FRACTAL-BASED IMAGE AND VIDEO CODING USING MATCHING PURSUIT

BY

MOHAMMAD GHARAVI-ALKHANSARI

B.S., University of Tehran, 1987
M.S., Iowa State University, 1990

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1997

Urbana, Illinois

FRACTAL-BASED IMAGE AND VIDEO CODING USING MATCHING PURSUIT

Mohammad Gharavi-Alkhansari, Ph.D.
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, 1997
Thomas S. Huang, Advisor

Fractal-based image coding is based on the theory of iterated function systems (IFS) which was originally developed for creation of a deterministically self-similar fractal in the 1980s. Using this theory, an image may be represented by a dynamical system, known as a fractal transform, whose attractor is close to the image being coded. For natural images, the parameters of such a dynamical system can usually be coded very compactly, which makes this method suitable for image compression.

In these methods, the problem of coding an image is that of designing such a dynamical system. To do this, an image is first partitioned into segments and each is approximated, in part, with another larger segment in the same image. These coding methods take advantage of the redundancies in the image at different scales. The decoding is done by finding the attractor of the dynamical system and is typically an iterative process.

In this thesis, we will first review the theoretical foundations and implementation issues of fractal-based image coding methods. The concepts of fractals, iterated function systems, and local iterated function systems are discussed and different implementations of compression of both still images and image sequences are reviewed.

The fractal/attractor coding systems will also be analyzed in light of linear systems theory and graph theory. We will show that the attractor decoder may be modeled as a linear system whose stability is a necessary and sufficient condition for its convergence. We will also address the relationship between the iterative nature of the fractal image decoders and the noncausality of their encoders and propose the use of a causal fractal encoder that results in a fast noniterative decoder.

A new extended fractal transform is then proposed. The new transform is designed to overcome some of the major shortcomings of most other types of fractal transforms designed for image coding. Using this method, the process of finding the fractal transform is based on approximating every block in the image by a linear expansion on an over-completed set of library vectors. The library is made up of a fixed and an adaptive part. The latter is made up of blocks taken from the original and the decimated image, and makes the algorithm able to exploit both the inter- and intra-scale redundancies in the image.

For a given image, the problem of finding the optimal transform of the above form is mathematically intractable. In fact, this problem is a special case of a more general optimization problem that has recently been of interest in signal analysis, coding, control theory and statistics. A general and flexible method called matching pursuit is developed that provides a solution to this problem. This new transform is then used for coding and resolution enhancement of still images, and coding of video sequences. Using this method, we will also evaluate the effectiveness of using inter-scale prediction inherent in the fractal coding.

In the context of coding, the new method provides a flexible and powerful generalized method by unifying the block transform coding, vector quantization, and most of the earlier fractal compression methods. In the context of video coding, the block prediction methods such as delta pulse code modulation (DPCM), adaptive block prediction methods such as block motion compensation methods, and hybrid coding methods such as motion compensation combined with transform coding of residual errors are also special cases of this algorithm.

This dissertation will also present a comparative study on the resolution enhancement of images using the inter-scale prediction inherent in the fractal transforms.

To my family

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my mentor and advisor, Professor Thomas Shi-Tao Huang, for his invaluable guidance, continuing encouragement, and support throughout this research. I would also like to thank Professor Kenneth Jenkins, Professor Kannan Ramchandran, and Professor Kenneth Zeger for their helpful comments and advice. My thanks also extend to my fellow members of the Image Formation and Processing Laboratory and the Robotics and Vision Laboratory at the Beckman Institute for Advanced Science and Technology at the University of Illinois for their generous help and stimulating discussions. I would also thank Robert DeNardo and Yoichi Tenda for their help in preparing the experimental results of Chapter 8, and for many discussions and helpful suggestions.

I wish to express my gratitude to my wife, Marjan, for her constant encouragement, patience, and support during the course of this work, and for her generous help and many valuable suggestions in preparing this dissertation. I also wish to thank my parents for their everlasting support and guidance throughout the course of my education.

This research was supported in part by Joint Services Electronics Program Grant N00014-90-5-1270 and ONR N00014-96-1-0129; in part, by Army Research Laboratory under Cooperative Agreement No. DAAL01-96-2-0003; and, in part, by a grant from Mitsubishi Electric.

PREFACE

Work on this research began in the summer of 1991 when I came to University of Illinois. At that time Professor Huang suggested that I look into fractal image coding. My first introduction to fractals was through the classic book entitled *The Fractal Geometry of Nature* by B. B. Mandelbrot [1]. Since then, the theory of fractals and its vast scientific and philosophical consequences have fascinated me.

In the field of image coding at that time, there were claims by Barnsley that compression ratios of 10000:1 were possible using a fractal compression method [2]. Later, Professor Kiyo Aizawa, who was visiting the University of Illinois at the time, gave me a copy of Jacquin's paper in ICASSP'90 [3] and I obtained a copy of Jacquin's thesis [4] soon after.

After studying the principles of iterated function systems and Jacquin's algorithm, I implemented a version of this algorithm and applied different variations. It soon became clear that neither the compression of this method nor simple variations could provide compressions extremely higher than those reported by Jacquin, and the extremely high compressions claimed by Barnsley were not realistic for natural images except for super-high resolutions. To ensure no omission of important facts, a comprehensive literature search was done on all of the papers published on fractal image compression and closely related subjects, specifically, papers by Barnsley and his group working at Georgia Institute of Technology, and later at Iterated Systems, Inc. Reviewing these literature findings confirmed my conclusions which was later summarized in a comprehensive study published in the form of a book Chapter [5].

Since then, I have continued the literature search which has resulted in more than 600 references at the time of this writing. The interest in the subject of fractal image compression seems to have grown during recent years. The majority, and the most promising, of algorithms proposed for fractal image compression are based on the algorithm by Jacquin.

There seemed to be several areas in the Jacquin algorithm for which major improvements were possible. Two of the most promising ones seemed to be a) using freely shaped regions instead of square blocks, b) using multiple domain blocks to encode each range block, and c) applying the transformation in the Wavelet domain. After some preliminary tests on these ideas, I focused my attention on the second area. I later realized that it could be further improved by including multiple fixed blocks. This idea made the basis for papers in ICASSP'93 and PCS'93 [6, 7], which approximated a range block with a near orthogonal set of basis blocks, some of which were transformed blocks selected from the output of an orthogonalization process applied on image blocks. However, this approach had a drawback that despite its novelty, the search in it was always limited to the dimension of the vectors being coded (which is the same as the number of pixels in the range blocks). We later realized that this algorithm could be further improved in flexibility and generality by using a series of extensions. However, to include these extensions, it required a basic type of mathematical optimization that had not been addressed in the signal processing literature, and seemed to have a large application in coding.

In June 1992, I developed a method to resolve this problem and discussed it with Professor Huang. This approach, which is discussed in Chapter 4, provides a powerful method for the solution to the very basic mathematical approximation problem of finding the optimal linear expansion of a vector over an over-complete set of library vectors. This solution has applications in the general area of coding and is used in the algorithm described in papers in ICIP'94 [8] and PCS'94 [9]. Its application was further developed for compression of still images in VCIP'96 [10] and on video in ICIP'96 [11].

However, in December of 1994, during discussions with my colleague and friend Aria Nosratinia and later with Professor Kannan Ramchandran, I was pointed to a recent work by Mallat et al., proposing a method named 'matching pursuit.' In fact, Mallat and Zhang in December of 1993 [12] and Davis et al. in July of 1994 [13] had published results on the application of such an algorithm in the context of pattern extraction in image analysis. However, our papers in PCS'94 and ICIP'94 were the first to apply this technique to image and video compression almost simultaneously with Neff and Zakhor

[14] and Vetterli and Kalker [15], who also published results on the application of such an approach to video coding.

The combination of the generality of the coding approach that we proposed and the powerful mathematical solution to the above-mentioned approximation problem made the proposed coding method capable of unifying the methods of block-based transform coding, standard and gain-shape VQ, and most of the earlier fractal compression methods making them only special cases of a general case. Due to the flexibility of the proposed method in the context of video coding, the method seamlessly covers the cases of block prediction methods such as DPCM, the adaptive block prediction methods such as block motion compensation, and hybrid coding methods such as motion compensation combined with transform coding of residual errors.

In addition to being among the first to develop the theory of orthogonal, nonorthogonal and rate-distortion optimized matching pursuit, we also, for the first time, proposed using multiple domain blocks in fractal coding and explicitly addressed the relationship between noncausality of the fractal encoder and the iterativeness of its decoder (simultaneously with [16]). In this light, we developed a very fast method for noniterative fractal decoding, successfully exploited intra-scale self-similarities of natural images in fractal coders, developed a lossless fractal coder, and studied fractal transform as a tool for spatial resolution enhancement of images. We also studied the fractal transform in the light of the linear system theory, addressed the issue of stability of the decoder in contrast to the contractivity of the fractal transform, and provided more general sufficient conditions for the stability of this system using a graph theory view of the fractal transform. The use of intra-scale self similarities was also viewed as an extension of the Lempel-Ziv algorithm from 1-D to 2-D.

Realization of the above ideas in a general framework was a difficult task and they were implemented using around twenty thousand lines of code, written by this author, which have been evolving constantly during the last few years.

The organization of this dissertation is as follows. Chapter 1 gives a review of the concept of fractals and its applications, especially in image compression. The mathematical principles of iterated function systems (IFS) and local IFS are studied in Chapter 2. Local IFS makes the basis for most fractal-based image compression methods. Different implementations based on this theory for compression of both still images and image sequences are also reviewed. In Chapter 3, a system/graph theoretical analysis of attractor/fractal coders is provided, and the close links between different concepts in attractor coding, systems theory and graph theory are established. In Chapter 4, the general mathematical optimization problem mentioned above and the solution provided by matching pursuit are studied. In Chapter 5, the new fractal transform is proposed. Chapter 6 addresses the application of this transform to coding of still images. Coding of video sequences using this transform is studied in Chapter 7. In Chapter 8, application of this fractal transform for enhancement of the resolution of images is investigated. Finally, in Chapter 9, conclusions are drawn.

Although the focus of this work is on images and video sequences, most of the theory and algorithms introduced are general and applicable to any kind of single or multidimensional digital data. However, the parameters of most of the algorithms are tuned to work well on digital photographic images of natural scenes.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Fractal-based image coding or *fractal image coding* is a new method of image compression. In this method, similarities between different scales of the same image are used for compression. The method is rooted in the work of Mandelbrot, who introduced the concept of fractals and the fractal dimension.

