Eurographics Symposium on Geometry Processing (2005)
M. Desbrun, H. Pottmann (Editors)

Subdivision Schemes and Attractors

S. Schaefer[1], D. Levin[2], R. Goldman[1]

[1] Rice University
[2] Tel Aviv University

Abstract

Subdivision schemes generate self-similar curves and surfaces. Therefore there is a close connection between curves and surfaces generated by subdivision algorithms and self-similar fractals generated by Iterated Function Systems (IFS). We demonstrate that this connection between subdivision schemes and fractals is even deeper by showing that curves and surfaces generated by subdivision are also attractors, fixed points of IFS's. To illustrate this fractal nature of subdivision, we derive the associated IFS for many different subdivision curves and surfaces without extraordinary vertices, including B-splines, piecewise Bezier, interpolatory four-point subdivision, bicubic subdivision, three-direction quartic box-spline subdivision and Kobbelt's $\sqrt{3}$-subdivision surfaces. Conversely, we shall show how to build subdivision schemes to generate traditional fractals such as the Sierpinski gasket and the Koch curve, and we demonstrate as well how to control the shape of these fractals by adjusting their control points.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Fractals

1. Introduction

The word *fractal* usually evokes images like the Koch snowflake and Sierpinski gasket shown in figure 1. Many fractal shapes are either totally disconnected or, like the Koch snowflake, continuous everywhere, but differentiable nowhere. Two geometric properties commonly characterize fractals: fractals are self-similar and fractals are attractors. Subdivision schemes are known to generate self-similar curves and surfaces. The purpose of this paper is to show that curves and surfaces produced by subdivision algorithms are also attractors. Thus, despite the fact that subdivision curves and surfaces are commonly smooth, these shapes are also fractals.

Recently, Goldman [Gol04] has shown that every Bezier curve is an attractor. Goldman presents a constructive procedure based on the de Cateljau subdivision algorithm for Bezier curves that builds an Iterated Function System (IFS) whose attractor is exactly the given Bezier curve. Here we shall extend his approach to curves and surfaces generated by more general subdivision algorithms.

To explain what we mean by an IFS, we begin with the definition of a contractive transformation. A transformation f is said to be *contractive* if there exists a constant s, $0 < s < 1$, such that $\|f(x_1) - f(x_2)\| \leq s\|x_1 - x_2\|$ for every pair of points x_1, x_2. An IFS F is a collection of contractive transformations $F = \{f_1, f_2, ..., f_n\}$. We can apply a transformation f to a set X by setting

$$f(X) = \{f(x) | x \in X\}.$$

To apply an IFS F to a set X, we simply take the union of each individual transformation f_i applied to X:

$$F(X) = f_1(X) \bigcup f_2(X) \bigcup ... \bigcup f_n(X). \qquad (1)$$

A set X is said to be an *attractor* of an IFS F if X is a fixed-point of F; that is, $F(X) = X$. For each IFS F composed of contractive maps, there exists a unique attractor for

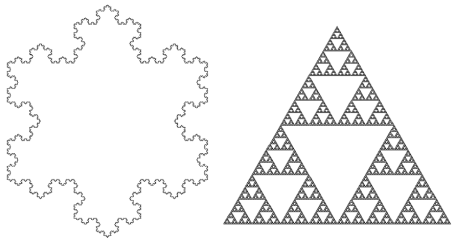

Figure 1: *Two examples of fractals. Koch snowflake (left) and Sierpinski gasket (right).*

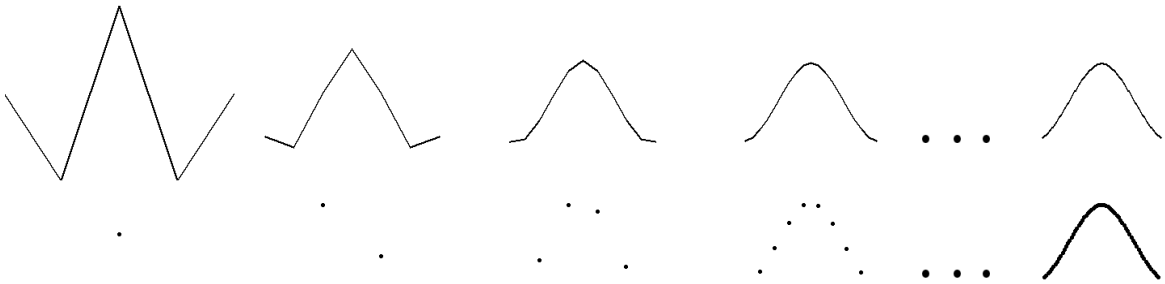

Figure 2: *A control polygon converging to a limit curve using uniform cubic B-spline subdivision with p^0, p^1, p^2, p^3 and p^∞ shown (Top). Starting with an arbitrary point and iterating the B-spline IFS also converges to the same curve (Bottom).*

F, which we denote by F^∞ [Bar93]. To generate F^∞, we can start with any compact set X_0 and iterate $X_{i+1} = F(X_i)$. Remarkably, any compact set X_0 will converge to F^∞ under iteration. Both the Koch snowflake and the Sierpinski triangle are attractors; that is, they are each fixed points of an IFS. Therefore, attractors are often synonymous with fractals [Bar93].

While Goldman concentrated on generating IFS's for Bezier curves, we are interested in curves generated by more general subdivision procedures such as B-splines. B-splines curves of degree n are curves specified over a set of knots $t_0 \ldots t_m$, where the curve in between t_i and t_{i+1} is a Bezier curve of degree n. If the knots are distinct, these curves are C^{n-1} everywhere and contain a discontinuity in the n^{th} derivative at each of the knots t_i.

