Signal Modeling With Iterated Function Systems

A THESIS

Presented to

The Academic Faculty

By

Greg Vines

In Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy in Electrical Engineering

Georgia Institute of Technology

May, 1993

Signal Modeling With Iterated Function Systems

Approved:

to my mother and in memory of my father

ACKNOWLEDGEMENTS

First I would like to thank my parents, and my siblings, who have always been supportive in whatever endeavor I have decided to undertake. In addition to my family, the support and intuitive suggestions of my advisor, Professor Hayes, were crucial in this effort. I would also like to thank reading committee members Professors Mersereau and Schafer as well as committee members Professors Williams and Geronimo for their thoughtful comments on my work. In addition, Professor Malassenet, a.k.a. *Bubba*, provided an invaluable sounding board for many ideas. Thanks also to Armin Kittel for the large variety of one-dimensional data used in Section 4.4.1.

My experience at Tech was enriched tremendously by the time I spent at Georgia Tech Lorraine. The opportunity to live in another country allowed me an educational experience far beyond that achievable in the halls of Van Leer. The gang at GTL, especially Mr. Hanet, whose infinite patience with my feeble attempts at French in the early months in Metz, will always be fondly remembered.

Five years of grad. school leaves one with many memories and friends. A one-page acknowledgement can hardly touch on all of the people and events which have had an influence. There are events and people that stand out, and they follow: Keepin' one wing in the sunshine, Aunt Charlie and her Brunswick Stew, Robbie for keeping the eight-hour rule of concern & going right when I went left. Dino for his driving, `Disturbance Man` and Sheila. Just how many Ph.D.'s does it take to change a tire? E466 – home of the real estate tycoons. And there is always that nagging question... *So so so so so what will you do with the baguette?*

Contents

List of Tables

List of Figures

SUMMARY

A new method of modeling signals has been proposed through the use of Iterated Function Systems (IFSs). Iterated Function Systems are capable of producing complicated functions, many of which closely resemble images and other waveforms that can be found naturally. The task of modeling a given data sequence with an IFS is an ill-posed problem and methods proposed to date have relied on iterative algorithms and using simple affine maps to determine an appropriate IFS. This research focuses initially on modeling one-dimensional signals. Variations of the basic affine map are explored, including the use of nonlinearities on the mapped data, as well as on the addressing of the data in the map. In addition, a condensation type map has been developed, as well as a non-iterative algorithm which is based on characteristics of the data to be modeled. The method used to determine the model parameters is improved through relaxing the boundary conditions which define the interpolation points for the model. All of the enhancements are evaluated with a set of test files and results are given quantifying the improvements with the new methods.

The second portion of this research concentrates on two-dimensional models, specifically the application of image coding. The nonlinear addressing and condensation type maps were extended to the two-dimensional case, as well as the relaxation of the boundary conditions. In addition, three new search techniques were developed based on the local fractal dimension of pixels in the image, a hierarchical block matching algorithm and the correlation between adjacent blocks in an image. An entirely new orthonormal basis approach was introduced which allows multiple domains to be used in each map. Several versions of this method are described, one of which is faster than all previous approaches, at a slight cost in SNR. Slower methods are also introduced, which offer higher reconstructed image quality.

CHAPTER 1

INTRODUCTION

With the ongoing computer revolution, we have become inundated with vast amounts of information. There is an ever present need to transfer and store copious volumes of data. Many methods have been developed to reduce the storage requirements for saving and moving this information. The purpose of this research was to investigate new methods for modeling and compressing data using a recently developed area of mathematics, namely fractals [1]. Fractals have seen much attention due to the pretty pictures and fascinating patterns which can be generated from these simple systems [2]. However, there is more to this field than just an application as a new art form. Preliminary research has shown several potential applications of fractal theory, including filtering [3], image segmentation [4], time-series modeling [5], and image coding [6]. This dissertation deals exclusively with a type of deterministic fractal which is generated from Iterated Function Systems [7].

Iterated Function Systems (IFSs) offer the possibility of high rates of data compression because they are capable of generating complex functions, yet the IFS itself is very simple in form. Initial work with IFSs was in the area of synthesizing images, which were very realistic, and as a consequence of these results much of the enthusiasm was born [8, 9, 10]. Because this data was synthesized, it was stored at extremely high compression rates, in excess of 10,000:1 [10], further fueling the interest in this emerging field. These results were very encouraging for applying this technology to creating new efficient compression algorithms for image data. Because this is a new technique, part of the research effort for this dissertation was directed towards finding the ideal type of data for this method. Previous work has utilized a variety of one and two-dimensional data sources [5, 11].

1.1 Objectives

Most of the research conducted to date has concentrated on IFS models with affine maps, which work well with self-affine data. Unfortunately, naturally occurring data is rarely self-affine. The demonstrated ability of IFSs to generate naturally *appearing* complex data justifies a more in-depth examination of this model. One of the objectives of this research was to derive an efficient IFS model for coding data which is not self-affine. In addition to alleviating the self-affinity constraint, the method by which the map parameters and the maps were constructed was examined and improved.

Another stumbling block in the application of IFSs to signal modeling has been the lack of an efficient algorithm to determine the map parameters. Previous algorithms relied on long searches to find the interpolation points on which to base the model [12]. The development of a faster algorithm to determine the IFS parameters was another goal of this research. The combination of a fast algorithm and the improvements to the IFS maps results in an efficient IFS model.

In summary, the three main objectives were to

- Create an IFS model for non-affine data
- Examine and improve the method by which the maps are constructed
- Create fast algorithm(s) for determining model parameters

This research was divided into two major parts, modeling of one-dimensional and modeling of two-dimensional data, and consequently this dissertation is organized along similar lines. An overview of IFS theory is given in Chapter 2, followed by chapters on one-dimensional modeling background and improvements to these models in Chapters 3 and 4. Several modifications to the standard IFS were explored for the one-dimensional model. A variety of nonlinear maps were evaluated as well as a new algorithm and method to determine the map parameters. Chapter 5 begins the second part of this dissertation with background on two-dimensional modeling. The primary emphasis with the two-dimensional model was the application of image coding, which is covered in Chapter 6. Those modifications that worked well for the one-dimensional case were extended to the two-dimensional case. Because of the increased amount of data in two-dimensions, the necessity of an intelligent and efficient algorithm is crucial. Three new search-based techniques were introduced based on the correlation of adjacent blocks, a hierarchical block matching algorithm, and the fractal dimension of the domain and range blocks. However, all search-based approaches tend to be time-consuming. An entirely new view of the IFS coding problem was proposed based on interpreting each of the terms in the map as basis vectors in a finite dimensional vector space. Based on this viewpoint, a coding scheme was developed which eliminates the time-consuming search required in previous methods. Finally, conclusions and recommendations for future research are given in Chapter 7.

CHAPTER 2

IFS THEORY

Iterated Function Systems were first conceived by Michael Barnsley in 1985 as a method to generate deterministic fractals [13]. The basic theory behind IFSs is very straightforward, and an overview is provided here. Additional material can be found in a number of texts, such as [7] or [14].

2.1 Iterated Function Systems

An IFS is a finite set of contraction mappings, $\{\mathrm{w}_i\}$, defined on a complete metric space, $\mathcal{U}$, with a distance function ρ, i.e.,

$$\mathrm{w}_i : \mathcal{U} \to \mathcal{U} \tag{2.1}$$

for $i = 1, 2, \ldots, P$ with

$$\rho\,[\mathrm{w}_i(\mathbf{x}), \mathrm{w}_i(\mathbf{y})] \leq s_i \cdot \rho[\mathbf{x}, \mathbf{y}] \tag{2.2}$$

for all $\mathbf{x}, \mathbf{y} \in \mathcal{U}$ with $0 \leq s_i < 1$. The notation $\{\mathcal{U} : \mathrm{w}_i, i = 1, 2, \ldots, P\}$ will be used to represent the IFS, and the transformation by all of the maps will be written:

$$W(B) = \bigcup_{i=1}^{P} \mathrm{w}_i(B), \tag{2.3}$$

where $B \subset \mathcal{U}$.

Because W is a contraction, every IFS has a unique fixed point given by [15]

$$A = W(A), \tag{2.4}$$

which is called the attractor and is defined below.

Definition 2.1 (Attractor) *The fixed point, $A = W(A)$, is called the attractor of the IFS.*

A measure of how close an IFS fits a particular data set can be estimated without computing the attractor and is provided by the Collage Theorem [16].

> **Theorem 2.1 (Collage Theorem)** *Let $\{\mathbf{R}^T : \mathrm{w}_i, i = 1, 2, \ldots, P\}$ be an IFS code of contractive T-dimensional affine maps, where $\mathbf{R}$ denotes the set of real numbers. Let $s < 1$ denote the largest contractivity factor for the maps. Let $\epsilon > 0$ be any positive number. Let V be a given closed bounded subset of $\mathbf{R}^T$, and suppose the maps, w_i, have been chosen so that*
>
> $$h[V, W(V)] < \epsilon,$$
>
> *then*
>
> $$h[V, A] < \frac{\epsilon}{1 - s},$$
>
> *where: A denotes the attractor of the IFS, and $h(A, B)$ is the Hausdorff metric, which is a measure useful for comparing binary sets*[1].

The Collage Theorem provides an upper bound on the closeness of fit between a data set, V, and the attractor of an IFS. While this theorem does not help us directly determine the appropriate mappings for a particular data set, it does assure us that if our mappings are individually 'close', then the composite IFS will also closely resemble the data set. This implies the ability to construct the IFS in pieces, as in a collage.

2.1.1 Condensation Maps

A condensation map is one which converges to its fixed point in one iteration. Thus if the map w_0 is a condensation map with fixed point A, then $\mathrm{w}_0(B) = A$, for all B.

[1]The Hausdorff metric is the maximum separation of the furthest elements (either in A or B) from the opposite set.

Essentially the condensation map generates a fixed data set, regardless of the data being transformed by the map. Another way to view the condensation map is not as a map at all, but rather as a stored data sequence which is available to improve the performance of the IFS model. For example, creating a circle is difficult with an IFS, yet by using a condensation map which synthesizes a circle, the problem is easily resolved. An example of the use of a condensation map is given in Figure 2.1 which was created with only two maps, one of which was a condensation map used to create the circle, and the other map rotates and shrinks the entire image.

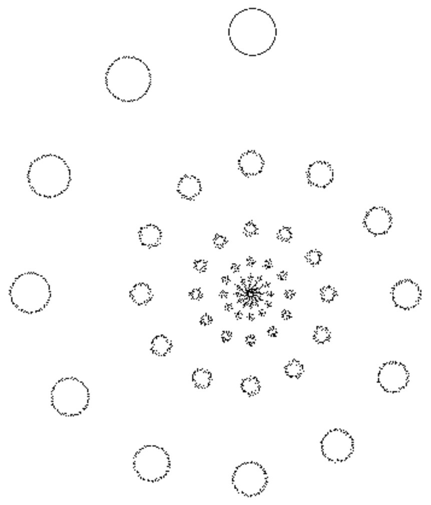


Figure 2.1: Condensation map example

2.2 Attractor Generation

The attractor for an IFS is given by[2]

$$A = \lim_{n \to \infty} W^{\circ n}(B) \tag{2.5}$$

for any $B \subset \mathcal{U}$. There are two methods to take an IFS and generate the attractor, the Deterministic Algorithm and the Random Iteration Algorithm [7].

The Deterministic Algorithm is based on the contractivity of the W mapping and begins by selecting any compact set $A_0 \subset \mathbf{R}^T$, where T is the topographical dimension of the data set. After successively computing $A_n = W^{\circ n}(A)$, the sequence $\{A_j\}_{j=1}^{\infty}$ will converge to the attractor of the IFS. Figure 2.2 shows the first nine iterations in the generation of the attractor for an IFS, and the final attractor.

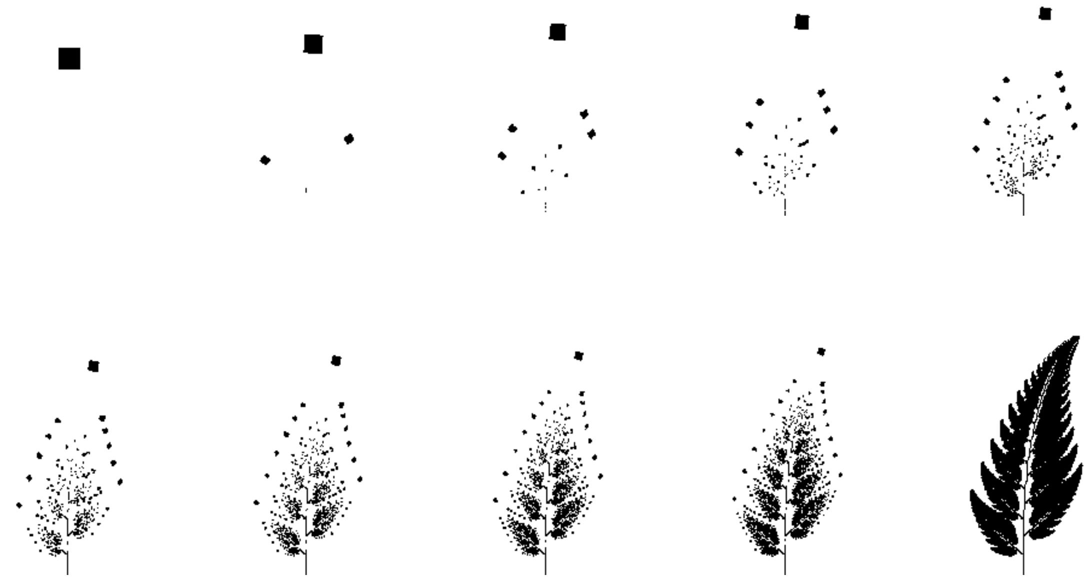


Figure 2.2: Deterministic algorithm example

[2]The notation $W^{\circ n}(A)$ means the n^{th} iterate of A by the function $W()$. For example: $W^{\circ 3}(A) = W(W(W(A)))$.

The Random Iteration Algorithm uses the properties of the dynamical system associated with the w_i's and proceeds with an initial $x_0 \in \mathbf{R}^T$ and then chooses recursively and independently,

$$x_n \in \{\mathrm{w}_1(x_{n-1}), \mathrm{w}_2(x_{n-1}), \ldots, \mathrm{w}_P(x_{n-1})\} \qquad \text{for n} = 1, 2, 3, \ldots \tag{2.6}$$

where the probability of selecting w_i is p_i. The sequence $\{x_n\}_{n=1}^{\infty}$ will converge to the attractor of the IFS [17]. Thus, in each iteration, a map is selected at random and the point x_n is transformed by that map. In the limit, the path of x_n will trace out the attractor. The probabilities, p_i, need to be assigned to each map before this algorithm can be used. Because these probabilities are used in distributing the generated points, they should relate how much of the total attractor comes from each mapping. A simple method to assign probabilities is to look at the relative scaling performed by each mapping and assign a fraction for each mapping:

$$p_i = \frac{|\det T_i|}{\sum_{j=1}^{N} |\det T_j|}, \tag{2.7}$$

where T_i is the transformation matrix for the i^{th} map using the following notation. For the two-dimensional case, such as in the examples given here, the affine maps may be written,

$$\mathrm{w}_i(\mathbf{x}) = \mathrm{w}_i \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix} = T_i\mathbf{x} + \mathbf{t}_i, \tag{2.8}$$

where the T_i matrix performs a linear transformation on the input vector, $\mathbf{x}$, and $\mathbf{t}_i$ provides a translation.

If for some i the value of p_i is 0, then the Random Iteration Algorithm will never allocate any points to that map, therefore, a small value should be assigned to this p_i to insure that this map is used. Figure 2.3 illustrates several intermediate steps in the generation of an attractor with the Random Iteration Algorithm.

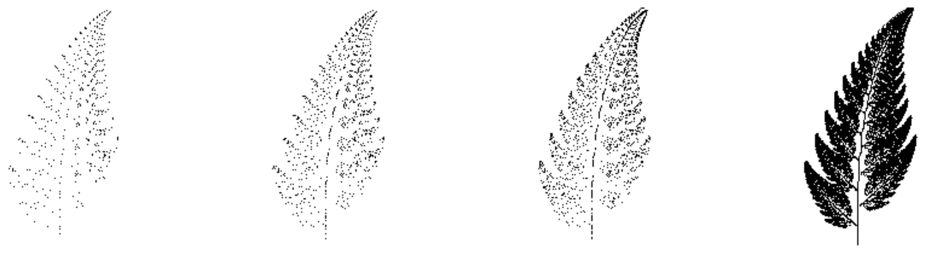


Figure 2.3: Random iteration algorithm example

While both of these algorithms are capable of synthesizing an attractor given a set of maps, there are implementation differences worth noting. Because the Random Iteration Algorithm follows a single point through the attractor, the memory requirements are minimal. The disadvantage of this method is that the attractor is only traced in the limit. For the discrete case, to insure that every required point has been determined, the algorithm must run for many iterations. In addition, some implementations of an IFS intentionally restrict the amount of data that is transformed by each map. In this case, because not every point is transformed by each map, the selection of the next map is dependent upon the address of the present point. Therefore, besides complicating the algorithm, it is also possible for the sequence of selected maps to become limited to a subset of the entire collection of maps. Maps which do not operate on the entire data set are called piecewise maps and are discussed in Section 3.1.1.

On the other hand, with the Deterministic Algorithm, because every point on the attractor is calculated at every iteration, it is possible to calculate with certainty when the algorithm can stop for a given size and resolution of the reconstructed attractor. For example, with maps having a contractivity factor of 0.5, to insure convergence in generating an attractor with eight intensity levels would require three iterations. Each iteration comes within one-half of the final value of the attractor. The disadvantage of the Deterministic Algorithm is the increased memory requirement – the entire attractor must be stored in memory. In this work, the Deterministic Algorithm was used almost exclusively. The largest quantity of data being worked with at one time was a 512×512 image, which easily fit in the memory of the Sun workstations that were used. In a commercial implementation of this method, this memory requirement issue may become a factor, in which case the Random Iteration Algorithm could be adapted to overcome the shortcomings mentioned here with some simple checks on the path of the calculations.

2.3 Conclusion

In this chapter it was shown that an IFS is a simple collection of contraction mappings and that the Collage Theorem provides a hint of how to construct maps to model a given data sequence. Finally, two methods were discussed to render the attractor given an IFS. What is needed now, and is covered in the next chapter, is a method to construct the maps to model time-series, or one-dimensional data.

CHAPTER 3

BACKGROUND FOR ONE-DIMENSIONAL MODEL

In the one-dimensional case, the data must be interpreted in a manner conducive to IFS modeling. In this chapter, a generic finite length sequence, $\{x[n]\}_{n=0}^{N-1}$ is considered. After introducing the one-dimensional model itself, the issue of constructing an IFS for discrete data is addressed. Variations on the basic model are discussed as well as an algorithm for determining a set of maps.

3.1 Model

One-dimensional IFS modeling is based on selecting points from the data samples, $\{x[n]\}_{n=0}^{N-1}$, and creating IFS maps to interpolate between these points. If we consider the set of data to be modeled as a set of points of the form $\{(n, x[n]) : n = 0, 1, \ldots, N-1; x[n] \in \mathbf{R}\}$, then a function could be created to interpolate between selected sample points, $\{(m_i, x[m_i]) : m_0 < m_1 < \ldots < m_P\}$, closely following the shape of the original data sequence. By using an IFS to perform this interpolation, the data could be reconstructed from the IFS maps. In this manner, the IFS interpolation function is set up such that the graph of this function will pass through the interpolation points, $(m_i, x[m_i])$. A graph is defined below.

Definition 3.1 (Graph) *Let $I, Y \subset \mathbf{R}$, Let $f : I \rightarrow Y$ be a function. The* graph *of f is the set of points*

$$G = \{(x, f(x)) \in \mathbf{R} \times Y : x \in I\}.$$

Thus, one-dimensional data is set up as a two-dimensional graph for the IFS to model. This is accomplished by constructing the maps in a manner that creates a single-valued function that passes through the interpolation points, using P affine maps of the form

$$\mathrm{w}_i\left(\begin{bmatrix} n \\ x[n] \end{bmatrix}\right) = \begin{bmatrix} a_i & 0 \\ c_i & d_i \end{bmatrix}\begin{bmatrix} n \\ x[n] \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix}, \tag{3.1}$$

for $i = 1, 2, \ldots, P$. The zero term in the above matrix forces the function to be single valued in the sense that for any n, there is only one $x[n]$ [18], and therefore the reconstructed data is forced to be a graph. This equation can also be expressed as two separate equations, one for $x[n]$ and the other for n,

$$\mathrm{w}_{ix}(x[n]) \quad = \quad c_i \cdot n + d_i \cdot x[n] + f_i, \tag{3.2a}$$

$$\mathrm{w}_{in}(n) \quad = \quad a_i \cdot n + e_i \ . \tag{3.2b}$$

Each map, being two-dimensional, must be a contraction in both the n and the $x[n]$ directions. A contraction in n is assured because the i^{th} map transforms N samples into M_i samples, where $N > M_i$. The contractivity in $x[n]$ is controlled by the parameter d_i. Thus, the map w_i will be a contraction mapping if $|d_i| < 1$ and d_i is called the *contraction factor* for the i^{th} map. The parameter d_i is independent of the interpolation points and is used to control the shape of the interpolation function. The remaining parameters, a_i, c_i, e_i, and f_i, are constrained so that $x[0]$ and $x[N-1]$ are mapped by w_i to the selected sample points for this map, $x[M_{i1}]$ and $x[M_{i2}]$ respectively; the ith map projects the N samples onto the M_i samples $[M_{i1}, M_{i2}]$.

The interpolation point constraints can be written as:

$$\mathrm{w}_i\left(\begin{bmatrix} 0 \\ x[0] \end{bmatrix}\right) = \begin{bmatrix} M_{i1} \\ x[M_{i1}] \end{bmatrix} \text{ and } \mathrm{w}_i\left(\begin{bmatrix} N-1 \\ x[N-1] \end{bmatrix}\right) = \begin{bmatrix} M_{i2} \\ x[M_{i2}] \end{bmatrix}, \tag{3.3}$$

for $i = 1, 2, \ldots, P$. As a result of the end point constraints, the following map parameters may be written as a function of d_i:

$$a_i = \frac{(M_{i2} - M_{i1})}{N-1} \quad , \quad e_i = M_{i1}, \tag{3.4a}$$

$$c_i = \frac{(x[M_{i2}] - x[M_{i1}])}{(N-1)} - d_i \frac{(x[N-1] - x[0])}{(N-1)}, \tag{3.4b}$$

$$f_i = x[M_{i1}] - d_i \cdot x[0] \; . \tag{3.4c}$$

The final parameter to be determined is the contraction factor, d_i, which is chosen to minimize the distance between the original data sequence and the transformed data sequence for each map. Two methods have been proposed for selecting the contraction factor, a geometric approach and a least squares approach [19]. Because both methods provide similar results, we will concentrate on the least squares approach which determines the contraction factor by minimizing the error function

$$\xi = \sum_{n=M_1}^{M_2} (\mathrm{w}_{ix}(x[n]) - x[n])^2, \tag{3.5}$$

where $\mathrm{w}_{ix}(x[n])$ is the equation from the map w_i which transforms $x[n]$, as defined in equation (3.2a).

Figure 3.1 illustrates the generation of an attractor for a four map IFS. The initial data set consisted of a large square in the space K shown in the figure. Each of the maps transforms all of the points in K during each iteration. With each iteration, the data is mapped in a contractive fashion closing in on the attractor. Because the maps have been constructed in the form of equation (3.1), the resulting attractor is a graph. This figure illustrates how any initial data set would yield the same attractor due to the contractive nature of the IFS. In addition, the self-affinity characteristic of this model can be seen in this example. Data which is constructed with this model will be self-similar. This means that subintervals of the data sequence are equal to a scaled and translated version of the entire data sequence. This self-similar characteristic is one of the identifying traits of fractals [7].

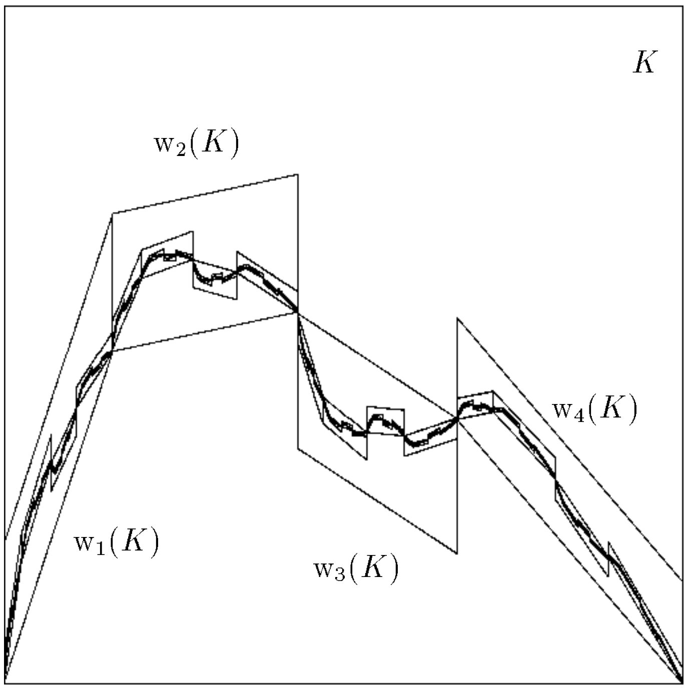


Figure 3.1: Four map IFS

3.1.1 Piecewise Maps

The linear fractal interpolation model produces self-affine data using a set of affine functions that map the entire data sequence onto sections of itself. A more general form of similarity, known as *piecewise self-affinity*, is a property in which *intervals* of the data sequence are mapped onto intervals [20]. Specifically, given a set of N sample points, $\{x[n]\}_{n=0}^{N-1}$, the interpolation function is constructed with P affine maps of the form of equation (3.1). However, in place of the end point constraints for the self-affine model given by equation (3.3), for the piecewise self-affine model, a subset of the interval $[0, N-1]$ is mapped between each pair of sample points:

$$\mathrm{w}_i\left(\begin{bmatrix} N_{i1} \\ x[N_{i1}] \end{bmatrix}\right) = \begin{bmatrix} M_{i1} \\ x[M_{i1}] \end{bmatrix} \text{ and } \mathrm{w}_i\left(\begin{bmatrix} N_{i2} \\ x[N_{i2}] \end{bmatrix}\right) = \begin{bmatrix} M_{i2} \\ x[M_{i2}] \end{bmatrix}, \qquad (3.6)$$

for $i = 1, 2, \ldots, P$.

Thus, for each i, the function w_i maps the signal $x[n]$ over the interval $[N_{i1}, N_{i2}]$ onto the interval $[M_{i1}, M_{i2}]$. The interval $[N_{i1}, N_{i2}]$ is called the address interval because it can be thought of as the address of the source data for this map. The contraction factor for each map is again constrained to be bounded by one in magnitude: $|d_i| < 1$. In addition, to insure a contraction mapping, the width of each address interval associated with a map is constrained to be strictly greater than the width of the section associated with the map:

$$N_{i2} - N_{i1} > M_{i2} - M_{i1} \text{ for } i = 1, 2, \ldots, P. \tag{3.7}$$

Once the contraction factors, addresses and interpolation points have been chosen, the remaining map parameters are easily determined from the end point constraints given by equation (3.6). Synthesis of the piecewise self-affine fractal interpolation function may be accomplished in a manner similar to the Deterministic Algorithm with some slight modifications [20].

3.1.2 Hidden Variable Fractal Interpolation

The Hidden Variable Fractal Interpolation (HVFI) method is based on modeling the data in a higher dimensional space. The data is then recovered by taking a projection of the reconstructed higher dimensional data [21]. The advantage of this method is that data which is not self-similar can be created due to the additional flexibility of the added dimension.