1.1 Fractals and Self-Similarity at Different Scales

In late 1970s and early 1980s, Mandelbrot showed that many natural and man-made phenomena have the very fundamental characteristic of invariance under change of scale [1, 17, 18]. Mandelbrot coined the name *fractal* for the geometry of these phenomena.

Intuitively, fractals are sets that reveal details at every scale. These sets are in contrast to regular sets like lines, curves and planes that are typically studied in Eucledian geometry and become smooth when sufficiently magnified. To define such sets mathematically, the concept of dimension is used. There are several types of dimensions defined for sets in mathematics, and for nonfractal sets they typically coincide. However, for fractal sets, they give different values.

The mathematical definition of fractals suggested by Mandelbrot is that they are sets for which the Hausdorff-Besicovitch dimension D is strictly larger than their topological dimension D_T [1]. However, computing the Hausdorff-Besicovitch dimension is often difficult, and in many cases the *fractal dimension* [19] is used instead. Fractal dimension is defined as

$$D = \lim_{d \to 0} \frac{\ln N(d)}{\ln(\frac{1}{d})} \tag{1.1}$$

where $N(d)$ is the minimum number of balls of diameter d which are needed to cover the set.[1]

This definition implies that if d is small enough, we can write the approximate power law

$$N(d) = K(1/d)^D \tag{1.2}$$

where K is a constant. This means that as d decreases, $N(d)$ grows with the Dth power of $1/d$, no matter how small d is. D can be considered as a measure of the roughness of a set, where rougher sets have larger Ds [22]. A classical example of a natural fractal set is the coastline of an island, and an example of an artificial fractal set is the Koch curve [1]. In practice, a natural set is considered fractal if its D is stable over a wide range of scales.

Figure 1.1 shows different steps in the construction of the Koch curve. In this construction, we begin with a line segment of length 1 (Figure 1.1(a)). Then, divide the line into three equal parts and replace the middle part with two line segments of length 1/3 as shown in Figure 1.1(b). If we apply the operation that generated Figure 1.1(b) from Figure 1.1(a) on every line segment in Figure 1.1(b), we will get Figure 1.1(c). Repeating this process once more results in Figure 1.1(d), and continuing to apply this process infinitely, results in the set shown in Figure 1.1(e), which is known as the Koch curve.

It is easy to show that the set shown in Figure 1.1(e) has a fractal dimension of

$$D = \frac{\ln 4^n}{\ln 3^n} = \frac{\ln 4}{\ln 3} \approx 1.26 > 1 = D_T.$$

The power law in Equation (1.2) states that dividing d by any factor a, always increases $N(d)$ by a factor a^D, for any small value of d. In the case of integer Ds, this result seems trivial, but when D is not an integer this means that, for example, magnifying any part of a fractal curve does not result in a curve that is smoother than the original curve.

This power law means that a fractal set has the same roughness, independent of scale. Therefore, one can say that a fractal set is self-similar at different scales in the above

[1]For more detailed information on the definition of fractals and different dimensions, see [20] or [21].

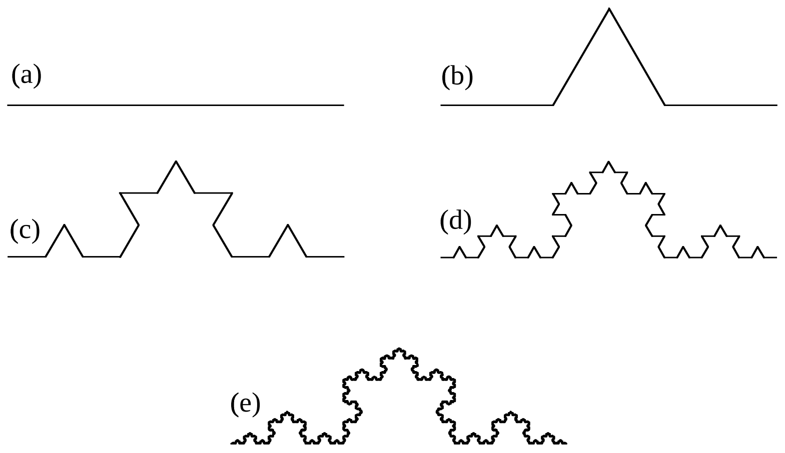


Figure 1.1 Different stages of construction of the Koch curve.

general sense. Of course, many simple, nonfractal geometrical sets, such as lines and planes are also self-similar in this sense, but are not fractals because they are not 'rough' sets and always become smooth when sufficiently magnified. Fractals always reveal more details under magnification which do not diminish by future magnification.

Some fractals possess self-similarities in a more restricted sense. The self-similarity of this class of fractals can be either deterministic or statistical. The deterministic self-similarity is when the shape of the set is similar to itself in a deterministic way as the scale changes. Particular kinds of fractals of this class are those with exact self-similarity that are not altered at all under change of scale, e.g., the Koch curve.

The statistical self-similarity is when the fractal set retains all of its statistical parameters at different scales. For these fractals, a deterministic relationship does not necessarily exist between different scales of the set.

1.2 Applications of Fractals

Fractal geometry in nature is more the rule than the exception. Since the introduction of the concept of fractals by Mandelbrot, the concept has been used in many different branches of science including mathematics, physics, chemistry, geophysics, botany, biology, computer graphics, computer vision, and image processing.

Fractals with statistical self-similarity have been of great interest in the area of computer graphics, where the concept was used to generate complex and strikingly natural-looking graphics of natural scenes using simple rules [1, 23].

In the area of computer vision, the fractal dimension has been used for image modeling, segmentation and shape extraction for natural scenes by Pentland [22, 24] and others [25]–[33]. The fractal dimension of natural objects can be different from each other or from those of man-made objects. On the other hand, under certain conditions, 2-D images taken from some 3-D fractal geometries are also fractals. Pentland [22] used the fractal dimension as a parameter for segmenting images. This method can be used to discriminate between different natural objects in a scene or between man-made and natural objects [30, 31].

1.3 Fractal-Based Image Compression

Fractal geometry has been used for image compression in a few, basically different ways.

Fractal curves, especially the Peano curve, were used for scanning images instead of the standard raster scanning [34]–[37].

A 'yardstick' method was used for image compression [38]–[40] and for shape classification [41].

Fractal dimension has been used as a tool in different aspects of image compression algorithms.

- Fractal dimension was used in a fractal image coder for adjusting error thresholding [42]. Also, in [43, 44], fractal dimension was used for image segmentation. After segmentation, fractal dimension was used as a measure of the complexity of the segment to determine how the segments should be coded.

- In the context of image coding, fractal dimension was also used for selecting the optimal scale parameter in an edge detector [45].

Wavelet decomposition has also been used to exploit self-similarities of images at different scales. In this approach, a wavelet decomposition is applied to an image and the similarity of same-size blocks in different subbands is used to reduce the size of the code needed for representing them [46]–[49]. Pentland and Horowitz [46] report typical compression ratios of 38:1 with 33 dB PSNR for 256×256 pixel images and Rinaldo and Calvagno [47] report compression ratios of about 54:1 (PSNR of 31.4 dB) to 20:1 (PSNR of 35.5 dB) for 512×512 test image Lena.

However, the method that has attracted the most attention, and will be discussed in more detail in this chapter, is based on the work by Barnsley et al. [2], [19], [50]–[68], summarized in [67].[2] His work was based on that of Hutchinson [70], who set up a theory for deterministically self-similar sets and studied transformations that can generate these kinds of sets. These transformations, which Barnsley later named *Iterated Function Systems (IFS)*, were originally used for generating fractals, but because many nonfractal sets can also be deterministically self-similar, these sets can also be generated by IFS. Iterated function systems will be discussed in detail in Section 2.1.

Barnsley's early work was based on the following assumptions:

- The images of many natural objects can be approximated by members of a class of deterministically self-similar sets.
- These sets can be generated by IFS transformations which have a relatively small number of parameters.

Barnsley observed that even very simple IFS transformations with very short codes can generate complex sets with infinite details that resemble natural objects. The IFS transformations describe the relationship between the whole image and its parts, and exploit the similarities that exist between an image and its smaller parts.

Given an IFS, generating the image corresponding to it is quite straightforward and easy. However, the *inverse problem* of finding the IFS, which can generate (or closely

[2]The methods based on Barnsley's work have a strong relationship with some of the wavelet methods mentioned earlier. For studies on this, see [49, 69].

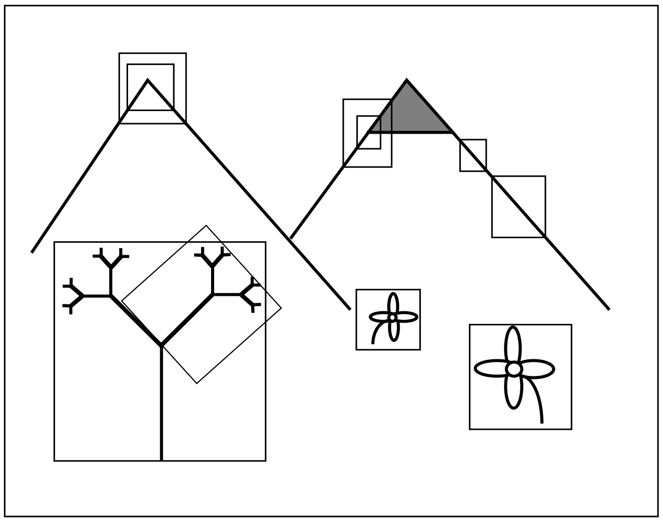


Figure 1.2 The essence of fractal coding methods is to try to approximate each segment of the image by applying some (contractive) transformation on some larger segment(s) in the image.

approximate) a given image, has yet to be solved. In other words, the problem was that of how the similarities between the whole image and its parts could be found automatically. Another problem was finding what could be done for images where the smaller parts do not resemble the whole image. To solve the second problem, in 1988, Barnsley generalized the theory of IFS to the theory of *Local Iterated Function Systems* which exploited similarities between parts of the image which were of different sizes. Using this theory, image parts were not required to resemble the whole image; they only needed to be similar to some other bigger parts in the image, as shown in Figure 1.2. But there was still the first problem of how these similarities could be found automatically. In the early implementations of this theory, these similarities were found by human interaction and, hence, the images were encoded by interactive computer programs. This resulted in codes for images which were extremely compact in size, but their decoded images had very low quality [19]. This was the result until the work of Arnaud E. Jacquin (a student of Barnsley) who automated this method for the first time [3, 4, 71, 72]. The code generated by Jacquin's program for an image was not as compact as before, but the compression ratio and the quality of the decoded images looked promising. The work by Jacquin provided a platform for others to continue this line of research. Since then, several extensions and generalizations of this method have been found, and many of its properties are better understood, which has resulted in more efficient algorithms. Some of these methods will be discussed in Section 2.2.

