On Approximating Rough Curves With Fractal Functions

Wayne O. Cochran John C. Hart Patrick J. Flynn

School of EECS
Washington State University
Pullman, WA 99164-2752
{wcochran,hart,flynn}@eecs.wsu.edu

Abstract

Fractal functions are explored as a representation for rough data in computer graphics. Two new techniques for using fractal interpolation functions to approximate rough functions and curves are introduced. The first is based on a Hough transform of fractal function transformation parameters. The second is based on previous techniques in fractal image compression. These techniques are then demonstrated on the task of recovering the parameters of a fractal function, approximating a rough function and approximating the boundary of a leaf.

Keywords: approximation, fractals, fractal functions, Hough transform, interpolation, iterated function system.

1 Introduction

Methods for approximating piecewise smooth curves and surfaces are abundant. "Rough" curves (*e.g.,* mountain range silhouettes) that are differentiable almost nowhere yet possess C^0 continuity are generally more difficult to model accurately with an economy of parameters. Fractal interpolation and approximation functions are attractive for their compact representation of rough curves.

Parametric curves $\mathbf{p}(t)$ are generally descibed by coordinate functions $\mathbf{p}(t) = (x(t), y(t))$. These coordinate functions are commonly formed by the sum of basis functions. A family of fractal functions has been developed [15] that represents such a graph not as the sum of basis functions, but out of smaller transformed copies of the graph. Using such functions, this paper explores new techniques for using these functions to approximate existing data, and applies these approximations to various input shapes, including a digitized leaf boundary.

1.1 Previous Work

Although fractal interpolation functions were formalized over a decade ago, they are only seeing recent use in computer graphics. They have been used to represent uncertainty in the visualization of scientific data [16] and to imply the shape of a dropping leaf [18]. The use of iterated function systems to represent curves has been previously explored [17] but not applied to reproducing a specific curve. The techniques explored in Section 3.1 are an extenion of previous techniques designed to detect self-similar structure in shapes [10].

1.2 Overview

Section 2 describes existing techniques for forming fractal functions that interpolate a given set of data points. Section 3 introduces two new techniques for using such functions to approximate an input data set, one based on the Hough transform, the other based on fractal image compression. Section 4 applies these approximation techniques to recover the parameters of a fractal function, approximate a rough but non-self-affine function and efficiently represent the boundary of a leaf.

2 Fractal Interpolation

We pose the interpolation problem with a set of input points $\{(t_i, x_i)\}_{i=0}^{N}$ with nodes $0 = t_0 < t_1 < \cdots < t_N = 1$ and ordinates $x_i = F(t_i) \in \mathbb{R}$ assuming some continuous function $F : [0, 1] \to \mathbb{R}$.

Classically, F is assumed smooth, and the input points are interpolated with a single degree N polynomial, or piecewise with a low-degree polynomial. Recent research has provided an alternative assumption that the interpolation function F is self-similar, and typically not smooth but fractal.

We note that a function $F : [0, 1] \to \mathbb{R}$ is well defined by its graph, and use the same symbol to denote the set of points in its graph. Hence a point $(t, x) \in F$ if and only if $x = F(t)$. We also use the notation $F[t_1, t_2]$ to denote the graph of F over the interval $[t_1, t_2]$. Hence a point $(t, x) \in F[t_1, t_2]$ if and only if $(t, x) \in F$ and $t \in [t_1, t_2]$.

2.1 Fractal Interpolation Functions

We construct an *iterated function system* (IFS) whose attractor is the graph of a function $F : [0, 1] \to \mathbb{R}$. Such a function F is called a *fractal interpolating function* (FIF) [2].

For $i = 1, \ldots, N$ let $T_i : [0, 1] \times \mathbb{R} \to [0, 1] \times \mathbb{R}$ be shears of the form

$$T_i : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} a_i & 0 \\ b_i & c_i \end{bmatrix} \begin{bmatrix} t \\ x \end{bmatrix} + \begin{bmatrix} d_i \\ e_i \end{bmatrix} \qquad (1)$$

where $|c_i| < 1$ is given as a parameter controling the roughness of the function, and a_i, b_i, d_i and e_i are determined either by the constraints

$$\begin{aligned} T_i(0, x_0) &= (t_{i-1}, x_{i-1}), \\ T_i(1, x_N) &= (t_i, x_i), \end{aligned}$$

or the "reflected" constraints

$$\begin{aligned} T_i(1, x_N) &= (t_{i-1}, x_{i-1}), \\ T_i(0, x_0) &= (t_i, x_i). \end{aligned}$$

An example of a FIF is shown in Figure 1.