Since B-splines are composed of piecewise Bezier curves, each of which is an attractor, it is natural to wonder whether B-splines are also attractors. However, constructing an IFS whose attractor is the union of two or more attractors is not so simple. Moreover, there exists arguments to suggest that an IFS for B-splines does not exist. Fractals are self-similar, composed of smaller copies of themselves. For example, the Sierpinski gasket in figure 1 is composed of three smaller Sierpinski gaskets. A B-spline composed of two Bezier curves is a piecewise n^{th} degree polynomial that contains a discontinuity in the n^{th} derivative at the central knot. The self-similarity of such a curve is not readily apparent, since the discontinuity must vanish on the self-similar pieces. Nevertheless, despite this apparent obstruction, in section 2.1 we shall show how to construct an IFS for uniform B-splines as well as for more general subdivision schemes.

Contributions

Inspired by Goldman's work, we propose to bridge the gap between subdivision curves and surfaces and the world of fractals by showing that the shapes generated by subdivision are themselves attractors of an IFS. For any spline curve with uniform knots, we shall present a constructive method for generating an IFS whose attractor is the given spline. Furthermore, we will show how any curve generated by an arbitrary stationary subdivision scheme can be represented by an IFS. We also examine subdivision surfaces in the ordinary case (no extraordinary vertices) and show how several surface subdivision schemes, including rectangular methods like bicubic spline subdivision [CC78] and triangle methods such as three-direction quartic box-spline subdivision [Loo87] and $\sqrt{3}$-subdivision [Kob00], can also be represented using an IFS. Finally, we also consider the converse question: can traditional fractals be generated through subdivision using a set of control points? We provide a general paradigm for introducing control points and subdivision rules for arbitrary fractals generated by an IFS consisting of affine transformations. We then illustrate our approach by introducing control points and subdivision rules for the Sierpinski gasket and the Koch curve. We also show how to adjust the shape of these fractals in an intuitive fashion by moving their control points.

Previous Work

In addition to Goldman's work on IFS's for Bezier curves [Gol04], Prautzsch and Micchelli [PM87] study fractal-like curves generated by subdivision. However, the curves they generate are not smooth. Kocić [Koc96] also considers Bernstein polynomials as fractals.

Kocić and Simoncelli [KS98] introduce Affine Iterated Function Systems, which use barycentric rather than rectangular or homogeneous coordinates to generate fractal shapes. The purpose of their method is to extend the definition of an IFS to include control points in order to govern the shape of the attractor. In contrast, we show how traditional fractals can be generated by subdivision rules. Kocić and Simoncelli also anticipate some of our work on fractal curves by constructing IFS matrices for uniform B-splines. However, they provide no general insight to show that an arbitrary binary subdivision scheme is necessarily an attractor; nor do they consider subdivision surfaces.

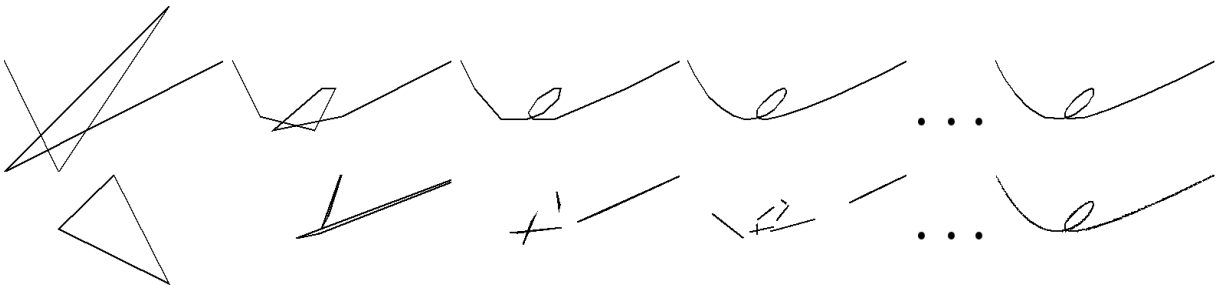

Figure 3: *Subdivision of a cubic B-spline curve with tripled knots at the end points to force interpolation (Top). Starting with a triangle and iterating the associated B-spline IFS generates the same curve despite self-intersection (Bottom).*

2. Curves

We begin by presenting a method for constructing an IFS for any curve defined by an arbitrary, stationary subdivision scheme. We show that the attractor of this IFS is exactly the curve defined by the subdivision scheme and a given set of control points, independent of the starting set X_0 used for the IFS. Next, we illustrate our construction by generating IFS's for uniform cubic B-spline curves. We then extend this method to show that we can handle end-point conditions, where the knot spacing is non-uniform. We also show that this method can generate an IFS for other types of subdivision curves such as non-polynomial curves produced by the four-point subdivision scheme [DGL87].

Subdivision schemes for curves are defined by a set of rules that take in a set of control points p^k as input and produce a new, refined set of control points p^{k+1} as output. Subdivision is a recursive procedure and repeating this process yields a limit curve p^∞. For our purposes, we consider binary subdivision schemes for curves with two sets of rules of the form

$$p_{2i}^{k+1} = \sum_j \alpha_j p_{j+i}^k$$

$$p_{2i+1}^{k+1} = \sum_j \beta_j p_{j+i}^k$$

where α_j and β_j are numerical coefficients and $\sum\alpha_j = \sum\beta_j = 1$. For instance, uniform cubic B-splines have an associated subdivision scheme with rules given in equation 4. Notice that these binary subdivision rules may also be written in matrix form as

$$S = \begin{pmatrix} \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \beta_0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 & \dots \\ \dots & \alpha_{-1} & \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \dots \\ \dots & \beta_{-1} & \beta_0 & \beta_1 & \beta_2 & \beta_3 & \dots \\ \dots & \alpha_{-2} & \alpha_{-1} & \alpha_0 & \alpha_1 & \alpha_2 & \dots \\ \dots & \beta_{-2} & \beta_{-1} & \beta_0 & \beta_1 & \beta_2 & \dots \\ \dots & \alpha_{-3} & \alpha_{-2} & \alpha_{-1} & \alpha_0 & \alpha_1 & \dots \\ \dots & \beta_{-3} & \beta_{-2} & \beta_{-1} & \beta_0 & \beta_1 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix}.$$