In 1987, Barnsley and Sloan founded Iterated Systems, Inc., for the development of products based on the fractal theory and patented some of the basic algorithms in fractal coding [64, 65]. This company has made various hardware and software products for image and video compression/decompression, especially for personal computers. Although many articles have been published on the basics of Barnsley's theory, many of the details of the algorithms used in these products have not been revealed.

1.4 Fractal Techniques in Second Generation Image Coding

Second generation image coding methods 1) take special advantage of the properties of the human visual system and 2) many of them are segmentation-based. In this section, we will briefly look at how fractals are related into these two features of second generation coding methods.

- *Fractals and the Human Visual System*

  Many researchers have studied the relationship between fractals and the human visual system [22, 24, 73].

  - To the human eye, many fractal curves and surfaces, look very similar to natural curves and surfaces, and for this reason they have been extensively used in computer graphics. In model-based coding, this similarity has the potential to be used for coding of natural images by modeling the underlying processes that generate parts of these images.
  - Experiments have shown that the fractal dimension of a curve or set is closely related to human perception of its roughness [22]. Although fractal dimension

alone is not enough for generating a visually good approximation of a set [25, 74], it may be used as one of the parameters for its representation.

  – It is a known fact that the human visual system's sensitivity to details in any part of an image is dependent on the amount of activity in the background surrounding that part. Fractal dimension of image regions has been used as an objective measure of this activity [42].

- *Segmentation Using Fractal Dimension*

  Many researchers have used fractal dimension for image segmentation [22, 27, 43, 75, 76, 77, 78]. Fractal dimension is usually computed locally [79, 80] and is used as the texture feature for segmentation.

- *Edge Detection Using Fractal Dimension*

  In the context of image coding, fractal dimension was also used for selecting the optimal scale parameter in a multiscale edge detector [45]. In this method, edge points were detected by wavelet transform and the dilation parameter was controlled by the fractal dimension.

- *Fractal Coding of Contours*

  One of the first applications of the theory of iterated function systems proposed by Barnsley and Jacquin, was in contour coding [4, 55]. A similar method was later used for this purpose by Jacobs et al. [81].

- *Jacquin's Method as a Second Generation Method*

  Fractal coding methods based on Jacquin's method use redundancies in an image at different scales, i.e., they use the fact that different parts of the image at different scales are similar. Due to computational complexity limitations, most fractal coding methods find similarities between image *blocks* after applying limited transformations, even though the most natural choice is finding similarities between *objects* or *segments* with more free deformations. The use of blocks instead of segments is more a matter of speed than anything else, especially because fractal coders are usually computationally intensive. In the basic theory, the shape of the domain segments is not restricted in any sense. Simple block splitting methods have been used by many researchers, including Jacquin, for adjusting the size of the blocks to the feature sizes of the image.

Thomas and Deravi [82, 83] devised a method for merging of blocks using a region growing procedure based on fractal coding. This method results in range *regions* with rather free shapes that are adapted to the content of the image. Using this method, a region in the image is approximated with another larger region in the same image, but with a similar shape.

Franich et al. [84] proposed a method for merging the quadtree block splitting method for shape description with the quadtree block splitting method used for fractal coding and used it in an object-based video coding system.

From this viewpoint, the fractal coding techniques originated by the work of Jacquin can be well adapted to both first and second generation image coding techniques, although during recent years, most of the advancements of fractal coding techniques have been in the direction of combining them with waveform-based coding methods.

CHAPTER 2

FRACTAL IMAGE CODING

2.1 Basic Theory

In essence, fractal image coders take advantage of across-scale redundancies in natural images. In most fractal coders, this is done by approximating each segment of the image by applying a contractive transformation on some larger segment(s) in the image. These approximations define a dynamical system, known as a fractal transform, whose attractor is close to the image being coded. For natural images, the parameters of such a dynamical system can usually be coded very compactly, which makes this method suitable for image compression. One can then reconstruct the image (with some error) by finding the attractor of this dynamical system [19, 67]. In these methods, most of the information in the image is basically encoded by coding relations among different segments of different sizes of the image. The mathematical framework of this theory is presented in the following sections.

2.1.1 Iterated function systems

We begin with a complete metric space (X, d), where $d(.,.)$ denotes the metric.[1] Now, consider a transformation $w: X \mapsto X$, for which there is a constant s such that for all $x, y \in X$,

$$d(w(x), w(y)) \leq s\ d(x, y).$$

[1]For more details on the basic theory brought in this section, see [19], [66], or [67].

If $0 \leq s < 1$, then w is said to be *contractive* (or a *contraction*) with *contractivity factor* s. If w is contractive, then according to the *Contraction Mapping Theorem*,

(1) w possesses a unique fixed point $x^* \in X$, i.e., $w(x^*) = x^*$.

(2) For any $x \in X$, $\lim_{n\to\infty} w^{(n)}(x) = x^*$.

The transformation w defined on X also induces a transformation on subsets of X. This can be done by defining

$$w(B) = \{w(x), \forall x \in B\} \quad \forall B \subseteq X.$$

Let $(H(X), h)$ denote the metric space whose points are nonempty, compact subsets of X, and h is the *Hausdorff Distance* [67]. An *Iterated Function System (IFS)* consists of a complete metric space (X, d) and a number of contractive mappings w_i defined on X, i.e., $\{X; w_i, i = 1, \ldots, N\}$. The *fractal transformation* associated with an IFS is the transformation $W : H(X) \mapsto H(X)$ as defined by

$$W(B) = \bigcup_{i=1}^{N} w_i(B) \tag{2.1}$$

for all $B \in H(X)$. If the mappings w_i are contractive with contractivity factors s_i, $i = 1, 2, \ldots, N$, then W is also contractive with contractivity factor $s = \max_i s_i$, and W has a unique fixed point $A \in H(X)$ for which

$$A = W(A) = \bigcup_{i=1}^{N} w_i(A),$$

and for all $B \in H(X)$ we have $\lim_{n\to\infty} W^{(n)}(B) = A$. A is called the *attractor* of IFS. The w_i's are usually chosen to be affine transformations. For the two-dimensional case, this becomes

$$w_i\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix} \tag{2.2}$$

defined on points in R^2. For the three-dimensional case (grayscale images), this becomes

$$w_i\left(\begin{bmatrix} x \\ y \\ I(x,y) \end{bmatrix}\right) = \begin{bmatrix} a_{1,1,i} & a_{1,2,i} & a_{1,3,i} \\ a_{2,1,i} & a_{2,2,i} & a_{2,3,i} \\ a_{3,1,i} & a_{3,2,i} & a_{3,3,i} \end{bmatrix} \begin{bmatrix} x \\ y \\ I(x,y) \end{bmatrix} + \begin{bmatrix} b_{1,i} \\ b_{2,i} \\ b_{3,i} \end{bmatrix}, \tag{2.3}$$

where $I(x, y)$ denotes the grayscale value at location (x, y). For an image, the *fractal code* is made up of the parameters of the fractal transformation W, which consists of the number N and the parameters of w_i's. The mappings w_i defined by (2.2) and (2.3) are contractions under suitable constraints on the parameters and, therefore, the resulting Ws are also contractions.

As an example of an IFS and its attractor in R^2, let us consider an IFS of the form

$$\{R^2; w_1, w_2, w_3\},$$

where

$$w_1\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \tag{2.4}$$

$$w_2\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \tag{2.5}$$

$$w_3\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix}. \tag{2.6}$$

The attractor of this IFS can be found by iteratively applying the induced W on any nonempty, compact subset of X. Figure 2.1 shows how a sequence of sets generated by the iterative application of W on an arbitrary initial set B converges to the attractor of W and how the attractor is dependent only on W and not on the initial set.

In general, A is completely described by W and is independent of B. Therefore, W gives a complete representation of A, and the set of parameters that represent W can be considered as a code for A. In the above example, it can be seen that A has a visually complex shape, but W has a very simple mathematical form which can be specified by three affine transformations. Considering the plot of A as a black and white image, the parameters of W make the code for this image.

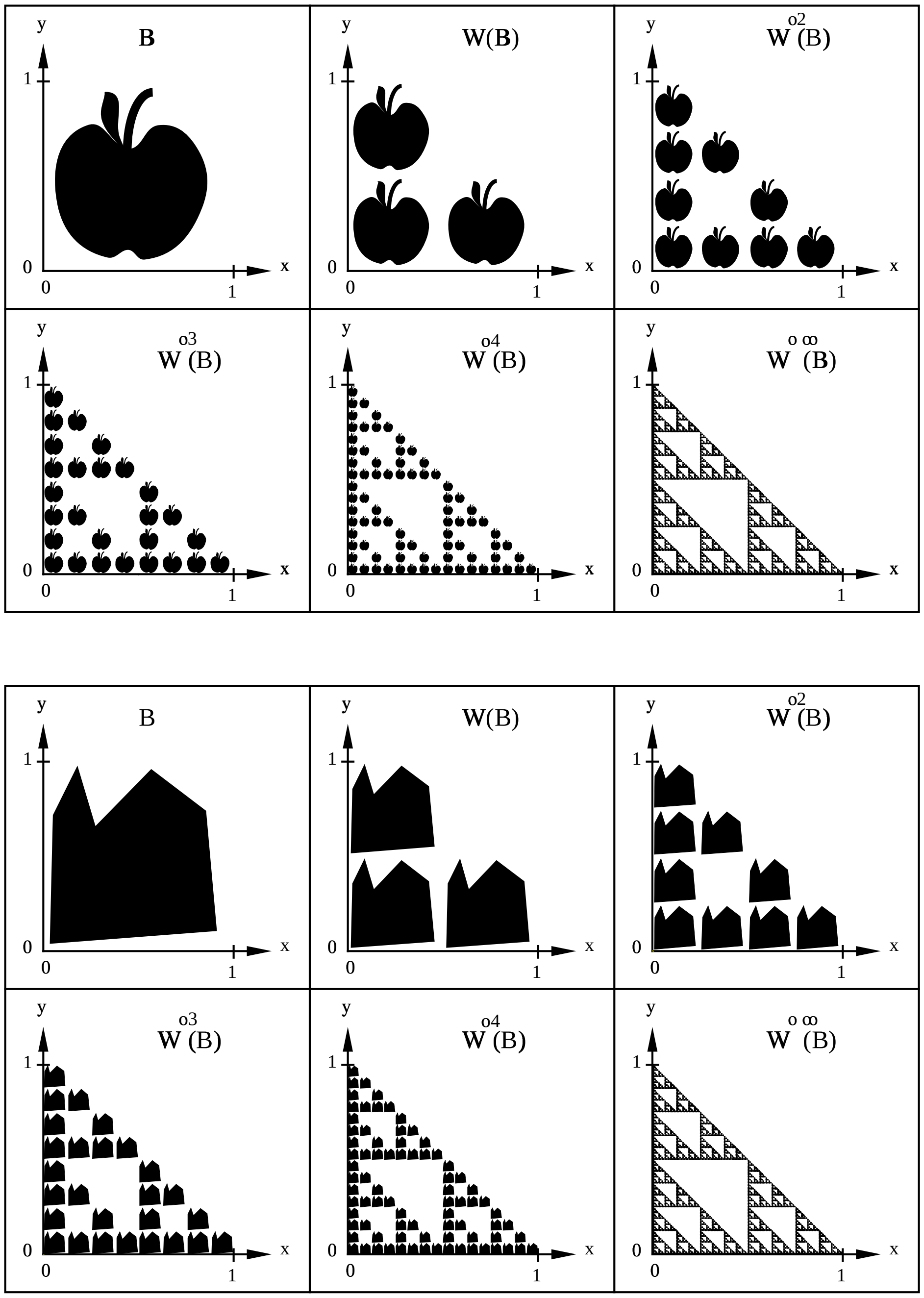


Figure 2.1 Sequences of sets generated by iterative application of the IFS transformation W defined by (2.1), (2.4), (2.5), and (2.6) on two different arbitrary initial sets (B) converging to the attractor of IFS.