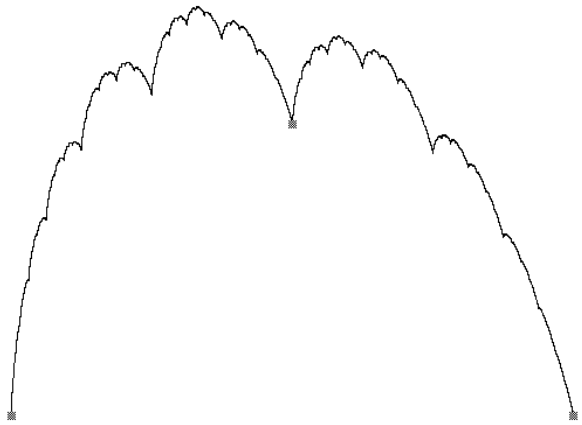

Figure 1: Graph of a fractal interpolation function of the points $(0, 0), (0.5, 1), (1, 0)$ using $c_1 = 0.6$ and $c_2 = 0.4$

Given the metric

$$d((t_1, x_1), (t_2, x_2)) = |t_1 - t_2| + \frac{(1 - A)}{2B}|x_1 - x_2| \quad (2)$$

where $A = \max_i |a_i|$ and $B > \max_i |b_i|$, it can be shown that each T_i has contractivity $s = \max\{(1+A)/2, C\} < 1$, where $C = \max_i |c_i|$. Hence, by the fixed point theorem, there exists one and only one function F satisfying the invariance

$$F = \bigcup_i T_i(F). \quad (3)$$

2.2 Recurrent Fractal Interpolation Functions

A generalization of the FIF, called the *recurrent fractal interpolating function* (RFIF) [3], provides more flexibility in representing rough curves. Like the FIF, the RFIF represents a function F with a set of shears T_i. However, whereas the FIF models the graph F out of smaller copies of F, the RFIF models the graph of F out of smaller copies of sections of F.

For each i, we have indices j_i and k_i such that

$$T_i(t_{j_i}, x_{j_i}) = (t_{i-1}, x_{i-1}), \quad (4)$$

$$T_i(t_{k_i}, x_{k_i}) = (t_i, x_i), \quad (5)$$

and $|t_{k_i} - t_{j_i}| > t_i - t_{i-1}$. If all of the shears T_i are contractive under (2) then there is one and only one graph F that satisfies the invariance

$$F = \bigcup_i T_i(F[t_{j_i}, t_{k_i}]). \quad (6)$$

Given a sequence of points (t_i, x_i) we can construct a RIFS that interpolates these points by forcing the curve that lies between (t_i, x_i) and (t_{i+1}, x_{i+1}) to be a contracted copy of the curve that begins at (t_{i-1}, x_{i-1}) and ends at (t_{i+2}, x_{i+2}).

2.3 Partitioned Fractal Interpolation Functions

The RFIF model is general and can be used to describe a wide variety of shapes, but it also requires that the "domain" intervals $[t_{j_i}, t_{k_i}]$ of each map T_i be the union of "range" intervals, since $t_{j_i}, t_{k_i} \in \{t_i\}_{i=0}^N$.

The *Partitioned Fractal Interpolation Function* (PFIF) relaxes this constraint by allowing the domain interval of each map to be arbitrary. A PFIF is defined by a finite set of maps $\{T_i\}$ and an associated domain interval $[\alpha_i, \beta_i]$ such that $F[t_{i-1}, t_i] = T_i(F[\alpha_i, \beta_i])$ and $|\beta_i - \alpha_i| > t_i - t_{i-1}$. As before, when T_i is contractive, then there is one and only one graph F that satisfies the invariance

$$F = \bigcup_i T_i(F[\alpha_i, \beta_i]). \quad (7)$$

We are currently unaware of any prior publication of partitioned fractal interpolation functions, although it is a rather obvious extension of the ideas from fractal image compression to fractal interpolation functions. The PFIF is the FIF analogy of the partitioned iterated function system [7] and the local iterated function system [6].

2.4 Fractal Interpolation Curves

A parametric version of the curve can be constructed by assigning a value of t to each knot and using two functions $x(t)$ and $y(t)$ which interpolate appropriately.