For a one-dimensional sequence, the HVFI method would model the data in $\mathbf{R}^3$. Given the original data sequence, $x[n]$, which consists of N data samples, the data for the additional dimension, $z[n]$, would have to be determined prior to modeling. With a given $x[n]$ and $z[n]$, the i^{th} map out of P maps is set up in the following manner:

$$\mathrm{w}_i\left(\begin{bmatrix} n \\ x[n] \\ z[n] \end{bmatrix}\right) = \begin{bmatrix} a_i & 0 & 0 \\ c_i & d_i & h_i \\ k_i & l_i & m_i \end{bmatrix} \begin{bmatrix} n \\ x[n] \\ z[n] \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ g_i \end{bmatrix}. \tag{3.8}$$

The zero terms are again necessary to insure the resulting function is single valued. The i^{th} map covers M_i data samples for $M_{i1} \leq n \leq M_{i2}$. The set of P maps are constructed to cover the entire data sequence. Figure 3.2 shows three views of the attractor for a hidden variable IFS. This method has seen little use due to the difficulty in determining the appropriate $z[n]$ data to use. One example, given in [22], uses a circularly shifted version of $x[n]$ for the data set $z[n]$. Although it is also possible to construct piecewise HVFI functions, because of the large number of parameters, no one has yet attempted to model data with this method.

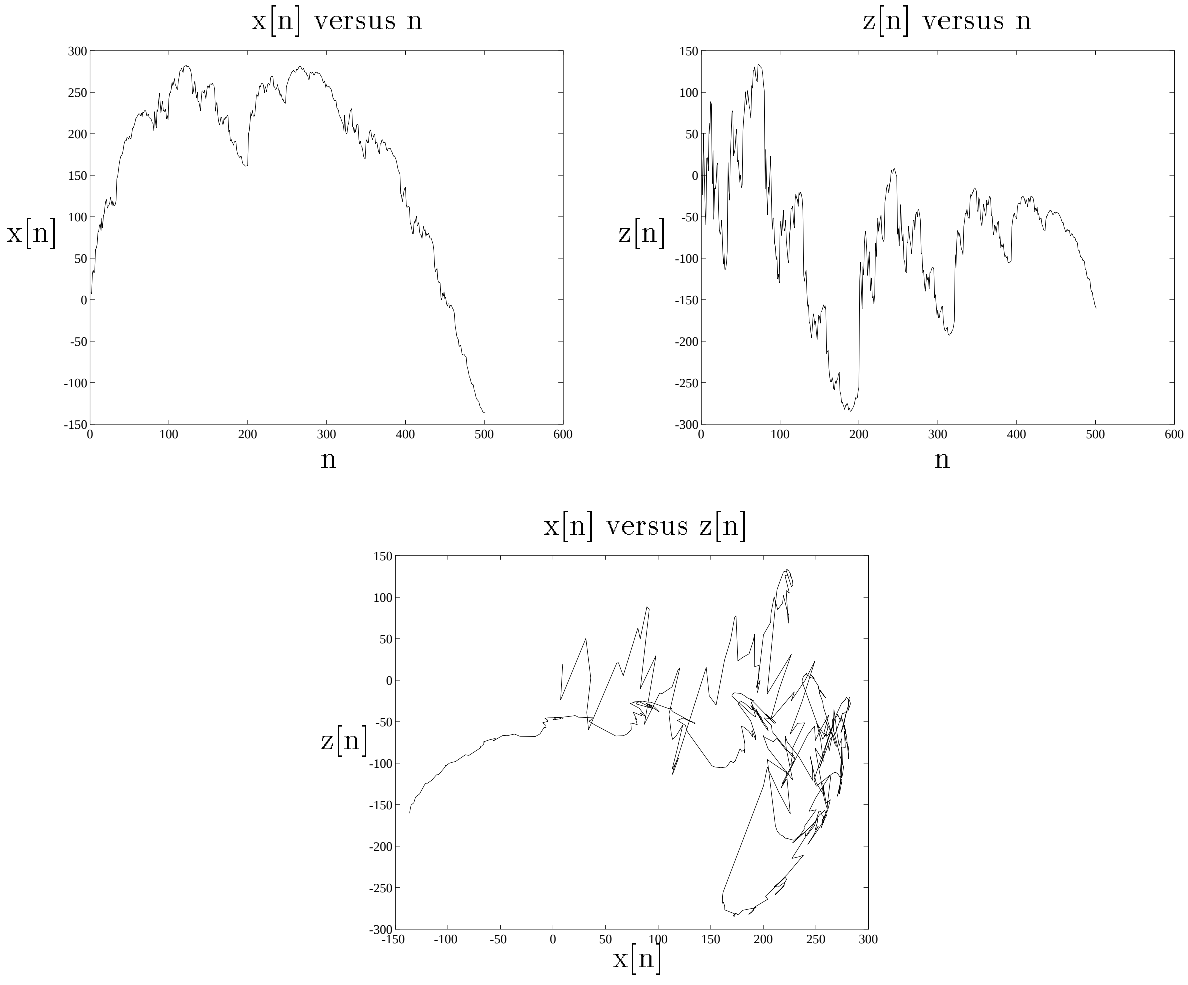


Figure 3.2: Attractor for hidden variable IFS

3.2 Discrete Data

In the continuous case, the operation of the maps on a data set is well defined. Each point in a data set is transformed by the maps in the IFS to another point. The location and value of the mapped data is exactly specified by the map equation. However, when the transition to discrete data is made, some potential problems can arise. Because the location for each sample in the data set must be quantized as it is transformed by the maps of the IFS, there can be cases where the address for the transformed sample does not correspond exactly to any of the actual samples.

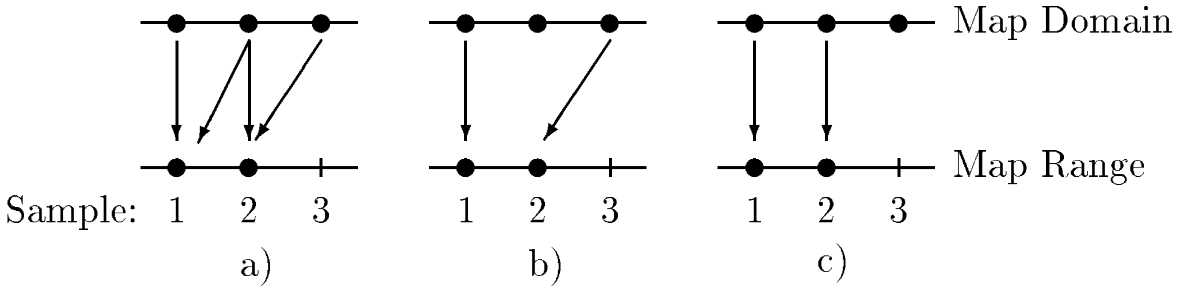


Figure 3.3: Discrete mapping example for 3:2 contraction mapping using: a) averaged, b) non-averaged, and c) non-averaged and non-contractive sampling.

Consider Figure 3.3, illustrating a map that transforms three samples into two. The addressing portion of this map can be written as

$$\mathrm{w}_{in}(n) = \frac{2}{3} \cdot n \; . \tag{3.9}$$

Note that the actual scaling of the samples is not important for this example. The question is which samples to use in the transformation equation (3.2a). The sample is referenced by a ubiquitous "$x[n]$". However, in this case the address n is not an integer. There are numerous ways to perform the transformation, three of which are shown here. The problem stems from the fact that when the contraction ratio, which is equal to 1/contraction factor, is not integral, an approximation must be made in determining the address for the transformed data. In this example, the transformation should take the j^{th} sample to the $\frac{2}{3}j^{th}$ sample, which results in non-integral indices when transforming the data. In Figure 3.3 a) an averaging scheme is used, that has been shown to work well [6]. Here the samples which correspond to `ceil`(n) and `trunc`(n) are averaged, where `ceil`(n), the ceiling operator, returns the smallest integer $\geq n$, and `trunc`(n), the truncation operator, returns the largest integer $\leq n$. It is also possible to avoid averaging if care is taken in determining the index, as is shown in b) and c). Note that c) is *not* a contraction because the first and second samples are always mapped to themselves. This can be avoided with careful use of rounding when calculating the index in the map calculations. Furthermore, this entire issue can be avoided by restricting the contraction ratios to be integer, as is done in the two-dimensional case, as will be seen in Chapter 5.

3.3 Inverse Problem

The *Inverse Problem* is that of how to take a given data set and determine the set of maps whose attractor will closely approximate the original data. This problem has turned out to be extremely difficult to solve, primarily because the determination of the map interpolation points is an ill-posed problem. While an IFS will produce a unique attractor, for a given attractor there are many IFSs which can produce this attractor. An example of this is the Cantor Middle Thirds set as shown in Figure 3.4. This set can be produced by an infinite number of IFSs. For example two such systems

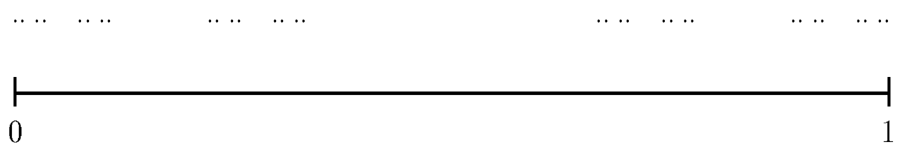


Figure 3.4: Cantor middle thirds set

are:

$$\{[0,1] : \mathrm{w}_1(x) = \frac{1}{3}x, \mathrm{w}_2(x) = \frac{2}{3} + \frac{1}{3}x\} \tag{3.10}$$

and

$$\{[0,1] : \mathrm{w}_1(x) = \frac{1}{9}x, \mathrm{w}_2(x) = \frac{2}{9} + \frac{1}{9}x, \mathrm{w}_3(x) = \frac{2}{3} + \frac{1}{3}x\}. \tag{3.11}$$

Therefore, given a data sequence, for an efficient model it is desirable to find the IFS that has the fewest number of maps and also accurately represents the data. Several obvious approaches have serious drawbacks, as discussed below.

A brute force approach, checking every possible map combination, turns out to be computationally impractical. For a fixed number of maps, the number of possible mappings is

$$\frac{n!}{k!(n-k)!}, \tag{3.12}$$

where $n = N - 2$, N is the number of points in the data set, and k is the number of maps minus one. For example, with a 128 point data sequence and five maps, the evaluation of just over ten million map combinations is required. As the number of maps is increased by one, the number of computations goes up by approximately two orders of magnitude for this example. Obviously, an exhaustive search for the best map is not feasible. However, exhaustive searches are used in the next chapter for evaluating the IFS model enhancements when the number of maps is small.

Another logical method is to use a gradient search for the best interpolation points. Unfortunately the error surface is extremely rough, and not conducive to gradient methods. A series of test files was created for evaluating the enhancements to the IFS model, a complete description of which is given in Section 4.4.1. To illustrate the error surface as a function of the map interpolation points, an exhaustive search was performed for a three map IFS modeling the first of these test files. The error was then plotted for each of the sets of interpolation points, thereby illustrating the error surface.

The maps were constructed so as to have two free variables: the two interpolation points for the second map. The first map, w_1, used the interpolation points $\{0, M_{21}\}$, and the second map's interpolation points were denoted as $\{M_{21}, M_{22}\}$. The third map had interpolation points $\{M_{22}, N-1\}$ to complete the IFS model. Thus for this three map IFS, there are two free variables, M_{21} and M_{22}. The remaining map parameters were determined by minimizing the error equation (3.5) as discussed in Section 4.1.1, the specifics of which are not important at this point. Using these two variables, an error can be defined as,

$$\text{error} = -\log(f(M_{21}, M_{22})), \tag{3.13}$$

where the negative log of the error was used because we are interested in minimizing the error, and visually seeing the low point on a plot is difficult. This error surface is shown from an isometric view in Figure 3.5 and as a contour plot in Figure 3.6.

This error surface does have a certain regularity to it; there are definite valleys and ridges, which suggests that good results may be obtained by searching along each axis. In [19] such a search method has been proposed. The method begins with one interpolation point fixed and advances the other interpolation point through the data to be mapped looking for the best map. This would correspond to searching along the error surface along one axis for the maximum. The best map is selected and the data covered by that map is not examined during subsequent searches. The remaining data is searched again by fixing one interpolation point at the edge of the remaining data to be mapped and repeating the search. In this manner, a set of maps is determined to model the data. This method has the disadvantage that there is no way to specify the number of maps to use, and it also does not take into consideration the interaction of the maps; each map is chosen independently. That is to say, by picking the peak of the error surface as we scan along one direction, we are limiting ourselves to a small subset of all the local extrema.

Figure 3.5: Error surface for three map IFS of test file 1, isometric view

3.4 Conclusion

In this chapter an IFS model was introduced for modeling one-dimensional sequences. Several variations on the model were introduced, including piecewise maps and the Hidden Variable Fractal Interpolation method. An algorithm for determining the model parameters was discussed, as well as some of the issues involved in determining a set of maps. In the next chapter, this model is expanded, and two new algorithms are introduced.

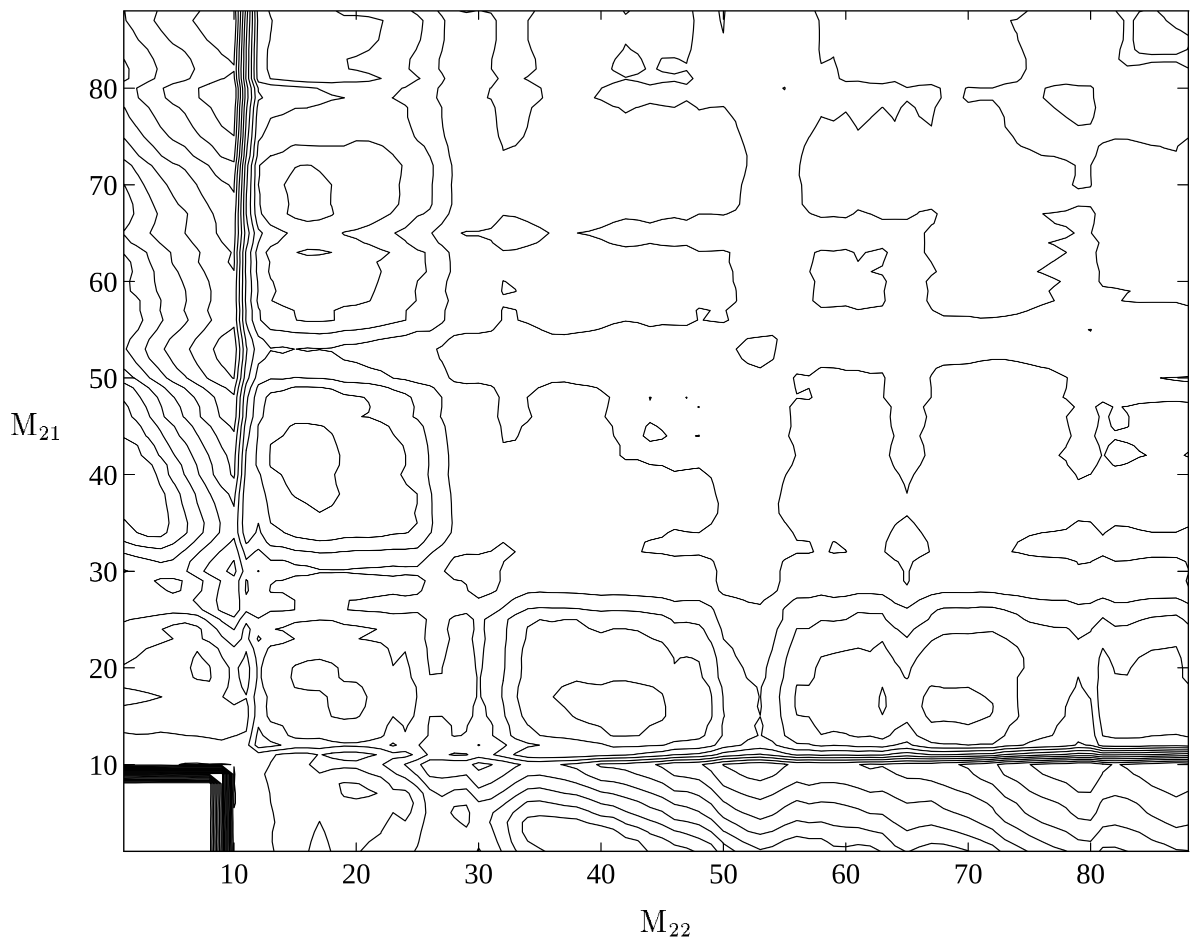


Figure 3.6: Error surface for three map IFS of test file 1, contour view

CHAPTER 4

ONE-DIMENSIONAL MODELING

This chapter examines a series of modifications to the basic IFS affine map, as well as algorithms to determine the necessary model parameters. In addition, the method by which the IFS model is constructed is explored.

4.1 Improvements to IFS Interpolation Method

In addition to the modifications to the individual maps, the following changes were investigated in the structure of the IFS model itself. The first concerns how the maps are set up to interpolate specific data points, and the second examines the selection criteria for the maps.

4.1.1 Relaxing the Boundary Conditions

In the fractal interpolation methods proposed to date, the IFS parameters have been calculated by forcing the model to intersect the interpolation points exactly. In modeling a one-dimensional sequence, these boundary conditions have been used to reduce the number of free variables, leaving only d_i, the *contraction factor*, to be calculated, as discussed in Section 3.1. By not imposing the boundary conditions, the number of variables increases from one, d_i, to three, c_i, d_i and f_i, but the model will be more flexible to fit the data [23]. These three variables can be determined with a least squares minimization of the error equation (3.5). Substituting equation (3.2a) into

(3.5) yields,

$$\xi = \sum_{j=0}^{M_i-1} [x[M_{i1}+j] - (c_i \cdot n + d_i \cdot x[n] + f_i)]^2, \tag{4.1}$$

where

$$n = \text{int}(\frac{N-1}{M_i-1}j). \tag{4.2}$$

Taking the partial derivatives of ξ with respect to each of these three variables, and setting these partials to zero,

$$\frac{\partial \xi}{\partial c_i} = 0 \ , \ \frac{\partial \xi}{\partial d_i} = 0 \ , \ \frac{\partial \xi}{\partial f_i} = 0, \tag{4.3}$$

provides the following set of linear equations:

$$\sum_{j=0}^{M_i-1} A \cdot \chi = \sum_{j=0}^{M_i-1} K_0 \cdot \mathbf{K}, \tag{4.4}$$

where

$$A = \mathbf{K} \cdot \mathbf{K}^T, \tag{4.5}$$

$$\mathbf{K} = [n, x[n], 1]^T, \tag{4.6}$$

$$\chi = [c_i, d_i, f_i]^T, \tag{4.7}$$

n is defined in equation (4.2) and ^{T} signifies the transpose operator.

This method has the advantage of not forcing the synthesized data to intercept the sampled data at any point exactly, instead the synthesized data is made to fit as well as possible *overall.* As a result, the overall error is not sacrificed in order to meet the interpolation point constraints. The choice of interpolation points is somewhat arbitrary as they have been chosen in order to minimize the error with the constraint that the model actually touches these points. The existence of this constraint is not necessary and in judging the fidelity of the modeled data, one data sample typically is no more important than another in achieving a good fit.

4.1.2 Map Selection Criteria

The Collage Theorem has been used extensively as a basis for map selection [6, 24, 25]. However, this is usually done by simply taking the map with the smallest distance measure as given by equation (3.5). The Collage Theorem also shows that a small contraction factor is desirable for an accurate model. For a one-dimensional signal, the IFS model is two-dimensional, so there is a contraction along the two axes of the graph. There must be a contraction in n, and the contraction factor along this axis is N_i/M_i, and there must also be a contraction in $x[n]$, which is equal to d_i. These contraction factors are presently ignored in the map selection process, other than the binary decision of whether or not the maps are contractions. To test the improvement of including these values in the map selection process, a variable α was defined,

$$\alpha = k_1 \cdot \frac{N_i}{M_i} + k_2 \cdot d_i + k_3 \cdot \xi_i, \tag{4.8}$$

and maps were selected based on a minimum α. For a given weighting scheme, more emphasis could be placed on the contractivity of the map. Numerous combinations of weights were tested, from setting all of the weights to the same value, $k_1 = k_2 = k_3$, to individually emphasizing each term separately, for example, $k_1 = 1, k_2 = k_3 = 0$. No improvements were noticed with any of the weighting schemes used.

The Collage Theorem also dictates the use of the Hausdorff metric in specifying the error bound. This is not a convenient metric because it involves comparing many points in the two sets, which can become computationally intensive for large sets. In order to find a suitable metric, the L_1, L_2 and L_∞ norms were tested, with similar results obtained for each.

Because the results obtained were unaffected by different selection criteria and distance measures, all maps were selected based on the Euclidean distance, ξ, from the error equation (3.5) alone. The Euclidean distance is a convenient metric when solving for parameters with least squares.

4.2 Non-Affine Maps

The basic affine map limits the type of data which can be generated to be self-affine in nature. As mentioned before, this is not a common characteristic of naturally occurring data. While the HVFI and the piecewise methods offer ways to eliminate this restriction, modifications to the basic affine maps are also possible. The affine map, equation (3.2a), is repeated here for reference,

$$\mathrm{w}_{ix}(x[n]) = c_i \cdot n + d_i \cdot x[n] + f_i. \tag{4.9}$$

There are many possible nonlinearities which can be introduced into this equation. We are interested primarily in two opposing desires: the first is to allow maximum flexibility in transforming a data set to fit another data set, and the second is to keep the number of map parameters to a minimum, or to construct the map so as to minimize the storage requirements. Additionally, it is desirable to be able to easily determine the best map parameters. In order to achieve this goal, nonlinear equations were implemented which are linear in the coefficients, thereby allowing least squares to be used to determine the parameters.

4.2.1 Polynomial Nonlinearities

The affine map of equation (4.9) was modified with the addition of various nonlinear terms in both $x[n]$ and n. Each of these equations was linear in the coefficients. In addition, all of the derivations here are done for piecewise maps. Because the piecewise case is more complicated, these results have been provided, with the understanding that the non-piecewise case can easily be derived with a simple substitution of variables:

$$N_{i1} = 0 \quad \text{and} \quad N_{i2} = N - 1, \tag{4.10}$$

thereby fixing the end points for the map addresses to the entire data sequence.

A) Simple Quadratic IFS

The first nonlinear map was created by adding a quadratic term to the affine map equation (4.9) to get:

$$\mathrm{w}_{ix}(x[n]) = c_i \cdot n + d_i \cdot x[n] + h_i \cdot x[n]^2 + f_i. \tag{4.11}$$

As with the affine maps, the boundary conditions can be used to eliminate some of the variables, leaving two unknowns: d_i and h_i, which can be solved for using a least squares minimization of the error.

Least Squares Solution

The two boundary equations can be written as:

$$x[M_{i1}] = c_i \cdot N_{i1} + d_i \cdot x[N_{i1}] + h_i \cdot x[N_{i1}]^2 + f_i \tag{4.12a}$$

and

$$x[M_{i2}] = c_i \cdot N_{i2} + d_i \cdot x[N_{i2}] + h_i \cdot x[N_{i2}]^2 + f_i. \tag{4.12b}$$

These equations can be solved for c_i and f_i to get:

$$c_i = \frac{x[M_{i2}] - x[M_{i1}]}{N_{i2} - N_{i1}} + d_i \cdot \frac{x[N_{i1}] - x[N_{i2}]}{N_{i2} - N_{i1}} + h_i \cdot \frac{x[N_{i1}]^2 - x[N_{i2}]^2}{N_{i2} - N_{i1}} \tag{4.13a}$$

and

$$\begin{aligned} f_i \;=\; & \frac{N_{i2} \cdot x[M_{i1}] - N_{i1} \cdot x[M_{i2}]}{N_{i2} - N_{i1}} + d_i \cdot \frac{N_{i1} \cdot x[N_{i2}] - N_{i2} \cdot x[N_{i1}]}{N_{i2} - N_{i1}} \\ & + h_i \cdot \frac{N_{i1} \cdot x[N_{i2}]^2 - N_{i2} \cdot x[N_{i1}]^2}{N_{i2} - N_{i1}}. \end{aligned} \tag{4.13b}$$

The error equation (3.5), with the simple quadratic map substituted, may be written as

$$\xi = \sum_{j=0}^{M_i - 1} [x[M_{i1} + j] - (c_i \cdot n + d_i \cdot x[n] + h_i \cdot x[n]^2 + f_i)]^2, \tag{4.14}$$

where

$$n = N_{i1} + \operatorname{int}\left(\frac{N_i - 1}{M_i - 1} j\right), \tag{4.15}$$

$$M_i = M_{i2} - M_{i1} + 1 \quad \text{and} \quad N_i = N_{i2} - N_{i1} + 1. \tag{4.16}$$

By substituting equations (4.13a) and (4.13b) into (4.14), then calculating the partial derivatives of ξ with respect to the two unknowns, d_i and h_i, and setting the partials equal to zero:

$$\frac{\partial \xi}{\partial d_i} = 0 \quad , \quad \frac{\partial \xi}{\partial h_i} = 0, \tag{4.17}$$

the following equation is obtained:

$$\sum_{j=0}^{M_i-1} A \cdot \chi = \sum_{j=0}^{M_i-1} K_0 \cdot \mathbf{K} \tag{4.18}$$

where

$$A = \mathbf{K} \cdot \mathbf{K}^T \tag{4.19}$$

$$\mathbf{K} = [K_1, K_2]^T \tag{4.20}$$

$$\chi = [d_i, h_i]^T \tag{4.21}$$

and

$$K_0 = -\left(x[M_{i1} + j] + \frac{x[M_{i1}](n - N_{i2}) + x[M_{i2}](N_{i1} - n)}{N_{i2} - N_{i1}} \right) \tag{4.22a}$$

$$K_1 = \frac{x[N_{i1}](N_{i2} - n) + x[N_{i2}](n - N_{i1})}{N_{i2} - N_{i1}} - x[n] \tag{4.22b}$$

$$K_2 = \frac{x[N_{i1}]^2(N_{i2} - n) + x[N_{i2}]^2(n - N_{i1})}{N_{i2} - N_{i1}} - x[n], \tag{4.22c}$$

and n is defined in equation (4.15).

Test for a Contraction Mapping

Whereas the contractivity of an affine map may be easily checked, the above quadratic is a little more complicated. One condition which will indicate a contraction mapping is if the fixed point is an attracting hyperbolic fixed point of prime period one [26][1]. This is true when, for a given function $f(x)$, $|f'(x)| < 1$. Applying this to equation (4.11) yields:

$$|d_i + 2h_i \cdot x[n]| < 1. \tag{4.23}$$

[1]Prime period one means that the fixed point lies on an orbit of period one, and an attracting hyperbolic fixed point ensures convergence.

In order to facilitate calculations, all of the data to be modeled is normalized. With a maximum magnitude of one, the inequality can be reduced by considering only the extreme values for $x[n]$. Therefore the inequality (4.23) can be written as

$$|d_i \pm 2h_i| < 1. \tag{4.24}$$

Each mapping that is generated must satisfy the inequality (4.24) in order to be a contraction mapping.

Relaxed Boundary Conditions

The method of relaxing the boundary conditions, Section 4.1.1, can also be utilized here. Taking equation (4.11) and using least squares without the boundary equations, yields the same equations, (4.18) and (4.19) with the new variables:

$$\mathbf{K} = [\ n\ ,\ x[n]\ ,\ x[n]^2\ ,\ 1\ ]^T \tag{4.25}$$

$$\chi = [\ c_i\ ,\ d_i\ ,\ h_i\ ,\ f_i\ ]^T \tag{4.26}$$

with

$$K_0 = x[M_{i1} + j]. \tag{4.27}$$

The contraction mapping test remains the same.