2.1.2 The Collage Theorem

Although the theory of generating the attractor of an IFS is well developed, the inverse problem of finding the IFS code for approximating an arbitrary given set, like many other inverse problems in mathematics, has proven to be a rather difficult problem.

Several studies have been made to find the exact mathematical solution to this inverse problem using tools such as the Fourier transform [85], wavelet transform [86, 87], moment method [88]–[101], chaotic optimization [95, 102], genetic algorithms [103], combination of the wavelet transform and the moment method [104]–[106], fuzzy sets [107] and other methods [108, 109]. However, this problem, in the general sense, has not yet been solved.

As discussed before, given W, the decoding process is based on the Contraction Mapping Theorem. The transformation W is applied iteratively on an arbitrary initial image until the transformed image does not change significantly. As W is contractive, the convergence of this sequence of images is guaranteed by the Contraction Mapping Theorem.

However, for a given set C, the encoding problem of finding a contractive transformation W such that its attractor A is close to C is a rather difficult problem. The *Collage Theorem* [19, 53] provides a guideline for solving this problem. It says that for a set C and a contraction W with attractor A,

$$h(C, A) \leq \frac{h(C, W(C))}{1-s}. \tag{2.7}$$

This means that in order for C and A to be close, it is sufficient that C and $W(C)$ be close, i.e., W may be found in such a way that $W(C)$ is as close to C as possible. $W(C)$ is sometimes called the *collage* of C.

In terms of w_i, we have

$$\left.\begin{array}{l} W(C) \approx C \\ W(C) = \bigcup\limits_{i=1}^{N} w_i(C) \end{array}\right\} \Rightarrow \bigcup_{i=1}^{N} w_j(C) \approx C.$$

This can be done by partitioning C into parts C_i,

$$C = \bigcup_{i=1}^{N} C_i$$

such that each C_i can be closely approximated by applying a contractive affine transformation w_i on the whole C, i.e.,

$$C_i = w_i(C).$$

If we denote $h(C, W(C))$ by ε_E and call it the *encoding error* or *collage error*, and denote $h(C, A)$ by ε_D and call it the *decoding error*, then according to (2.7),

$$\varepsilon_D \leq \frac{1}{1-s}\varepsilon_E \tag{2.8}$$

which gives an upper bound for ε_D in terms of ε_E.

2.1.3 Local iterated function systems

For most natural images, it is not possible to closely approximate all parts of the image by a small number of transformations applied on the *whole* image. To solve this problem, the theory of Iterated Function Systems was extended to *Local Iterated Function Systems* [66] and its associated fractal transform. In contrast to an Iterated Function System, which approximates each part of the image by a transformed version of the whole set, in the Local Iterated Function System, each part of the image is approximated by applying a contractive affine transformation on another *part* of the image. In this case, the image C is partitioned into range segments C_i, where $C = \bigcup_{i=1}^{n} C_i$. Then, each range segment C_i is approximated by a transformed version of a bigger domain segment D_i, i.e., $C_i \approx w_i(D_i) \Rightarrow C \approx W(C) = \bigcup_{i=1}^{N} w_i(D_i)$ as shown in Figure 2.2. The decoding process for Local IFS is very similar to that of IFS.

2.1.4 Resolution independence

When the above theory is used for image compression, it is implemented in a discrete setting. However, the fractal code generated by encoding a digital image describes relationships, in the form of affine functions, between various segments of the image and is independent of the resolution of the original image. In other words, the fractal code is a *resolution independent* representation of the image and theoretically represents a

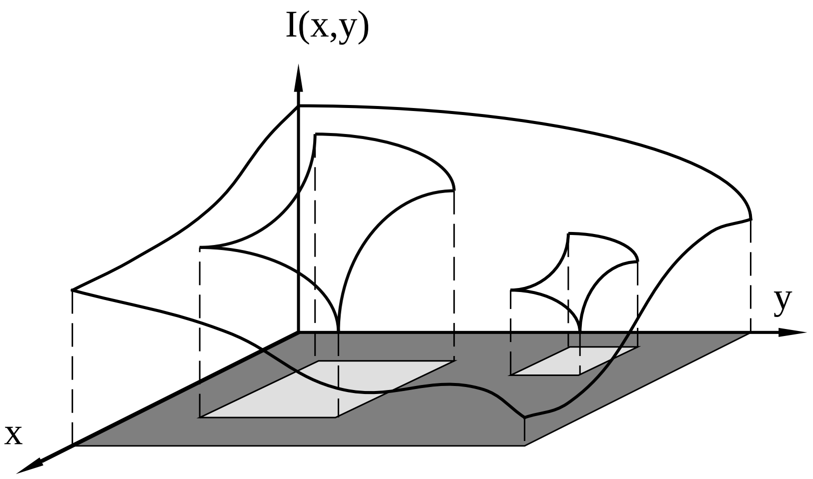


Figure 2.2 Approximation of a range block by a transformed domain block in a local iterated function system.

continuous image approximating the original image. A decoder may decode this code to generate a digital image at any resolution. The resolution of the decoded image may as well be higher than the resolution of the original image. This increase of resolution is sometimes referred to as *fractal zoom*.

The higher resolution obtained is not created by a simplistic technique such as repeating the pixels of the image, but more detail is actually generated in the decoded images. In fact, the additional higher resolution information is generated using information from the image at a lower resolution. When an image is reconstructed at the same resolution as the original encoded image, in the decoding process domain blocks of the image are shrunk (lowpass filtering followed by subsampling), which eliminates some of the details of the domain blocks. However, if the image is reconstructed at a higher resolution, in the shrinking of the domain block, the details of the domain block are only shrunk to generate the extra resolution in the range block. In fact, details of the domain blocks are used for missing details of the range block. The details in the domain block are also generated to some extent from details of other domain blocks used for encoding each part of it. In other words, it is implicitly assumed that if the range block is similar to its corresponding domain block, then the details of the range block, which are *beyond* the resolution of the originally encoded image, are also similar to the details of the domain block, which are *within* the resolution of the encoded image. This assumption is a typical property of self-similarity of fractal sets at different scales and the resolution independence is a property of the code generated by fractal-based methods. This issue is studies further in Chapter 8.

2.2 Implementations

In view of the theory discussed in the previous section, some of the basic questions to be answered are:

- how to segment the image,
- what transformations to use,
- how to find the parameters of the transformations, and
- where to find the matching segments.

These issues will be discussed in this section along with compression results reported for both still images and video sequences.

2.2.1 Compression of still images

In 1989 and 1990, Jacquin [3, 4, 71, 72] developed an automatic implementation of the Local IFS method by restricting B_is to squares of two fixed sizes, and restricting the affine transformation to the following special case,

$$w_i\left(\begin{bmatrix} x \\ y \\ I(x,y) \end{bmatrix}\right) = \begin{bmatrix} a_{1,1,i} & a_{1,2,i} & 0 \\ a_{2,1,i} & a_{2,2,i} & 0 \\ 0 & 0 & p_{i,0} \end{bmatrix} \begin{bmatrix} x \\ y \\ I(x,y) \end{bmatrix} + \begin{bmatrix} b_i \\ c_i \\ p_{i,1} \end{bmatrix}.$$

where

$$\begin{bmatrix} a_{1,1,i} & a_{1,2,i} \\ a_{2,1,i} & a_{2,2,i} \end{bmatrix} = \begin{bmatrix} \pm a & 0 \\ 0 & \pm a \end{bmatrix} \text{or} \begin{bmatrix} 0 & \pm a \\ \pm a & 0 \end{bmatrix},$$

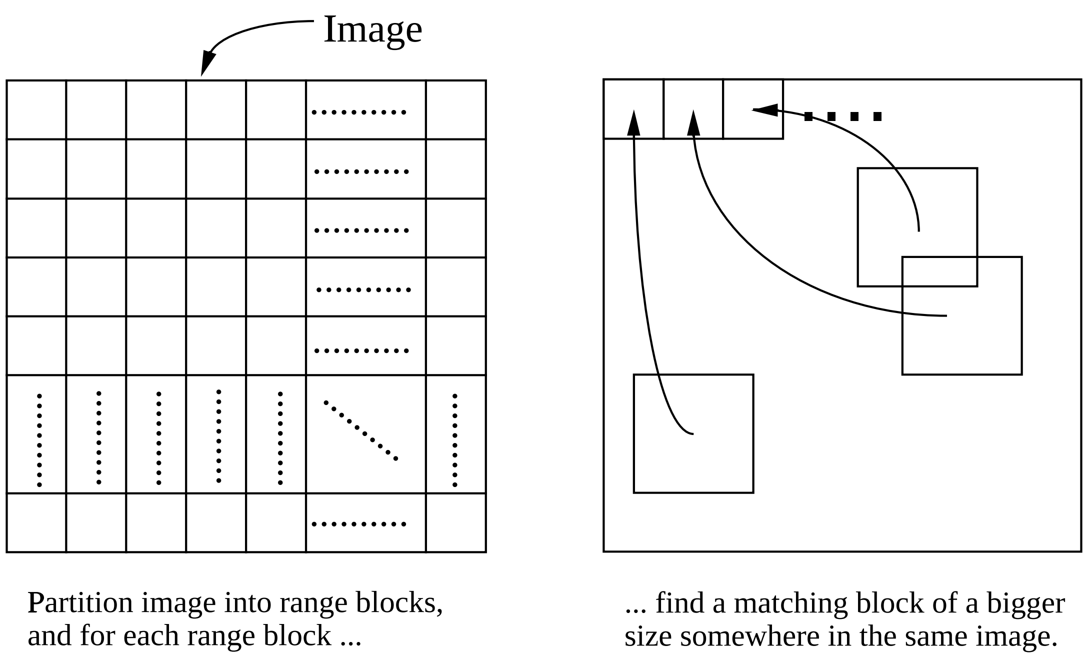


Image
Partition image into range blocks, and for each range block ...
... find a matching block of a bigger size somewhere in the same image.