Hence, we are given a collection of input data points $\{(x_i, y_i, t_i)\}_{i=0}^N$ where (x_i, y_i) are the co-ordinates to be interpolated, and t_i is the parameter value of the curve as it passes through that point. As before, we require $0 = t_0 < t_1 < \cdots < t_N = 1$.

We now have two FIF's, $\{X_i\}_{i=1}^N$ and $\{Y_i\}_{i=1}^N$ where for each i we have either

$$\begin{aligned} X_i(0, x_0) &= (t_{i-1}, x_{i-1}), \\ X_i(1, x_N) &= (t_i, x_i), \end{aligned}$$

or

$$\begin{aligned} X_i(1, x_0) &= (t_{i-1}, x_{i-1}), \\ X_i(0, x_N) &= (t_i, x_i). \end{aligned}$$

We also have either

$$\begin{aligned} Y_i(0, y_0) &= (t_{i-1}, y_{i-1}), \\ Y_i(1, y_N) &= (t_i, y_i), \end{aligned}$$

or

$$\begin{aligned} Y_i(1, y_0) &= (t_{i-1}, y_{i-1}), \\ Y_i(0, y_N) &= (t_i, y_i). \end{aligned}$$

We combine the two FIF's by using a single set of maps of the form

$$P_i : \begin{bmatrix} t \\ x \\ y \end{bmatrix} \mapsto \begin{bmatrix} a_i & 0 & 0 \\ b_{xi} & c_{xi} & 0 \\ b_{yi} & 0 & c_{yi} \end{bmatrix} \begin{bmatrix} t \\ x \\ y \end{bmatrix} + \begin{bmatrix} d_i \\ e_{xi} \\ e_{yi} \end{bmatrix} \quad (8)$$

where, for example, b_{xi} is the b_i component of X_i. Note that the a_i and e_i components of X_i equal their corresponding components in Y_i. While $x(t)$ and $y(t)$ are independent, they share a common "hidden variable" t that allows a single transformation P_i to represent both for a given segment of the curve [4].

Equation 8 is a separable function in that the $X(t)$ and $Y(t)$ component functions are unrelated, except for sharing the same parameter t. The form

$$P_i : \begin{bmatrix} t \\ x \\ y \end{bmatrix} \mapsto \begin{bmatrix} a_i & 0 & 0 \\ b_{xi} & c_{xi} & c_{xyi} \\ b_{yi} & c_{yxi} & c_{yi} \end{bmatrix} \begin{bmatrix} t \\ x \\ y \end{bmatrix} + \begin{bmatrix} d_i \\ e_{xi} \\ e_{yi} \end{bmatrix} \quad (9)$$

includes additional factors c_{xy} and c_{yx} that allow the $X(t)$ and $Y(t)$ curves to affect each other. While the representation is no longer separable ($X(t)$ and $Y(t)$ are mutally dependent), the added factors increase the flexibility of the curve.

3 Fractal Approximation

Like the previous interpolation methods, we pose the approximation problem with a set of input points $\{(t_i, x_i)\}_{i=0}^{N}$. Unlike interpolation, we seek to represent the input data with a fractal function (FIF, RFIF, PFIF) with significantly fewer than N transformations. In the general approximation case, there are more data points than degrees of freedom in the representation, hence an error minimization over the space of transformation parameters is performed to find the best fit of the fractal function to the input data.

3.1 Hough Transform

A Hough transform detects organization in input data by assuming the input data is organized according to some model. It then gathers all groups of the input points and fits each group to the model, yielding the model's parameters for that group of points. These model parameters are plotted in a parameter space, and the Hough transform detects the presence of the organization of the input points according to a given model by the clustering of model parameters plotted in this parameter space.

For example, the Hough transform can detect lines in a collection of 2-D points [1]. For every pair of 2-D points in the input data, the equation $y = mx+b$ of the line passing through the two points is determined, and the parameters m and b are plotted in a 2-D parameter space. Clusters of points in this parameter space indicate that numerous points in the input data lie on the same line, and the parameters for this line can be found at the center of this cluster.

Given a set of $N + 1$ points $\{(t_i, x_i)\}_{i=0}^{N}$ we seek to find a small number of shears that approximate the data points out of smaller copies of other points. These shears may then be organized into a FIF, RFIF or PFIF. We find all shears that take collections of points to other collections of points, and plot the parameters of the shears. These parameters cluster around parameters of the self-affinity of the input data.