B) Additional Quadratic Terms

The simple quadratic equation (4.11) was supplemented with additional terms as follows:

$$\mathrm{w}_{ix}(x[n]) = c_i \cdot n + d_i \cdot x[n] + g_i \cdot n^2 + h_i \cdot x[n]^2 + k_i \cdot x[n] \cdot n + f_i. \tag{4.28}$$

Again the boundary conditions can be used to eliminate some of the variables, leaving four unknowns: d_i, g_i, h_i, and k_i, which can be determined, as before, with a least squares minimization of the error.

Following the above procedure, equations (4.18) and (4.19) would remain the same, with the new variables:

$$\mathbf{K} = [\ K_1\ ,\ K_2\ ,\ K_3\ ,\ K_4\ ]^T \tag{4.29}$$

$$\chi = [\ d_i\ ,\ g_i\ ,\ h_i\ ,\ k_i\ ]^T \tag{4.30}$$

with

$$K_0 = -\left(x[M_{i1}+j] + \frac{x[M_{i1}](n-N_{i2}) + x[M_{i2}](N_{i1}-n)}{N_{i2}-N_{i1}} \right) \tag{4.31a}$$

$$K_1 = x[n] + \frac{x[N_{i1}](n-N_{i2}) + x[N_{i2}](N_{i1}-n)}{N_{i2}-N_{i1}} \tag{4.31b}$$

$$K_2 = n^2 + \frac{N_{i1}^2(n-N_{i2}) + N_{i2}^2(N_{i1}-n)}{N_{i2}-N_{i1}} \tag{4.31c}$$

$$K_3 = x[n]^2 + \frac{x[N_{i1}]^2(n-N_{i2}) + x[N_{i2}]^2(N_{i1}-n)}{N_{i2}-N_{i1}} \tag{4.31d}$$

$$K_4 = n \cdot x[n] + \frac{N_{i1} \cdot x[N_{i1}](n-N_{i2}) + N_{i2} \cdot x[N_{i2}](N_{i1}-n)}{N_{i2}-N_{i1}} \tag{4.31e}$$

and n is defined in equation (4.15).

Test for a Contraction Mapping

The conditions under which we have a valid map are determined again by ensuring that the fixed point is a attracting hyperbolic fixed point of prime period one. Taking the derivative of equation (4.28) with respect to $x[n]$, and again incorporating the normalized data we are dealing with, yields the constraint

$$|d_i + k_i \cdot n \pm 2h_i| < 1. \tag{4.32}$$

Each map that is used must satisfy this inequality over the range of $N_{i1} \leq n \leq N_{i2}$. This provides four inequalities to check:

$$|d_i + k_i \cdot N_{i1} + 2h_i| < 1 \tag{4.33a}$$

$$|d_i + k_i \cdot N_{i1} - 2h_i| < 1 \tag{4.33b}$$

$$|d_i + k_i \cdot N_{i2} + 2h_i| < 1 \tag{4.33c}$$

$$|d_i + k_i \cdot N_{i2} - 2h_i| < 1 \tag{4.33d}$$

Relaxed Boundary Conditions

Again the method of relaxing the boundary conditions, Section 4.1.1, can be applied. Solving equation (4.11) with least squares yields the same equations, (4.18) and (4.19) with the new variables:

$$\mathbf{K} = [\ n\ ,\ x[n]\ ,\ n^2\ ,\ x[n]^2\ ,\ n \cdot x[n]\ ,\ 1\ ]^T \tag{4.34}$$

$$\chi = [\ c_i\ ,\ d_i\ ,\ g_i\ ,\ h_i\ ,\ k_i\ ,\ f_i\ ]^T \tag{4.35}$$

and

$$K_0 = x[M_{i1} + j]. \tag{4.36}$$

The contraction mapping test remains the same.

C) Higher Order Polynomials

As more terms were added to the map equation, it was found that fewer and fewer maps were contraction mappings. Because so few maps could be used with the addition of cubic terms, no benefit in model accuracy was seen.

The disadvantage of the previous sections method of adding terms to the maps is that all of the additional coefficients must be stored. The next method provides a less expensive way to permute the data as it is transformed.

4.2.2 Nonlinear Addressing

The idea behind nonlinear addressing is to take the source data sequence and add a nonlinearity in the addressing of these points as they are mapped onto the region $[M_1, M_2]$ [27]. Let this nonlinear function be called $g(n)$, so that $g(n)$ returns another index into $x[n]$ within the range $[0, N-1]$. We can then write equation (3.2a) as

$$\mathrm{w}_{ix}(x[n]) = c_i \cdot n + d_i \cdot x[g(n)] + f_i. \tag{4.37}$$

In an interest of keeping the additional information required to describe the nonlinearity as small as possible, the number of nonlinear functions was limited. Thus each map only requires a code to specify which function is used in that map. The set of functions given in Table 4.1 was used. Note that the first function is the standard linear addressing method. The second flips the data in n, while the remaining stretch and compress various portions of the data. For those functions which return non-integer values, the nearest integer is used as the address. The appropriate addressing function would be selected by evaluating the map error equation (3.5) for each of the functions and selecting the nonlinearity that gives the smallest error.

Table 4.1: Nonlinear addressing functions

#	Function
0	n
1	$N - n$
2	$N \cdot sin(\pi/2 \cdot n/N)$
3	$N \cdot cos(\pi/2 \cdot n/N)$
4	$N \cdot [1.0 - sin(\pi/2 \cdot n/N)]$
5	$N \cdot [1.0 - cos(\pi/2 \cdot n/N)]$
6	$N \cdot sin(\pi \cdot n/N)$
7	$N \cdot [1.0 - sin(\pi \cdot n/N)]$

4.2.3 Condensation Type Maps

Because the attractor of an IFS will have a built-in roughness to it, it is difficult to model smooth curves [28]. Conversely, although the condensation map with sinusoidal data allows a map to code a segment of smooth data with a known and fixed error, it does not take advantage of any self-affinity in the data. A hybrid approach to the condensation map was evaluated which allowed the incorporation of the advantages of both the pure condensation map and the normal IFS map [29]. The addition of the condensation data function to the map itself combines these two features. A sinusoidal condensation function was used, and was added to the map equation (3.2a) to get:

$$\mathrm{w}_{ix}(x[n]) = c_i \cdot n + c1_i \cdot sin(\frac{2\pi n}{N-1}) + d_i \cdot x[n] + f_i. \tag{4.38}$$

The coefficients for the condensation terms were determined in a similar manner to the other map parameters. In addition, the boundary conditions were relaxed to improve the model's performance.

4.3 Quantization

By carefully constructing the coded maps, the number of required bits can be reduced. First, bit allocation is examined for the standard affine map, followed by the alternative maps covered thus far.

4.3.1 Affine Maps

In the basic affine map there are five parameters: a, e, c, d, and f. The a and e values can be determined from equation (3.2b) using the map end point addresses, M_1 and M_2, whose size depends on the number of samples in the data being modeled. If each map is adjacent and not overlapping, then only one address will have to be stored for each map because M_2 for one map will be M_1 for the adjacent map. In addition, the first and last map will have known end point addresses – the first and last data sample. Therefore, to determine a and e for P maps will require $(P-1) \times N_B$ bits, where N_B is the number of bits in the address.

Because d is constrained by $|d| < 1.0$ for a contraction mapping, this suggests storing d as a signed fixed-point fraction. This requires one bit for the sign, and additional bits for the binary fraction portion.

The data transformation equation (3.2a) may be interpreted in two parts, a linear interpolation function,

$$\mathrm{w}(x[n]) = c \cdot n + f, \tag{4.39}$$

combined with the mapped data portion, $d \cdot x[n]$. This is shown in Figure 4.1 as the mapped data being laid on a linear interpolation between the map end points. If the data is normalized prior to modeling, the bias in the linear interpolation portion of the map, f, must have sufficient range to bias the smallest $d \cdot x[n]$ value to the highest $x[n]$, and also the highest $d \cdot x[n]$ value to the lowest $x[n]$. Because d is restricted to $|d| < 1$, and $x[n]$ is restricted to the range $0 \leq x[n] \leq 1$, the maximum range of f is $-2 \leq f \leq 2$, and f can be stored in a similar manner as d with the addition of a single bit for the 1's place. The slope, c, depends on the smallest map width: $|c| < 2.0/M$. Examination of many IFS models has shown that most c's are within the range $10^{-2} < |c| < 10^{-7}$. This range limitation is taken advantage of by using an exponent with a binary offset of 7, so c was stored in the format $\pm 0.bbb \times 2^{-(7+exp)}$, where exp is a four bit number, giving c the desired range. With this coding method a minimum of 6 bits are required for c: four for the exponent, one for the sign, and one or more for the mantissa.

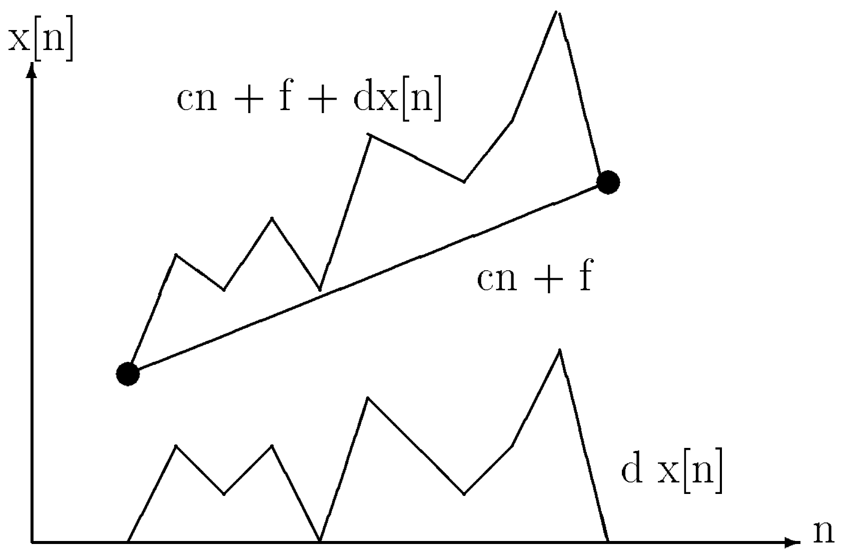


Figure 4.1: Example of affine map illustrating how the transformation can be interpreted as a linear interpolation combined with the mapped data.

Tests were run on a series of IFSs and there were no noticeable quantization effects when using eight bits for c, six bits for d and seven for f. In none of these tests were any underflow or overflow seen for the c exponent.

4.3.2 Non-affine Maps

Unlike the affine case, with polynomial terms the magnitude of the parameters is not strictly limited. Therefore a more liberal quantization scheme was used. For each of the parameters, both the polynomial and the condensation terms, an eight bit format, like the one discussed above for c, was implemented. For each parameter the exponent ranges were determined from a set of IFSs.

4.4 Results

All of the above mentioned one-dimensional models were tested with a series of data sequences. A set of standard test files was created to test all of the various algorithms and modifications to the basic IFS. Before running any tests, it was first necessary to determine a suitable type of data to model with an IFS. Once this was determined, the test files were constructed and a uniform testing methodology was used to obtain results which facilitated comparisons.

4.4.1 Test Files and Methodology

A known characteristic of fractal data is a spectral shape of an exponential decay [4]. This shape is commonly seen in nature with stochastic processes and has been given the name of $1/f$ noise [30]. In determining an appropriate class of data to apply IFS modeling techniques to, it is important to use a compatible type of data. Fortunately, such a type of data exists and is the focus of many modeling and compression techniques – image data. Because image data is an ideal application for IFS modeling, and is so prevalent, this research effort has focused on this application. For the one-dimensional models, scan lines from images were used, and in the two-dimensional case, the entire image was used.

For one-dimensional data samples, scan lines were extracted from a series of image files, such as LENA and the CAMERA MAN from the USC database. A set of five files was gathered and all tests were conducted with these files. Figures 4.2a through 4.2e show the original files.

In order to evaluate each of the modifications, exhaustive searches were conducted with a fixed number of maps. This was performed with two through five maps for most of the tests. Occasionally the computational burden of five maps was too much, and the tests were halted at four. Table 4.2 gives the true SNR values for the test files coded with each of the nonlinear modifications covered thus far, as well as with the original affine IFS for comparison. The SNR is defined as

$$10 \log \left(\frac{\sum_{n=0}^{N-1} x[n]^2}{\sum_{n=0}^{N-1} (x[n] - \hat{x}[n])^2} \right), \tag{4.40}$$

where $\hat{x}[n]$ is the noise corrupted version of the signal.

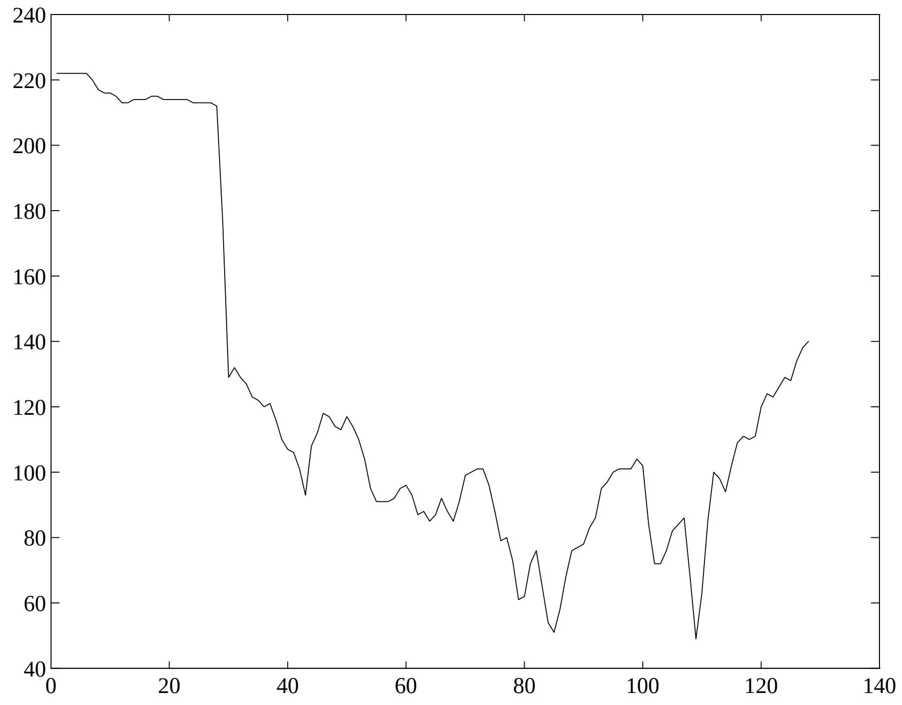


Figure 4.2a: Test file 1

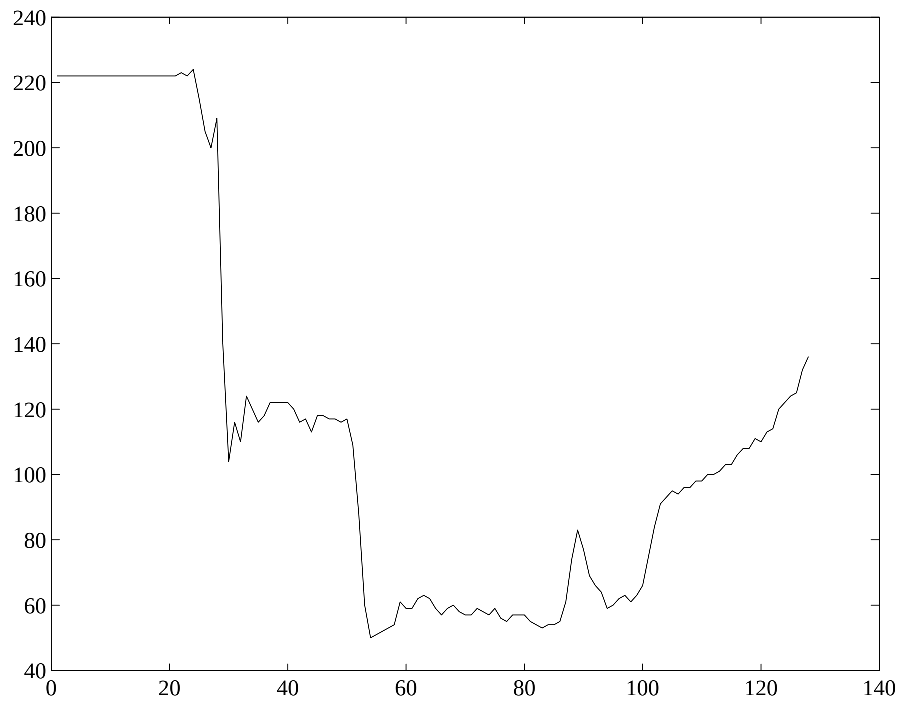


Figure 4.2b: Test file 2

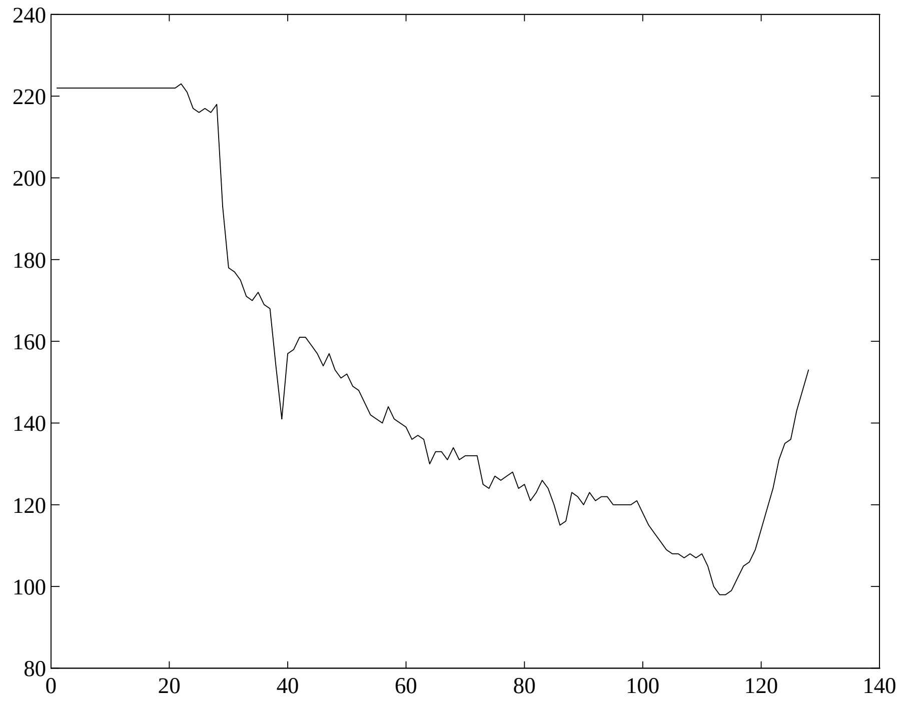


Figure 4.2c: Test file 3

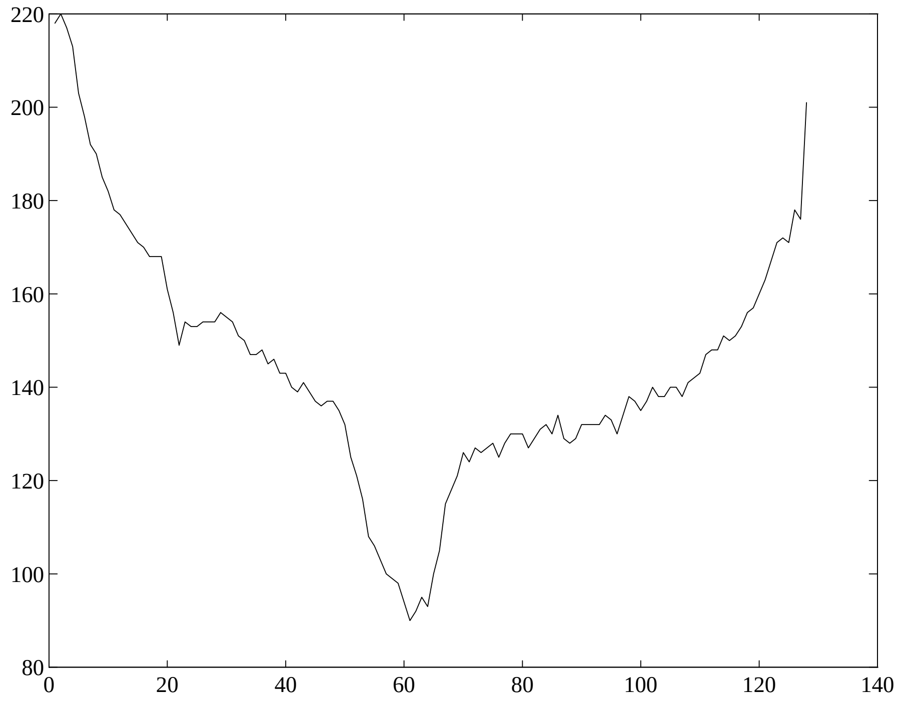


Figure 4.2d: Test file 4

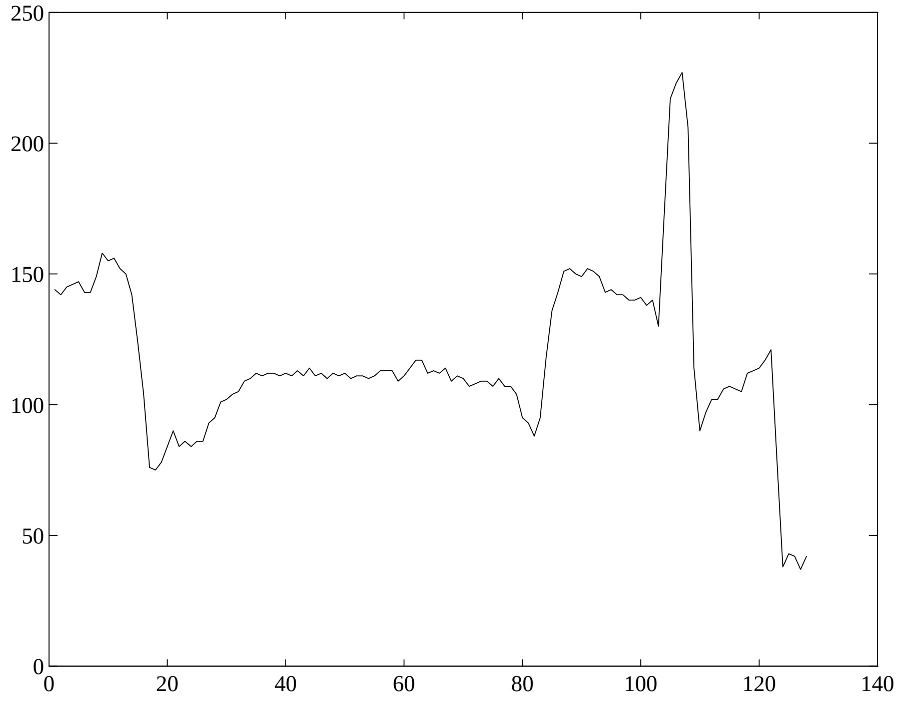


Figure 4.2e: Test file 5

Table 4.2: 1D IFS nonlinear map comparison

File	Method	2 Maps SNR	3 Maps SNR	4 Maps SNR	5 Maps SNR
1	Affine	19.4	21.2	23.4	25.0
	Relaxed Affine	20.0	23.2	24.5	26.0
	Simple Quadratic	19.4	21.2	23.5	25.2
	Full Quadratic	19.9	22.5	24.4	27.5
	Relaxed Full Quadratic	20.2	23.9	25.3	27.6
	Nonlinear Addressing	21.5	24.2	26.0	
	Condensation Maps	21.1	23.5	25.0	
2	Affine	19.5	24.6	24.7	26.9
	Relaxed Affine	22.6	25.6	26.8	28.5
	Simple Quadratic	19.5	24.5	24.9	27.1
	Full Quadratic	20.3	24.6	28.3	28.9
	Relaxed Full Quadratic	22.6	25.9	27.6	30.0
	Nonlinear Addressing	22.6	25.8	27.3	
	Condensation Maps	23.2	25.9	27.4	
3	Affine	24.2	27.8	32.4	35.0
	Relaxed Affine	25.4	31.0	33.3	36.1
	Simple Quadratic	24.2	28.3	33.3	35.7
	Full Quadratic	26.4	31.5	35.7	36.7
	Relaxed Full Quadratic	25.4	33.4	36.2	37.2
	Nonlinear Addressing	27.8	33.9	34.9	
	Condensation Maps	26.1	33.5	35.8	
4	Affine	24.3	26.5	30.2	32.9
	Relaxed Affine	26.4	30.4	33.0	34.4
	Simple Quadratic	24.2	26.6	30.3	33.3
	Full Quadratic	30.5	35.3	37.1	39.3
	Relaxed Full Quadratic	27.8	35.9	37.4	39.6
	Nonlinear Addressing	28.5	32.7	34.4	
	Condensation Maps	26.7	32.1	34.2	
5	Affine	11.9	15.6	16.9	18.5
	Relaxed Affine	14.2	17.2	19.4	21.1
	Simple Quadratic	11.9	15.6	17.1	18.7
	Full Quadratic	15.2	17.3	21.5	22.3
	Relaxed Full Quadratic	15.8	18.5	20.9	22.9
	Nonlinear Addressing	14.3	17.6	20.1	
	Condensation Maps	16.2	18.5	21.1	

Examination of these results shows that all of the nonlinearities improved the performance of the model. In addition, relaxing the boundary conditions improved each of the maps even further. However, it is important to take into consideration the increased storage requirements for these new and more complicated maps. Table 4.3 shows the calculated compression ratios for each of the methods with two through five maps. Further analysis of these results is given in Section 4.7.

Table 4.3: 1D IFS map compression ratios

Method	2 Maps	3 Maps	4 Maps	5 Maps
	CR	CR	CR	CR
Affine	20.9	13.3	9.8	7.7
Relaxed Affine	20.9	13.3	9.8	7.7
Simple Quad.	15.8	10.1	7.5	5.9
Relaxed Simple Quad.	15.8	10.1	7.5	5.9
Full Quad.	10.6	6.9	5.1	4.0
Relaxed Full Quad.	10.6	6.9	5.1	4.0
Nonlinear Addressing	18.6	11.9	8.8	6.9
Condensation Maps	15.8	10.1	7.5	5.9

4.5 Algorithms for Interpolation Points

Presently there are two methods for determining the interpolation points; first, the exhaustive search, which has only been used here for evaluating the various improvements to the IFS model, and second, the previously mentioned search algorithm [19]. Using exhaustive searches is too time consuming to be of practical use. The disadvantage of the search-based method, is that there is no firm control over the number of maps used in modeling a data sequence. In this section we will introduce two new approaches to determining the interpolation points, which allow the number of maps in the model to be fixed.

It has been shown that in two-dimensional binary self-affine data, the vertices of the convex hull enclosing the attractor are also the fixed points of the maps for the IFS which generated the attractor [31]. This result was used as the basis for an algorithm for determining the interpolation points for one-dimensional data [32]. The convex hull is expensive to compute, and instead, the local extrema of the data sequence were used as the fixed points of the maps. The extremum points are obtained by comparing a low-pass filtered version of the data to the original data and selecting those points which differ the most between the two data sequences. The first and last points in the data sequence are vertices of the convex hull and must also be fixed points for the first and last map, respectively. Therefore, in order to determine the interpolation points for P maps, an additional $P-2$ extremum points are required. The locations of the fixed points do not uniquely define the interpolation points, as is shown in the derivation below. However, they greatly reduce the number of possible sets of interpolation points, thereby simplifying the map selection process.