Let P be a matrix of control points for p^0, where the i^{th} row of P contains the control point p_i^0. To build an IFS corresponding to the curve with control points p^0 and subdivision matrix S, we break the matrix S apart into multiple square matrices $S_1, S_2, \dots, S_n$ such that applying all products of S_i of length k to the control points P converges to p^∞ as k approaches ∞. Note that, in order to form square matrices, some of the rows of the matrices S_i may overlap (for example, see the B-spline matrices in section 2.1). This decomposition of the matrix S into the matrices S_i is the same as the decomposition used in the joint spectral radius calculation introduced by Levin and Levin [LL03] to study the smoothness properties of various subdivision schemes. Now construct an IFS by letting

$$f_i(X) = XP^{-1}S_iP. \tag{2}$$

Notice that P may not actually be invertible or even a square matrix. To construct a form of P suitable for equation 2, we shall lift the points in P to a higher dimension. To make the matrix P square and invertible, we concatenate the columns of P with rows from an identity matrix to make P size $n \times n-1$; then we change the coordinates into homogeneous form by adding a column of ones to P, making P a square matrix. This new square matrix is invertible as long as the original control points form an affine basis (i.e., the vertices do not all lie on a straight line).

Now if we choose our starting set X_0 to be P, then applying equation 1 simulates subdivision and generates exactly the curve p^∞. Since the fractal generated by an IFS is unique if the f_i are contractive maps, iteration on any compact set X_0 would yield the same fractal. Therefore, the attractor of this IFS is exactly the limit curve p^∞ as long as the f_i are contractive maps.

This independence of the starting set X_0 is somewhat counter-intuitive. Certainly if we begin with the control points P, this process should converge to the limit curve p^∞. However, uniqueness asserts that iterating on *any* compact starting set will converge to the exact same fractal. Figures 2, 5 and 3 show examples that start the iteration with a single point, a line and even a triangle. In these examples, iteration converges to the attractor regardless of the initial shape.

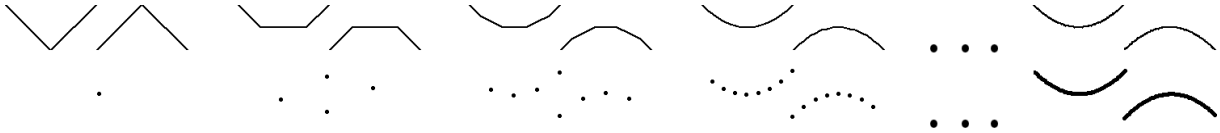

Figure 4: *Subdivision of two disjoint quadratic Bezier curves (Top). Starting with the point* $(2, \frac{3}{2}, 1, 2, \frac{3}{2}, 1)$*, and iterating the IFS whose attractor is the union of the attractors of two individual IFS's one for each Bezier curve (Bottom).*

To show that the maps f_i are contractive, we first note that the matrices $P^{-1}S_iP$ are always of the form

$$P^{-1}S_iP = \begin{pmatrix} . & \dots & . & 0 \\ \vdots & \ddots & \vdots & \vdots \\ . & \dots & . & 0 \\ . & \dots & . & 1 \end{pmatrix}. \tag{3}$$

This result holds because convergent subdivision schemes always have a constant right eigenvector of ones corresponding to the eigenvalue one. Therefore, the last column of S_iP is a column of ones corresponding to the homogeneous component of P. Finally, P^{-1} multiplied by this vector of ones will produce the desired column of the identity matrix because the last column of P is a column of ones and $P^{-1}P = I$.

For convergent subdivision schemes, the S_i have eigenvalues of the form $1 > \lambda_1 \geq \lambda_2 \ldots$. To build f_i, we perform a change of basis on S_i by $P^{-1}S_iP$. This change of basis does not alter the eigenvalues of the matrices. The eigenvector associated with eigenvalue 1 from the matrix in equation 3 corresponds to the translational component of the transformation matrix. Since the remaining eigenvalues are less than 1, the maps f_i are contractive. Therefore, the IFS composed of the f_i has a unique attractor, which is exactly the curve p^∞ defined by recursive subdivision.

2.1. B-splines

Uniform cubic B-splines are piecewise cubic polynomials that have a very simple subdivision scheme [LR80]. Given an initial set of control points p^0, the control points at level p^{k+1} are given by the rules

$$\begin{aligned} p_{2i}^{k+1} &= \tfrac{1}{8}p_{i-1}^k + \tfrac{3}{4}p_i^k + \tfrac{1}{8}p_{i+1}^k \\ p_{2i+1}^{k+1} &= \tfrac{1}{2}p_i^k + \tfrac{1}{2}p_{i+1}^k \end{aligned}. \tag{4}$$

We now construct an IFS whose attractor is exactly a cubic B-spline curve defined by five control points. The limit curve in this case is two cubic polynomial segments that meet with C^2 smoothness. To start, we split the 8×5 subdivision matrix S into two 5×5 subdivision matrices S_1 and S_2 that build the left and right halves of the curve.

$$S_1 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Note that the last 3 rows of S_1 are identical to the first 3 rows of S_2. Next, we construct a matrix P of control points in homogeneous form and extend the coordinates to build a square invertible matrix.

$$P = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 1 \\ x_2 & y_2 & 0 & 1 & 1 \\ x_3 & y_3 & 0 & 0 & 1 \\ x_4 & y_4 & 0 & 0 & 1 \\ x_5 & y_5 & 0 & 0 & 1 \end{pmatrix}$$

Finally, we construct the functions f_1 and f_2 using equation 2.