From the set of input points, we enumerate all four-tuples of points $(t_i, x_i), (t_k, x_k), (t_l, x_l)$ and (t_n, x_n) such that

1. $i + 1 < k$,
2. $l + 1 < n$, and
3. $t_k - t_i > t_n - t_l$.

For each such four-tuple, we then solve for the two sets of values a and d such that T maps the t-domain $[t_i, t_k]$ onto the t-range $[t_l, t_n]$. The first set of values is found by solving

$$at_i + d = t_l, \quad (10)$$
$$at_k + d = t_n, \quad (11)$$

and the second set by solving

$$at_k + d = t_l, \quad (12)$$
$$at_i + d = t_n. \quad (13)$$

For each set of values a, d we then find the value b, c and e. For each index j such that $i < j < k$ we find the index m strictly between l and n that minimizes $|(at_j + d) - t_m|$. That is, we find the t_m between t_l and t_n closest to the image of t_j under T. We then solve

$$\begin{bmatrix} t_i & x_i & 1 \\ t_j & x_j & 1 \\ t_k & x_k & 1 \end{bmatrix} \begin{bmatrix} b \\ c \\ e \end{bmatrix} = \begin{bmatrix} x_l \\ x_m \\ x_n \end{bmatrix}. \quad (14)$$

Each set of parameters a, b, c, d, e generated by the above processes is plotted in a 5-D space. If the input data contains self-affine structure, then these plotted points will cluster around the parameters of the self-affinity.

Due to the enormous memory required for storing an $O(N^5)$ array we use a two pass algorithm. During the first pass we use a 3-D array corresponding to the map parameters a, b and c. For the second pass, a set of K 2-D parameter plots are created for the K most popular a, b, c parameter clusters discovered during the first pass. All of the parameters a, b, c, d, e are then re-generated in a second pass of the input data and d and e are plotted in the corresponding 2-D array if a, b and c correspond to one of the K most popular parameter clusters.

Candidate maps are extracted from the array entries corresponding to the most popular a, b, c, d, e parameters. At this point, no attempt has been made to automatically create a full description of the approximating RFIF. This would involve selecting the appropriate set of maps T from the candidates map parameters, refining them (the maps may be very crude due to quantization), and defining the proper domain segments. Instead, a program was designed that allowed the user to select candidate maps and determine their appropriate domains manually.

3.2 Least Squares Minimization

Following the technique popularized by fractal image compression [11], given the set of input points $\{(t_i, x_i)\}_{i=0}^{N}$ we seek to find an PFIF with significantly fewer than N maps that approximates the input data.

We first partition the input data into K non-overlapping range segments $\{R_k\}_{k=1}^{N}$. Each range segment contains M input points. We also create a coarser partition of possibly-overlapping domain segments D_j each containing more than M points. For each range segment R_k we iterate through the domain segments, and subsample each such domain segment D_j to produce M samples. We then use least squares minimization to find the parameters of a transformation T that minimizes

$$d(T(D_j), R_k)^2 = \sum_{m=1}^{M} ||T(D_j(m)) - R_k(m)||^2. \quad (15)$$

For each range segment, we save the domain index l and the transformation parameters a, b, c, d and e that best approximate it, forming a PFIF approximation of the input data containing K transformation-domain pairs. If the domain segments can be formed as the union of range segments, then the PFIF structure can be converted to an RFIF.

Fractal curves formed by the separable basis (8) can be approximated by performing the above operation independently on its coordinate functions. However, this results in a range segment of the curve being influenced by two domain segments: one influencing $X(t)$ and another influencing $Y(t)$. Hence, fractal curves are best approximated by partitioning the curve into range curve segments, and again into larger domain segments. Then to determine the fitness of a given domain segment for approximating the current range segment, least squares minimization is used to minimize (15) for the eight parameters in the separable case (8) or the ten parameters in the non-separable case (9).

4 Results

We applied the Hough transform and least squares minimization techniques to an inverse problem and two approximation problems.

4.1 Recovering Parameters of a Fractal Function

The inverse problem for fractal interpolation functions is given the graph of a FIF, recover the parameters of the FIF.