Figure 2.3 A demonstration of Jacquin's algorithm.

and the origin of the x and y axes is the center of the domain block, $a = 0.5$ and $p_{i,0} < 1$. For each i, the $p_{i,0}$, b_i, and c_i are found by search, and $p_{i,1}$ is computed. The essence of Jacquin's method can be summarized as partitioning the image into square range blocks and searching the image for matching domain blocks of twice the size of the range block, as shown in Figure 2.3. In finding a matching block, we are allowed to apply simple transformations on the domain block, which include shrinking, adding a single value to the grayscale of the pixels in the block, and scaling by a number less than one. Some shuffling of the pixel locations (*isometric* transformations) are also allowed, which include rotation by multiples of 90 degrees, and/or reflection against vertical or horizontal axes. The encoding process is also enhanced by a two-level hierarchical block splitting method and a range and domain block classification scheme for a faster search. For the 512×512 standard Lena image, PSNRs of 30.1 dB and 31.4 dB were reported at bitrates of 0.57 and 0.6 bits per pixel (bpp) [71, 110].

In 1991, Øien et al. [111] extended this method to

$$w_i\left(\begin{bmatrix} x \\ y \\ I(x,y) \end{bmatrix}\right) = \begin{bmatrix} a_{1,1,i} & a_{1,2,i} & 0 \\ a_{2,1,i} & a_{2,2,i} & 0 \\ d_i & e_i & p_{i,0} \end{bmatrix} \begin{bmatrix} x \\ y \\ I(x,y) \end{bmatrix} + \begin{bmatrix} b_i \\ c_i \\ p_{i,1} \end{bmatrix}.$$

In this case, for each i, values of d_i, e_i, $p_{i,0}$, $p_{i,1}$ are found by least squares methods, and b_i, c_i are again found by search. Using this method, the 512×512 Lena image was encoded at a bitrate of 0.5 bpp with a 30.8 dB PSNR.

In 1992, Monro and Dudbridge [112, 113] suggested partitioning an image into small square images and for each small image, an IFS (and not a Local IFS) was to be found. This is equivalent to a Local IFS with the domain block for each range block being a predetermined block which contains the range block.

Fisher [114, 115] and Jacobs, Fisher, and Boss [116] studied the effect of using blocks of different shapes including squares, rectangles, and/or triangles combined with a multilevel hierarchical block splitting method. They also compared the trade-offs between compression ratio and signal-to-noise-ratio (SNR) for their method [116]. In two of their experiments, the 512×512 Lena image was coded at 0.22 bpp with PSNR of 30.71 dB and at 0.45 bpp with PSNR of 33.40.

In 1993, Gharavi-Alkhansari and Huang [6, 7] extended Jacquin's method and showed that one can use a linear combination of a series of fixed and transformed domain blocks instead of only a single domain block and one or a few fixed blocks. They also provided an algorithm for making a selection among all of the possible domain and fixed blocks.

Thomas and Deravi [82] used blocks of relatively free shapes in Jacquin's algorithm and showed that this could improve the performance of Jacquin's method for simple images. For the 512×512 image Lena, they obtained a PSNR of 27.7 dB at 0.30 bpp.

Lepsøy et al. [117] introduced a noniterative decoding algorithm for fractal-based image compression.

Also, Vines and Hayes [118] suggested limiting the search on b_i and c_i by looking for matching domain blocks only in the neighborhood of the corresponding range blocks.

Using this method and a multilevel block-splitting scheme, the 512×512 Lena image could be compressed at a bitrate of 0.47 bpp with PSNR of 31.5 dB.

Up until 1993, most of the attention of the papers published on fractal coding were concentrated on the fractal transform. Since then, more attention has been paid to the entropy coding stages following the fractal transform and on the problem of how the fractal transform parameters could be best modeled for entropy coding. This has resulted in more efficient algorithms.

In 1994, Barthel et al. published results on a fractal-based coding method with a performance of 35 dB PSNR at 0.35 bpp for the 512×512 Lena image [119]. In this method, after approximating a range block with a domain block, any spectrum coefficient of the range block in DCT domain that is not well-approximated by the domain block, is individually coded using transform coding. These spectral coefficients are then excluded from being approximated by the domain block. Rate-distortion optimality is also used as the criteria for selecting the best possible choice in places where there are several possible alternatives in the encoding process.

Also, in 1994, Gharavi-Alkhansari and Huang proposed a generalized image block coding method for unifying the three methods of block transform coding, vector quantization, and fractal-based coding methods [8, 9]. In this method, every block in the image is approximated by a linear combination of one or more blocks selected from a possibly large dictionary of (not necessarily orthogonal) library blocks. In the case of video coding, block prediction methods such as DPCM and adaptive block prediction methods like block motion compensation methods are also special cases of this algorithm. They also proposed that the iterative nature of the fractal image decoders is related to the noncausality of the encoder and using a causal encoder results in a noniterative decoder that converges in one iteration.

Lin [16] also studied fractal image coding as a generalized predictive coding method and showed how noncausal prediction in fractal coders necessitates an iterative decoding.

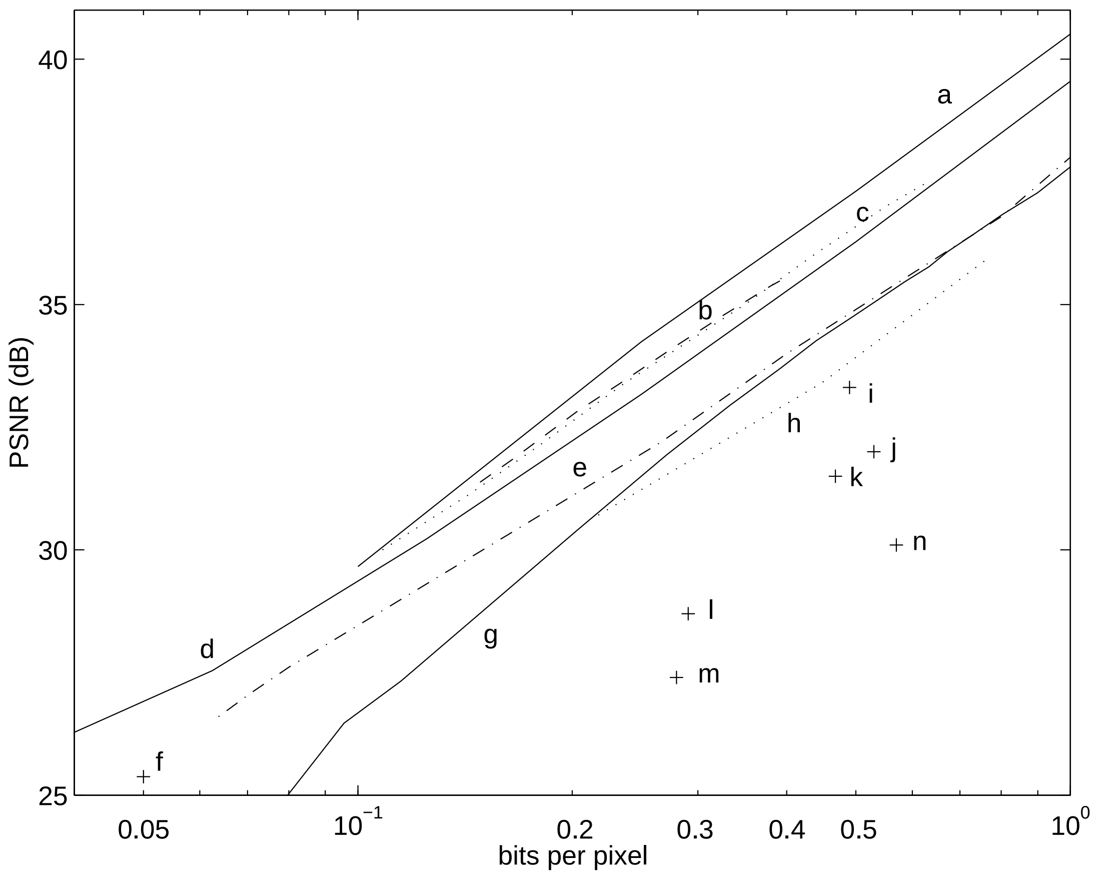


Figure 2.4 PSNR vs. bitrate for compression of 512 × 512 Lena image with some fractal-based and nonfractal-based methods. See Table 2.1 for references.

In 1994 and 1995, Rinaldo and Calvagno [47, 48] used similarities between blocks in different subbands of image for image coding and reported a performance of 32.78 PSNR at 0.26 bpp [47].

Figure 2.4 shows the reported performance of some fractal compression methods, along with some nonfractal compression methods, in terms of PSNR and bitrate for the 512 × 512 Lena image.[2] Different curves in this plot are assigned letters which are described in Table 2.1. Curves "g," "d," and "a" are for JPEG, wavelet-based zerotree, and improved wavelet-based zerotree methods which are nonfractal methods and are

[2]The data shown in this plot are brought here only for a rough comparison. PSNR is not always a good measure of image quality. Also, in regards to the 512 × 512 8 bits per pixel grayscale test image Lena, authors are aware of at least two versions of this image that may have been used by researchers for obtaining these results. Some of the mentioned methods also did not use optimal entropy coders for coding of the fractal transform parameters.

Table 2.1 References for Figure 2.4

	Year	Researchers	Reference	Method
a	1994	Xiong et al.	[120]	(Nonfractal) Wavelets
b	1994	Rinaldo and Calvagno	[47]	Fractal-Wavelet
c	1994	Barthel et al.	[119]	Fractal-DCT
d	1993	Shapiro	[121]	(Nonfractal) Wavelets
e	1995	Fisher and Menlove	[122]	Fractal
f	1995	Culik and Kari	[123]	Fractal
g	1991	JPEG	[124, 125]	(Nonfractal) JPEG
h	1992	Fisher et al.	[114, 115, 116]	Fractal
i	1994	Kim and Park	[126]	Fractal
j	1993	Lepsøy et al.	[117]	Fractal
k	1993	Vines and Hayes	[118]	Fractal
l	1994	Lu and Yew	[127]	Fractal
m	1993	Thomas and Deravi	[82, 83]	Fractal
n	1990	Jacquin	[71]	Fractal

introduced here only for comparison. The JPEG results brought here are based on the "Independent JPEG Group's free JPEG software" implementation of JPEG. For an implementation based on JPEG standard with a significantly better performance, see Crouse and Ramchandran [128].