To display the attractor of our IFS, we can start with any compact set X_0. For our example, we choose a random homogeneous point in five dimensions. Next, we iterate equation 1, each time doubling the number of points. To render these points, we simply drop the extra coordinates and display the points in the plane using only the x, y coordinates. Figure 2 shows an example of this process. Notice that we can start with any set such as lines or polygons and iterating this process will still converge to the same attractor. In contrast, normal subdivision cannot start with an arbitrary set as input and converge to the correct curve. Instead, subdivision requires that we start only with the control polygon for the given curve.

Our strategy for constructing an IFS does not require that the B-spline subdivision rules be uniform throughout the entire curve as they are in the previous example. To illustrate this point, we now build an IFS for a cubic B-spline that interpolates its end-points. We begin with a non-uniform knot spacing with knots $\{0,0,0,1,2,2,2\}$. Using blossoming [Ram89], we then build the two subdivision matrices S_1,

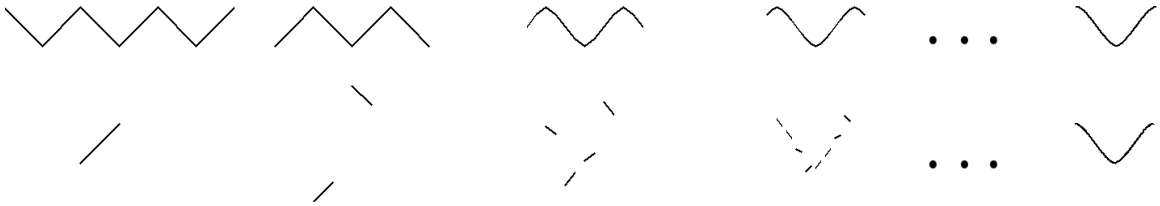

Figure 5: *Subdivision using the four-point rule (Top). Starting with a line and iterating the IFS also converges to the same curve (Bottom).*

S_2 that correspond to inserting knots to form the knot sequence $\{0,0,0,\frac{1}{2},1,1,1,\frac{3}{2},2,2,2\}$.

$$S_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{3}{4} & \frac{1}{4} & 0 & 0 \\ 0 & \frac{3}{8} & \frac{1}{2} & \frac{1}{8} & 0 \\ 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ 0 & \frac{1}{8} & \frac{1}{2} & \frac{3}{8} & 0 \\ 0 & 0 & \frac{1}{4} & \frac{3}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3 shows a triangle converging under the action of this IFS to a cubic B-spline with interpolating endpoint conditions. Similar subdivision algorithms exist for any knot sequence where knot insertion generates a self-similar sequence of knots.

2.2. Bezier Curves

Though Goldman showed how a single Bezier curve could be generated from an IFS, he did not discuss how to build an IFS for multiple Bezier curves that meet with various levels of smoothness. Using our technique, we can construct a single IFS whose attractor is two disjoint Bezier curves. Since each Bezier curve can be represented by an IFS, our method generates a single IFS whose attractor is the union of the attractors from the two individual IFS's.

For a single quadratic Bezier curve, the two subdivision matrices for the corresponding IFS are

$$S_1 = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix}$$

$$S_2 = \begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix}.$$

Given two quadratic Bezier curves with control points P, Q written as rows in homogeneous form, we construct the IFS for the union of the two Bezier curves using block matrices as

$$f_1(X) = X \begin{pmatrix} P^{-1}S_1P & 0 \\ 0 & Q^{-1}S_1Q \end{pmatrix}$$

$$f_2(X) = X \begin{pmatrix} P^{-1}S_2P & 0 \\ 0 & Q^{-1}S_2Q \end{pmatrix}.$$

While similar to the curve matrices developed in section 2.1, there is a complication that arises when using this union technique. Each matrix S_1, S_2 contains an eigenvalue of 1 that corresponds to the homogeneous component in the control points. However, in block form, the new matrix contains two eigenvalues of 1 because there are now two homogeneous components. Therefore, to make sure the f_i are still contractive maps, we require that the starting set X_0 for equation 1 contain 1's in each of the two homogeneous components.

Figure 4 shows an attractor consisting of two Bezier curves generated using this technique. For the starting set X_0, we used a single point of the form $\{x_1,y_1,1,x_2,y_2,1\}$. To render the fractal, we draw the points $\{x_1,y_1\}$ and $\{x_2,y_2\}$.

While this method generates an IFS whose attractor is the union of two Bezier curves, a B-spline curve is composed of multiple Bezier curves that meet with C^{n-1} continuity for a degree n curve. Therefore, for Bezier curves meeting with C^{n-1} smoothness, we get similar results in a more compact form using the B-spline matrices from section 2.1. However, the technique that we present here is general enough to construct an IFS whose attractor is the union of the attractors of any two IFS's whose transformations consist of matrix multiplication.

2.3. Four-Point Subdivision

Until now, all of our examples using this IFS construction have converged to curves that are piecewise polynomial. However, there is nothing special about polynomials and our method can operate on arbitrary subdivision schemes. For instance, the four-point subdivision scheme is an interpola-