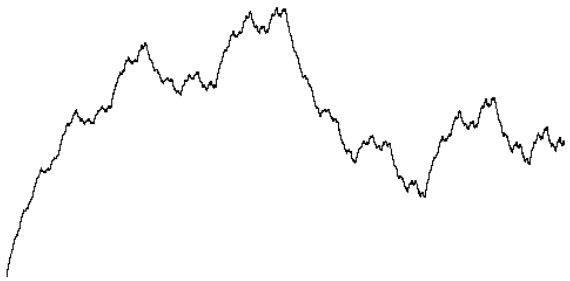

Figure 2: Fractal function with two maps.

Figure 2 is the attractor generated from the parameters in Table 1.

	a	b	c	d	e
T_1	0.5	.746	0.5	-0.5	0.246
T_2	0.5	-0.199	-0.6	0.5	0.189

Table 1: FIF Parameters for Figure 2.

We generate a set of ideal feature points as the set of fixed points of the maps T_i and their images under various compositions. Two such ideal feature point sets were generated containing 65 and 129 points. The resolution of the Hough transform plot was 128^3 for the a, b, c plot used during the first pass and 256^2 for the d, e plot used during the second pass. Each pass, generating 38,519,585 valid maps and took approximately 3 minutes 20 seconds on a MIPS R10000. The first five maps extracted from the Hough transform are shown in Table 2. The top two discovered maps correspond to the transformations in Table 1 within quantization error.

Rank	Hits	a	b	c	d	e
#1	43,657	0.496	0.746	0.496	-0.488	0.251
#2	24,587	0.496	-0.201	-0.600	0.488	0.211
#3	5,637	0.496	-0.911	0.496	0.016	0.512
#4	5,373	0.496	1.337	-0.420	-0.992	-0.571
#5	3,251	0.496	1.456	-0.600	0.016	1.607

Table 2: Five most popular parameter clusters for Figure 2.

4.2 Approximation of a Rough Function

Figure 3 shows an example of a hidden-variable fractal interpolation function. Its projection on the "top" plane is shown in Figure 4 and is claimed to be not self-affine [4]. The goal of our second experiment is to discover a PFIF that approximates this function using the hashing method as a guide.

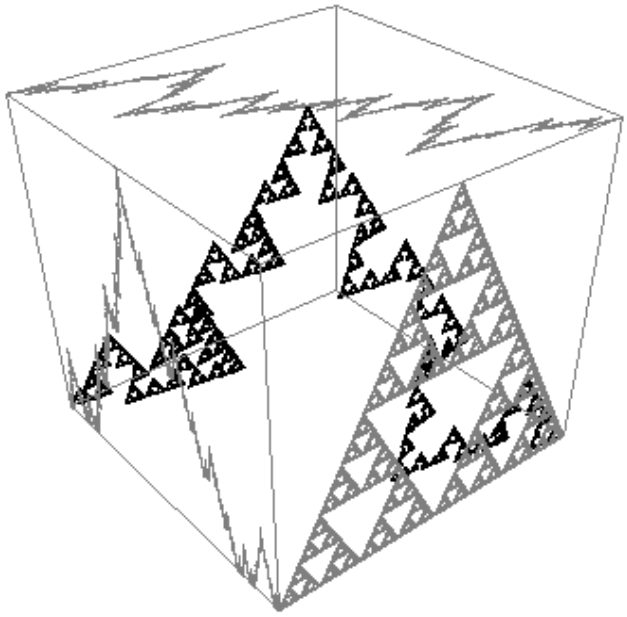

Figure 3: A 3-D Sierpinski space curve and its projections onto three coordinate planes.

A set of low and high resolution features points were constructed by the method described in Section 4.1 containing 55 and 163 feature points respectively.

Figure 5 illustrates the hashing process. The top figure shows the distribution of clusters for the top 100 parameter clusters. A total of $31,032,427$ parameters were plotted. The middle figure shows the original feature points and their image under the most popular extracted map. The lower figure illus-

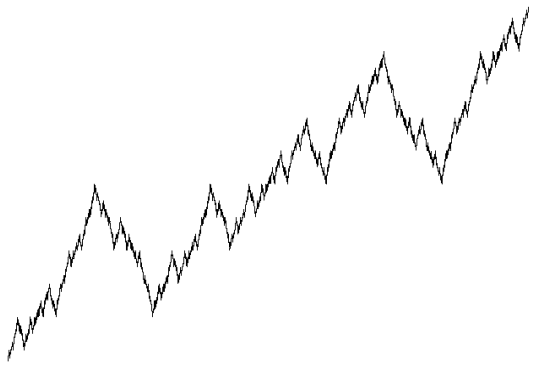

Figure 4: The "top" projection of the Sierpinski space curve.

trates the 5-D hash table projected onto the axes represented by the a and c "scale" parameters.