2.2.2 Compression of video sequences

Fractal-based techniques have also been explored for coding image sequences.

In 1991, Beamount [129] used fractal-based techniques for video compression. He tried two different approaches for this purpose. In one method, he extended Jacquin's method and used three dimensional blocks, i.e., rectangular cubes, of video sequences instead of the 2-D blocks in still images. He reported that although the high compression could be achieved using this method, the quality of the decoded images was not good. Using another method, Beamount applied the 2-D Jacquin method on individual frames, but for each frame (except for the first frame) took the domain blocks from the previous frame instead of the same frame. It was reported that 10 frame-per-second 352×288

grayscale video sequences could be coded at a data rate of 80 Kbits/s with "reasonable quality."

In 1992, Hurd et al. [130] from Iterated Systems, Inc., published results on fractal-based video compression claiming compression ratios from 21:1 (average PSNR of 39.2 dB) to 79:1 (average PSNR of 30.8 dB) for a 160×120, 8-bit grayscale video sequence. In their method, they encoded the first frame using regular fractal coder. For the following frames, they always used the previous frame as the source of domain blocks. To approximate each range block in one frame they either (1) applied motion compensation and found a matching same-size domain block from previous decoded frame (no contraction) or (2) a single matching larger size domain block with a contractive transformation applied on it from the previous decoded frame was found. No residual error was sent for the frames. As the coding of this method was causal, the decoding process was noniterative. In fact, due to the low complexity of the decoding algorithm, this method had a very fast decompression.

In 1993, Hürtgen and Büttgen [131] applied fractal techniques for low-bitrate video coding. They applied prediction by frame differencing with no motion compensation. Then, for each frame, they applied the fractal transform only to regions of the frame where prediction failed. For these regions, they used a still fractal coding scheme. For range blocks located in these regions, domain blocks from the entire same frame were searched. In contrast to previous methods, for these regions they did not use previous frames. They also used a 3-level block splitting method in their algorithm. The 352×288, 8 1/3 Hz (25 Hz subsampled by 3) Miss America video sequence was reported to be coded at 128 Kbits/sec with an average PSNR of 36–37 dB, and at 64 Kbits/sec with an average PSNR of 34–35 dB, and at 32 Kbits/sec with an average PSNR of 30–32 dB. As the domain blocks for each range block were selected from the same frame, the decoder is iterative in this method.

Also, in 1993, Li et al. [132] tried an extension of the still image compression method developed by Monro and Dudbridge [113] to video compression and showed how compression ratios from 25:1 (average PSNR of 36.2 dB) to 51:1 (average PSNR of 27.2 dB) can be achieved for the 256×256, 15Hz Miss America sequence. In this method, the video sequence is partitioned into 3-D blocks. Each block is then partitioned into eight 3-D sub-blocks each of which is approximated by a contractive transformation applied on the block that contains it.

In 1994, Lazar and Bruton [133] also extended Jacquin's 2-D algorithm to 3-D and used 3-D range and domain blocks for image compression. They also used a 3-D block splitting method and the search for selecting domain blocks is done only in the neighborhood of the range block. They reported an average compression ratio of 74.39 at an average PSNR of 32–33 dB for the 360×280, 8 bit/pixel, 30 Hz Miss America video sequence.

Other researchers have also contributed to the theory and implementation issues of fractal video coding [84], [134]–[141].

2.2.3 Complexity

In terms of complexity, fractal-based image coding is asymmetric, i.e, the complexity of the encoder is typically much higher than that of the decoder. Complexities of these encoders are typically much higher than those of transform coders and vector quantizers. The most time consuming part of the encoding procedure is usually the search for finding the best matching domain blocks. Different techniques have been studied for limiting, structuring, and approximating the search procedure [142]–[144].

In many implementations of fractal image coders, the search is limited to the neighborhood of the range block where finding a good match is more likely. In the extreme case, the search may be totally avoided by using a predetermined domain block at the location of the range block. The search in Jacquin's original method included searching domain blocks that were generated by applying some isometric transformations on the image blocks (e.g., rotation by multiples of 90 degrees or reflection against horizontal or vertical axes). It has been found that it is more likely that the best match is the domain block taken from the image rather than from among the isometrically transformed versions and, therefore, in many fractal image and video coders these transformations are not used.

In other implementations, the domain and range blocks are classified based on some criteria of structure of the blocks. Then, for each range block, the block matching search is done only among the domain blocks that are in the same class as that of the range block.

Another approach to reducing the search is by doing a coarse-to-fine search. The search is done first using a coarse measure of the similarity of the blocks, then another search with a finer measure of similarity is done among the blocks that had high similarity in the coarser measure.

On the other hand, complexity of the fractal image decoders is usually much lower than their corresponding encoders, and, in some cases, even less than some transform coding methods. This makes this method more suitable for publishing or broadcasting where the image is compressed once by a central processor and decompressed several times by smaller receiving processors.

CHAPTER 3

A SYSTEM/GRAPH THEORETICAL ANALYSIS OF ATTRACTOR CODERS

In Chapter 2, we saw how some types of deterministically self-similar fractals may be constructed as attractors of fractal transformations. These attractors typically have visually complex structures, while the mathematical description of the IFS generating them is simple. On the other hand, some of these attractors have similarities to natural objects. This motivated the use of IFS for image compression. The theory of IFS was further extended to Local IFS, which eventually provided a tool for practical image compression.

The fractal transformations associated with Local IFS are only a small subset of all transformations with attractors and the above study provides a motivation for the more general theory of *attractor coding*. In this theory, an image is represented as the attractor of a transformation, or equivalently, the steady-state of a dynamical system. The encoding process is done by encoding the parameters of this dynamical system.

In the literature, the expressions *attractor coding* and *fractal coding* are used interchangeably. This is due to the fact that nearly all of the study on attractor coding has been concentrated on the fractal coders. In this dissertation, we emphasize the distinction between these two expressions and the fact that one is only a subset of the other. In fractal coders, the compression is done by using the basic property of fractals, namely, the property that different segments of different sizes in the signal can be similar in some sense. In fact, in this dissertation, we will study attractor coders that do not use this type of self-similarity.

In this chapter, we will look at the theory of attractor coding from a general point of view. We will begin the study from a broad point of view and narrow it down to more special cases as we progress. We will specifically focus our attention on linear systems, and make use of some of the results of linear systems theory. From this novel approach, it will soon become clear how the concept of the stability of a dynamical system provides a much more direct path to the convergence of attractor coders compared to the concept of contractivity, which is being widely used in the literature of fractal coders.

3.1 Mathematical Model of Images

The theory of IFS is based on transformations defined on the metric space $(H(x), h)$. The attractors of these transformations are compact, nonempty subsets of the metric space (X, d). In order to use an IFS for representing a natural image, the image must be represented by a compact, nonempty *set*. Black and white continuous-space images may be represented by subsets of R^2 where black points are members and white points are not members of the set. Grayscale images may be represented by subsets of R^3, where for each point with (x, y, z) coordinates, the x and y are the spatial coordinates and z is the intensity of the image at (x, y). To apply this theory to discrete-space images, one possible approach is to assume that a discrete-space image is a sampled version of a continuous-space image, which is the attractor (or close to attractor) of an IFS. This representation of grayscale images has several difficulties. For the continuous-space images, we need to put the following restrictions on the sets representing images:

(1) The projection of all points in the set onto the xy plane completely covers a subset of R^2 of the form $I \times J$ where I and J are closed intervals in R.

(2) For each (x, y) in $I \times J$, there must be one and only one point of the form (x, y, z) in the set.

In the discrete-space case, there is the additional problem that this method does not provide a *clean* representation and, in practice, is more difficult to implement and analyze.

The theory of IFS has been extended to measures and IFS with probabilities [3, 19] in which images may be represented as measures. This provides a more suitable approach for modeling grayscale images in terms of attractors of IFS.

However, an alternative approach, which we will use in this dissertation, is to model continuous-space images as functions from $I \times J \mapsto R$, and discrete-space images as functions from $\{1, 2, \ldots, N_1\} \times \{1, 2, \ldots, N_2\} \mapsto R$, where N_1 and N_2 are some positive integers. In either case, the set of all such functions makes a vector space and each image is a vector in the space. In the case of continuous-space images, the space is infinite-dimensional, while the dimension of discrete-time case is finite.

3.2 An Operator Point of View

Consider a complete metric space X, and an operator $\mathcal{T} : X \mapsto X$. If we apply $\mathcal{T}$ iteratively on an initial $x_0 \in X$, we obtain a sequence (x_n) described by the recursive equation

$$x_{n+1} = \mathcal{T}(x_n), \qquad n \geq 0. \tag{3.1}$$

If for any arbitrary initial $x_0 \in X$, the sequence (x_n) converges to the same point x^* in the space, then x^* is called the *attractor* of $\mathcal{T}$. In such case, x^* is completely described by $\mathcal{T}$, and given $\mathcal{T}$, x^* may be obtained by iteratively applying $\mathcal{T}$ on any $x \in X$, i.e.,

$$x^* = \mathcal{T}^{\infty}(x), \qquad \forall x \in X.$$

We call $\mathcal{T}$ an attractor representation of x^*. We consider two special cases.

Case 1: The trivial case of $\mathcal{T}$ being constant

$$\forall x \in X, \qquad \mathcal{T}(x) = a,$$

then

$$\forall x \in X, \qquad \mathcal{T}^{\infty}(x) = a.$$

In this sense, any process that generates a specific image may be interpreted as an attractor representation of that image. However, by default, when we refer to attractor representation of an image, we mean representation by a nonconstant operator.

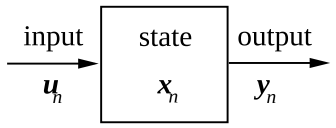


Figure 3.1 A discrete-time system.

Case 2: X is a vector space and $\mathcal{T}$ has the form

$$\forall x \in X, \qquad \mathcal{T}(x) = Ax + B \tag{3.2}$$

where A is a linear operator and B is a constant operator. This is the most common form of operator used in attractor coding and will be discussed further in the following sections.