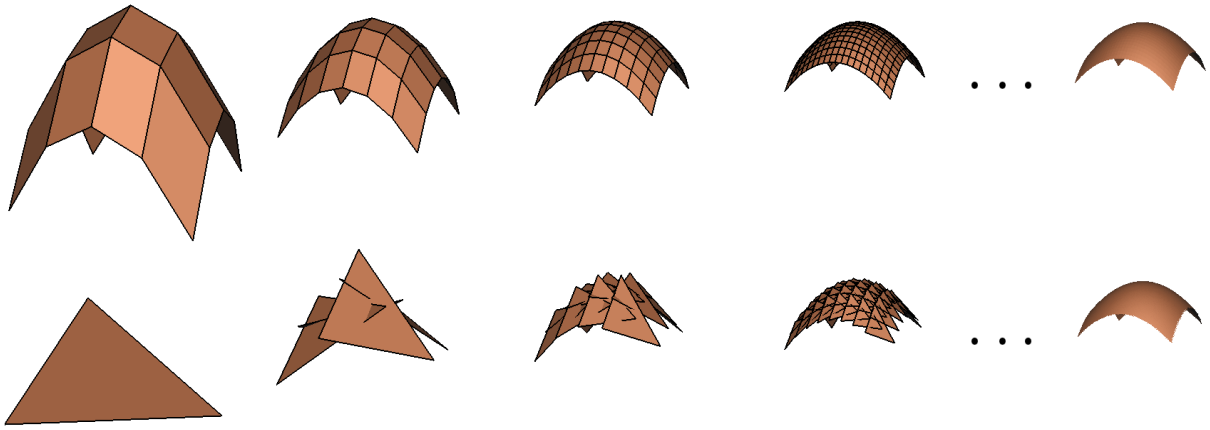

Figure 6: *Subdivision for a bicubic spline surface (Top). Generating the same surface using an IFS (Bottom).*

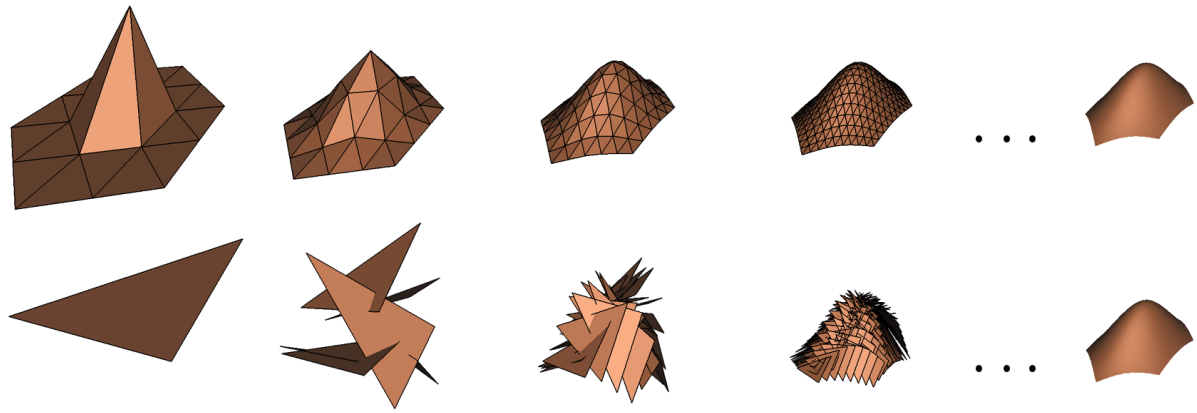

Figure 7: *Three-direction quartic box-spline subdivision for a hexagonal patch (Top). Starting with a single triangle and iterating the IFS converges to the same surface (Bottom).*

tory scheme whose rules are

$$p_{2i}^{k+1} = p_i^k$$

$$p_{2i+1}^{k+1} = \tfrac{-1}{16} p_{i-1}^k + \tfrac{9}{16} p_i^k + \tfrac{9}{16} p_{i+1}^k - \tfrac{1}{16} p_{i+2}^k.$$

This subdivision scheme can reproduce cubic polynomials, though in general the four-point method generates smooth curves that are not polynomial. Nevertheless, we can still construct an IFS that converges to these curves. Below are the subdivision matrices S_1 and S_2 used to construct this IFS.

$$S_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{-1}{16} & \frac{9}{16} & \frac{9}{16} & \frac{-1}{16} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{-1}{16} & \frac{9}{16} & \frac{9}{16} & \frac{-1}{16} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{-1}{16} & \frac{9}{16} & \frac{9}{16} & \frac{-1}{16} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{-1}{16} & \frac{9}{16} & \frac{9}{16} & \frac{-1}{16} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{-1}{16} & \frac{9}{16} & \frac{9}{16} & \frac{-1}{16} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{-1}{16} & \frac{9}{16} & \frac{9}{16} & \frac{-1}{16} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 5 depicts the fractal process starting with a line segment, which converges to the curve defined by this subdivision scheme.

3. Surfaces

While the IFS construction in section 2 is specific to curves, the concepts are general enough to apply as well to subdivision surfaces. In this section, we show several examples of IFS's whose attractors are surfaces defined via uniform subdivision. Though it is straightforward to extend the curve

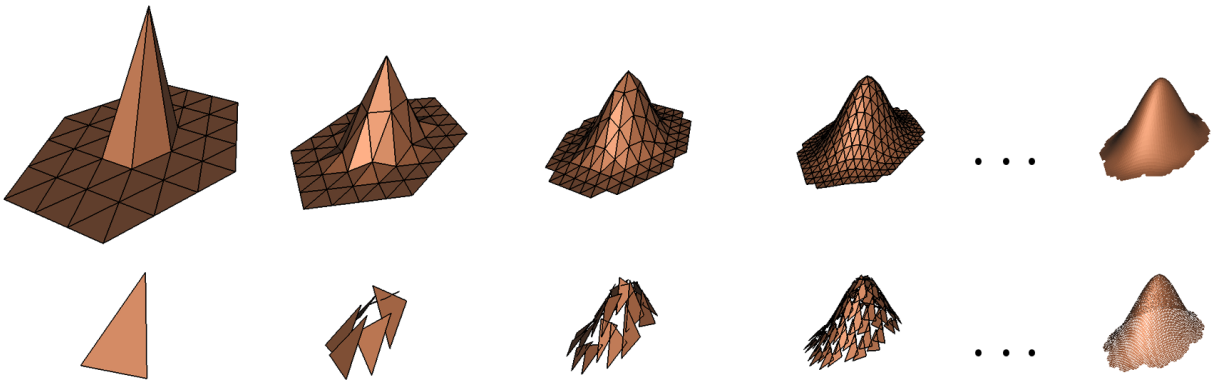

Figure 8: *Kobbelt's $\sqrt{3}$-subdivision for a hexagonal surface (Top). The same surface generated using an IFS (Bottom).*

methods from section 2 to tensor product surfaces, it may not be immediately apparent that other subdivision schemes such as Kobbelt's $\sqrt{3}$-subdivision can also be generating from an IFS.