The update equation (4.15) for n will have a fixed point, $\hat{n}_i$, for the i^{th} map when

$$\hat{n}_i = a_i \cdot \hat{n}_i + e_i. \tag{4.41}$$

Each w_i must map the end points of the data sequence, 0 and $N-1$, onto the map end points, M_{i1} and M_{i2}. From equation (4.15), this gives

$$e_i = M_{i1} \quad , \quad a_i = \frac{M_{i2} - M_{i1}}{N}, \tag{4.42}$$

and therefore equation (4.41) can be written as

$$\hat{n}_i = \frac{M_{i2} - M_{i1}}{N} \cdot \hat{n}_i + M_{i1}. \tag{4.43}$$

Thus, given the fixed point of a map and either end point, the remaining end point can be determined.

The first map has the additional known constraints $M_{11} = 0$ and $\hat{n}_1 = 0$, which, when used in equation (4.43), give no additional information about M_{12}. In addition, the last map has the known constraints, $\hat{n}_P = N-1$ and $M_{P2} = N-1$, which also yield no information about M_{P1}. The intermediary maps have the two interpolation points as unknowns, and one equation for each map, as given by equation (4.43). Because each map is adjacent, for the remaining $P-2$ maps, there are $P-1$ unknowns and we have $P-2$ equations. In addition, the interpolation points are constrained so that the fixed point must lie between the interpolation points for that map, and the interpolation points must be increasing:

$$M_{i1} < \hat{n}_i < M_{i2} \quad , \quad M_{i2} = M_{(i+1)1}. \tag{4.44}$$

Unfortunately this does not give a unique solution to the problem. However, there are enough restrictions on the interpolations points, so that the number of possible solutions is greatly reduced. Because the selection of the interpolation points is an ill-posed problem, this leaves the option of evaluating all possible sets of interpolation points, which leads to the first of the two algorithms.

4.5.1 Fixed-Point Algorithm

The first method begins by selecting M_{12} for the first map. All possible interpolation points are evaluated. While at first this may appear to be a large number, in fact it typically is quite small, being equal to $\hat{n}_2-1$. This is because M_{12} must be less than the next fixed-point, $\hat{n}_2$, which must be in the next map. Once M_{12} is selected, then the remaining interpolation points can be determined from equation (4.43) and knowing the remaining $\hat{n}_i$'s. If a valid set of interpolation points can not be determined from M_{12}, then that value is skipped and the search continues. Once there is a complete set of interpolation points, then the map parameters are determined. Subsequent sets of interpolation points are compared by the squared error between the original data and the transformed data, as in equation (3.5), with the map set which produced the lowest error being saved as the final set. If a minimum map size is imposed, such as five samples, then the search is reduced further.

4.5.2 Geometric Algorithm

The second algorithm is not based on the aforementioned proof, instead it is motivated by a geometric construction of the model. The map equation (3.2a) may be viewed as a linear interpolation portion,

$$\mathrm{w}_{in}(x[n]) = c_i \cdot n + f_i, \tag{4.45}$$

combined with a mapped data portion, $d_i \cdot x[n]$. The maps are initially set up to approximate the original data with a linear interpolation. This is implemented by taking the extremum points as the interpolation points exactly. In this manner, even without the effects of the IFS, the model will roughly follow the original data using a series of straight-line segments. Then, the mapped data portion can be used to further increase the accuracy of the model. As with the previous algorithm, the map parameters are determined through a least squares error minimization. This method avoids most of the computations of the first method completely, and consequently is extremely fast, depending on the number of maps, the coding time can be ten to twenty times faster than the fixed-point algorithm.

4.5.3 Results

The results are given in Table 4.4, which shows the best possible performance through the exhaustive search followed by each of the two algorithms introduced here.

Table 4.4: IFS algorithm comparison

File	Method	2 Maps SNR	3 Maps SNR	4 Maps SNR	5 Maps SNR
1	Exhaustive search	20.0	23.2	24.5	26.0
	Fixed-point alg.	14.0	20.7	21.2	22.9
	Geometric alg.	18.2	19.8	21.6	23.0
2	Exhaustive search	22.6	25.6	26.8	28.5
	Fixed-point alg.	8.2	18.9	22.3	22.8
	Geometric alg.	22.6	21.0	22.9	24.0
3	Exhaustive search	25.4	31.0	33.3	36.1
	Fixed-point alg.	22.8	27.2	32.9	33.9
	Geometric alg.	24.0	25.7	27.0	29.1
4	Exhaustive search	26.4	30.4	33.0	34.4
	Fixed-point alg.	11.9	22.8	31.0	32.4
	Geometric alg.	25.1	27.3	31.1	31.8
5	Exhaustive search	14.2	15.6	16.9	21.1
	Fixed-point alg.	11.8	14.9	15.5	15.8
	Geometric alg.	12.9	14.6	15.1	16.4

It should be noticed that the method of scanning for the best initial interpolation point occasionally performed poorly. For example, the two map system with the second file gave only 8.2dB SNR. It turns out that when there are *very* few maps, say two or three, the use of equation (3.5) to select the best map set could lead to a poor choice. This is caused by the fact that there is a small error when M_i is very small, and also when it is close to the size of the entire data sample. If there are only two maps, when equation (3.5) is evaluated for the case $M_0 = 2$ and $M_1 = 126$, it is possible that this might appear to be a good mapping, when in fact it is not. By experimenting with a minimum map size, for example of five data samples, this problem can be alleviated.

4.6 Hidden Variable Fractal Interpolation

The HVFI method is based on using an IFS to interpolate between selected sample points in R^3. We are given the original data sequence, $x[n]$, which consists of N data samples for $0 \leq n \leq N-1$, and have to determine the additional dimension's data, $z[n]$, prior to modeling. Unlike the previous IFS models, the equations here will be limited to the simpler non-piecewise case. The HVFI method has not seen much use because of the lack of a method to determine the hidden variables and no work has been done on the piecewise case. A variety of methods can be used to determine the $z[n]$ data [33]. With a given $x[n]$ and $z[n]$, the i^{th} map out of P maps is set up as given in equation (3.8). The i^{th} map covers M_i of the data samples for $M_{i1} \leq n \leq M_{i2}$. The set of P maps are constructed to cover the entire data sequence. First a method for determining the map parameters is discussed, followed by several approaches for generating the hidden data.

4.6.1 Map Parameter Determination

As with the simple IFS, an error equation can be defined based on the Euclidean distance between the original data and a transformed version of this data. Because there is the additional hidden data, the sum of these two distances is used to get:

$$\xi_i = \sum_{j=M_{i1}}^{M_{i2}} \left[\mathrm{w}_{ix}(x[n], z[n]) - x[j])^2 + (\mathrm{w}_{iz}(x[n], z[n]) - z[j] \right]^2, \tag{4.46}$$

for each map, where

$$\begin{aligned} \mathrm{w}_{ix}(x[n], z[n]) &= c_i \cdot n + d_i \cdot x[n] + h_i \cdot z[n] + f_i, & \text{(4.47a)} \\ \mathrm{w}_{iz}(x[n], z[n]) &= k_i \cdot n + l_i \cdot x[n] + m_i \cdot z[n] + g_i, & \text{(4.47b)} \end{aligned}$$

and

$$n = \mathrm{int}\left(\left(\frac{N-1}{M_i - 1} \right) j \right). \tag{4.48}$$

The total error from all of the maps is defined as

$$\xi = \sum_{i=1}^{P} \xi_i. \tag{4.49}$$

Once the $z[n]$ data is known, the map parameters can be determined through a least squares minimization of the error equation (4.49), giving the following sets of equations for each map,

$$\sum_{j=0}^{M_i-1} \left[S_j \cdot S_j^T \right] \begin{bmatrix} c_i \\ d_i \\ h_i \\ f_i \end{bmatrix} = \sum_{j=0}^{M_i-1} S_j \cdot x[M_{i1} + j], \tag{4.50a}$$

and

$$\sum_{j=0}^{M_i-1} \left[S_j \cdot S_j^T \right] \begin{bmatrix} k_i \\ l_i \\ m_i \\ g_i \end{bmatrix} = \sum_{j=0}^{M_i-1} S_j \cdot z[M_{i1} + j] \tag{4.50b}$$

where

$$S_j = \begin{bmatrix} n & x[n] & z[n] & 1 \end{bmatrix}^T, \tag{4.51}$$

and n is defined in equation (4.48).

Once the map parameters have been calculated, the map must be checked to see if it is a contraction mapping. This can be done by examination of the sub-matrix

$$A_i = \begin{bmatrix} d_i & h_i \\ l_i & m_i \end{bmatrix}. \tag{4.52}$$

If the determinant of A_i has a magnitude less than one, then the map is a contraction mapping.

4.6.2 Hidden Data Determination

The remaining tasks in using the HVFI method are the determination of the $z[n]$ data and selection of the interpolation points. Several methods were developed to determine the $z[n]$ data based on a least squares minimization of the error for each map.

For continuous functions, the IFS maps are constructed to be 'just touching', which means that adjacent maps share end points. For the discrete case this is not necessary and would result in an overdetermined system of equations. Therefore, the maps were set up with adjacent, instead of overlapping, end points. With adjacent end points, the P maps provide N equations for the N unknowns. The nice feature of this system of equations is the sparseness of the matrix; most rows have only two non-zero coefficients. The system of equations can be solved using an iterative method, such as the Gauss-Seidel or the Conjugate-Gradient method, both of which were implemented with similar results.

An Optimal Approach

The optimal $z[n]$ data was obtained in a least squares sense through the error equation (4.49) for all of the maps. Because the $z[n]$ data is used by all of the maps together, it is not possible to work with each map individually. Therefore the $z[n]$ data must be derived from the composite error equation (4.49). Examination of equation (4.46) with regard to $z[n]$ shows that there are three distinct ranges depending on the value of j or n, where n is defined in equation (4.48).

1. $M_{i1} \leq n \leq M_{i2}$ and $j \neq M_{i1}\frac{(M_i-1)}{M_i-N}$
2. $M_{i1} \leq n \leq M_{i2}$ and $j = M_{i1}\frac{(M_i-1)}{M_i-N}$
3. n outside of $\{M_{i1} \cdots M_{i2}\}$

The range $M_{i1} \leq n \leq M_{i2}$ can be written in terms of j:

$$M_{i1}\frac{(M_i-1)}{N-1} \leq j \leq M_{i2}\frac{(M_i-1)}{N-1}. \tag{4.53}$$

Taking the partial derivative of equation (4.49) with respect to $z[n]$ for each of these three cases yields three equations:

$$m \cdot z[n] - z[M_{i1}+j] = -k_i \cdot n - l_i \cdot x[n] - g_i, \tag{4.54a}$$

$$\begin{aligned}(h_i^2 + m_i^2 - m_i) \cdot z[n] + (1-m_i) \cdot z[M_{i1}+j] = \\ (1-m_i)(k_i \cdot n + l_i \cdot x[n] + g_i) \\ +h(x[M_{i1}+j] - c_i \cdot n - d_i \cdot x[n] - f_i),\end{aligned} \tag{4.54b}$$

$$\begin{aligned}-(h_i^2 + m_i^2) \cdot z[n] + m_i \cdot z[M_{i1}+j] = \\ h_i(c_i \cdot n + d_i \cdot x[n] + f_i - x[M_{i1}+n]) + m_i(k_i \cdot n + l_i \cdot x[n] + g_i).\end{aligned} \tag{4.54c}$$

Determination of the map parameters and the $z[n]$ data proceeds as follows.

1. Initialize $z[n]$
2. Calculate the map parameters using the method from Section 4.6.1
3. Determine the $z[n]$ data to minimize the error

4. Calculate error and compare to previous; if the change is small then stop, otherwise go to step 2

Obviously the last step offers some flexibility in that the iterations may be allowed to continue if the error is decreasing, or stopped after a certain number of iterations, and in addition, a check should be made for divergence. The initial value chosen for $z[n]$ will affect the convergence of the algorithm. There is no guarantee of convergence for a given initial value. The only value which should not be used is $x[n]$ itself; which will lead to a singular set of equations when the map parameters are calculated.

Two Fast Iteration Methods

Rather than solve the system of equations for all cases, if instead $z[n]$ and $z[M_{i1}+j]$ are treated as different variables then a simpler solution is derived. When the partial of equation (4.46) with respect to $z[n]$ and $z[M_{i1}+j]$ is set to zero, the following two equations are obtained:

$$\begin{aligned}(h_i^2+m_i^2)z[n]+(-m_i)z[M_{i1}+j] = \\ h_i(c_i\cdot n+d_i\cdot x[n]+f_i-x[M_{i1}+j]) \\ +m_i(k_i\cdot n+l_i\cdot x[n]+g_i)\end{aligned} \tag{4.55a}$$

and

$$(m_i)z[n]+(-1)z[M_{i1}+j]=k_i\cdot n+l_i\cdot x[n]+g_i. \tag{4.55b}$$

These equations can be solved for either $z[n]$ or $z[M_{i1}+j]$ resulting in a single equation to calculate $z[n]$ directly from $x[n]$. The two equations obtained are:

$$\begin{aligned}z[M_{i1}+j]=\frac{m_i}{h_i}(x[M_{i1}+j]-c_i\cdot n-d_i\cdot x[n]-f_i) \\ +k_i\cdot n+l_i\cdot x[n]+g_i,\end{aligned} \tag{4.56}$$

and

$$z[n]=\frac{x[M_{i1}+j]-c_i\cdot n-d_i\cdot x[n]-f_i}{h_i}. \tag{4.57}$$

While these two equations are not optimal, they can generate the necessary $z[n]$ much more quickly than solving the previous system of equations.

The above iterative algorithm is again used to determine the best set of maps and corresponding $z[n]$ data.

Independent Hidden Variables

With the aforementioned methods one problem which occurs with the interdependence between $x[n]$ and $z[n]$ during the reconstruction process is that the model has built in additional performance requirements. The reconstructed $z[n]$ data will depend on the proper reconstruction of the $x[n]$ data. Errors tend to propagate and in an interest of stopping this, the link from $x[n]$ to $z[n]$ was broken. This was implemented by setting the l_i variable in all of the maps equal to zero. With this modification equation (4.50b) becomes

$$\sum_{j=0}^{M_i-1} \left[\hat{S}_j \cdot \hat{S}_j^T\right] \begin{bmatrix} k_i \\ m_i \\ g_i \end{bmatrix} = \sum_{j=0}^{M_i-1} \hat{S}_j \cdot z[M_{i1}+j], \tag{4.58}$$

where

$$\hat{S}_j = \begin{bmatrix} n & z[n] & 1 \end{bmatrix}^T.$$

This method was implemented with each of the methods for determining $z[n]$ and the results are given in Section 4.6.3.

Another variation on this idea is to generate the $z[n]$ data as suggested previously, create an IFS to model the $z[n]$ data, use this IFS to recover $\hat{z}[n]$, and finally use $\hat{z}[n]$ when the map parameters are determined for $x[n]$. This method offers the advantage that $\hat{z}[n]$, which is used in determining the $x[n]$ map parameters, can be reconstructed without error. This method did not show any improvements over the others and will not be discussed further.

Non-Iterative Method

All of the aforementioned methods rely on iteratively calculating $z[n]$ and the map parameters in order to find a solution to the nonlinear system of equations. The computational requirements can be greatly reduced by avoiding these iterations. The system was tested after one iteration of the independent hidden variable method as described previously and the results are provided in the next section. The single iteration was set up by first solving for the map parameters, then determining the $z[n]$ data and finally with this new $z[n]$ data, recalculating the map parameters.

4.6.3 Results

Table 4.5 shows results for the different methods discussed. The tests were conducted with the test files as described in Section 4.4.1. Again, in order to facilitate comparisons, exhaustive searches were conducted with a fixed number of maps.

4.6.4 HVIFS Conclusion

Each of the methods provided similar results, which suggests using the simplest method to reduce the computational requirements. It turned out that the selection for the initial $z[n]$ made a difference and in general the best results were obtained with a constant value of 0.01, when the $x[n]$ data was normalized prior to modeling.

The simple method for calculating $z[n]$ with independent hidden variables and initial $z[n] = 0.01$ gave the best results of all the methods presented. Of all the iterative approaches, this method also requires the fewest computations for solving a particular map. The non-iterative version of the independent hidden variables method also provided very good results with significantly fewer computations required. The difference in the results of these two methods is so small that the non-iterative method is probably the better approach overall.

Table 4.5: HVIFS model performance

File	Method	2 Maps SNR	3 Maps SNR	4 Maps SNR
1	Regular IFS	20.0	23.2	24.5
	Optimal	20.2	23.7	25.2
	Quick: $z[n]$	20.3	23.3	25.1
	Quick: $z[M_1 + j]$	20.3	22.9	25.1
	Independent $z[n]$	20.3	23.7	25.2
	Non Iterative	20.3	23.7	25.2
2	Regular IFS	22.6	25.6	26.8
	Optimal	22.7	26.4	28.1
	Quick: $z[n]$	22.8	25.2	27.5
	Quick: $z[M_1 + j]$	18.0	24.9	27.4
	Independent $z[n]$	22.9	26.3	28.2
	Non Iterative	22.4	26.3	28.2
3	Regular IFS	25.4	31.0	33.3
	Optimal	25.4	31.3	33.4
	Quick: $z[n]$	25.3	30.9	33.1
	Quick: $z[M_1 + j]$	25.2	31.1	33.6
	Independent $z[n]$	25.4	31.3	33.6
	Non Iterative	25.4	31.2	33.6
4	Regular IFS	26.4	30.4	33.0
	Optimal	26.4	30.4	33.0
	Quick: $z[n]$	26.2	30.4	33.1
	Quick: $z[M_1 + j]$	24.8	30.1	32.8
	Independent $z[n]$	26.2	30.4	33.3
	Non Iterative	26.3	30.4	33.3
5	Regular IFS	11.9	15.6	16.9
	Optimal	14.0	17.0	19.8
	Quick: $z[n]$	14.0	17.3	19.6
	Quick: $z[M_1 + j]$	13.6	16.8	18.9
	Independent $z[n]$	14.3	17.3	19.9
	Non Iterative	14.2	17.0	19.9

It is important to compare these results to the regular affine IFS of equation (3.2a), and these results are also given in Table 4.5. While the HVIFS method did improve the SNR of the reconstructed data, the storage cost of the additional coefficients weighs heavily against using this method in this application.

Complex Data

Another possible application of the HVIFS method that was evaluated is modeling complex data. The real and imaginary portions of a signal were used for the $x[n]$ and $z[n]$ data respectively. The motivating idea being to take advantage of any similarities between these two parts of the data.

In modeling real data, another approach evaluated for determining the hidden variables for the HVIFS was to take the FFT of the data, then use the imaginary portion of the frequency representation as the hidden variable, and then model the transformed data using an HVFI function. The data was then recovered by using an inverse FFT of the reconstructed data. Both of these ideas were implemented without positive results.

4.7 One-Dimensional Conclusions

In this chapter two different types of IFS models have been presented, the normal IFS where the maps are constructed to interpolate in two-dimensions, and the HVIFS approach, which constructs a three-dimensional model and uses a two-dimensional projection. In order to facilitate comparing the various models introduced here, the SNR gains were averaged over the complete set of test files, as well as over the complete set of exhaustive tests with two through four maps, using the data from Tables 4.4 and 4.5. The average gain for each of the methods is given in Table 4.6. In comparing the results between the two approaches, it is clear that more SNR gain is seen through the addition of nonlinearities. For the HVIFS case, the greatest gain over the standard IFS was 3.0 dB for test file 5, the remainder of the gains were closer to a fraction of a dB. In fact, most of the gain seen in the HVIFS approach can be attributed to the relaxation of the boundary conditions. The second column of results in Table 4.6 gives the SNR gain over the standard affine IFS with relaxed boundary conditions. It is interesting to note that the largest performance gain was achieved by just relaxing the boundary conditions. The only cost associated with this change was the additional computational cost of determining more parameters through least squares. There was no increased storage cost for this approach.

Table 4.6: 1D IFS model average gains

Method	Average Gain	Over Relaxed Method
Relaxed Affine	2.1	–
Simple Quad.	0.14	–
Full Quad.	3.3	–
Relaxed Full Quad.	3.8	1.7
Relaxed Nonlinear Addressing	3.2	1.2
Relaxed Condensation Maps	3.3	1.2
HVIFS: Relaxed Optimal	2.4	0.3
HVIFS: Relaxed Quick z[n]	2.2	0.086
HVIFS: Relaxed Quick z[m+j]	1.6	-0.52
HVIFS: Relaxed Independent	2.5	0.38
HVIFS: Relaxed Non Iterative	2.4	0.31

Because of the large number of parameters required for the HVIFS model, and the relatively small increase in SNR, it is not a favored approach for increasing model performance. The largest SNR gain was seen with the relaxed full quadratic maps, which also have the largest number of parameters. On the other hand, using nonlinear addressing, which has a very small incremental storage cost per map, resulted in only 0.5 dB less SNR than the relaxed full quadratic maps. In evaluating all of these options, it is also important to consider using a larger number of maps instead of a more complicated map. In order to allow all of the methods to be evaluated fairly, comparisons were made with each of the methods and using a range of number of maps. Figure 4.3 shows the compression ratio versus SNR for test file 1 using from two through four maps. This figure allows all of the methods discussed so far to be compared. Those points represented by an $\times$ are from the methods which gave the best results in the sense of maximizing SNR for a given compression ratio.

These points represent the three methods:

- Relaxed boundary conditions with affine maps
- Nonlinear addressing
- Condensation type maps

It turns out that the various polynomial nonlinear maps, while achieving higher SNR values, do so at a cost of greatly decreasing the compression ratio, which effectively neutralizes the gain, as compared to using additional maps. In addition to separately indicating the points from the superior methods with an $\times$ symbol, a line was fitted through these points to indicate the trend in the SNR as the compression ratio is changed. A straight line would be the proper curve for a memoryless, zero mean, Gaussian signal when plotting the bit rate versus SNR [34]. While the data here arguably does not fit this description very well, we are only interested in seeing the trend for these models, thus a straight line approximation works well. The results shown here are typical for each of the test files, and a composite plot of all of the files, using only the three methods above, is given in Figure 4.4, which illustrates the trade–off between compression ratio and SNR.

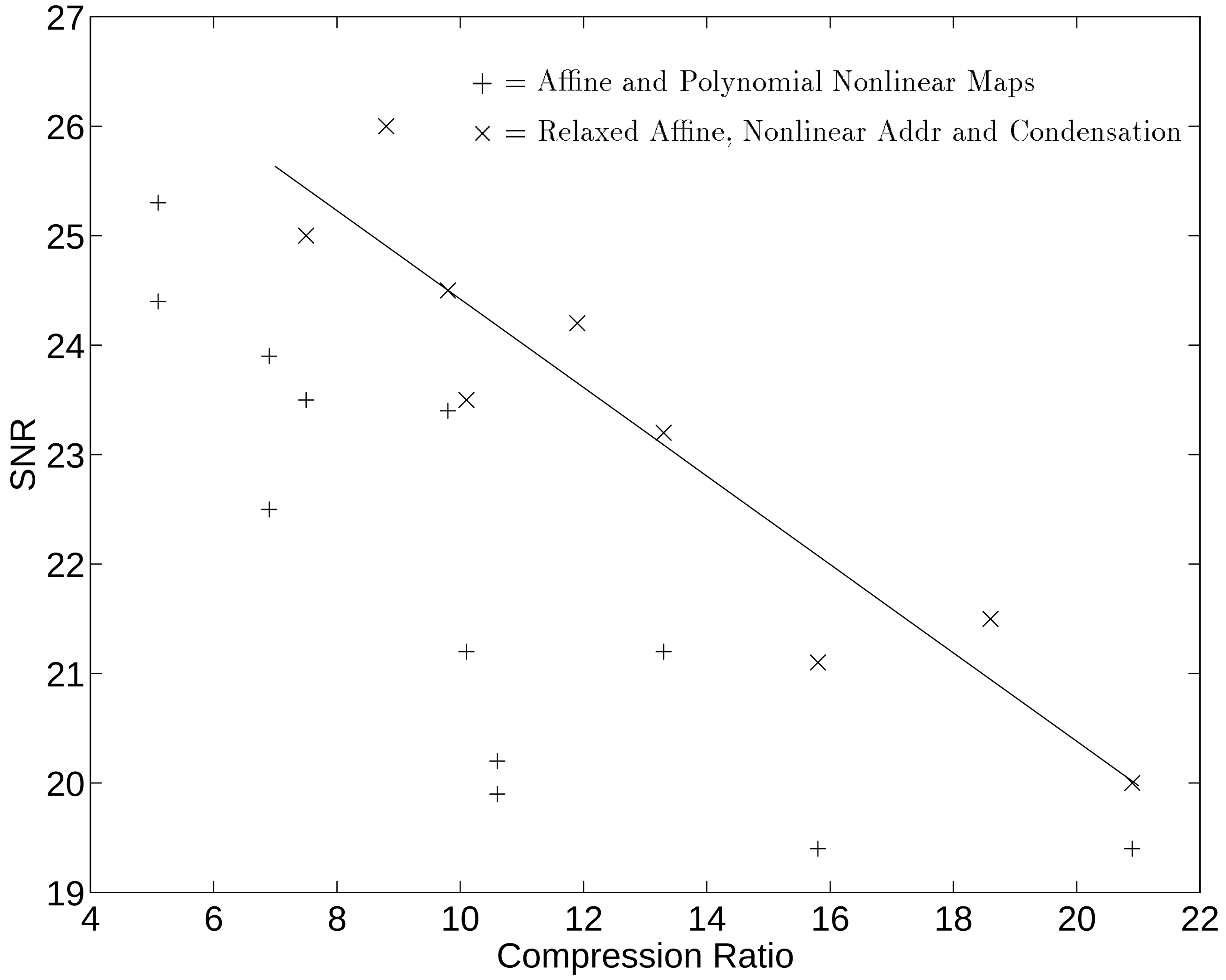


Figure 4.3: SNR versus compression ratio for test file 1 with two through four maps with all methods

In this comparison of the different methods introduced in this chapter, we have seen that the three methods which provide significant improvements over the standard affine maps are the relaxation of the boundary condition with affine maps, nonlinear addressing, and the condensation type maps. The use of the HVFI method and the addition of various quadratic terms to the map did not perform as well. Next these three types of maps were used in the two algorithms introduced in Section 4.5, and the results are given in Table 4.7. Included in this table for comparison are the results from using the exhaustive search for each of the three types of maps tested. Because it is difficult to discern a trend, the average SNR cost for each of the two algorithms with each of the three types of maps is given in Table 4.8. From this table it is seen that the nonlinear addressing method does not work well with these algorithms. Further, it is interesting to note that the geometric algorithm performed better on average than the fixed point algorithm. This is in part due to the poor performance seen with an occasional file using only two maps. For example, the reconstructed data for file 2 with relaxed affine maps achieved only 8.2 dB SNR. For this reason, the average SNR costs were also calculated for three and four maps. On average, the method which performed the best was the geometric algorithm using condensation type maps. However, as with any of the nonlinear maps, there is a cost associated with the nonlinear terms. Determining which type of map to use for a specific application would depend on the desired compression ratio and SNR values. As was seen, any of the three types of maps discussed here: relaxed affine, relaxed nonlinear addressing, and relaxed condensation type maps, will perform better than the standard affine map typically used in an IFS model.