Many of the maps extracted from the plot were similar due to quantization "spill over" in adjacent bins. These spurious maps could be removed by only considering very localized maxima in the hash table. Also, many of the maps are "echoes" or compositions of other more popular maps. These maps can also be readily identified and removed.

domain	a	b	c	d	e
[0.278, 0.667]	-0.333	0	-0.500	0.222	0.375
[0.167, 0.500]	0.333	0	-0.500	0.074	0.563
[0.167, 0.444]	0.333	0	0.500	0.185	0.063
[0.000, 1.000]	0.333	0	0.500	0.333	0.250
[0.167, 0.444]	-0.333	0	-0.500	0.815	0.938
[0.167, 0.500]	-0.333	0	0.500	0.926	0.437
[0.277, 0.667]	0.333	0	0.500	0.778	0.625

Table 3: PFIF parameters used to approximate the curve in Figure 4.

Seven maps were used to construct the approximating PFIF listed in Table 3. Maps 1, 7, 8, 13, 14, 25, and 30 from the top 100 hash table bins were refined to build the PFIF whose attractor overlays the original in Figure 6.

It is not obvious whether this PFIF can be redefined in terms of a RFIF or if there are other map sets that could be extracted from the hash table that could be used to define an appropriate RFIF.

4.3 Approximation of a Leaf Boundary

We approximated the boundary of the Maple leaf shown in Figure 7 (top) with a fractal curve based on PFIF component functions. First, 261 feature points (x_i, y_i) were assigned to the boundary of the leaf (which contained 4,068 pixels) using the Polyline Splitting technique [12], and each feature point was assigned a uniformly spaced parameter t_i increasing from 0 to 1.

The technique introduced in Section 3.2 was applied to the silhouette curve of the maple leaf, and two resulting approximations of the leaf appear in Figure 7. The more accurate curve on the left was generated by a non-separable RFIF consisting of of 52 maps whereas the less accurate curve on the right was