Similar to the curve case, we begin by constructing a subdivision matrix S that encodes the rules for subdividing a control polyhedron. Next, we separate this matrix into square matrices S_i such that taking all possible combination of these matrices of length k converges to the subdivision surface as k approaches ∞. Unfortunately, the subdivision matrices used to generate these surfaces are too large to include in the body of this paper. However, we have placed a Mathematica notebook online at http://www.cs.rice.edu/ sschaefe/splineifs.nb, which includes our implementation of the IFS's used for all the curves and surfaces in this paper.

3.1. Bicubic Spline Subdivision

In the uniform case, Catmull-Clark subdivision [CC78] operates on quadrilateral polygons and generates a tensor product scheme for uniform cubic B-splines. Since we can represent a cubic B-spline as an IFS, we can also represent this surface subdivision scheme as an IFS. For the surface IFS, we have four subdivision matrices instead of the two matrices from the curve examples in section 2.1. Figure 6 illustrates a bi-cubic patch constructed from a 4×4 grid of control points using this subdivision scheme. Starting with a single triangle, we apply the fractal process using the IFS we have constructed and generate the same bi-cubic patch.

3.2. Three-Direction Quartic Box-Splines

Three-direction quartic box-splines correspond to Loop subdivision [Loo87] in the uniform case. The surfaces that this scheme takes as input are composed of triangles. In contrast to bicubic subdivision in section 3.1, this scheme cannot be written as a tensor product; however, this scheme may still be used to generate an IFS. Figure 7 shows a hexagonal surface created from a hexagonal grid of control points.

Like many fractals, there is more than one way to write down the transformations that generate this attractor. Even though this subdivision scheme is triangle-based, we could have split S into four matrices instead of the six we used in figure 7. However, the attractor in that case would look rectangular and similar to that of figure [CC78]. We chose six matrices in this case to create an attractor with a hexagonal shape to distinguish this attractor from the tensor product scheme in section 3.1.

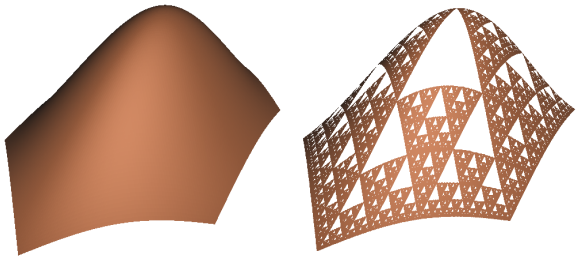

Figure 9: *Three-direction quartic box-spline surface generated by an IFS with six transformations (Left). The same IFS with two of the six transformations removed creates a fractal-like surface, which is a subset of the original surface (Right).*

3.3. Kobbelt's $\sqrt{3}$-subdivision

Finally, surfaces generated by Kobbelt's $\sqrt{3}$-subdivision [Kob00] can also be represented by an IFS. Like quartic box-spline subdivision in section 3.2, this scheme operates on meshes of triangles. However, two rounds of subdivision using this scheme correspond to a ternary split of the triangle polygons (hence the name $\sqrt{3}$-subdivision), whereas three-direction quartic box-spline subdivision performs a binary split at each level of subdivision.

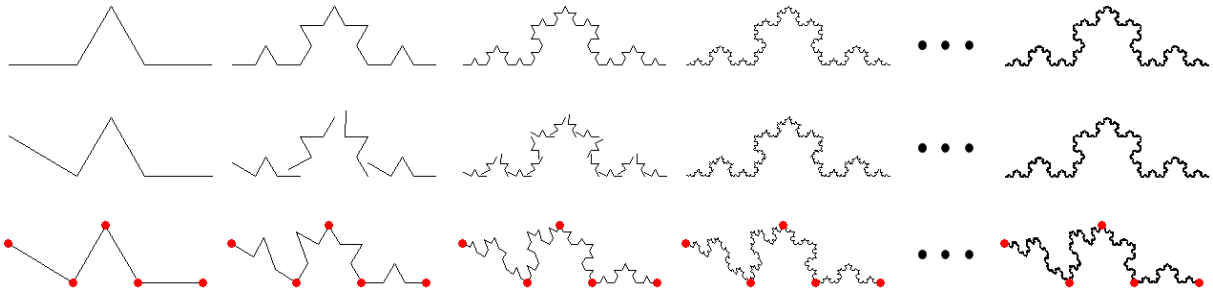

Figure 10: *Applying the standard IFS for the Koch curve to a control polygon produces the fractal (Top). Starting with a different control polygon and applying the standard IFS still converges to the same fractal (Middle). With subdivision, moving a control point changes the shape of the Koch curve (Bottom).*

Though $\sqrt{3}$-subdivision is a somewhat exotic subdivision scheme, the rules as well as the structure of the control points behave in a self-similar fashion, which allows us to construct the subdivision matrices necessary to build an IFS for these surfaces. Figure 8 (top) contains a surface generated through uniform $\sqrt{3}$-subdivision on polygons. On the bottom of the figure, we use the associated IFS to recover the same surface starting from a single triangle.