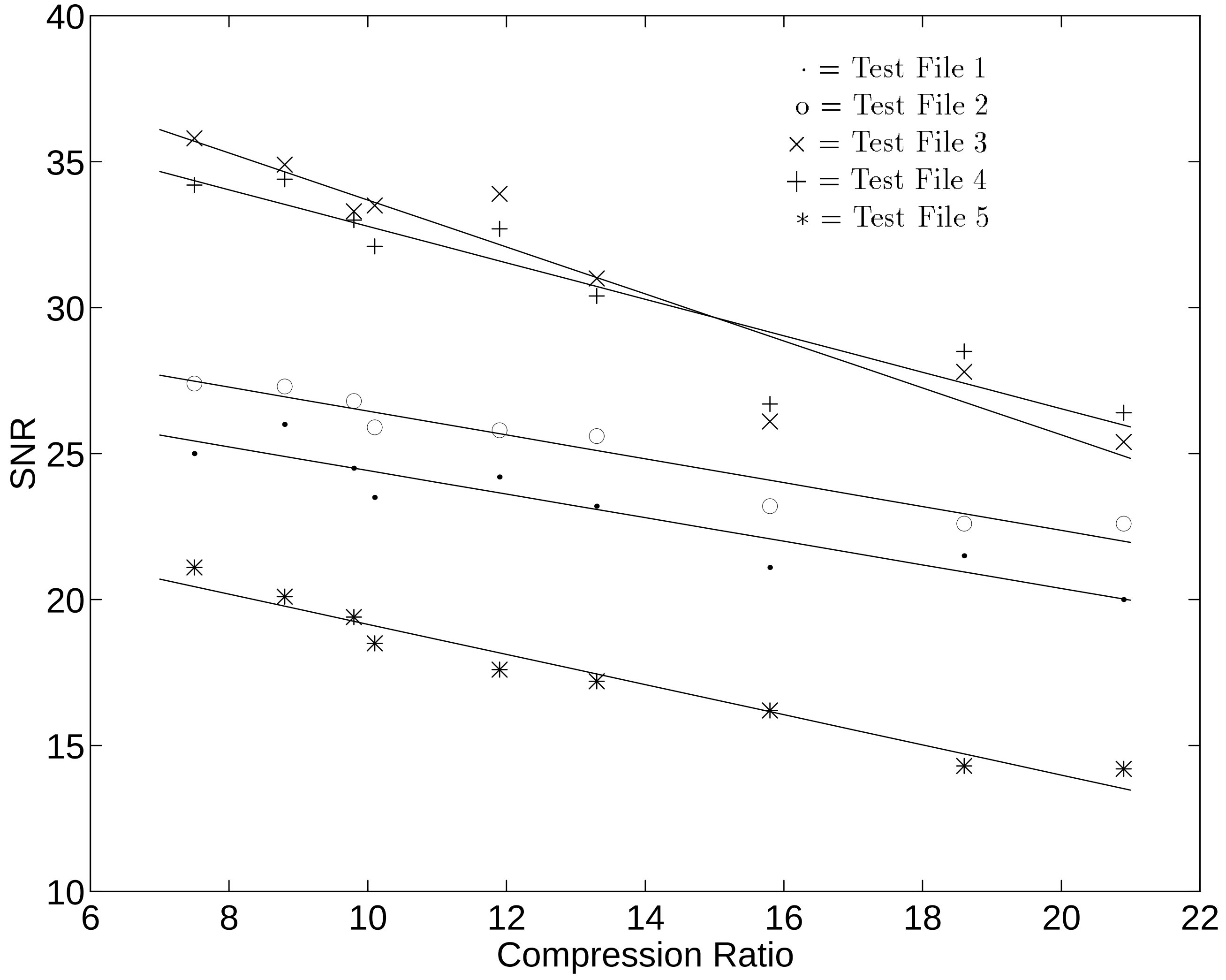


Figure 4.4: SNR versus compression ratio for all test files with two through four maps using only relaxed affine, nonlinear addressing and condensation type maps

Table 4.7: 1D IFS model algorithm comparison with nonlinear maps

File	Method	2 Maps SNR	3 Maps SNR	4 Maps SNR
1	Exhaustive Relaxed	20.0	23.2	24.5
	Fixed Point: Relaxed	14.0	20.7	21.2
	Geometric: Relaxed	18.2	19.8	21.6
	Fixed Point: Nonlinear Address	14.0	21.4	21.9
	Geometric: Nonlinear Address	18.2	20.4	21.7
	Fixed Point: Condensation	17.5	22.3	22.8
	Geometric: Condensation	20.4	22.2	23.8
2	Exhaustive Relaxed	22.6	25.6	26.8
	Fixed Point: Relaxed	8.2	18.9	22.3
	Geometric: Relaxed	22.6	21.0	22.9
	Fixed Point: Nonlinear Address	8.2	18.9	22.4
	Geometric: Nonlinear Address	22.6	21.1	23.0
	Fixed Point: Condensation	9.9	24.0	24.6
	Geometric: Condensation	23.7	22.1	24.5
3	Exhaustive Relaxed	25.4	31.0	33.3
	Fixed Point: Relaxed	22.8	27.2	32.9
	Geometric: Relaxed	24.0	25.7	27.0
	Fixed Point: Nonlinear Address	22.8	27.4	32.9
	Geometric: Nonlinear Address	23.9	26.1	27.2
	Fixed Point: Condensation	23.8	28.9	34.5
	Geometric: Condensation	25.5	28.1	31.0
4	Exhaustive Relaxed	26.4	30.4	33.0
	Fixed Point: Relaxed	11.9	22.8	31.0
	Geometric: Relaxed	25.1	27.3	31.1
	Fixed Point: Nonlinear Address	11.8	22.9	31.3
	Geometric: Nonlinear Address	25.0	27.2	31.1
	Fixed Point: Condensation	16.3	23.5	34.3
	Geometric: Condensation	31.8	33.9	34.6
5	Exhaustive Relaxed	11.9	15.6	16.9
	Fixed Point: Relaxed	11.8	14.9	15.5
	Geometric: Relaxed	12.9	14.6	15.1
	Fixed Point: Nonlinear Address	11.8	14.8	15.7
	Geometric: Nonlinear Address	14.3	15.6	16.3
	Fixed Point: Condensation	13.9	18.1	17.9
	Geometric: Condensation	15.9	16.5	18.5

Table 4.8: 1D IFS algorithm average SNR costs over exhaustive search

Algorithm	Average SNR Cost	
	2 – 4 Maps	3 – 4 Maps
Geometric Algorithm: Relaxed	2.51	3.42
Geometric Algorithm: Nonlinear Address	3.86	4.72
Geometric Algorithm: Condensation Type Maps	1.19	2.18
Fixed Point Algorithm: Relaxed	4.70	3.29
Fixed Point Algorithm: Nonlinear Address	6.22	4.73
Fixed Point Algorithm: Condensation Type Maps	3.57	2.11

CHAPTER 5

BACKGROUND FOR TWO-DIMENSIONAL MODEL

The main thrust of research efforts in applying IFS theory to signal modeling has been in the area of image coding. This is a natural application for IFSs as we have seen from the spectral characteristics of images and of IFS attractors. In light of this fact, all of the efforts of this research with regard to two-dimensional modeling have been geared towards image coding and the remainder of this dissertation concentrates on this area also.

5.1 Early IFS Coding Efforts

The initial work with fractals and with IFSs was in generating pictures that looked 'realistic.' These efforts resulted in many images and scenes which contained amazing realism and detail. Because they were constructed by hand, using IFSs to generate the objects in the images, the coded versions of the scenes were extremely compact with compression ratios on the order of 10,000:1 [10]. Early efforts at coding arbitrary images centered around the use of libraries of images and a pattern matching type of algorithm to find the various objects in the image. From this object oriented approach, the necessary maps for the IFS were gleamed. Consequently, initial reports spoke of the tremendous potential of this new technique. Unfortunately, the creation of an image which is realistic in appearance and the coding of an arbitrary image are different problems entirely. Aside from one U.S. Patent [35], detailing the concept and describing a hardware system for implementing the algorithm, little of the results from these early efforts has been published. Presumably the complex task of segmenting the objects in an image combined with the new IFS technique gave poor results. A block-oriented approach was introduced in [36] and all subsequent efforts have been based on this method.

5.2 Search-Based Block Coding

In order to provide a common notation for comparing the different approaches to the IFS block coding problem, a standard framework for the problem will be presented. Following this overview, each of the different proposed variations and algorithms will be covered.

5.2.1 Block Coding Background

Given an image, V, of size $L \times L$ pixels, divided into a set of

$$N_R = (L/L_R)^2 \tag{5.1}$$

range blocks of size $L_R \times L_R$, the goal is to code each range block with a transformation of a domain block. The basic affine map can be written as

$$\mathrm{w}_i\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ b_i \end{bmatrix}, \tag{5.2}$$

for $i = 1, 2, \ldots, N_R$, where x and y are the coordinate addresses in the image, and z corresponds to the image intensity. Essentially the image is coded with a three-dimensional version of the piecewise maps given in Section 3.1.1. The submatrix

$$\begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \tag{5.3}$$

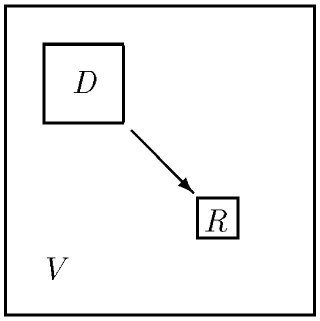


Figure 5.1: Two-dimensional IFS mapping

in the above equation along with the variables e_i and f_i control the mapping of the domain block pixels to those of the range block in a contractive fashion, as shown in Figure 5.1. The map is only defined over the pixels in the range block, and does not alter neighboring pixels outside of the range block.

In order to insure a contraction in the spatial domain, the domain blocks are of size $2L_R \times 2L_R$, and are then decimated to form blocks the same size as the range blocks with $L_R \times L_R$ pixels. By fixing the contraction factor in each of the spatial directions at 0.5, the issue of sampling is avoided as discussed in Section 3.2. The set of potential domain blocks is not restricted to a non-overlapping tiling of the image as the range blocks are. Instead, the domain blocks can overlap one-another. Therefore, there are many more potential domain blocks than range blocks. With this potential overlap in the domain blocks, there are

$$N_D = (L - 2L_R + 1)^2 \tag{5.4}$$

domain blocks to choose from for each range block. The two remaining terms in equation (5.2), s_i and b_i, provide scaling for the domain pixel intensities, and a bias value for the data as it is mapped, respectively.

For notational convenience, the image will be addressed with $V[0,0]$ being the upper left corner, and each range and domain block referenced by the upper left corner of the block. Thus a typical domain and range block address would be written as (x_D, y_D) and (x_R, y_R), respectively. Using this notation, the i^{th} domain and range blocks for the image V can be written as $d_{x_{D_i},y_{D_i}}[x,y]$ and $r_{x_{R_i},y_{R_i}}[x,y]$, respectively, where

$$d_{x_{D_i},y_{D_i}}[x,y] = V[x_{D_i}+x, y_{D_i}+y], \quad \text{for} \quad x,y = 0,\ldots,2\cdot L_{R_i}-1, \tag{5.5a}$$

$$r_{x_{R_i},y_{R_i}}[x,y] = V[x_{R_i}+x, y_{R_i}+y], \quad \text{for} \quad x,y = 0,\ldots,L_{R_i}-1. \tag{5.5b}$$

The domain blocks have been decimated from $2L_R \times 2L_R$ pixels to $L_R \times L_R$ pixels in one of two ways. In [36] the decimated pixels were taken as the average of the domain pixels. Thus for a decimated domain block, $\hat{d}_{x_{D_i},y_{D_i}}[x,y]$,

$$\hat{d}_{x_{D_i},y_{D_i}}[x,y] = \frac{1}{4}\left(\sum_{m=0}^{1}\sum_{n=0}^{1} d_{x_{D_i},y_{D_i}}[2\cdot x+m, 2\cdot y+n]\right), \tag{5.6}$$

and in [37] no averaging was performed,

$$\hat{d}_{x_{D_i},y_{D_i}}[x,y] = d_{x_{D_i},y_{D_i}}[2\cdot x, 2\cdot y]. \tag{5.7}$$

The transformation of the image data in equation (5.2) can be written as

$$\hat{r}_{x_{R_i},y_{R_i}}[x,y] = b_i + s_i \cdot \hat{d}_{x_{D_i},y_{D_i}}[x,y], \tag{5.8}$$

for $x,y = 0,\ldots,L_R-1$, where $\hat{r}_{x_{R_i},y_{R_i}}$ is the transformed domain data. In order to satisfy the contraction mapping requirement, the scaling factor, s_i, must be strictly less than one in magnitude. A more complicated map was used in [38], which included terms for tilt in the x and y directions as follows:

$$\hat{r}_{x_{R_i},y_{R_i}}[x,y] = b_i + g_{i1}\cdot x + g_{i2}\cdot y + s_i \cdot \hat{d}_{x_{D_i},y_{D_i}}[x,y]. \tag{5.9}$$

This modification corresponds to replacing the two zeros in the last row of the matrix in equation (5.2) with the variables g_{i1} and g_{i2}.

Each potential map can be evaluated by calculating the distance between the range data and the transformed domain data. By selecting the map which minimizes this distance, the accuracy of the model should be improved based on the Collage Theorem. Typically the L_2 distance is used:

$$\xi_i = \sum_{m=0}^{L_R-1}\sum_{n=0}^{L_R-1}\left(\hat{r}_{x_{R_i},y_{R_i}}[m,n] - r_{x_{R_i},y_{R_i}}[m,n]\right)^2. \tag{5.10}$$

With this basic overview and notation, the proposed approaches can be reviewed and compared.

5.2.2 Initial Approach

In [6, 36] the fundamental method for a block coding IFS was introduced. A range block size of 8×8 was used with corresponding domain blocks of size 16×16. In order to reduce the search to find acceptable domain blocks for each range, both the range and the domain blocks were classified into one of the following types:

1. Shade block
2. Edge block
3. Midrange block

After classification, a search was performed to find the best domain block for each range block based on minimizing equation (5.10). In decimating the domain block, the four pixels corresponding to every address were averaged to arrive at the downsampled version of the domain block as in equation (5.6). In addition to searching over the domain blocks, each domain block was transformed by each of the eight isometric transformations shown in Figure 5.2 in evaluating equation (5.10).

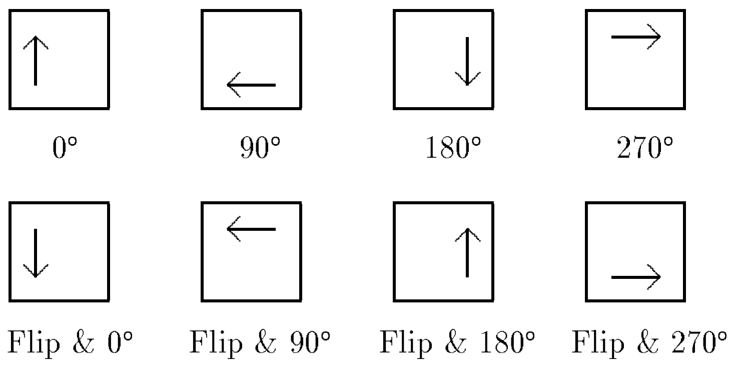


Figure 5.2: Eight isometries for 2D maps

Once the best domain block was determined for each range block, the error from equation (5.10) was evaluated for each quarter of the entire block. If the error for any 4×4 subblock was greater than some threshold, then that subblock was coded separately. In this manner, each 8×8 block was coded with from one to four blocks. If two or three blocks were required, then there would be one larger 8×8 block in addition to one or two 4×4 blocks. Using this approach, the eleven ways that 4×4 subblocks can be appended to an 8×8 block are shown in Figure 5.3.

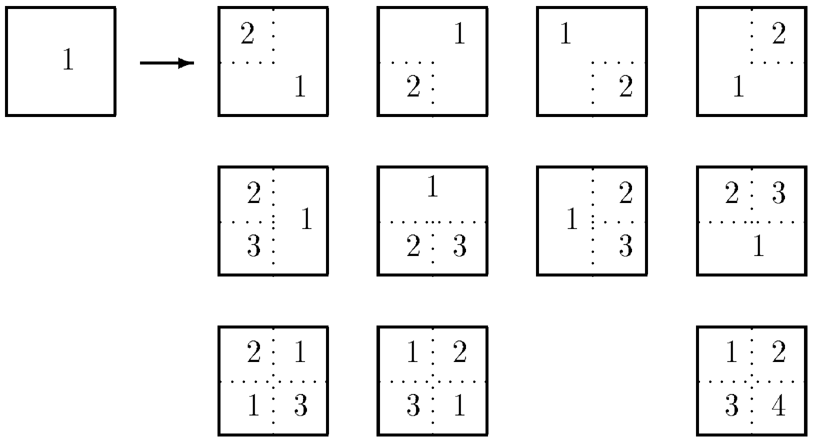


Figure 5.3: Eleven ways to append subblocks to a larger block

The parameters for the domain blocks were quantized prior to evaluating the error equation (5.10). The shade blocks were quantized with six bits, representing a constant value over the entire block. The edge blocks used seven bits for the bias value, three bits for the scaling, and three bits for the isometry used. Midrange blocks, which could not be classified as shade or edge, used seven bits for the bias and three for the scaling factor. The transformation for the midrange block was of the form of equation (5.8), using a decimated domain block from equation (5.6). For the edge block transformation, appropriate index changes in equation (5.8) were made for the eight isometries.

Using this approach, the 256×256 six-bit gray scale version of the LENA image was coded at 0.68 bpp and the reconstructed image had an PSNR of 27.7 dB. In order to reduce the number of domain blocks to search over, the image was first divided into four 128×128 sub-images which were each coded independently, thus reducing the search to 12769 potential domain blocks. While the quality of the reconstructed image was quite good, there were still some artifacts from the blocking technique used, as well as a smoothing of some of the finer textures in the image. In addition, this method still suffered from the computational complexity of the search.

Following this introduction to image coding with a block-based IFS, several authors have proposed enhancements to this method to improve the performance, both in reconstructed image quality and in the computational cost of coding an image. Each of these modifications is discussed below.

5.2.3 Improvements to the IFS Block Coding Method

In [24], the search was greatly reduced by only using nonoverlapping domain blocks, thus for a 512×512 image with 8×8 pixel range blocks, there are only 1024 nonoverlapping domain blocks of size 16×16. In addition, each range block was first approximated by a quadratic polynomial. If the error was below some threshold, then the block was classified as smooth and coded with the coefficients for the polynomial. Otherwise the block was classified as rugged, and coded with a transformed domain block. In addition to reducing the search time, the smaller set of domain blocks also reduced the storage for the maps because only 10 bits were required to store the address of the domain as opposed to 18 bits for the complete address. This method also took advantage of the eight isometries as before. Using this approach, the 512×512 eight-bit gray scale version of LENA was coded at 0.5 bpp and the reconstructed image had a PSNR of 30.8 dB.

It is difficult to compare results between different sizes of the same image because the reconstructed image SNR will typically be higher for a larger image. There is also a difference of several dB between the SNR and the PSNR values for a given image. Going from the 256×256 to the 512×512 version of LENA, typically the SNR will increase by about four dB. In addition, there is approximately a four dB increase in going from the SNR to the PSNR for this image. Finally, the original work used a six-bit gray scale image, which further complicates any comparisons between it and subsequent works using eight-bit gray scale images. For this reason, comparisons will be avoided unless the same image was used and the reconstruction error is reported in equivalent forms.

A series of Naval Ocean Systems Center Technical Reports and correspondences discuss numerous enhancements to the block coding approach [39, 40, 41, 42, 37, 43], culminating in a journal paper [25]. The basic square domain and range blocks were expanded to a Horizontal-Vertical (HV) partitioning scheme, where each block is divided either horizontally or vertically and also at a variable location. Thus the resulting block sizes are completely variable. In addition, a triangular partitioning method was proposed, however, neither one of these ideas were carried to the implementation stage [41].

In [43] a quadtree approach was used with block sizes ranging from $L_R = 4$ to $L_R = 32$ pixels. The quadtree method begins with the largest block size and divides those blocks which do not meet a set error threshold into four equally sized subblocks. Each subblock is recursively examined until either the error threshold is met, or the minimum block size is reached.

In addition, the domain and range blocks were grouped into a fixed number of classes [40]. Prior to grouping the blocks, they were first transformed based on using the previously given eight isometries to place the brightest corner of the block in the upper left hand corner. Then the next brightest corner was placed, if possible, in the upper right corner. Thus, during all comparisons, every block was prealigned in the same orientation with respect to image intensities over the four quadrants of each block. The blocks were then classified based on the amount of variation in the brightness of each quadrant. The total number of classes was also varied from 1 through 72 classes.

Another significant contribution of this work was the use of what was termed "Eventual Contraction Mappings" in the IFS [37, 43]. By taking advantage of the fact that often the domain for a given range is located in another part of the image, it was determined that *some* of the maps do not have to be contraction mappings. As long as the composite transformation of all of the maps is a contraction, then the IFS will converge. Using the definition for a contraction mapping given in Section 2.1, an IFS is defined as Eventually Contractive if there exists a positive integer m, such that $W^{\circ m}$ is contractive [43]. To understand how $W^{\circ m}$ can be contractive when some of the individual maps, w_i, are expansive, it is instructive to recall that $W^{\circ m}$ is composed of a union of the individual maps,

$$W^{\circ m} = \mathrm{w}_{i1} \circ \mathrm{w}_{i2} \circ \ldots \circ \mathrm{w}_{im}, \tag{5.11}$$

thus as long as the *net* effect of all of the maps is a contraction, then $W^{\circ m}$ will be contractive. By allowing the constraint on the scaling factor to be loosened, the reconstructed image fidelity was increased. Using rectangular maps, with the relaxed restriction on the scaling factor magnitude, resulted in coding the 512×512 version of LENA at 0.5 bpp. The reconstructed image had a PSNR of 32.1 dB.

An extremely fast coder was created in [44, 38] by eliminating the search for the domain for each range block. By taking each group of four range blocks as a domain block for those ranges, the search was avoided entirely. In addition to the map equation (5.9), the eight isometries were also utilized to allow some permutations to the fixed domain block. In using least squares to solve for the map parameters, the relaxation of the boundary conditions introduced in Section 4.1.1 was extended to two dimensions by this work. Results were given for a 576×720 GOLD HILL image coded with 8×8 range blocks at 1.0 bpp resulting in a reconstructed image with a PSNR of 31.8 dB.

In [45] a novel approach to determining the map parameters was introduced. In this paper, the range block size was fixed at 8×8 pixels and the selection of the domain blocks was restricted as in [24] to nonoverlapping blocks. Each range block and decimated domain block was interpreted as a vector in a 64-dimensional vector space. The transformation equation (5.9) was used, and the three terms, b_i, $g_{i1} \cdot x$ and $g_{i2} \cdot y$ were interpreted as a set of three orthonormal basis vectors and their respective weights. The task of determining the best domain block, or vector, to use could then be viewed as that of finding the domain vector whose component orthogonal to the three *a-priori* basis vectors best represents the orthogonal component of the range vector. By performing a Gram–Schmidt procedure on these three fixed vectors, and then on each of the domain vectors individually, a four element orthonormal basis was generated for each of the domain vectors. With this set of four orthonormal basis vectors, $\{\mathbf{v}_i\}_{i=1}^{4}$, the weights, w_i, for each range vector, $\mathbf{r}_j$, could be determined with a simple, and fast, inner product,

$$w_i = < \mathbf{r}_j, \mathbf{v}_i > . \tag{5.12}$$

Therefore the encoding process was reduced to two main steps, first, determining the Gram–Schmidt orthonormal basis vector for each domain vector, and second, determining the weights for each orthogonalized domain vector and keeping the domain vector with the largest weight. The computational cost of the search over the domain blocks was reduced to a single inner product calculation for each domain block. Using this method, the 512×512 version of LENA was coded at 0.66 bpp and the reconstructed image had a PSNR of 32.0 dB.

5.3 Conclusion

In this chapter the basic two-dimensional IFS model was reviewed for coding images, in addition, several search-based techniques to find the necessary domain blocks for each range block were discussed. Two basic approaches were taken to reduce the search time for the appropriate domain block for a given range block. The first was to classify the domain and range blocks, then restrict the search over similarly classified blocks and the second involved reducing the number of potential domain blocks, somewhat arbitrarily, by restricting the domain blocks to be nonoverlapping. In addition, the performance of the IFS model was improved with the use of "Eventual Contraction Mappings" by allowing the scaling factors to be larger than one in magnitude, and by the use of the quadtree approach of allowing the range blocks that were difficult to model to be divided in size until either a minimum size was reached, or some error threshold was achieved.

While much progress has been reported on image coding with IFSs, the results still involve long searches for the domain blocks. In addition, the quality of the reconstructed image seems to reach an upper limit of around 32 dB PSNR for the 512×512 LENA image. This trend was illustrated in a recent report comparing the IFS image coding technique to the JPEG standard [46]. In this comparison, it was seen that the IFS coding technique seems to have a limit in the accuracy that an image can be coded. Lowering the compression ratio does not increase the reconstructed image quality significantly. This has also been noticed in our research. This is due to the nature of IFS coding methods, in contrast to the JPEG standard, which is a DCT based approach, the affine IFS is not a paradigm for an arbitrary waveform. A typical SNR versus compression ratio curve is shown in Figure 5.4. In the next chapter, several new search-based techniques and modifications to the affine map are introduced, as well as a new type of IFS model based on using multiple domains in each map.

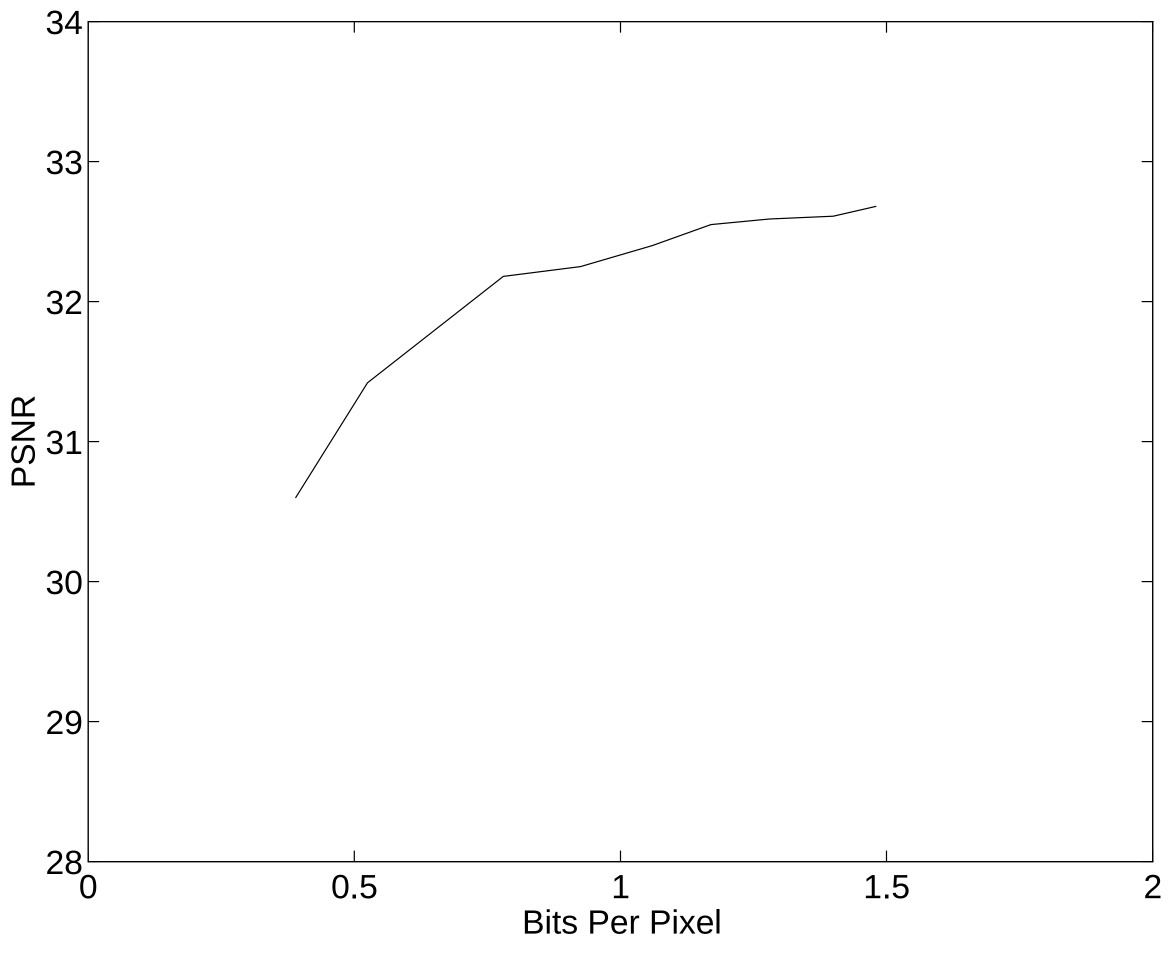


Figure 5.4: Typical SNR versus compression ratio for IFS coded image

CHAPTER 6

TWO-DIMENSIONAL MODELING

Because the added dimension dramatically increases the problem of searching for the map domains and ranges, the two main thrusts of the two-dimensional model development have been, first, to increase the flexibility of the maps without increasing the search requirement, and second, to eliminate portions of the search in a manner which enhances the odds of finding good maps. In attempting to improve the model's performance, those modifications which worked well for the one-dimensional case were extended to the two-dimensional model. In particular, relaxation of the boundary conditions, nonlinear addressing maps, and condensation type maps were converted and tested. In addition to these modifications to the map structure, several new search techniques were developed, based on characteristics of the image being coded. Finally, an entirely new approach to IFS modeling is introduced, which is based on allowing a variable number of domain blocks to be used in each map. This method also eliminates the search for the domains for each range block.

