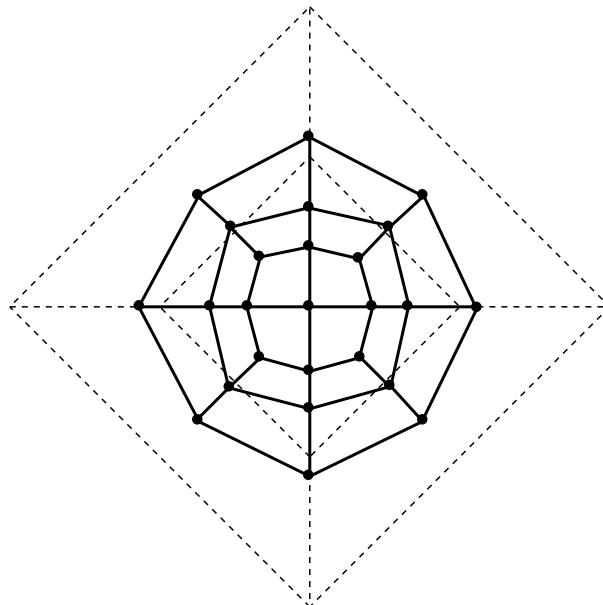


We now construct the vertex points. These points are shown in the figure below, with the face points and

edge points indicated with white centers.



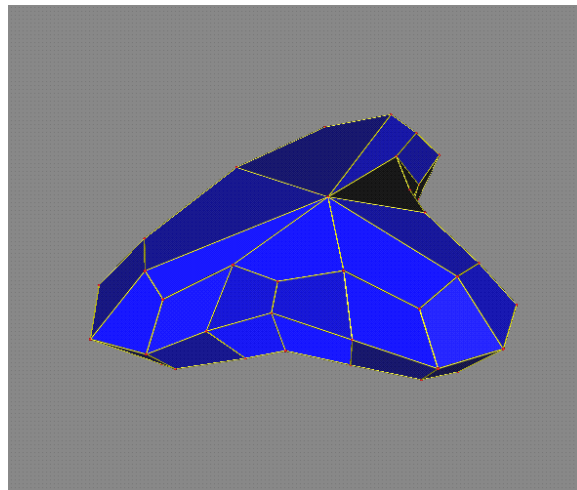
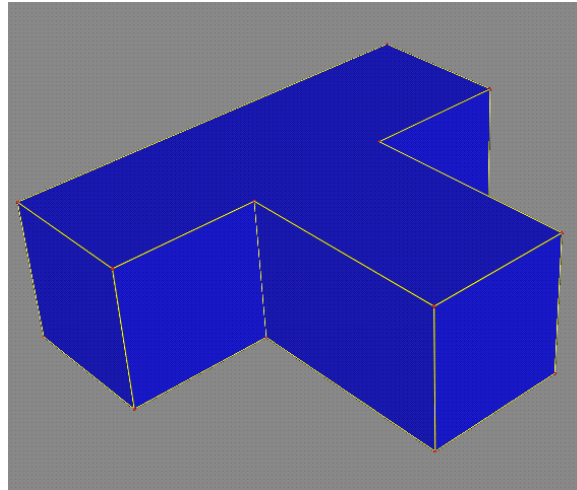
Finally, we connect the edges to the points we have generated: First connecting the face points to the edge points that correspond to edges on the face, and then connecting the vertex point to the edge points.



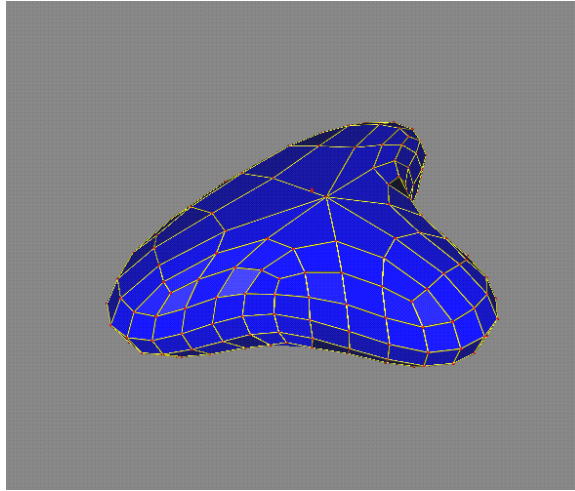
We note now that the each face of the refined mesh has four edges, and in fact, this is true in all cases no matter how many sides the original figure has.

---

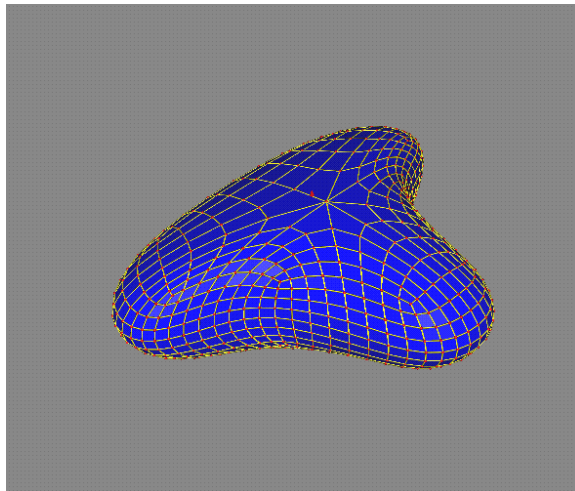
Four steps in the Catmull-Clark refinement process are shown in the illustrations below. Note what happens to the corner control points under the refinement process.



We note that the new set of control points has the property that all faces have four sides. However also note that the vertices corresponding to the original control points retain the *valence* (the number of edges that are adjacent to the vertex). One of the quadrilaterals is shaded incorrectly, since it is non-planar and the rendering algorithm used cannot process these correctly.