The surfaces generated in this section are developed through an IFS; however, they certainly do not resemble fractal shapes such as the Sierpinski gasket from figure 1. Nevertheless, we can observe their underlying fractal structure by the following device. If F is an IFS and $G \subset F$, the fractal generated by G is a subset of the fractal generated by F - that is, if $G \subset F$, then $G^{\infty} \subseteq F^{\infty}$. Thus if we were to omit some of the matrices from an IFS that generates a subdivision surface, we would generate a fractal on the surface. Figure 9 shows a picture of the three-direction quartic box-spline subdivision surface from figure 7 and a subset of the same surface generated by an IFS with two of the six transformations omitted. The resulting fractal surface very much resembles a Sierpinski gasket, but all of the points lie on the subdivision surface. This figure illustrates the underlying fractal structure of these subdivision surfaces.

4. Fractals with Control Points

Subdivision schemes have advantages over more traditional fractal methods because the user can control the shape of a subdivision curve or surface by manipulating a set of control points that govern the shape of the attractor. Since many common subdivision schemes generate fractals, perhaps many common fractals can also be generated by subdivision. The purpose of this section is to show that this insight is indeed correct – that many standard fractals can be generated by subdivision rules applied to control points. We shall present a general strategy for converting fractals into subdivision schemes and we will provide several examples to illustrate the method. As with subdivision schemes, our strategy is to build the control points into the very fabric of the iterated function system of the fractal. The advantage of this control point representation for fractals is that control points provide much finer control over the shape of a fractal than standard IFS's.

Given an IFS defined by a set of affine transformations $F = \{f_1, \ldots f_n\}$, we construct a subdivision scheme by choosing a set of control points $P = \{P_1 \ldots P_m\}$. These control points may be any points in space. However, for many fractals such as the Koch curve there are natural choices for the control points (see figure 10).

To build the subdivision rules for our fractal, we apply F to P. For each vertex p_j in $f_i(P)$, we represent p_j as an affine combination of the original control points P:

$$p_j = \Sigma_k \alpha_{j,k} P_k. \tag{5}$$

We construct the subdivision matrix S_i corresponding to the transformation f_i by setting the j,k entry of S_i to $\alpha_{j,k}$. The subdivision matrix S for this fractal is then simply the concatenation of the S_i. Notice that the decomposition in equation 5 does not yield a unique set of weights $\alpha_{j,k}$. In fact, many different choices for the weights are possible because the problem is underdetermined.

4.1. Koch curve

To construct the Koch curve, we begin by selecting five control points (see figure 10). To generate the subdivision rules, we note that the Koch curve is composed of four transformations. The first two transformations scale about the left and right end-points of the curve by $\frac{1}{3}$. The remaining two transformations scale about the end-points by $\frac{1}{3}$, rotate upwards by 60 degrees about the outside corners and translate inwards by $\frac{1}{3}$. Since there are four transformations, there will be four subdivision matrices. We apply each transformation to the control points and then build our subdivision rules using barycentric coordinates with respect to the three closest

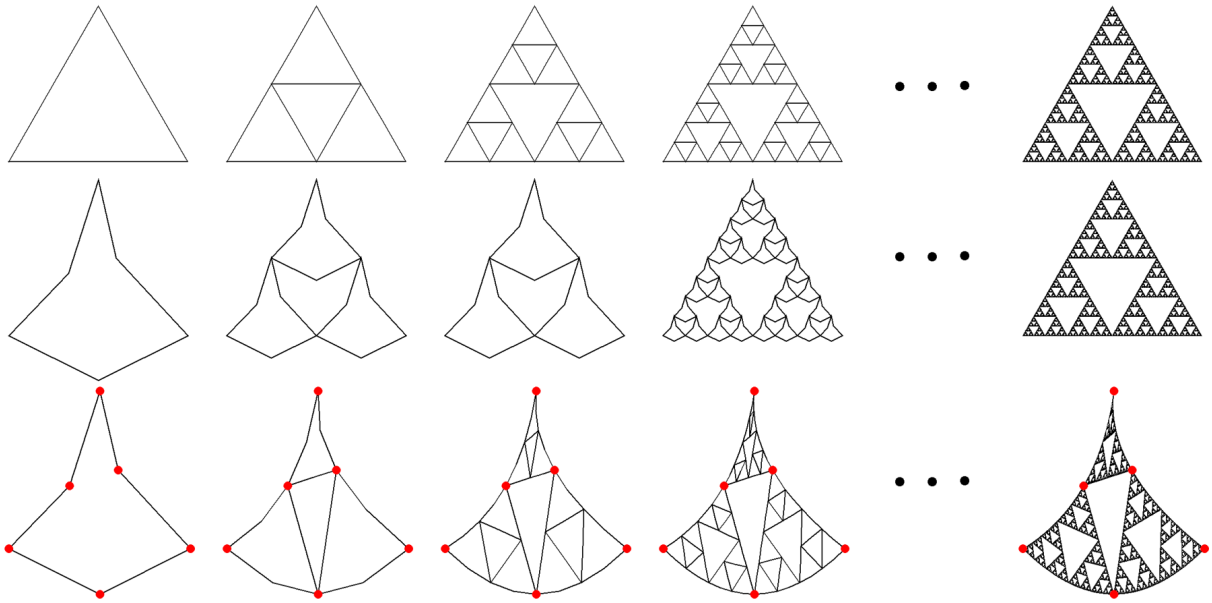

Figure 11: *Applying the standard IFS for the Sierpinski gasket to a triangle as the starting set produces the gasket (Top). Starting with a different polygon and applying the standard IFS converges to the same fractal (Middle). With subdivision, moving the control points changes the shape of the final fractal (Bottom).*

control points. The four subdivision matrices are

$$S_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{2}{3} & \frac{1}{3} & 0 & 0 & 0 \\ \frac{2}{3} & 0 & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & \frac{2}{3} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{2}{3} & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & 0 & \frac{2}{3} & 0 & 0 \\ 0 & \frac{1}{3} & \frac{2}{3} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$S_3 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & 0 & \frac{2}{3} & 0 & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$S_4 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{2}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & 0 & \frac{2}{3} \\ 0 & 0 & 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 10 illustrates the difference between the ordinary IFS for the Koch curve and the Koch curve built via subdivision. On top, we start with the control polygon and apply the standard IFS to produce the Koch curve. The standard IFS converges to the same shape even if we start with a different control polygon. However, if we use the subdivision matrices S_i, the altered control points generate a deformed version of the Koch curve. Notice that this deformation is local to the left side of the curve.