6.1 Extension of 1D Approaches

The basic affine two-dimensional map from equation (5.9) is repeated below for convenience.

$$\hat{r}_{x_{R_i},y_{R_i}}[x,y] = b_i + g_{i1} \cdot x + g_{i2} \cdot y + s_i \cdot \hat{d}_{x_{D_i},y_{D_i}}[x,y] \tag{6.1}$$

The one-dimensional modifications which provided superior results: relaxation of the boundary conditions, nonlinear addressing, and condensation type maps, were implemented in this equation, and are discussed below.

6.1.1 Relaxing the Boundary Conditions

As mentioned in Chapter 5, concurrent to our implementation of not imposing the boundary conditions, it was reported in [44]. As with the one-dimensional implementation, the map parameters are determined by minimizing the L_2 distance between the range block data, $r_{x_{R_i},y_{R_i}}[x,y]$, and the transformed domain block data, $\hat{r}_{x_{R_i},y_{R_i}}$, as given in equation (5.10). For the case with the map from equation (5.9), the error equation can be written as

$$\xi_i = \sum_{m=0}^{L_R-1} \sum_{n=0}^{L_R-1} (r_{x_{R_i},y_{R_i}}[m,n] - (b_i + g_{i1} \cdot m + g_{i2} \cdot n + s_i \cdot \hat{d}_{x_{D_i},y_{D_i}}[m,n]))^2. \tag{6.2}$$

Taking the partial derivatives of ξ_i with respect to each of the four variables, and setting these partials to zero results in the following set of four linear equations to solve for the parameters:

$$\sum_{m=0}^{L_R-1} \sum_{n=0}^{L_R-1} A \cdot \chi = \sum_{m=0}^{L_R-1} \sum_{n=0}^{L_R-1} K_0 \cdot \mathbf{K}, \tag{6.3}$$

where

$$A = \mathbf{K} \cdot \mathbf{K}^T, \tag{6.4}$$

$$\mathbf{K} = [\hat{d}_{x_{R_i},y_{R_i}}[m,n]\ ,\ 1,\ m,\ n]^T, \tag{6.5}$$

$$\chi = [s_i\ ,\ b_i\ ,\ g_{i1}\ ,\ g_{i2}]^T, \tag{6.6}$$

and

$$K_0 = r_{x_{R_i},y_{R_i}}[m,n]. \tag{6.7}$$

Because this method resulted in consistent improvements, it was used for all of the two-dimensional methods discussed in this chapter.

6.1.2 Condensation Maps

Following the form of the condensation type map used for the one-dimensional case, the map equation (6.1) was modified with the addition of sinusoidal terms in both x and y:

$$\begin{aligned} \hat{r}_{x_{R_i},y_{R_i}}[x,y] &= b_i + g_{i1}\cdot x + g_{i2}\cdot y + s_i \cdot \hat{d}_{x_{D_i},y_{D_i}}[x,y] \\ &\quad + h_{i1}\cdot \sin(\pi\frac{x}{L_R-1}) + h_{i2}\cdot \sin(\pi\frac{y}{L_R-1}). \end{aligned} \tag{6.8}$$

The parameters were again determined through the solution of equations (6.3) and (6.4) with the new variables:

$$\mathbf{K} = [\hat{d}_{x_{R_i},y_{R_i}}[m,n]\ ,\ 1,\ m\ ,\ n\ ,\ \sin(\pi\frac{m}{L_R-1})\ ,\ \sin(\pi\frac{n}{L_R-1})]^T, \tag{6.9}$$

and

$$\chi = [s_i\ ,\ b_i\ ,\ g_{i1}\ ,\ g_{i2}\ ,\ h_{i1}\ ,\ h_{i2}]^T. \tag{6.10}$$

The coefficients for the sinusoidal terms do not affect the contractivity of the map, therefore the s_i term is the only variable which needs to be examined in performing the contractivity check.

6.1.3 Nonlinear Addressing

Because the two-dimensional domain block data is already permuted via the eight isometries, the same eight nonlinear addressing functions as in Table 4.1 were not used for the two-dimensional case. Instead a subset of these functions was implemented, eliminating the redundancies generated because of the isometries. The new set of four functions is given in Table 6.1.

The nonlinear addressing function, $g(\cdot)$, is inserted into the map equation (6.1) to get:

$$\hat{r}_{x_{R_i},y_{R_i}}[x,y] = b_i + g_{i1}\cdot x + g_{i2}\cdot y + s_i \cdot \hat{d}_{x_{D_i},y_{D_i}}[g(x),g(y)], \tag{6.11}$$

where each $g(\cdot)$, can use a different function number. Thus, four bits are required to store the function indices for each map.

Table 6.1: Two-dimensional nonlinear addressing functions

#	$g(n)$
0	n
1	$L_R \cdot sin(\pi/2 \cdot n/L_R)$
2	$L_R \cdot (1.0 - sin(\pi/2 \cdot n/L_R))$
3	$L_R \cdot sin(\pi \cdot n/L_R)$

With the addition of the nonlinear addressing function to the two-dimensional maps, the number of ways that each domain can be permuted increases dramatically. The four functions implemented in each of the two axes, combined with the eight isometries, results in 128 different transformations to test. This computational expense may need to be taken into consideration in evaluating this approach.

6.2 Other 2D Maps

In addition to the previously mentioned extensions from the one-dimensional case, it is also possible to use other modifications to the basic affine transformation.

6.2.1 Multiscale Maps

All of the methods to date have restricted the domain blocks to be fixed at twice the size of the range blocks in both x and y. By changing this scaling factor, and keeping it integer to avoid the sampling issues discussed in Section 3.2, a variety of domain sizes can be used. With this modification, and using a scaling factor of S, equation (5.7) can be written:

$$\hat{d}_{x_{D_i},y_{D_i}}[x,y] = d_{x_{D_i},y_{D_i}}[S \cdot x, S \cdot y]. \tag{6.12}$$

Because no performance increase was seen with the use of averaging in the decimation stage of transforming the data from the domain blocks, this method was only implemented for the non-averaged case.

6.2.2 Multiple Domain Maps

As was seen in the previous chapter, the IFS coding technique seems to be limited in terms of how accurately an image can be coded. In coding a more complicated range block, it would be beneficial to be able to combine more than one domain block, because each additional domain block would increase the dimensionality of the map and therefore its ability to model the range block accurately. A transformation allowing multiple domain blocks can be written as a modified version of equation (5.9),

$$\hat{r}_{x_{R_i},y_{R_i}}[x,y] = b_i + g_{i1} \cdot x + g_{i2} \cdot y + \sum_{j=1}^{N} s_{i_j} \cdot \hat{d}_{x_{D_{i_j}},y_{D_{i_j}}}[x,y], \qquad (6.13)$$

where N is the number of domain blocks used, and $(x_{D_{i_j}}, y_{D_{i_j}})$ is the address of the j^{th} domain block for the i^{th} map. The parameters can again be determined through least squares as before. While this approach should certainly improve the accuracy of the maps, the search becomes absurdly long for as few as two domains. What is needed is an alternative approach to selecting the domains, which will be covered in Section 6.4.

6.2.3 Adaptive Block Size Coder

Concurrently with [25] we developed a coder which used adaptive block sizes, with range block sizes from 16×16 down to 4×4 pixels. This range of block sizes was found to provide the best looking reconstructed image. Using larger 32×32 range blocks resulted in objectionable blocking artifacts.

The image was initially divided on a uniform grid of 16×16 pixel range blocks, then each range block was coded and the error was determined based on equation (5.10) for each block. If the error for any range block exceeded a predetermined threshold, then that block would be divided. The decision to divide a block was determined based on the Mean Squared Error (MSE) for each of the four quadrants

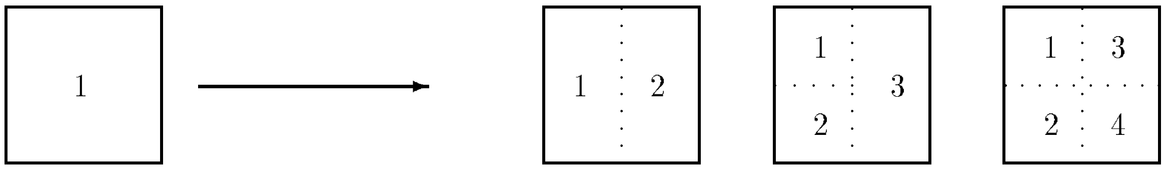


Figure 6.1: The three ways that a range block was divided into two through four blocks. With all possible rotations, there are seven distinct divisions.

in the range block. If two quadrants whose error was below the threshold error were adjacent, then they were grouped together to form a larger block, while the subblocks which had a larger error were coded separately. In addition, in subsequent divisions, the smallest created block was 4×4 pixels. This differs from the standard quadtree algorithm in that each block is not always divided into four smaller blocks. Therefore if just half of the block contained data which was difficult to model, it could be separated and coded independently. Thus, for example, a 16×16 block could be divided into a 16×8 and two 8×8 blocks. Figure 6.1 illustrates how one range block can be divided into three different configurations using two through four blocks. With the various rotations, there are seven possible ways to divide a range block using this approach. In Figure 6.2, the LENA image has been partitioned using this method and the concentration of smaller blocks can be seen around the more complex portions of the image.

6.3 Search Strategies

In addition to improving the accuracy of the maps in the IFS, it is also beneficial to examine other ways to search for the proper domains for each range block. The search has been the most computationally intensive portion of the IFS coding technique.

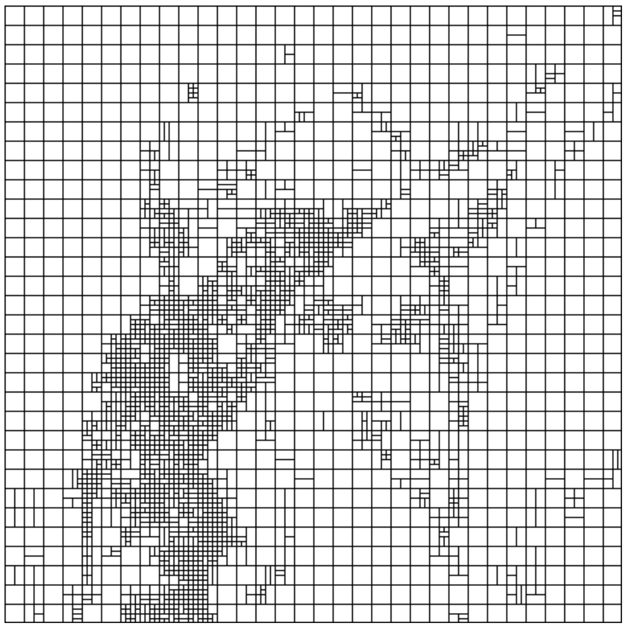


Figure 6.2: Adaptive block size partitioning of LENA image

6.3.1 Proximity Maps

In an image there is significant correlation between adjacent pixels, and also between adjacent blocks of the image. In an effort to take advantage of this correlation, the search was limited to the immediate area around the map's range block. In these tests the search was implemented over the closest 256 domain blocks. In addition, the address storage requirements are reduced to a total of only eight bits for each map because the domain address can be stored as an offset from the range address. One disadvantage of the proximity map is that it is not possible to use the eventually contractive mappings as discussed in the previous chapter. Because the domains and ranges for each map are all spatially close, any expansive map will most likely prevent the attractor from converging.

6.3.2 Fractal Dimension

In coding an image with an IFS, we are modeling the image with a deterministic fractal. One characteristic of a fractal is that the object will exhibit the properties of self-similarity or scale invariance [1]. This measurable quantity, the fractal dimension, was used as the basis for classifying the domain and range blocks. First some background on the fractal dimension is given, followed by a description of the algorithm used to compute the local fractal dimension of the range and domain blocks.

Mathematically the properties of self-similarity and scale invariance are manifested by a non-integer dimension, and a fractal is therefore defined as any set for which the Hausdorff-Besicovich (HB) dimension is greater than the topological dimension [1]. The HB dimension of an object, E, is defined:

Definition 6.1 (Hausdorff-Besicovich Dimension)

$$D(E) = \sup\{0 \leq d < \infty : H_d(E) > 0\}$$

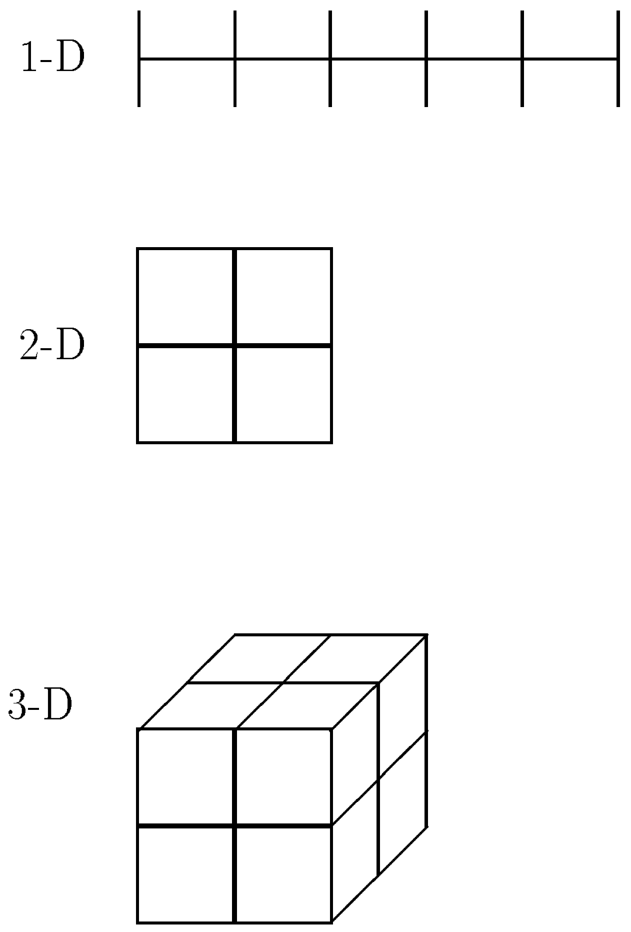
with

$$H_d(E) = \lim_{\epsilon \to 0} \left(\inf_U \sum_i |U_i|^d \right)$$

where the inf *is taken over all countable covers U of E such that $|U_i| \leq \epsilon$ and $|U_i| = \sup\{|x - y| : x, y, \in U_i\}$.*

The HB dimension is difficult to work with and is not generally used for the determination of fractal characteristics of data. Because it is also not an intuitive definition, a more useful definition can be derived from our conceptual notion of the dimension of an object.

When an object is described as being one-dimensional, this means that the object may be divided into N equally sized parts, each of which is $r = \frac{1}{N}$ of the size of the entire object. Figure 6.3 illustrates this for several objects with integer dimensions. A two-dimensional object may similarly be divided into N pieces, each of which is $r = \frac{1}{N^{1/2}}$ of the entire object. The final example is a three dimensional object, again divided into N parts, each of which is $r = \frac{1}{N^{1/3}}$ of the original.



Divide object into $N = 5$ parts
Each scaled by ratio $r = \frac{1}{5}$
$Nr^D = 5(\frac{1}{5})^1 = 1$

Divide object into $N = 4$ parts
Each scaled by ratio $r = \frac{1}{4^{\frac{1}{2}}} = \frac{1}{2}$
$Nr^D = 4(\frac{1}{2})^2 = 1$

Divide object into $N = 8$ parts
Each scaled by ratio $r = \frac{1}{8^{\frac{1}{3}}} = \frac{1}{2}$
$Nr^D = 8(\frac{1}{2})^3 = 1$

Figure 6.3: Example of intuitive definition of dimension for objects with $D = 1, 2$ and 3. Concept can be extended to fractional dimensions.

The trend is that a D-dimensional object can be divided into N equally sized parts that are

$$r = \frac{1}{N^{\frac{1}{D}}} \tag{6.14}$$

of the original. The relationship between the ratio of the size of the part to the whole, r, the number of blocks, N, and the dimension of the object, D, can be written as

$$Nr^D = 1. \tag{6.15}$$

Solving for D gives:

$$D = \frac{\log(N)}{\log(\frac{1}{r})}. \tag{6.16}$$

This equation can be applied to the Cantor Middle Thirds set seen in Section 3.3. The Cantor fractal was characterized by replacing the object by $N = 2$ smaller objects, each of which is $r = \frac{1}{3}$ of the whole. Thus the fractal dimension for the Cantor Middle Thirds set can be determined to be:

$$D = \frac{\log(N)}{\log(\frac{1}{r})} = \frac{\log(2)}{\log(3)} = 0.631. \tag{6.17}$$

The dimension equation (6.16) is the basis for many fractal dimension measurement algorithms, and is the basis for the Box-Counting Algorithm which was implemented to calculate the local fractal dimension for comparing potential domain blocks for each range block.

By interpreting the 512×512 eight-bit gray scale image as consisting of a grid of $512 \times 512 \times 256$ cubes, it becomes a simple task to count the number of cubes intersected by the image and determine the fractal dimension using equation (6.16). This idea can be extended to compute what is referred to as the local fractal dimension. By restricting the counting of the cubes to a fixed size neighborhood around each point, a measure of the fractal dimension in the proximity of each point can be determined. Because we are interested in matching domain and range blocks, the fractal dimension was computed for $L_R \times L_R$ and $2L_R \times 2L_R$ sized blocks.

It is instructive to look at the synthesized image consisting of the local fractal dimensions scaled to the range of $[0, 255]$. Figure 6.4 shows this image for the 512×512 version of LENA. The edges in the image have a higher fractal dimension than the smooth areas, and from this image it is easy to see why the local fractal dimension has been used as the basis for edge detection algorithms [4, 47, 48].

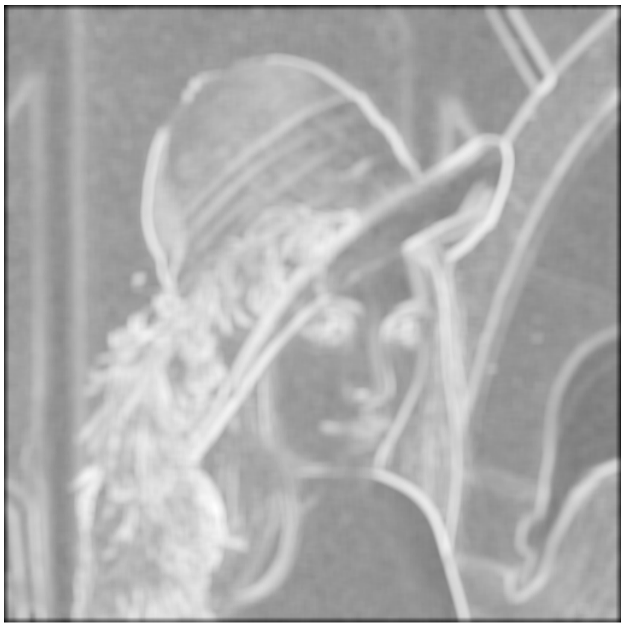


Figure 6.4: Local fractal dimension of LENA image

The local fractal dimension was used as a basis for the domain block search by first computing the local fractal dimension for each range and domain block. Then for each range block, a search was performed over the 256 domain blocks whose local fractal dimension was close to the range block's local fractal dimension. The range of the local fractal dimension used for each range block was adjusted to ensure that the pool of domain blocks contained at least the desired 256 blocks. In this manner, each range block had the same size pool of potential domain blocks to choose from, and these blocks were similar in a fractal sense. Additionally, because these domain blocks were selected from all over the image, it is possible to use the eventually contractive mappings as discussed previously.

6.3.3 Hierarchical Block Matching

Typically in an image there are regions where the texture will be similar. A Hierarchical Block Matching (HBM) type of approach was implemented to look iteratively for a region similar to the range block being coded. This method proceeds by initially checking a widely scattered collection of potential domain blocks, then the domain block which has the smallest error is taken as the center for a smaller collection of potential domain blocks, and the search is repeated. In this manner, a number of domain blocks are examined, which are progressively focused in an area where the closest match was found in the previous search.

For each range block, sixteen surrounding domain blocks were tested in a grid fashion, as is shown in Figure 6.5. The initial distance between horizontal and vertical test points was set to 128 pixels. Each inter–testpoint distance was divided by four to avoid any redundant tests. A typical series of center points for the domain maps to test is shown in this figure. By starting with an initial step size of 128 in both x and y and performing four iterations of this algorithm, one quarter of the total number of domain blocks are reachable, with the unreachable domain blocks being those with odd numbered rows and columns. This allows far more domain blocks to be addressed with a minimal number of checks than the proximity maps discussed previously. Whereas the proximity maps were implemented with 256 domains examined, the HBM approach, as specified here, requires only 64 domains to be evaluated. In addition, because the domain block may be located far from the range block, the use of eventual contraction mappings is also possible.

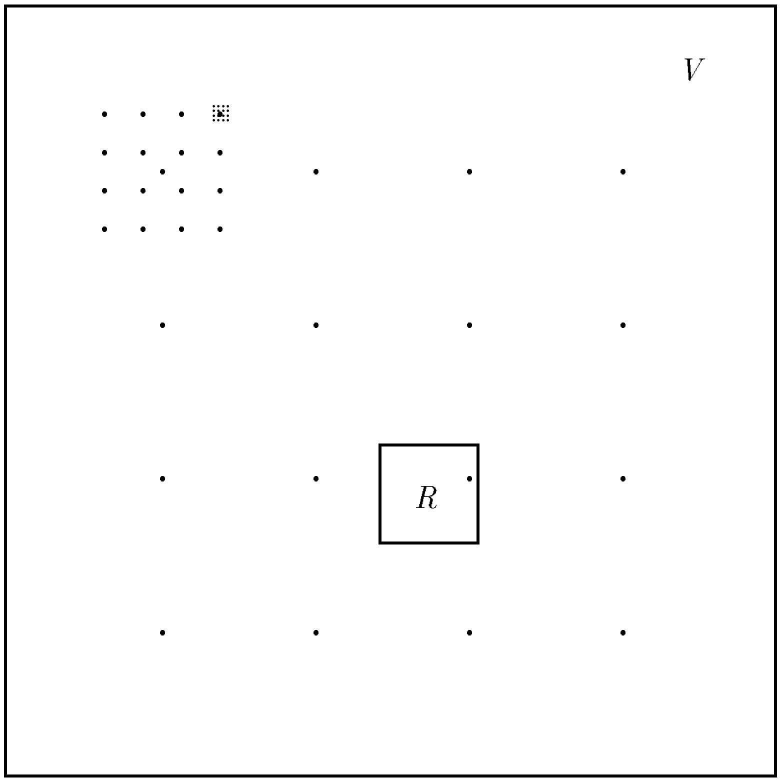


Figure 6.5: Hierarchical block matching example. Each point represents the center of a domain to be tested.

6.4 Searchless Based Approach

As noted previously, occasionally it is desirable for some maps in the IFS model to contain more information. Presently there are two ways to address this problem. The first is to use a quadtree type of approach and reduce the size of the block until the block is able to handle the amount of information to be coded, and the other method is to use a more complicated map in transforming the domain block. Both of these methods suffer from the additional complexity and overhead of having to evaluate several different types and sizes of maps in modeling a range block. It would be nice to have a method which allows a rapid determination of the necessary parameters to model a block. In addition, it would be advantageous for the method to be more flexible in modeling both simple and more complicated blocks.

The idea of multiple domain blocks offers the potential of using a larger variety of information in modeling the range block, however the implementation of such an idea is not computationally feasible as it was presented in Section 6.2.2. In this section a new type of IFS is introduced which attempts to alleviate this limitation of the IFS model.

6.4.1 Orthonormal Basis Approach

All of the previous approaches to creating maps are search-based and require a set of domain blocks be evaluated for each range block. In this orthonormal basis approach, a set of orthonormal basis vectors will be created by the Gram–Schmidt procedure and the range blocks will be coded by projecting the blocks onto this basis. The advantage of using an orthonormal basis is two-fold. First, there is only one search for a set of domain blocks. Thus, once a suitable collection of domain blocks have been chosen, the actual encoding of the range blocks is based on using this fixed subset of the domain blocks. The second advantage is that the actual encoding process is relatively fast because the encoding of the range blocks only requires a simple projection onto the orthonormal basis. As with all compression techniques, we are concerned with reducing the dimensionality of the data to be stored. Thus we wish to find a smaller subspace in which to accurately represent each range block. The goal in determining the orthonormal basis will be to create a basis which allows each range block to be accurately represented with a minimum number of the basis vectors. By reducing the dimensionality of each of the range blocks, compression can be achieved. In addition, the amount of compression will vary automatically with the complexity of the range block. More complicated range blocks will require a higher dimensional subspace to be represented. In this manner, this new model will be able to accommodate widely differing range blocks.

First a discussion of how the range and domain blocks are viewed as vectors is presented, followed by the method of generating the orthonormal basis vectors and the encoding and decoding processes. Once the basic algorithm has been introduced, several approaches are discussed for selecting a set of domain blocks to use.

If each of the elements in the range block, $r_{x_{R_i},y_{R_i}}[x,y]$, are ordered sequentially to form a vector of length L_R^2,

$$\mathbf{r}_i = \left[r_{x_{R_i},y_{R_i}}[0,0]\ ,\ r_{x_{R_i},y_{R_i}}[1,0]\ ,\cdots\ ,\ r_{x_{R_i},y_{R_i}}[L_R-1,L_R-1]\right]^T, \tag{6.18}$$

then each range block can be interpreted as a vector in an L_R^2-dimensional vector space. A similar interpretation can be applied to the decimated domain blocks to form domain vectors, $\hat{\mathbf{d}}_i$. Thus, for the typical 8×8 block size, we are working in a 64-dimensional vector space. As with all compression techniques, we are concerned with reducing the dimensionality of the problem. Thus we wish to find a smaller subspace in which to accurately represent each range vector.

Because the domain blocks are decimated early in the process, we are assured of a contraction in two of the three dimensions present in the block. To achieve a contraction in all three dimensions, domain vectors will be chosen which are *large* in an L_2 sense.