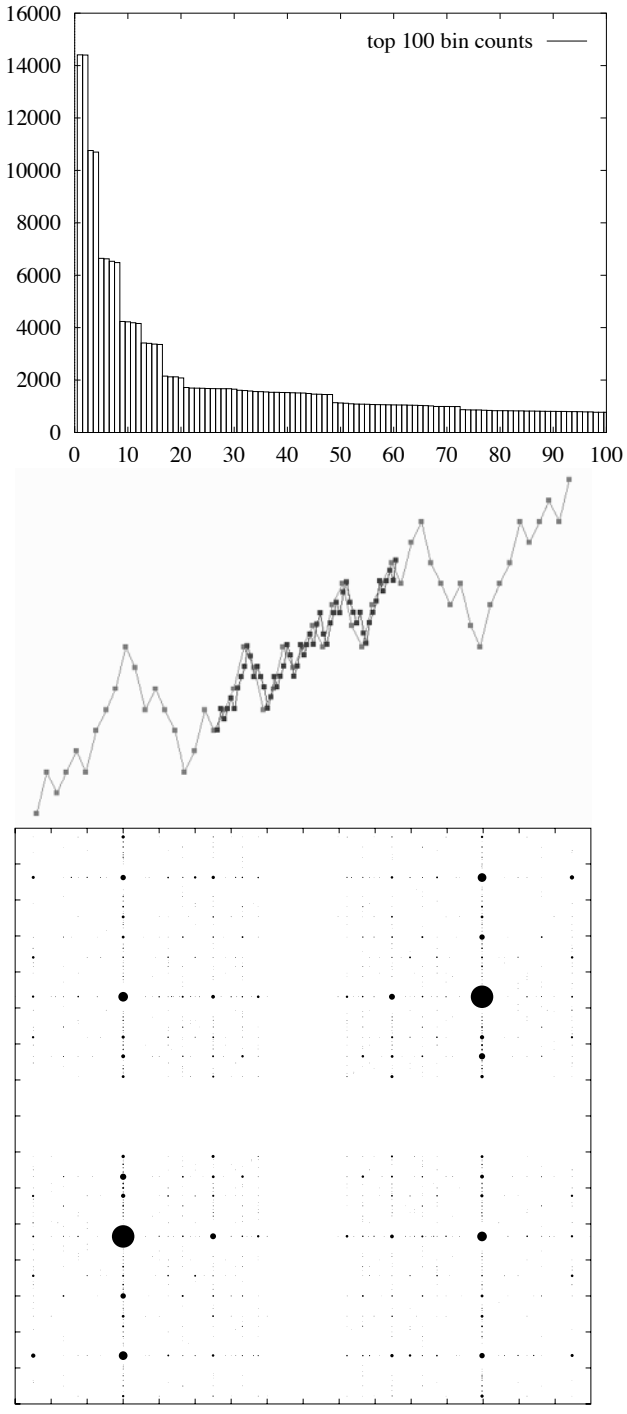

Figure 5: Top: Histogram of the top 100 clusters. Middle: Feature points and a contracted image under the most popular map. (The slight misalignment is due to quantization error.) Bottom: Projection of 5-D hash table for parameters $-0.8 \leq a, c \leq +0.8$. (The size of each circle indicates the popularity of the associated parameters.)

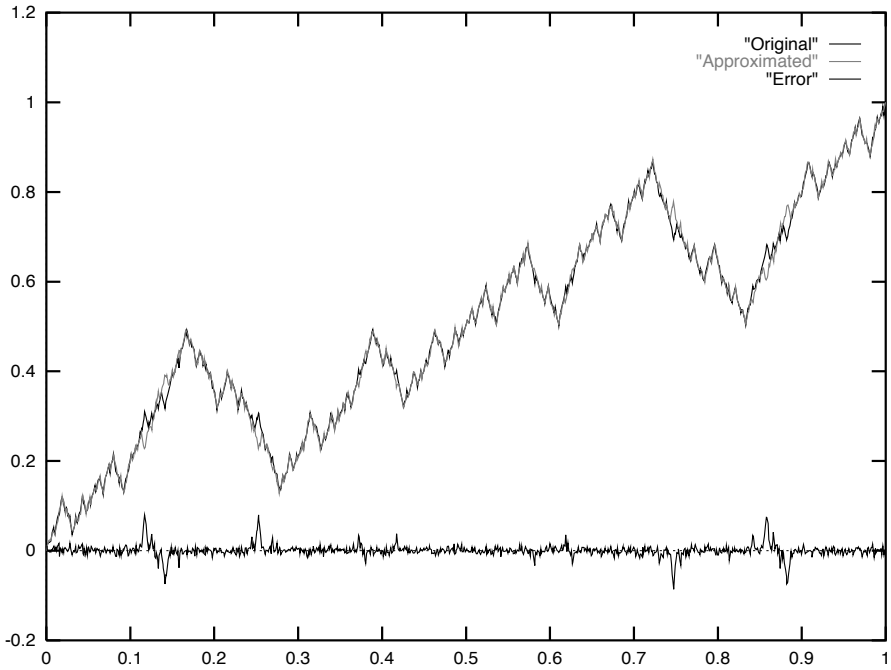

Figure 6: Error of the approximation of the hidden-variable fractal function by a partitioned fractal interpolation function. Maximum error is 0.0858 and mean squared error is 0.000180.

generated by a non-separable RFIF consisting of 29 maps. Notice that the less accurate approximation had fewer maps but its rough surface still appears leaf-like. Fewer maps yields an approximation with the same rough quality but without the same specific roughness as the input.

5 Conclusion

Fractal functions have been shown the potential to be a useful tool in the modeling of 2-D shapes with fractal boundaries. Two automatic techniques were developed to aid the modeler with the unintuitive task of finding self-affinity in a rough curve. These techniques were demonstrated on the tasks of recovering fractal function parameters, approximating rough functions and modeling natural shape boundaries.

5.1 Further Research

This paper served to explore the application of fractal functions on a variety of graphics modeling tasks. While the results suffice to demonstrate the utility of fractal functions, the immediate next step is to perform a detailed error analysis of the models to determine their accuracy.

The Hough transform technique is very sensitive to noise in the input dataset, and techniques for making it more robust will need to be developed before it can become a useful method for detecting self-affinity in natural object boundaries.

Preliminary principal component analysis experiments on the parameter space plotted by the Hough transform shows that typically most of the information happens in a two or three-dimensional subspace. Plotting these subspaces would provide a visual perception of the clustering, a better understanding of the method and lower the memory requirements of the technique.