3.3 A System Point of View

3.3.1 Discrete-time system

An alternative point of view for analyzing attractor coding systems is to look at them as *discrete-time systems.* Let us consider a discrete-time system with input $\mathbf{u}_n$, output $\mathbf{y}_n$, and state $\mathbf{x}_n$, as shown in Figure 3.1. The set of equations that describe the relation between input, output, and state are called *dynamical equations.* We are interested in discrete-time systems whose dynamical equations are in the form

$$\mathbf{x}_{n+1} = h(\mathbf{x}_n, \mathbf{u}_n, n) \tag{3.3}$$

$$\mathbf{y}_n = g(\mathbf{x}_n, \mathbf{u}_n, n). \tag{3.4}$$

Equation (3.3) describes the behavior of the state of the system and is called the *state equation.* Equation (3.4) describes the output of the system and is called the *output equation.*

If the system is *time-invariant*, these equations become

$$\mathbf{x}_{n+1} = h(\mathbf{x}_n, \mathbf{u}_n)$$

$$\mathbf{y}_n = g(\mathbf{x}_n, \mathbf{u}_n).$$

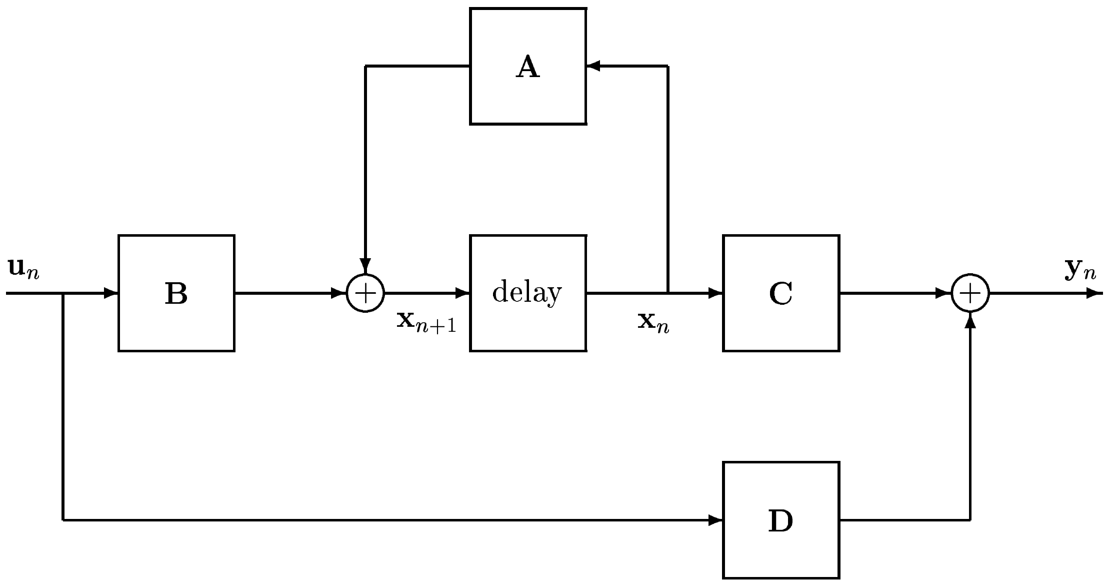


Figure 3.2 Block diagram of a linear, time-invariant, discrete-time system.

If the system is *linear*, then (3.3) and (3.4) may be written as

$$\begin{aligned}\mathbf{x}_{n+1} &= \mathbf{A}_n\mathbf{x}_n + \mathbf{B}_n\mathbf{u}_n \\ \mathbf{y}_n &= \mathbf{C}_n\mathbf{x}_n + \mathbf{D}_n\mathbf{u}_n\end{aligned}$$

where $\mathbf{A}_n$, $\mathbf{B}_n$, $\mathbf{C}_n$, and $\mathbf{D}_n$ are linear, time-dependent operators, and $\mathbf{A}_n$ is called the *state-transition* operator.

If the system is both linear and time-invariant, then the dynamical equations may be written as

$$\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{B}\mathbf{u}_n \tag{3.5}$$

$$\mathbf{y}_n = \mathbf{C}\mathbf{x}_n + \mathbf{D}\mathbf{u}_n. \tag{3.6}$$

This linear, time-invariant (LTI), discrete-time system may be represented by a block diagram as shown in Figure 3.2.

3.3.2 Continuous-time system

Let us consider a *continuous-time* system with input $\mathbf{u}(t)$, output $\mathbf{y}(t)$, and state $\mathbf{x}(t)$, as shown in Figure 3.3. We are interested in continuous-time systems with dynamical

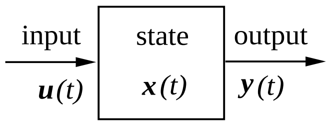


Figure 3.3 A continuous-time system.

equations in the form

$$\dot{\mathbf{x}}(t) = h(\mathbf{x}(t), \mathbf{u}(t), t) \tag{3.7}$$

$$\mathbf{y}(t) = g(\mathbf{x}(t), \mathbf{u}(t)), \tag{3.8}$$

where

$$\dot{\mathbf{x}}(t) \overset{\text{def}}{=} \frac{d\mathbf{x}(t)}{dt}.$$

If the system is time-invariant, then (3.7) and (3.8) become

$$\dot{\mathbf{x}}(t) = h(\mathbf{x}(t), \mathbf{u}(t))$$

$$\mathbf{y}(t) = g(\mathbf{x}(t), \mathbf{u}(t)).$$

If the system is linear, then (3.7) and (3.8) may be written as

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) \tag{3.9}$$

$$\mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t)$$

where $\mathbf{A}(t)$, $\mathbf{B}(t)$, $\mathbf{C}(t)$, and $\mathbf{D}(t)$ are linear, time-dependent operators.

If the system is both linear and time-invariant, then the dynamical equations may be written as

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{3.10}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t). \tag{3.11}$$

A linear, time-invariant, continuous-time system may be represented by a block diagram as shown in Figure 3.4.

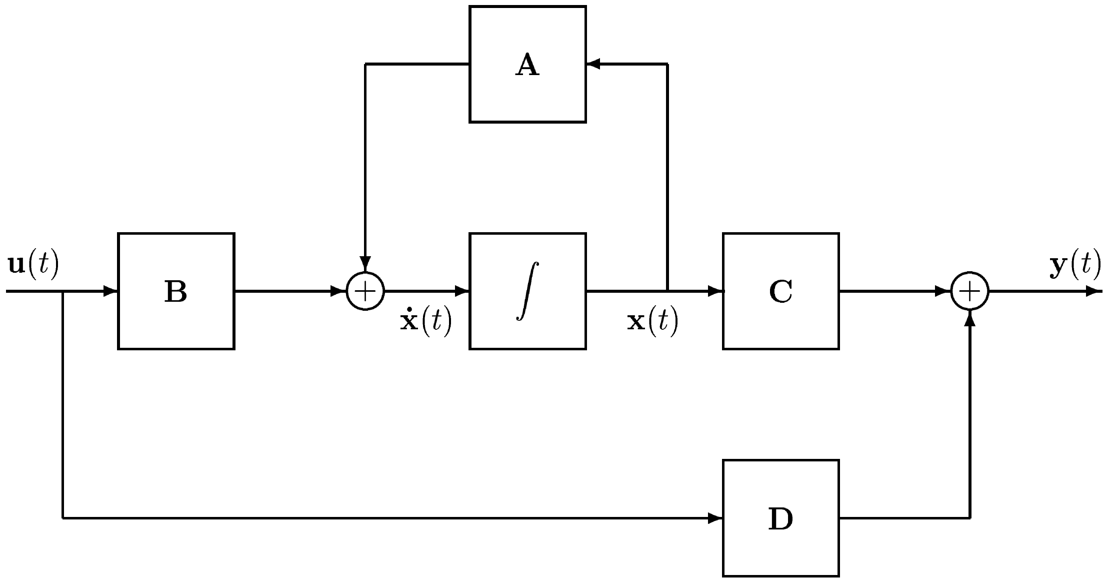


Figure 3.4 Block diagram of a linear, time-invariant, continuous-time system.

Table 3.1 Interpretation of continuity in time and space for images

	continuous time	discrete time
continuous space	continuous image evolving continuously in time	continuous image changing only in time steps
discrete space	discrete image evolving continuously in time	discrete image changing only in time steps

3.3.3 Representation of operators

If $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{u}$ are finite-dimensional signals (e.g., discrete-space images), then we may represent them by column matrices and $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ may be represented in matrix form. However, in the infinite-dimensional case with $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{u}$ being functions from $R^n \mapsto R$ (e.g., continuous-space images), $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ may be represented by their corresponding kernels. For the case of the images, we should emphasize the distinction between continuity in time and space, as shown in Table 3.1.

3.4 Comparison of the Two Viewpoints

Note that the operator point of view described by (3.1) and (3.2) is equivalent to Equation (3.5) of a linear, time-invariant system when $\mathbf{u}_n$ is constant. In other words, we may consider $x \in X$ as the state of a system which changes at each time step according to $\mathcal{T}$. The systems point of view, in this sense, has the operator point of view embedded in it. It also provides for a more general class of attractor coders and for systems that change continuously in time, as well as makes the connection between the young field of attractor coding and the well-established field of systems theory. Using the systems method, the output of the system may be defined to be different from (although a function of) the state of the system.

A comparison of Equations (3.2) and (3.5) also shows that the most common type of system being represented by attractor coders is, in fact, a linear system in which

- the image is the *state* of the system,
- the input is constant,
- the initial image is the *initial state* of the system, and
- the attractor is the *steady-state* of the system.

More interesting situations occur if we take the output Equation (3.6) into account and define $\mathbf{y}_\infty$ as the final decoded image in attractor coding. Then, if $\mathbf{C}$ is not the identity matrix, there may be state variables that are variable in time whose changes indirectly affect the output image. This has a strong relation to the concept of *hidden variables* defined in [19]. For example, $\mathbf{x}_n$ may be an evolving 3-D set and $\mathbf{y}$ a projection of that set into 2-D. The final decoded image is the projection of the attractor of the state into a 2-D plane.

We note that linearity is defined when the signals being studied are members of vector spaces. In the original theory of IFS, images are only nonempty compact sets and cannot be analyzed by linear systems. However, implementations of this theory

(Jacquin's method and those based on it) can be well analyzed using the linear systems theory.

The continuous-time system case introduced in Section 3.3.2 has no equivalent in the theory of attractor coding. Nevertheless, signals in a discrete-time system may be represented as samples of signals in a related continuous-time system, which brings up the concept of *continuous-time attractor coding.*

3.5 Modeling Fractal Coding with Linear Systems

In this section, we present a linear system model for fractal coding. As described in the previous chapter, fractal coders typically encode an image by representing it with the parameters of a dynamical system whose attractor is close to the given image. The analysis provided in this chapter is for the 1-D case. For the 2-D case, the analysis and properties will be very similar to that of the 1-D case.