The traditional map equation (5.9) may also be placed in this setting and viewed as the linear combination of four vectors in this space,

$$\hat{\mathbf{r}}_i = b_i \cdot \mathbf{v_1} + g_{i1} \cdot \mathbf{v_x} + g_{i2} \cdot \mathbf{v_y} + s_i \cdot \hat{\mathbf{d}}_\mathbf{i}, \tag{6.19}$$

where $\mathbf{v_1}$ is a vector with all ones in it, $\mathbf{v_x}$ and $\mathbf{v_y}$ are vectors which provide the 'tilt' in each of the x and y axis respectively,

$$\mathbf{v_x} = [\ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ \cdots 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ ]^T, \tag{6.20a}$$

$$\mathbf{v_y} = [\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ \cdots 7\ 7\ 7\ 7\ 7\ 7\ 7\ 7\ ]^T, \tag{6.20b}$$

and $\hat{\mathbf{d}}_\mathbf{i}$ is the decimated domain block in vector form.

The advantage of viewing the problem in this manner, is that now we can take advantage of well known vector space properties. In particular, we are interested in forming an orthonormal basis from the domain vectors. With these vectors, the range blocks can be encoded with a simple projection operation, and the map parameters will be the weights for this orthonormal basis. The set of basis vectors will consist of several which are known *a priori*, as well as those derived from the domain blocks. As long as the maps form contraction mappings we are assured of convergence and better control over the reconstructed image error.

For a range block of size $L_R \times L_R$, let $D = L_R^2$ be the length of the range and decimated domain vectors, which will be noted by $\{\mathbf{r}_i\}_{i=1}^{N_R}$ for the range vectors and $\{\hat{\mathbf{d}}_i\}_{i=1}^{N_D}$ for the decimated domain vectors. The set of three fixed basis vectors can be orthonormalized to form the first three of the required D orthonormal basis vectors. The remaining basis vectors will be chosen to span the $N_S = (D-3)$-dimensional subspace, S^0, orthogonal to the space spanned by these three vectors.

A projection operator for the subspace spanned by a vector, $\mathbf{v}_i$, can be written:

$$P_{\mathbf{v}_i} = \mathbf{v}_i(\mathbf{v}_i^T\mathbf{v}_i)^{-1}\mathbf{v}_i^T. \tag{6.21}$$

Therefore the operator to project a vector into the desired subspace, S^0, is given by

$$P_{S^0} = I - (P_{\mathbf{v}_1} + P_{\mathbf{v}_2} + P_{\mathbf{v}_3}), \tag{6.22}$$

where I is the identity matrix, and $\mathbf{v}_1, \mathbf{v}_2$ and $\mathbf{v}_3$ are the *a priori* basis vectors. Using this projection operator, the projected versions of the range vectors can be computed as:

$$\mathbf{s}_j^0 = P_{S^0}\mathbf{r}_j. \tag{6.23}$$

Figure 6.6 illustrates this process for a simple one-dimensional *a priori* vector subspace in a two-dimensional example.

The combination of the three fixed vectors, $\mathbf{v}_1, \mathbf{v}_2$, and $\mathbf{v}_3$ with the N_S domain vectors form a set of D vectors which will span the space of the range vectors. If the selected N_S domain vectors are denoted as $\{\mathbf{b}_i\}_{i=1}^{N_S}$, then this set of vectors can be written as a matrix B, where

$$B = [\,\mathbf{v}_1\,,\,\mathbf{v}_2\,,\,\mathbf{v}_3\,,\,\mathbf{b}_1\,,\,\dots\,,\,\mathbf{b}_{N_S}\,]\,. \tag{6.24}$$

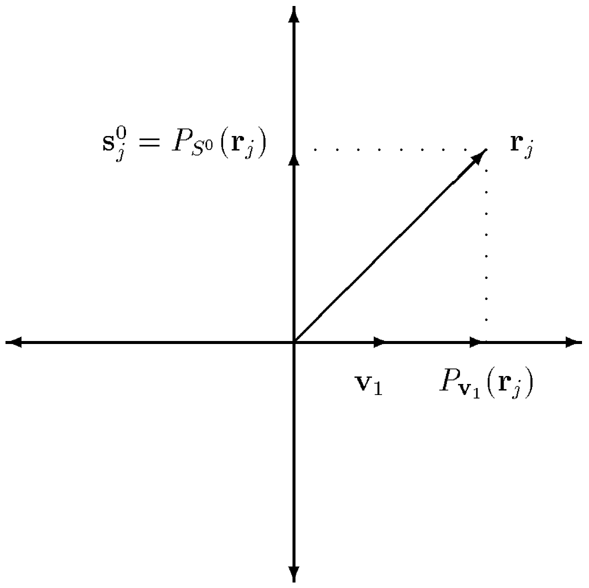

$\mathbf{s}_j^0 = P_{S^0}(\mathbf{r}_j)$
$\mathbf{r}_j$
$\mathbf{v}_1$
$P_{\mathbf{v}_1}(\mathbf{r}_j)$

Figure 6.6: Example of projection of a range vector onto a subspace orthogonal to an *a priori* basis vector subspace. This example uses only one *a priori* vector, $\mathbf{v}_1$, for a simplified two-dimensional example.

In order to allow rapid computation of the appropriate weights for these vectors to represent each range vector, these basis vectors are orthonormalized with the Gram–Schmidt procedure, resulting in a Q matrix containing the orthonormal basis vectors for the space.

Once the set of basis vectors are determined, the coding process is straightforward. Thus, to represent a given range vector, $\mathbf{r}_i$, with a set of weights, $\mathbf{w}_i$, the following is true,

$$\mathbf{r}_i = Q\mathbf{w}_i. \tag{6.25}$$

Therefore, premultiplying by Q^T yields

$$\mathbf{w}_i = Q^T\mathbf{r}_i, \tag{6.26}$$

which is used to determine the weights. These weights can quickly be calculated for each $\mathbf{r}_i$, and these two equations define the basic encoding and decoding process.

The remaining task to implement the coder is an algorithm to select the N_S domain vectors to use. The desired characteristics are that the vectors should:

- provide contraction mappings,
- be as nearly orthogonal as possible, and
- form a set of basis vectors, which best allows each range vector to be represented with as few vectors as possible.

Several methods were developed based on these three criteria and are discussed below.

Covariance Method

Rather than search through the relatively large set of domain vectors, the range vectors were analyzed to determine the optimal basis vector directions, then the domain vectors were searched to find the largest vector in each of these directions. This approach has two advantages: first we are primarily interested in representing the range vectors, thus the directions for the basis vectors should be based on the range vectors themselves; secondly, there are far fewer range vectors than domain vectors and the computational task of determining the basis vectors is reduced.

The range vectors were searched iteratively to find the "best" direction for the next basis vector. After each direction was selected, the remaining range vectors were projected into the subspace orthogonal to this vector. The algorithm begins by projecting all of the range vectors into the subspace, S^0, perpendicular to the subspace spanned by the *a priori* vectors. At the k^{th} iteration, the i^{th} projected range vector will be noted by $\mathbf{s}_i^k$ which resides in a corresponding subspace, S^k. The optimal basis vector direction was determined by taking the $\mathbf{s}_i^k$ vector with the largest correlation to all of the other $\mathbf{s}_i^k$ vectors. Thus the vector, $\mathbf{s}_l^k$, which maximizes the equation,

$$C_i = \sum_{j=1, j\neq i}^{N_R} | < \mathbf{s}_i^k, \mathbf{s}_j^k > | \tag{6.27}$$

was selected, where $| < \mathbf{s}_j^k, \mathbf{s}_i^k > |$ is the absolute value of the inner product of $\mathbf{s}_j^k$ and $\mathbf{s}_i^k$.

Once each basis vector direction has been determined, then the remaining $\mathbf{s}_i^k$ vectors are projected onto the subspace not spanned by $\mathbf{s}_l^k$, using a projection operator of the form:

$$P_{S_k} = I - \mathbf{s}_l^k(\mathbf{s}_l^{k^T}\mathbf{s}_l^k)^{-1}\mathbf{s}_l^{k^T}. \tag{6.28}$$

The chosen basis vector direction is saved as $\mathbf{t}_k$ and the process is repeated until the necessary N_S vectors are obtained. Essentially the Gram-Schmidt procedure is performed on the range vectors. However, the vector used in each step is the vector with the largest correlation with the other remaining vectors. In this manner, the set of N_S selected vectors, or *direction vectors*, $\{\mathbf{t}_i\}_{i=1}^{N_S}$, are determined which best represents the subspace S^0. In addition, these *direction vectors* are orthogonal. Thus the last two criteria in the list above are satisfied. Figure 6.7 illustrates this process for a simple two-dimensional, three vector case. In this example, $\mathbf{s}_3^k$ was chosen as the vector with the largest correlation. Therefore the remaining vectors are projected into the subspace S^{k+1}. Effectively, the component of the remaining vectors which can be represented by $\mathbf{s}_3^k$ is removed.

However, domain vectors are needed for contraction mappings, and a search must be performed through the domain vectors to find the best set of domain vectors for these *direction vectors*. Because the order in which the $\mathbf{t}_i$ vectors were selected was important, with the most significant vector coming first, the same order was used in finding the domain vectors. The domain vector with the largest component in the direction of the *direction vector* was used, where the projection of a domain vector, $\hat{\mathbf{d}}_j$, onto a *direction vector*, $\mathbf{t}_i$, may be written as:

$$P_{\mathbf{t}_i}(\hat{\mathbf{d}}_j) = \frac{|\hat{\mathbf{d}}_j \cdot \mathbf{t}_i|}{||\mathbf{t}_i||}, \tag{6.29}$$

where $|| \cdot ||$ indicates the L$_2$ norm. Because it is possible that one domain vector has the largest component on more than one *direction vector*, each domain vector

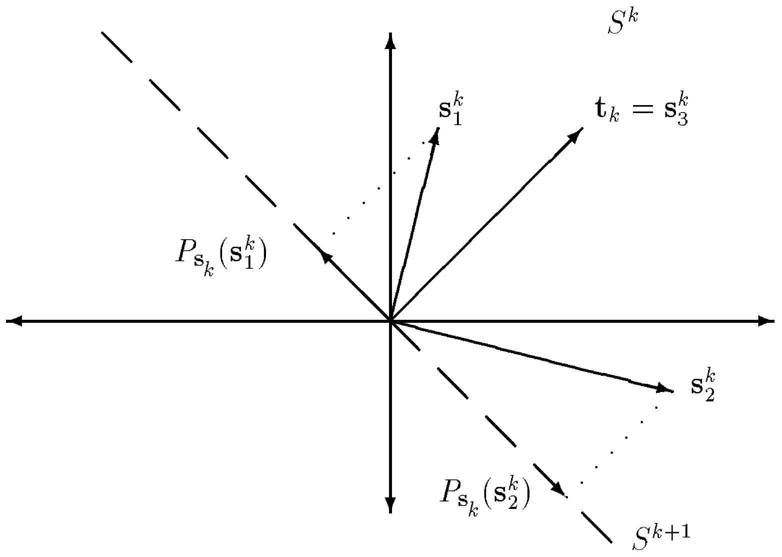

S^k
$\mathbf{s}_1^k$
$\mathbf{t}_k = \mathbf{s}_3^k$
$P_{\mathbf{s}_k}(\mathbf{s}_1^k)$
$\mathbf{s}_2^k$
$P_{\mathbf{s}_k}(\mathbf{s}_2^k)$
S^{k+1}

Figure 6.7: Example of choosing the vector with the largest correlation to the other vectors and projecting the remaining vectors into the orthogonal subspace. After the selection of $\mathbf{t}_k$, the remaining vectors are projected into the space S^{k+1}.

was only allowed to be used once. Figure 6.8 shows how the domain vector with the largest component in the direction of the *direction vector*, $\mathbf{t}_i$, is selected.

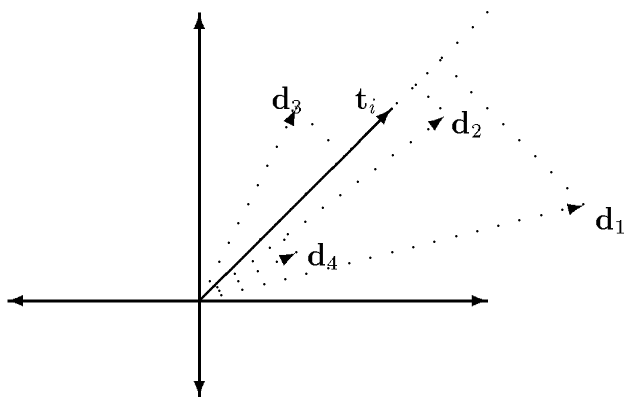


Figure 6.8: Selection of the domain vector with the largest projection on the *direction vector* $\mathbf{t}_i$. In this example, $\mathbf{d}_1$ is the selected domain vector.

In summary, this algorithm proceeds as follows:

1. Loop N_S times to find the *direction vectors*, $\mathbf{t}_i$
   (a) Find vector with largest correlation using equation (6.27)
   (b) Save vector as $\mathbf{t}_i$
   (c) Project remaining into subspace using equation (6.28)
2. Loop N_S times for each $\mathbf{t}_i$ to find best domain vector
   (a) Find domain vector with largest component in direction of $\mathbf{t}_i$ using equation (6.29)
   (b) Save vector and eliminate from list of domain vectors

The final result is the set of domain vectors to use for encoding the image. The complete encoding algorithm is provided in Section 6.4.1.

K-Means Based Approach

The previous approach is optimal in an incremental sense, that is, the optimal *additional* basis vector is selected at each iteration. This does not provide the best set of two, or n, vectors to represent the subspace S^0. For example, if each of the vectors were going to be coded with a single vector, then a K-means approach would be optimal, with $k = 64$. Essentially, K-means is a clustering algorithm which guarantees convergence to a local minimum in grouping a set of vectors into k clusters. A complete discussion of the K-means algorithm can be found in [49]. Better overall results might be obtained by looking at the range vectors in groups. Thus, we will not try and find the best single vector to represent the entire set of range vectors. Instead, the range vectors will first be clustered into a fixed number of groups, then each of these groups will be represented by a single range vector.

The K-means algorithm will group the vectors into clusters. Thus $\mathbf{s}_j$ and $-\mathbf{s}_j$ are very different as perceived by this algorithm. However, because we are interested in finding a set of basis vectors, it would be preferable to treat these two vectors as equivalent. A simple modification to the data prior to running the K-means algorithm eliminates this problem. All of the vectors were forced to reside in the same half-space. The selected half-space was the one which included the positive axes, and any vector which was outside of this half-space was set to its negative. This can easily be checked by examining the sign of the sum of the elements in each vector,

$$h_j = \sum_{i=1}^{D} \mathbf{s}_j[i]. \tag{6.30}$$

If $h_j < 0$, then the negative of that vector was used.

Figure 6.9 illustrates this modification for a simple two-dimensional example. In this example, vectors **c** and **d** on the left side of the dashed line are changed to their negative to form the corresponding vectors **c** and **d** in the right half-plane.

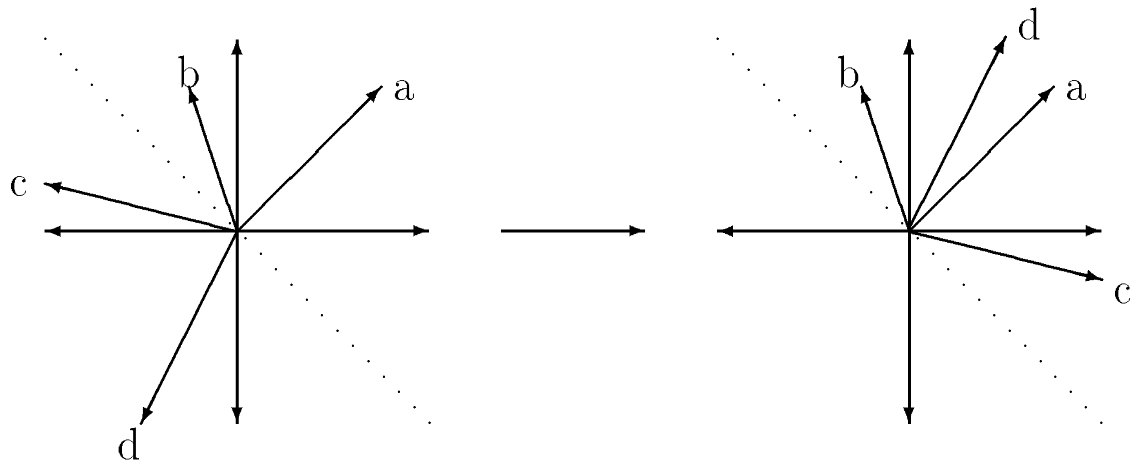


Figure 6.9: Example of moving vectors to the same half-space

In addition, the magnitude of each of the range vectors is not significant, therefore the vectors were normalized prior to running the K–means routine. Once the k means have been determined, the domain vectors were searched to find the set which match these *direction vectors*.

Search Method

While the covariance method is optimal in the sense that each additional basis vector best represents the remaining subspace, there is a cost associated with this optimality. Performing a Gram–Schmidt on the range vectors, and the ensuing search through the domain vectors can be computationally expensive for large images. Similarly, the K-means algorithm begins to be computationally taxing with larger numbers of vectors. A sub-optimal approach was developed, which is based on finding a smaller set of domain vectors, from which the basis vectors are selected.

The set of domain vectors was first reduced by discarding those vectors whose magnitude of the projection in the subspace S^0 was less than a threshold, T_e, to form a set of projected "large" vectors $\{\mathbf{e}_i\}$. Once the domain vectors were reduced to this subset, then the set of N_S vectors was chosen to span the subspace S^0. The first N_S vectors were selected,

$$\mathbf{b}_i = \mathbf{e}_i \tag{6.31}$$

for $i = 1 \ldots N_S$, and a *quality factor*, defined by

$$q = \sum_{i=1}^{N_S} \sum_{j=i+1}^{N_S} |\mathbf{b}_i \cdot \mathbf{b}_j|, \tag{6.32}$$

was determined for this set of basis vectors. Essentially the desired set of vectors should be as orthogonal as possible. This was accomplished by finding the set of vectors which minimize q. Once an initial set was collected, each of the remaining $\mathbf{e}_i$'s were substituted with the closest $\mathbf{b}_i$, and a new q was calculated. If the quality factor was reduced, then this new basis vector was kept, otherwise the next $\mathbf{e}_i$ was examined, until all of the $\mathbf{e}_i$ were checked. The final set of $\mathbf{e}_i$'s is a set of nearly orthonormal basis vectors for the subspace S^0, which can be combined with the *a priori* vectors to form the B matrix as in equation (6.24).

Searchless Method

Because any search over the domain vectors will tend to be time consuming for large images, a searchless approach was implemented which simply takes the required number of domain vectors from the domain blocks evenly spaced across the image. While this method does not insure that the set of vectors will span the space, and consequently the performance can be expected to be less than the other methods, it is extremely fast for small blocks. In coding an image with 8×8 blocks, after the Gram-Schmidt procedure is complete, there are 64 multiplies to code each pixel.

Encoding

The encoding method is based on equation (6.26), which provides the weights which specify the vector in the rotated coordinate system. In order to achieve a compression, the weights must be quantized. Ideally we would like to discard most of the weights. The encoding algorithm consists of the following steps.

1. Determine the domain vectors to use by one of the above methods.
2. Form the B matrix and use the Gram-Schmidt procedure to get Q.
3. Determine the map parameters, $\mathbf{w}_i$, with equation (6.26).
4. Save those weights which exceed a threshold T_w.

The final encoded image consists of the indices for the N_S domain vectors and the quantized weights for each map which exceeded the threshold. By adjusting the threshold, the accuracy of the reconstructed image can be controlled. In addition, the quantization approach can be different for the *a priori* vectors, which will tend to have a different distribution as compared to the weights for the $\mathbf{b}_i$ vectors.

Decoding

In order to reconstruct the $\mathbf{r}_i$, which will provide the reconstructed image, equation (6.25) is used. This requires the Q matrix, which can be obtained by performing a Gram-Schmidt procedure on the B matrix.

The reconstruction process is much simpler than the encoding procedure, and begins with any initial image, V_0, and iteratively performs the following steps.

1. Gather the basis vectors, $\hat{\mathbf{d}}_i$ from the image.
2. Form B as given in equation (6.24).
3. Perform the Gram-Schmidt procedure on B to get Q.
4. Compute each $\mathbf{r}_i = Q\mathbf{w}_i$ and save in the image.
5. Go to step 1.

The image will converge in a few iterations.

Larger Block Sizes

The previous IFS coding model was limited in the size of blocks that could be used and still accurately code each range block. When larger block sizes were used, the maps were unable to represent the range blocks accurately and the reconstructed image SNR rapidly decreased. Even the addition of the various nonlinearities was unable to prevent the image from degrading excessively. By using an orthonormal basis approach, a larger block size can be implemented. Those blocks which contain more information will simply require more basis vectors to represent them.

In addition to 8×8 block sizes, both 16×16 and 32×32 were implemented. However, several problems occur with the larger block sizes. With the 32×32 block size, the vector space is now of dimension 1024, and the B matrix alone has over one million elements. Performing the Gram-Schmidt procedure on a matrix of this size is extremely computationally intensive because the Gram-Schmidt procedure is of order N^3. For comparison purposes, this was performed only for the searchless method discussed previously. This larger block size also has the problem that there are only 256 range blocks. Thus we no longer have a large enough set of range blocks to determine an optimal set of direction vectors for the other algorithms.

6.5 Results

In order to compute the compression ratios for all of the possible combinations of map types with the different IFS models, the parameters need to be quantized for each map. After the quantization of the map parameters, the implementation of the different coders as well as a comparison of coding results will be given.

6.5.1 Quantization

Each of the maps for all of the methods consists of weights which must be quantized for storage. In general, the parameters to quantize include the bias value, b_j, as well as the tilt variables, g_{j1} and g_{j2}. A uniform quantization method was not always efficient due to the distribution of these parameters, and therefore, a Lloyd-Max quantizer was also implemented. The Lloyd-Max quantizer is basically a one-dimensional K-means algorithm [50], which was discussed in Section 6.4. Because the scaling factor for the domain block is limited in magnitude by the contraction requirement, a fixed decimal point storage method was used for s_j. With this method, typical storage requirements were nine bits for b_j, six for each g_{j1} and g_{j2}, and seven for s_j.

In addition to the standard affine IFS map, the condensation type map coefficients were quantized with a uniform quantizer, and good results were achieved with seven bits for each parameter h_{i1} and h_{i2}. The orthonormal basis approach was also tested with both uniform and Lloyd-Max quantization, with a slight improvement in using the Lloyd-Max approach. The other types of maps, nonlinear addressing and multi-scale, require additional bits to describe their operation, however no additional terms need to be quantized. The additional information required to complete the coded image is covered in the next section.

6.5.2 Construction of Coders

Three basic coders were implemented with the techniques outlined in this chapter. First was a basic, fixed block size coder, which used more complicated maps for those range blocks which could not be coded accurately with simple affine maps. A second coder proceeded by adapting the range block size to adjust for more complicated range blocks – dividing the range blocks until an acceptable error threshold was reached. The third coder used the orthonormal basis approach and included all basis vectors whose weights exceeded some threshold.

For each of these coders, the address for the domain block for each map must be stored. Depending on the search method used, the number of bits required will vary. Those methods which search over the entire image will require the full address, which is 18 bits for a 512×512 image. The proximity map with the closest 256 blocks will require only 8 bits for each address. The Hierarchical Block Matching approach required 16 bits for the four level – sixteen point method as discussed in Section 6.3.3, since it requires four bits for each level to specify which point was selected. Only the search over the fractal dimension requires that the entire address be saved. The orthonormal basis approach has the added overhead of the full addresses for each of the N_S domain blocks used to form the basis vectors. Then the weights in each map only require an index into the D basis vectors.

In the following paragraphs each of the coders are discussed in detail. For each of these methods, a threshold MSE is required to determine when the algorithm can stop with each range block. For the tests performed here, a value of 10.0 was used.

Fixed Block Size Coder

For the fixed block size coder, the goal is to use as simple a map as possible for those range blocks that do not contain much information. Because the more complicated blocks will pose a problem if the block size is too large, a block size of 8×8 pixels was chosen. Each range block was initially modeled with a constant value. If the error from equation (5.10) exceeded the threshold, then a more complicated map was used. If the constant block was inadequate, then a plane was used, followed by the normal affine map of equation (5.9) and then multiple scaling factors. Finally the nonlinear addressing and condensation type maps were used. In summary, the maps in the order that they were tried are given in Table 6.2.

Each of the various modifications to the affine map will require a different number of bits to code. As mentioned previously, the nonlinear addressing maps will require an additional four bits per map, and the eight isometries require three bits. The multiscale maps require enough bits to uniquely specify the scales available. In these tests scales ranging from 2 through 9 were implemented, which requires three bits per map. Finally, for each map a code is required to indicate which one of the six map types was used. By entropy coding these indicators, the storage requirements were minimized.

Table 6.2: Fixed block size coder – map functions

#	Type	$\hat{r}_{x_{R_i},y_{R_i}}[x,y]$
1	Constant	b_i
2	Plane	$b_i + g_{i1} \cdot x + g_{i2} \cdot y$
3	Basic Affine	$b_i + g_{i1} \cdot x + g_{i2} \cdot y + s_i \cdot \hat{d}_{x_{D_i},y_{D_i}}[x,y]$
4	Multiscale	$b_i + g_{i1} \cdot x + g_{i2} \cdot y + s_i \cdot d_{x_{D_i},y_{D_i}}[S \cdot x, S \cdot y]$
5	Nonlinear Addressing	$b_i + g_{i1} \cdot x + g_{i2} \cdot y + s_i \cdot \hat{d}_{x_{D_i},y_{D_i}}[g(x), g(y)]$
6	Condensation Type Maps	$b_i + g_{i1} \cdot x + g_{i2} \cdot y + s_i \cdot \hat{d}_{x_{D_i},y_{D_i}}[x,y]$ $+h_{i1} \cdot \sin(\pi \frac{x}{L_R-1}) + h_{i2} \cdot \sin(\pi \frac{y}{L_R-1})$

Adaptive Block Size Coder

The adaptive block size coder requires additional information to note which maps were divided, and how. The maps were stored sequentially, and one bit was used to indicate if the map had been split. Once a map was split, then an additional bit would indicate whether the split was vertical or horizontal, and the other half of the original map would follow immediately. If a map was divided, there would be a recursive check to see if there were further divisions, and of which type.

Orthonormal Basis Coder

The Orthonormal Basis Coder was implemented with 8×8, 16×16 and 32×32 pixel range blocks. The coded image consists of the index for the domain blocks that constitute the basis vectors, followed by the indices and weights for the basis which are used in each of the maps. Because the weights for the first three *a priori* basis vectors are always saved, the indices do not need to be saved for these weights.

Coding Results

With the fixed block size coder and the adaptive block size coder, the method to use to search for the domain block for each map is also a variable. The three search algorithms, proximity maps, fractal dimension and hierarchical block matching, must be evaluated also. Both the proximity maps and fractal dimension methods were implemented with 256 potential source domains. Thus the results can be compared fairly with regard to the computational cost. The HBM coder was implemented requiring only 64 domains to check, and consequently was four times faster than the previous two methods.

In implementing the various coders some characteristics were noticed. The use of eventually contractive mappings did not consistently improve the performance of the different models. This technique caused other problems as well. Because the scaling factor, d_i, could have a large magnitude, the range of bias values, b_i, seen for the maps with the relaxed constraint on d_i was greatly increased. This, in turn, caused problems in quantizing the bias parameters. In order to compensate for this, d_i was first quantized to get $\tilde{d}_i$, then if $|\tilde{d}_i| > 1.0$, the bias parameter was adjusted prior to quantizing to get $b_j/\tilde{d}_i$. This scaling effectively reduced the range of the bias terms and reduced the required bits for quantization. In the reconstruction stage, b_j was recovered by checking if $|\tilde{d}_i| > 1.0$, then adjusting the quantized bias term, $\tilde{b}_j$ to $\tilde{b}_j \cdot \tilde{d}_i$. This compensation technique was found to reduce the required number of bits by an average of five for the b_i term.