Notice still that the vertices corresponding to the original control points retain their valence. Notice the vertex of valence eight at the top of the solid.



Note that any portion of the surface where we have a  $4 \times 4$  array of control points in a rectangular topology, represents a bicubic uniform B-spline surface patch. As subdivision proceeds, the only place where such a topology will not exist is at the vertices that represent the original control points, and whose valence is not four (commonly called extraordinary points). If all vertices in the original had valence four, we would have a bicubic uniform B-spline surface patch to start with.



## Summary

Catmull and Clark have shown that the rules expressed for the cubic B-spline subdivision not only work for arbitrary rectangular meshes, but can also be extended to meshes of an arbitrary topology. This methodology produces a surface that is locally a bicubic uniform B-spline except at a finite number of points on the surface.

---

## References

- [1] CATMULL, E., AND CLARK, J. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design* 10 (Sept. 1978), 350–355.
- 

All contents copyright (c) 1996, 1997, 1998, 1999  
Computer Science Department, University of California, Davis  
All rights reserved.

## On-Line Geometric Modeling Notes

### LOOP SURFACES

Kenneth I. Joy  
Visualization and Graphics Research Group  
Department of Computer Science  
University of California, Davis

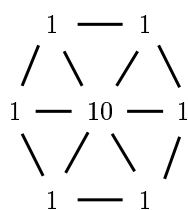
#### Overview

Loop surfaces are similar to Doo-Sabin or Catmull-Clark surfaces in that they are based upon the subdivision paradigm. However, as the Doo-Sabin and Catmull-Clark methods are based upon quadratic and cubic uniform B-spline surface subdivision, the Loop algorithm is based upon the subdivision of quartic uniform box splines – and therefore a mesh of triangles.

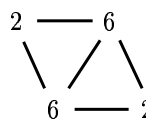
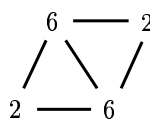
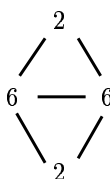
---

#### Loop Surfaces

Given a triangular mesh, the Loop refinement scheme generates both vertex points and edge points and utilizes the following subdivision masks



vertex mask



edge masks

The vertex mask generates new control points for each vertex, and the edge masks generate new control points for each edge of the original triangular mesh.

The edge masks compute the new edge points as the average of three values : the two centers of the faces that share the edge and the midpoint of the edge. The vertex mask can be stated as a convex combination of the points **V**, the original vertex, and **Q** the average of the original points that share an edge with **V**. This

convex combination can be seen to be the following: If  $\mathbf{V}^1$  is the new vertex point, then

$$\begin{aligned}
\mathbf{V}^1 &= \frac{10\mathbf{V} + \mathbf{Q}_1 + \mathbf{Q}_2 + \mathbf{Q}_3 + \mathbf{Q}_4 + \mathbf{Q}_5 + \mathbf{Q}_6}{16} \\
&= \frac{5}{8}\mathbf{V} + \frac{\mathbf{Q}_1 + \mathbf{Q}_2 + \mathbf{Q}_3 + \mathbf{Q}_4 + \mathbf{Q}_5 + \mathbf{Q}_6}{16} \\
&= \frac{5}{8}\mathbf{V} + \frac{6\mathbf{Q}}{16} \\
&= \frac{5}{8}\mathbf{V} + \frac{3}{8}\mathbf{Q}
\end{aligned}$$


---

### Specifying the Refinement Procedure

For an arbitrary triangular mesh we can apply the same rules to generate the new edge points and new vertex points for the refined mesh. However, Loop noted that with the above vertex rule, the surface exhibited some points for which a tangent plane was discontinuous. Upon further examination, he noted that a rule of the form

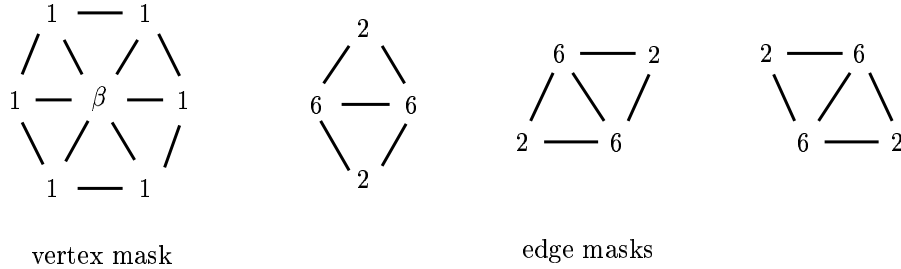
$$\mathbf{V}^1 = \alpha_n \mathbf{V} + (1 - \alpha_n) \mathbf{Q}$$

where

$$\alpha_n = \left( \frac{3}{8} + \frac{1}{4} \cos \frac{2\pi}{n} \right)^2 + \frac{3}{8}$$

gave a surface with a smooth tangent surface. (We note that when  $n = \frac{1}{2}$  then  $\alpha_n = \frac{5}{8}$ , as required).

These surfaces are known as *Loop surfaces* and are generated with the subdivision masks



where for a vertex of  $n$  edges,  $\beta$  is utilized to find the new vertex control point. That is

$$\alpha_n \mathbf{V} + (1 - \alpha_n) \mathbf{Q} = \frac{\beta \mathbf{V} + n \mathbf{Q}}{\beta + n}$$

or

$$\alpha_n = \frac{\beta}{\beta + n}$$

or

$$\beta = \frac{n\alpha_n}{1 - \alpha_n}$$

---

**All contents copyright (c) 1996, 1997, 1998, 1999, 2000  
Computer Science Department, University of California, Davis  
All rights reserved.**