4.2. Sierpinski gasket

To build the Sierpinski gasket using subdivision, we first choose our control points P. One natural choice for these control points are the three extremal vertices of the triangle. However, using only three vertices will result in a set of control points that can generate only affine transformations of the gasket. Therefore, in addition to the three corner points, we choose the three edge vertices as well (see figure 11) for the control points of the fractal.

In contrast to the Koch curve where we use barycentric coordinates, for the Sierpinski gasket we will choose a more interesting set of subdivision rules. The Sierpinski gasket is generated by three transformations f_i, each of which scales by $\frac{1}{2}$ about the corners of the outer triangle. If we treat each of the outer edges of the gasket as quadratic curves, we can define the new positions of the control points along each edge using Lagrange interpolation. For the image of the control points on the interior of the gasket, we simply linearly interpolate the vertices at the end of the interior edge. These

rules create the three subdivision matrices

$$S_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{8} & \frac{3}{4} & \frac{-1}{8} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{3}{8} & 0 & 0 & 0 & \frac{-1}{8} & \frac{3}{4} \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{-1}{8} & \frac{3}{4} & \frac{3}{8} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{3}{8} & \frac{3}{4} & \frac{-1}{8} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \end{pmatrix}$$

$$S_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-1}{8} & \frac{3}{4} & \frac{3}{8} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{-1}{8} & 0 & 0 & 0 & \frac{3}{8} & \frac{3}{4} \end{pmatrix}.$$

Figure 11 shows the difference between the standard IFS and subdivision for the Sierpinski gasket. The fractal generated by the standard IFS is independent of the starting set (Top and Middle). However, if the user changes the positions of the control points, the subdivision scheme will converge to a deformed version of the gasket (Bottom).

5. Conclusions

We have demonstrated that standard, stationary subdivision schemes generate fractals - that is, subdivision curves and surfaces are attractors of an IFS. In contrast to ordinary subdivision, an IFS can generate these curves and surfaces by starting with any compact set and not just their control polygons. Also, subdivision requires not only the control vertices, but the topology of those vertices as well. An IFS generates these curves and surfaces without directly storing the topology. We provided several examples of curves and surfaces generated using this method, including B-splines, piecewise Bezier curves and curves generated by the four-point scheme as well as surfaces created by subdivision schemes such as bicubic splines, three-direction quartic box-splines and Kobbelt's $\sqrt{3}$-subdivision. We ended by demonstrating that many traditional fractals such as the Koch curve and Sierpinski gasket can also be represented by subdivision schemes, which allows the user control over the shape of the fractal by adjusting a set of control points.

In section 2.2, we noted that the method for constructing an IFS consisting of two Bezier curves is a union operator. In the future we would like to consider whether other operators on attractors such as intersection have simple expressions as well.

Finally, our surface examples consist only of ordinary configurations of polygons and do not address issues associated with extraordinary vertices. We believe that we can construct an IFS whose attractor also includes extraordinary vertices. However, to do so, we may need to add a condensation set [Bar93] to the IFS.

Acknowledgements

We would like to thank Joe Warren and Wenping Wang for their many helpful discussions. We would also like to thank an anonymous referee for pointing out to us the papers by Kocić and Simoncelli.

References

[Bar93] BARNSLEY M.: *Fractals Everywhere (Second Edition)*. Academic Press, 1993. 2, 10

[CC78] CATMULL E., CLARK J.: Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design 10* (1978), 350–355. 2, 7

[DGL87] DYN N., GREGORY J., LEVIN D.: A four point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design 4* (1987), 257–268. 3

[Gol04] GOLDMAN R.: The fractal nature of bezier curves. In *Geometric Modeling and Processing* (2004). 1, 2

[Kob00] KOBBELT L.: $\sqrt{3}$-subdivision. In *Proceedings of SIGGRAPH 2000 (SIGGRAPH-00)* (New Orleans, USA, 2000), Akeley K., (Ed.), vol. 2000 of *Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, ACM Press, pp. 103–112. 2, 7

[Koc96] KOCIĆ L. M.: Fractals and bernstein polynomials. *Periodica Mathematica Hungarica 33* (1996), 185–195. 2

[KS98] KOCIĆ L. M., SIMONCELLI A. C.: Towards free-form fractal modelling. In *Mathematical methods for curves and surfaces, II (Lillehammer, 1997)*, Innov. Appl. Math. Vanderbilt Univ. Press, Nashville, TN, 1998, pp. 287–294. 2

[LL03] LEVIN A., LEVIN D.: Analysis of quasi uniform subdivision. *Applied and Computational Harmonic Analysis 15(1)* (2003), 18–32. 3

[Loo87] LOOP C.: *Smooth subdivision surfaces based on triangles*. Master's thesis, University of Utah, Department of Mathematics, 1987. 2, 7

[LR80] LANE J., RIESENFELD R.: A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence 2* (1980), 35–46. 4

[PM87] PRAUTZSCH H., MICCHELLI C.: Computing curves invariant under halving. *Computer Aided Geometric Design 4*, 1–2 (July 1987), 133–140. 2

[Ram89] RAMSHAW L.: Blossoms are polar forms. *Computer Aided Geometric Design 6* (1989), 323–358. 4