Table 6.3: Image coding results

Search Method	BPP	PSNR
HBM: Adaptive Block Size	0.55	30.5
with Eventually Contractive Maps	0.55	29.2
HBM: Fixed Block Size	0.81	31.5
with Eventually Contractive Maps	0.81	31.5
FD: Adaptive Block Size	0.62	30.5
with Eventually Contractive Maps	0.60	29.5
FD: Fixed Block Size	0.76	30.9
with Eventually Contractive Maps	0.85	31.7
Proximity: Adaptive Block Size	0.47	31.5
Proximity: Fixed Block Size	0.68	31.1
Orthonormal Basis: Covariance	0.44	30.5
Orthonormal Basis: K-means	0.47	30.2
Orthonormal Basis: Search Method	0.49	29.6
Orthonormal Basis: Searchless Method	0.50	27.0

6.5.3 Comparison of Results

Each of the coders was tested with the standard LENA image and the results are given in Table 6.3. While quantitatively the results are similar for each of the methods, there are some differences in the coding artifacts. Figure 6.10 shows the original 512×512 LENA image which was used in these tests. In Figure 6.12 the orthonormal basis approach results are given prior to quantizing. This illustrates the noise artifact which is introduced with this method, as well as the extremely high quality of the image. To see how this method is working, the basis vectors are shown in Figures 6.14a and 6.14b. In the first figure, the blocks are shown prior to the Gram–Schmidt process, and in the second figure the orthonormal vectors and their noise-like appearance is illustrated.

The methods which required the lowest number of bits per pixel were the search-based methods which used the adaptive block sizes as compared to the fixed block sized approach. In addition, the use of eventually contractive mappings gave inconclusive results with regards to SNR improvements, with half of the tests resulting in a lower SNR, and the other half having an increased SNR. The overall best performing method was the use of proximity maps with adaptive block sizes. This approach was also one of the faster methods implemented.

The orthonormal basis approaches also performed uniformly well with regard to the compression ratios achieved. The SNRs were slightly less for these methods. However it is important to visually examine the images for a subjective evaluation of this method. The reconstructed image, prior to quantizing is shown in Figure 6.12, and after quantizing in Figure 6.13.

It is interesting that the reconstructed image using all of the weights and without any quantization is not perfect. As it turns out, examination of the weights in the mappings shows that many are not contractions. While the existence of some expansion maps does not necessarily cause a problem [43], the maps here apparently have an excessive number. It is possible to look at the actual contraction factors for the maps through the relationship between the B and Q matrices. The matrix R relates the orthonormal basis to the original domain vectors: $B = QR$. Therefore $Q = BR^{-1}$, and equation (6.25) can be written

$$\mathbf{r}_i = BR^{-1}\mathbf{w}_i = B(R^{-1}\mathbf{w}_i). \tag{6.33}$$

Thus, the contraction factors are the elements of the vector $R^{-1}\mathbf{w}_i$.

In viewing the reconstructed images, the error signal is concentrated around the edges within the image, and is a "salt and pepper" type of high frequency noise, as can be seen in Figure 6.12. In an attempt to attenuate this noise, the weights which were not contraction mappings were scaled down to be contraction mappings during the reconstruction process. Then just prior to the last iteration of the reconstruction, the weights were restored to their full magnitude. Unfortunately, this reduced the final PSNR even further.

To facilitate comparisons with other coding methods, several different images were coded using the proximity map with the adaptive block size, as well as the orthonormal approach with the covariance method. Table 6.4 provides these results.

Table 6.4: Coding results for several images

Image	Proximity w/ Adaptive Size		Orthonormal Basis w/ Covariance	
	PSNR	bpp	PSNR	bpp
PEPPERS	29.0	0.367	30.8	0.551
BOAT	29.6	0.575	28.4	0.622
BABOON	20.3	0.528	19.7	0.672
MOFFETT	28.4	0.559	28.0	0.972

6.6 Conclusions

In this chapter we have seen several new types of maps for the two-dimensional IFS model. In addition several new search-based techniques were introduced and evaluated. Section 6.4 introduced a new approach to IFS modeling, the orthonormal basis IFS, which approaches the modeling problem from the perspective of considering the range and domain blocks as vectors in a vector space, and creating an orthonormal basis from which the range vectors are coded.

The different algorithms and models were evaluated using a standard test image, and the results show that the proximity maps combined with the adaptive block sizes produced the best overall performance. In addition the new orthonormal based approach with fixed basis vectors was the fastest coding method. However the resulting SNR was reduced slightly.

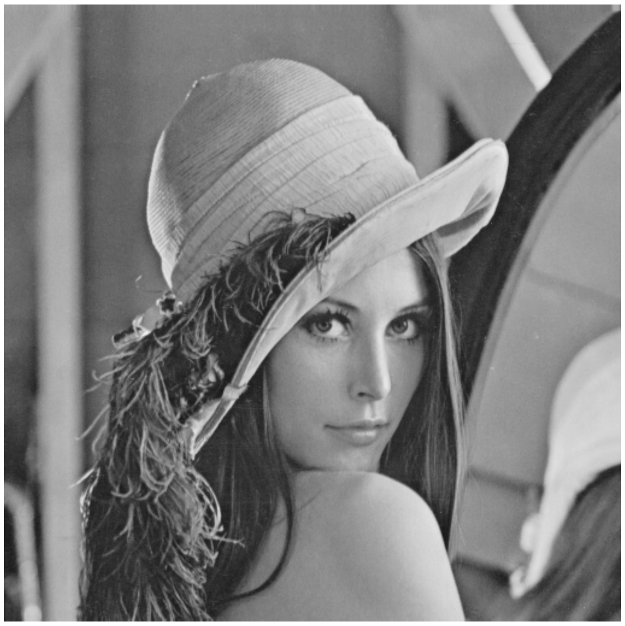


Figure 6.10: Original 512 × 512 eight bit gray scale LENA image

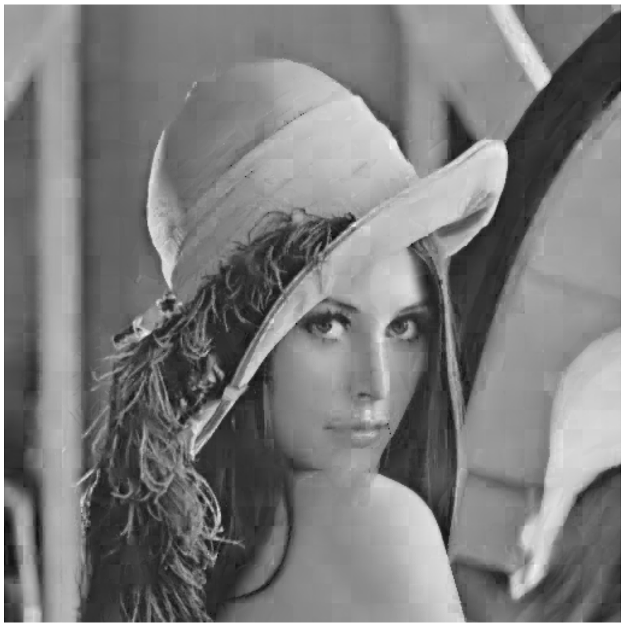


Figure 6.11: LENA image coded with proximity maps and adaptive block sizes at 0.47 bpp, PSNR = 31.5dB

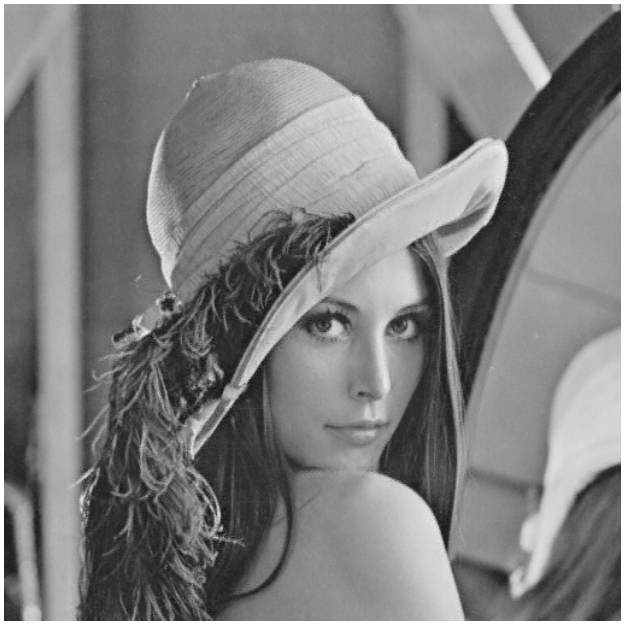


Figure 6.12: Lena image coded with orthonormal basis method prior to quantizing

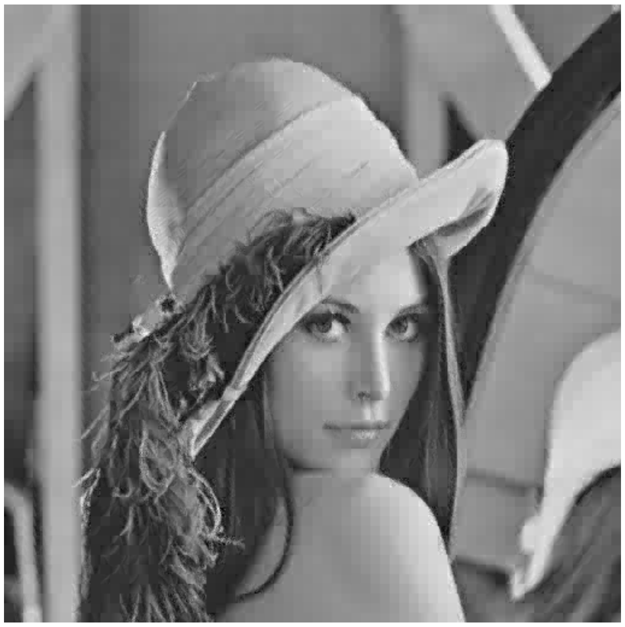


Figure 6.13: Lena image coded with orthonormal basis method at 0.44 bpp, PSNR = 30.5dB

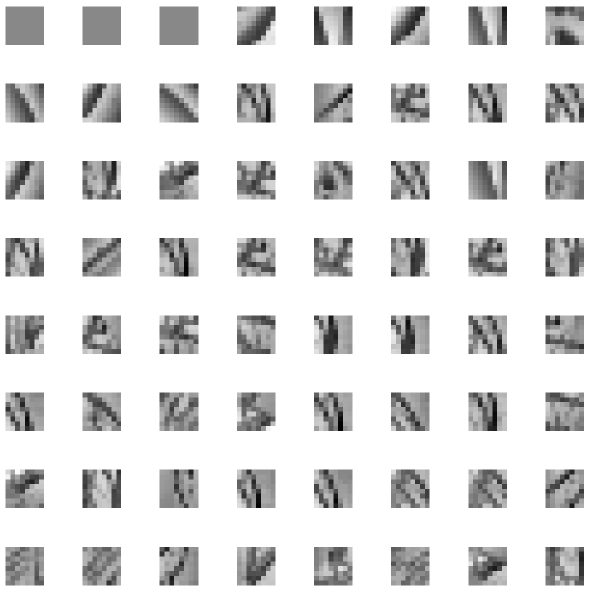


Figure 6.14a: Basis vector blocks for covariance method: before Gram–Schmidt

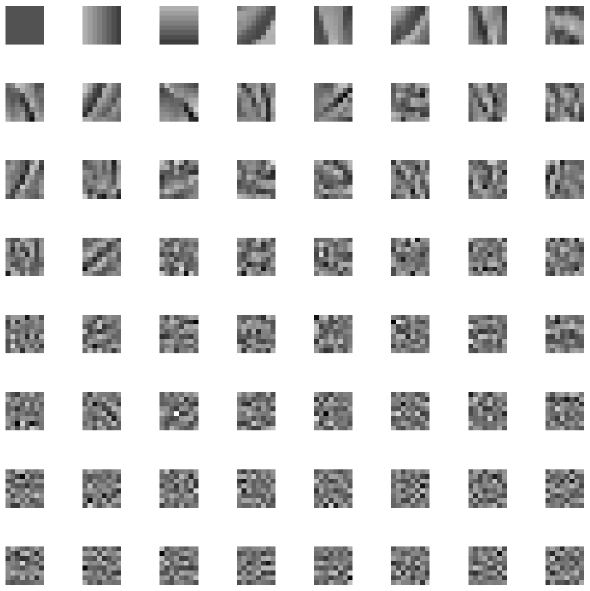


Figure 6.14b: Basis vector blocks for covariance method: orthonormalized vectors

CHAPTER 7

CONCLUSIONS

In this dissertation, many aspects of the IFS model have been investigated. The primary application area has been that of modeling image data, which is a natural type of data to generate with an IFS due to the common spectral characteristics between the attractors of IFSs and images. Many types of data were tested with a one-dimensional IFS model, and it was experimentally verified that the best results were obtained with image data.

For the one-dimensional model several new types of maps were examined. The addition of a variety of nonlinearities was evaluated, with the most significant improvements coming from the addition of nonlinear addressing, and condensation type maps. All of the different maps were tested and compared, taking into consideration the additional storage requirements of the more complicated maps.

In addition to the new map structures, the method by which the parameters were selected was investigated. It was determined that the performance of the model was improved by relaxing the boundary conditions in the interpolation function, and allowing the interpolation points to be determined with least squares along with the other map parameters.

Other IFS models were investigated, such as the hidden variable IFS, which did not yield significant improvements over the normal affine map IFS, as well as the use of other nonlinear terms in the map. The best results were obtained with the relaxed boundary conditions combined with the nonlinear addressing or the condensation type maps.

The final improvement to the one-dimensional model was the introduction of two new algorithms to determine the interpolation points on which to base the model. These algorithms were based on the location of the local extremum of the data being modeled.

For the two-dimensional model, the improvements to the one-dimensional model were extended to the two-dimensional case. Because the two-dimensional model is constructed differently due to the added complexity of the additional dimension, the efficiency of the search for the map domains takes on greater importance. Several new search-based strategies were introduced, including a proximity search, which takes advantage of the correlation between adjacent blocks in an image. A hierarchical block matching approach was also implemented, which allows a rapid search over almost the entire image, and a search based on the fractal dimension of the range and domain blocks was developed. Each of these search methods was evaluated for all of the types of maps introduced, with the best results obtained with the proximity searches.

In addition to the new search methods, several new maps were introduced for the two-dimensional model. First the nonlinear addressing and condensation type maps were extended to the two-dimensional case. Then a map with multiple scaling factors was implemented. Previous two-dimensional maps had used a fixed scaling factor of 2:1 in each coordinate direction.

As with the one-dimensional model, the boundary conditions were relaxed for the two-dimensional case and the interpolation points were determined along with all of the map parameters using least squares to minimize the map's error. This was found to improve the results of the model consistently by as much as several dB in the reconstructed signal's SNR.

In addition to the above changes to the standard affine IFS, a new kind of IFS model was introduced called the orthonormal basis IFS, (OBIFS). The OBIFS is based on interpreting the range and decimated domain blocks as vectors in a vector space and finding a set of orthogonal basis vectors for the space spanned by the range blocks from the domain blocks. The Gram–Schmidt procedure was used to generate an orthonormal basis for the space, and the range vectors were then quickly decomposed into weights for this basis. Several algorithms were introduced for determining the set of domain blocks to use in forming the basis vectors, including a searchless approach which resulted in an extremely fast coder.

Image coding has been the primary application area for IFS techniques, and a major portion of this research was directed towards using image data. While much progress has been accomplished by these efforts, there still remains work to be done in this area.

7.1 Future Work

The selection of orthonormal basis vectors still is not optimal. The covariance method succeeds in finding a set of optimal vectors in an incremental sense, however it is apparent that the contractivity of the maps is more important than the approaches introduced here allow. An algorithm which places more emphasis on the contraction factor for each domain vector should improve the results significantly. In addition, this problem would have to be solved in order to get satisfactory results with larger blocks.

The key to achieving low bit rate coding is the use of large block sizes. While the orthonormal basis approach offers the possibility of using larger blocks, the efforts in this area have been less than satisfactory. There are two main problems with this approach as it presently stands. The first is that of insuring that the maps will be contractions. It was found that the larger block sizes were more susceptible to problems in this area because of the large number of vectors involved. The second hurdle involves the computational complexity of searching through the domain vectors for an optimal set due to the increased dimensionality of the problem. An efficient search method is crucial for this approach. Some sort of classification scheme might prove effective here, as has been used in the search-based methods.

Unfortunately the coders which performed the best were still computationally intensive. There is still a great deal of work to be done in reducing the time required to code an image. In this work, the main emphasis has been in finding approaches to improve the model's performance in an SNR and compression sense, as opposed to trying to optimize the algorithm for speed. This is an area where much rapid progress is possible. The work of [25] has done much in this area, yet there is still room for significant performance improvements. The times quoted are in the neighborhood of four minutes to code an image, whereas the orthonormal basis approach with fixed vectors was just under two minutes to code a 512×512 image.

Bibliography

[1] B. B. Mandelbrot, *The Fractal Geometry of Nature.* New York, N. Y.: W. H. Freeman and Co., 1982.

[2] H.-O. Peitgen and P. H. Richter, *The Beauty of Fractals.* New York, N. Y.: Springer–Verlag, 1986.

[3] W. C. Strahle, "Adaptive Nonlinear Filter Using Fractal Geometry," *Electronics Letters*, vol. 24, pp. 1248–1249, September 1988.

[4] A. P. Pentland, "Fractal-Based Description of Natural Scenes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, pp. 661–674, November 1984.

[5] D. S. Mazel, *Fractal Modeling of Time Series Data.* PhD thesis, Georgia Institute of Technology, Atlanta, GA, 1992.

[6] A. E. Jacquin, "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations," *IEEE Trans. on Image Processing*, vol. 1, pp. 18–30, January 1992.

[7] M. Barnsley, *Fractals Everywhere.* New York, N. Y.: Academic Press, Inc., 1988.

[8] E. Corcoran, "Not Just a Pretty Face," *Scientific American*, March 1990.

[9] G. Zorpette, "Fractals: Not Just Another Pretty Picture," *IEEE Spectrum*, October 1988.

[10] M. F. Barnsley and A. D. Sloan, "A Better Way to Compress Images," *Byte*, January 1988.

[11] A. Jacquin, *A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding.* PhD thesis, Georgia Institute of Technology, Atlanta, GA, 1989.

[12] D. S. Mazel and M. H. Hayes, "Fractal Modeling of Time-Series Data," in *Proceedings of the Twenty-Third Asilomar Conference on Signals, Systems and Computers*, 1989.

[13] M. F. Barnsley and S. Demko, "Iterated Function Systems and the Global Construction of Fractals," in *Proceedings of the Royal Society of London A*, vol. 399, pp. 243–275, 1985.

[14] M. Barnsley and L. P. Hurd, *Fractal Image Compression.* Wellesley, MA: AK Peters, Ltd., 1993.

[15] J. E. Hutchinson, "Fractals and Self Similarity," *Indiana University Mathematics Journal*, vol. 30, no. 5, pp. 713–747, 1981.

[16] M. F. Barnsley, V. Ervin, D. Hardin, and J. Lancaster, "Solution of an Inverse Problem for Fractals and Other Sets," in *Proceedings of the National Academy of Sciences USA*, vol. 83, 1986.

[17] J. Elton, "An Ergodic Theorem for Iterated Maps," *Ergodic Theorems and Dynamical Systems*, no. 7, 1987.

[18] M. Barnsley, "Fractal Functions and Interpolation," *Constructive Approximation*, vol. 2, pp. 303–329, 1986.

[19] D. S. Mazel and M. H. Hayes, "Using Iterated Function Systems to Model Discrete Sequences," *IEEE Trans. on Signal Processing*, vol. 40, pp. 1724–1734, July 1992.

[20] M. F. Barnsley, J. H. Elton, and D. P. Hardin, "Recurrent Iterated Function Systems," *Constructive Approximation*, vol. 5, pp. 3–31, 1989.

[21] M. F. Barnsley, J. Elton, D. Hardin, and P. Massopust, "Hidden-Variable Fractal Interpolation Functions," *SIAM Journal of Mathematical Analysis*, vol. 20, pp. 1218–1242, September 1989.

[22] D. S. Mazel and M. H. Hayes, "Hidden-Variable Fractal Interpolation of Discrete Sequences," in *Proc. ICASSP*, 1991.

[23] M. H. Hayes, G. Vines, and D. S. Mazel, "Using Fractals to Model One-Dimensional Signals," in *Proceedings of the Thirteenth GRETSI Symposium*, vol. 1, pp. 197–200, September 1991.

[24] T. A. Ramstad, G. E. Øien, and S. Lepsøy, "An Inner Product Space Approach to Image Coding by Contractive Transformations," in *Proc. ICASSP*, vol. 4, pp. 2773–2776, 1991.

[25] E. W. Jacobs, Y. Fisher, and R. D. Boss, "Image Compression: A Study of the Iterated Transform Method," *Signal Processing*, vol. 29, pp. 251–263, December 1992.

[26] R. L. Devaney, *An Introduction to Chaotic Dynamical Systems.* New York, N. Y.: Addison Wesley, 1989.

[27] G. Vines and M. H. Hayes, "Nonlinear Address Maps in a One-Dimensional Fractal Model," *IEEE Trans. on Signal Processing*, vol. 41, pp. 1721–1724, April 1993.

[28] W. D. Withers, "Newton's Method for Fractal Approximation," *Constructive Approximation*, vol. 5, pp. 151–170, 1989.

[29] G. Vines and M. H. Hayes, "Nonlinear Interpolation in a One-Dimensional Fractal Model," in *Proceedings of the Fifth Digital Signal Processing Workshop*, pp. 8.7.1–8.7.2, September 1992.

[30] H.-O. Peitgen and D. Saupe, *The Science of Fractal Images.* New York, N. Y.: Springer–Verlag, 1988.

[31] R. Shonkwiler, J. Geronimo, and A. Deliu, "On the Inverse Fractal Problem for Two Dimensional Disjoint Polyhulled Attractors," 1992. Georgia Institute of Technology Preprint.

[32] G. Vines and M. H. Hayes, "Fast Algorithm for IFS Maps in a Fractal Model," in *Proceedings of the Visual Communications and Image Processing '92 Conference*, vol. 1818, pp. 944–949, November 1992.

[33] G. Vines and M. H. Hayes, "Using Hidden Variable Fractal Interpolation to Model One-Dimensional Signals," in *Proceedings of the European Signal Processing Conference EUSIPCO*, pp. 883–886, August 1992.

[34] N. S. Jayant and P. Noll, *Digital Coding of Waveforms.* Englewood Cliffs, NJ: Prentice-Hall Publishing Co., 1984.

[35] M. F. Barnsley and A. D. Sloan, "Methods and Apparatus for Image Compression by Iterated Function System," *U. S. Patent Number 4,941,193*, July 10 1990.

[36] A. E. Jacquin, "A Novel Fractal Block-Coding Technique for Digital Images," in *Proc. ICASSP*, vol. 4, pp. 2225–2228, 1990.

[37] Y. Fisher, "Fractal Image Encoding: SBIR Phase I Final Report," *Report to Dr. Michael Schlesinger, Office of Naval Research*, pp. 1–26, November 1990.

[38] D. M. Monro and F. Dudbridge, "Fractal Approximation of Image Blocks," in *Proc. ICASSP*, vol. 3, pp. 485–488, 1992.

[39] Y. Fisher, B. Bielefeld, A. Lawrence, and D. Greenwood, "Fractal Image Compression," *Netrologic Quarterly Report to Dr. Michael Shlesinger, Office of Naval Research*, pp. 1–6, August 1991.

[40] R. D. Boss and E. W. Jacobs, "Studies of Iterated Transform Image Compression and its Application to Color and DTED," *Naval Ocean Systems Center Techincal Report 1468*, pp. 1–40, December 1991.

[41] Y. Fisher, "Fractal Image Compression," *SIGGRAPH '92 Course Notes*, pp. 1–21, 1992.

[42] R. D. Boss and E. W. Jacobs, "Fractal-Based Image Compression," *NOSC Technical Report 1315*, pp. 1–35, September 1989.

[43] Y. Fisher, E. W. Jacobs, and R. D. Boss, "Iterated Transform Image Compression," *NOSC Technical Report 1408*, pp. 1–31, April 1991.

[44] D. M. Monro and F. Dudbridge, "Fractal Block Coding of Image Blocks," *Electronics Letters*, vol. 28, pp. 1053–1055, May 1992.

[45] G. E. Øien, S. Lepsøy, and T. A. Ramstad, "Reducing the Complexity of a Fractal-Based Image Coder," in *Signal Processing VI, Proc. EUSIPCO*, vol. 3, pp. 1353–1356, 1992.

[46] K.-M. Cheung and M. Shahshahani, "A Comparison of the Fractal and JPEG Algorithms," *TDA Progress Report 42-107*, pp. 21–26, November 15 1991.

[47] C.-C. Chen, J. S. Daponte, and M. D. Fox, "Fractal Feature Analysis and Classification in Medical Imaging," *IEEE Trans. on Medical Imaging*, vol. 8, no. 2, pp. 133–142, 1989.

[48] J. P. Rigaut, "Automated Image Segmentation by Mathematical Morphology and Fractal Geometry," *Journal of Microscopy*, vol. 150, pp. 21–30, April 1988.

[49] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles.* Reading, MA: Addison-Wesley Publishing Co., 1974.

[50] S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. on Information Theory*, vol. 28, pp. 129–137, March 1982.

Vita

Greg Vines was born in Memphis, Tennessee, on June 13, 1960. He received his B.S. and M.S. degrees in Electrical Engineering from the University of Virginia and the Georgia Institute of Technology in 1982 and 1990, respectively. While at the Georgia Institute of Technology, he was a Graduate Research Assistant from 1988 until 1993. His research interests include signal modeling, image processing, and image/video coding. Outside of school, when he scrapes enough money together, he can be found at 3000' AGL punching holes in the sky doing wingovers and other stomach-churning maneuvers.

Signal Modeling With Iterated Function Systems

Greg Vines

118 pages
Directed by Dr. Monson H. Hayes III

A new method of modeling signals has been proposed through the use of Iterated Function Systems (IFSs). Iterated Function Systems are capable of producing complicated functions, many of which closely resemble images and other waveforms that can be found naturally. The task of modeling a given data sequence with an IFS is an ill-posed problem and methods proposed to date have relied on iterative algorithms and using simple affine maps to determine an appropriate IFS. This research focuses initially on modeling one-dimensional signals. Variations of the basic affine map are explored, including the use of nonlinearities on the mapped data, as well as on the addressing of the data in the map. In addition, a condensation type map has been developed, as well as a non-iterative algorithm which is based on characteristics of the data to be modeled. The method used to determine the model parameters is improved through relaxing the boundary conditions which define the interpolation points for the model. All of the enhancements are evaluated with a set of test files and results are given quantifying the improvements with the new methods.

The second portion of this research concentrates on two-dimensional models, specifically the application of image coding. The nonlinear addressing and condensation type maps were extended to the two-dimensional case, as well as the relaxation of the boundary conditions. In addition, three new search techniques were developed based on the local fractal dimension of pixels in the image, a hierarchical block match-

ing algorithm and the correlation between adjacent blocks in an image. An entirely new orthonormal basis approach was introduced which allows multiple domains to be used in each map. Several versions of this method are described, one of which is faster than all previous approaches, at a slight cost in SNR. Slower methods are also introduced, which offer higher reconstructed image quality.