5.2 Acknowledgments

This research is part of the recurrent modeling project, funded in part by the National Science Foundation grant #CCR-9529809 and a gift from Intel. Thanks also to the GI reviewers for helpful comments.

REFERENCES

[1] Dana H. Ballard and Christopher M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[2] Michael F. Barnsley. Fractal functions and interpolation. *Constructive Approximation*, 2:303–329, 1986.

[3] Michael F. Barnsley, John H. Elton, and D. P. Hardin. Recurrent iterated function systems. *Constructive Approximation*, 5:3–31, 1989.

[4] Michael F. Barnsley, John H. Elton, D. P. Hardin, and Peter R. Massopust. Hidden variable fractal interpolation functions. *SIAM Journal on Mathematical Analysis*, 20(5):1218–1242, September 1989.

[5] Michael F. Barnsley and A. N. Harrington. The calculus of fractal interpolation functions. *Journal of Appoximation Theory*, 57:14–34, 1989.

[6] Michael F. Barnsley and Lyman P. Hurd. *Fractal Image Compression*. AK Peters, Wellesley, MA, 1993.

[7] Yuval Fisher, editor. *Fractal Image Compression: Theory and Applications to Digital Images*. Springer-Verlag, New York, 1994.

[8] Douglas P. Hardin, Bruce Kessler, and Peter R. Massopust. Multiresolution analysis based on fractal functions. *Journal of Appoximation Theory*, 71:104–120, 1992.

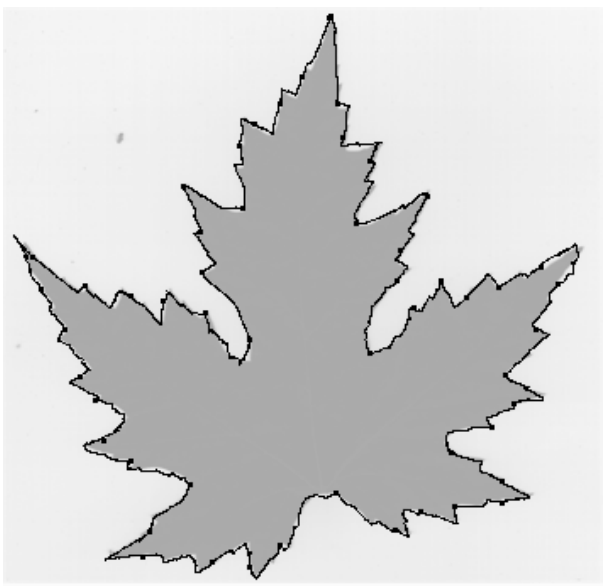

Figure 7: Maple leaf (top) and FIF approximations with 52 control vertices (left) and 29 control vertices (right).

[9] Douglas P. Hardin and Peter R. Massopust. The capacity for a class of fractal functions. *Communications in Mathematical Physics*, 105:455–460, 1986.

[10] John C. Hart, Wayne O. Cochran, and Patrick J. Flynn. Similarity hashing: A model-based vision solution to the inverse problem of recurrent iterated function systems. *Fractals*, 3, May 1997.

[11] Arnaud E. Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, 1(1):18–30, Jan. 1992.

[12] Ramesh Jain, Rangachar Kasturi, and Brian G. Schunck. *Machine Vision*. McGraw-Hill, New York, 1996.

[13] Peter R. Massopust. *Hidden Variable Fractal Interpolation Functions*. PhD thesis, Georgia Institute of Technology, 1986.

[14] Peter R. Massopust. Fractal surfaces. *Journal of Mathematical Analysis and Applications*, 151:275–290, 1990.

[15] Peter R. Massopust. *Fractal Functions, Fractal Surfaces, and Wavelets*. Academic Press, San Diego, California, 1994.

[16] Craig M. Wittenbrink. IFS fractal interpolation for 2D and 3D visualization. In *Proc*. IEEE Visualation '94, pages 77–83, Oct. 1995.

[17] C.E. Zair and E. Tosan. Computer-aided geometric design with IFS techniques. In *Fractal Frontiers (Proc. Fractals '97)*, pages 443–452, Singapore, 1997. World Scientific.

[18] Nailiang Zhao. Construction and application of fractal interpolation surfaces. *The Visual Computer*, 12:132–146, 1996.