We begin with real discrete-time signals of finite length N. These signals may be represented by N-dimensional vectors in R^N. Let

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

be such a signal. In the 2-D case, $\mathbf{x}$ may, for example, be generated by scanning an image row-by-row. Then, $\mathbf{x}$ may be coded by designing a transformation $\mathcal{T}$ such that the dynamical system described by the equation

$$\mathbf{x}_{n+1} = \mathcal{T}(\mathbf{x}_n) \tag{3.12}$$

has the property that for any initial vector $\mathbf{x}_0 \in R^N$,

$$\mathcal{T}^{\infty}(\mathbf{x}_0) \approx \mathbf{x},$$

and the number of bits needed for representing $\mathcal{T}$ is smaller than that of $\mathbf{x}$. The Contraction Mapping Theorem provides a sufficient condition for convergence of (3.12). A more relaxed sufficient condition is that $\mathcal{T}$ must be *eventually contractive* (see Section 3.9.1).

In practice, linear transformations are generally used, so $\mathcal{T}$ may be represented by matrices $\mathbf{A}$ and $\mathbf{B}$ such that

$$\mathcal{T}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{B}$$

and (3.12) may be written as

$$\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{B}. \tag{3.13}$$

In fractal image compression, each signal is typically partitioned into K range blocks $\mathbf{x}_i$, $i = 1, 2, \ldots, K$, each of size M, i.e.,

$$\begin{aligned} N &= KM, \\ \mathbf{x} &= \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_K \end{bmatrix}, \\ \mathbf{x}_i &= \begin{bmatrix} x_{(i-1)M+1} \\ x_{(i-1)M+2} \\ \vdots \\ x_{(i-1)M+M} \end{bmatrix}, \end{aligned}$$

and for each $\mathbf{x}_i$, a domain block $\mathbf{d}_i$ of size LM (with $L > 1$) and an operator $\mathcal{T}_i : R^{LM} \mapsto R^M$ is found such that

$$\mathbf{x}_i \approx \mathcal{T}_i(\mathbf{d}_i) = \alpha_i \mathbf{P}_i \mathbf{G} \mathbf{d}_i + \mathbf{B}_i, \qquad i = 1, 2, \ldots, K.$$

$\mathbf{B}_i$ is a shift in grayscale

$$\mathbf{B}_i = \begin{bmatrix} b_i \\ b_i \\ \vdots \\ b_i \end{bmatrix}_{M\times 1},$$

and α_i is a scalar factor. $\mathbf{P}_i$ is an $M \times M$ matrix generated by a permutation of the rows of the identity matrix. It corresponds to a shuffling of the elements of the domain blocks, and, in the 2-D case, can, among other things, apply rotation of blocks by multiples of 90 degrees, and/or reflection against vertical, horizontal, or diagonal axis.

$\mathbf{G}$ is a *shrinking* matrix, which, in the 1-D case, typically has the form

$$\mathbf{G} = \begin{bmatrix} \mathbf{w} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{w} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{w} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{w} \end{bmatrix}_{M \times LM},$$

where

$$\mathbf{w} = \begin{bmatrix} \frac{1}{L} & \frac{1}{L} & \cdots & \frac{1}{L} \end{bmatrix}_{1 \times L}.$$

However, $\mathbf{G}$ may be generalized to represent decimation, i.e., lowpass filtering followed by subsampling, in which case it may be written as

$$\mathbf{G}_{M \times LM} = \mathbf{S}_{M \times LM} \mathbf{F}_{LM \times LM}$$

where

$$\mathbf{F} = \begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_{L'-1} & a_{L'} & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & a_1 & a_2 & \cdots & a_{L'-2} & a_{L'-1} & a_{L'} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & a_1 & a_2 & \cdots & a_{L'} \\ a_{L'} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_1 & \cdots & a_{L'-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_2 & a_3 & a_4 & \cdots & a_{L'} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & a_1 \end{bmatrix},$$

and is a circular matrix.[1] L' is the length of the filter and, in general, may be different from L. $\mathbf{S}$ is the subsampling matrix and may be written as:

$$\mathbf{S} = \begin{bmatrix} \mathrm{s} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathrm{s} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathrm{s} \end{bmatrix}_{M \times LM},$$

where

$$\mathrm{s} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}_{1 \times L}.$$

In the 2-D case, $\mathbf{F}$ should represent a 2-D filtering operator and $\mathbf{S}$ should represent a 2-D subsampling by a factor L.

The operator $\mathcal{T}_i$ transforms a domain vector $\mathbf{d}_i$ to an approximation of a range vector $\mathbf{x}_i$. All of the operators $\mathcal{T}_i$, $i = 1, 2, \ldots, K$, together define a transformation $\mathcal{T}$ for $\mathbf{x}$ such that

$$\mathbf{x} \approx \mathcal{T}(\mathbf{x}) \tag{3.14}$$

where

$$\begin{aligned} \mathcal{T}(\mathbf{x}) &= \sum_{i=1}^{K} \mathbf{H}_i \left(\mathcal{T}_i \left(\mathbf{d}_i\right)\right) \\ &= \sum_{i=1}^{K} \mathbf{H}_i \left(\left(\alpha_i \mathbf{P}_i \mathbf{G} \mathbf{d}_i\right) + \mathbf{B}_i\right) \\ &= \sum_{i=1}^{K} \alpha_i \mathbf{H}_i \mathbf{P}_i \mathbf{G} \mathbf{d}_i + \sum_{i=1}^{K} \mathbf{H}_i \mathbf{B}_i \\ &= \left(\sum_{i=1}^{K} \alpha_i \mathbf{H}_i \mathbf{P}_i \mathbf{G} \mathbf{K}_i\right) \mathbf{x} + \left(\sum_{i=1}^{K} \mathbf{H}_i \mathbf{B}_i\right). \end{aligned} \tag{3.15}$$

$\mathbf{K}_i$, $i = 1, 2, \ldots, K$ are *fetch* operators, generating $\mathbf{d}_i$ from $\mathbf{x}$, i.e., $\mathbf{d}_i = \mathbf{K}_i\ \mathbf{x}$. If we denote the index of the first element of the domain block corresponding to $\mathbf{x}_i$ by I_i, then

$$\mathbf{K}_i = \begin{bmatrix} \mathbf{0}_{LM \times (I_i - 1)} & \mathbf{I}_{LM \times LM} & \mathbf{0}_{LM \times (N-(I_i-1)-LM)} \end{bmatrix}_{LM \times N}.$$

[1]There are many ways to treat boundaries. Here, the simplest case is presented.

$\mathbf{H}_i$, $i = 1, 2, \ldots, K$ are *put* operators generating a component of $\mathbf{x}$ from $\mathbf{x}_i$, i.e.,

$$\mathbf{H}_i\mathbf{x}_i = \begin{bmatrix} \mathbf{0}_{(i-1)M\times 1} \\ \mathbf{x}_i \\ \mathbf{0}_{(K-i)M\times 1} \end{bmatrix}_{N\times 1},$$

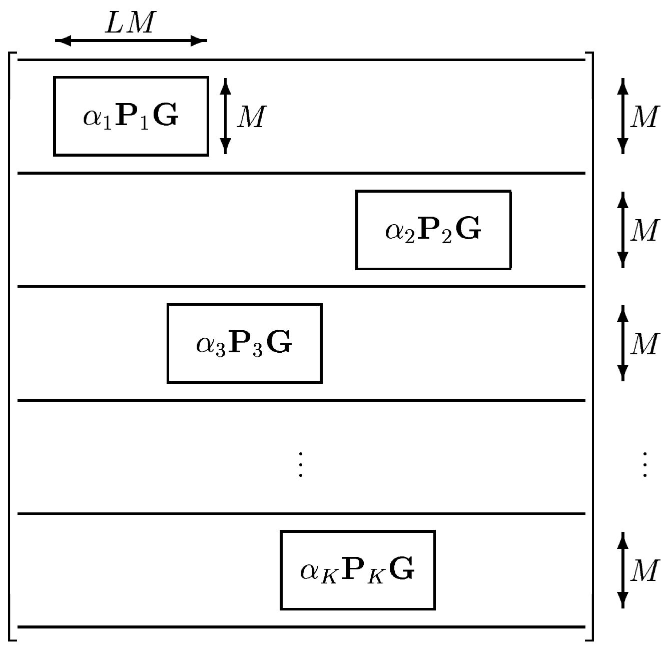
$$\mathbf{H}_i = \begin{bmatrix} \mathbf{0}_{(i-1)M\times M} \\ \mathbf{I}_{M\times M} \\ \mathbf{0}_{(K-i)M\times M} \end{bmatrix}_{N\times M}.$$

Note that (3.15) has the form

$$\mathcal{T}(\mathbf{x}) = \mathbf{A}_{N\times N}\mathbf{x} + \mathbf{B}_{N\times 1} \tag{3.16}$$

where $\mathbf{A}$ is of the form

$\mathbf{A} =$



LM
$\alpha_1\mathbf{P}_1\mathbf{G}$
M
M
$\alpha_2\mathbf{P}_2\mathbf{G}$
M
$\alpha_3\mathbf{P}_3\mathbf{G}$
M
$\vdots$
$\vdots$
$\alpha_K\mathbf{P}_K\mathbf{G}$
M

and the horizontal location of each matrix $\alpha_i\mathbf{P}_i\mathbf{G}$ is determined by I_i. $\mathbf{B}$ has the form

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \vdots \\ \mathbf{B}_K \end{bmatrix}_{N\times 1}.$$

In fractal coders, the matrices $\mathbf{G}$ and $\mathbf{H}_i$, $i = 1, 2, \ldots, K$ are typically independent of $\mathbf{x}$ and are fixed both at the encoder and the decoder, and are not part of the code. But matrices $\mathbf{B}_i$, $\mathbf{K}_i$, $\mathbf{P}_i$ and values α_i, for $i = 1, 2, \ldots, K$ need to be specified for the decoder through the code.

Due to the structure of $\mathbf{B}_i$, it can be specified by sending b_i, $i = 1, 2, \ldots, K$, i.e., for each range block specify the offset of the mean of the range block with respect to the mean of the domain block. To specify $\mathbf{K}_i$, one needs only to specify I_i, $i = 1, 2, \ldots, K$, i.e., for each range block specify the address of its domain block. $\mathbf{P}_i$ typically represents one of a few types of permutations and one needs to specify which one is used for each range block. Many researchers use only $\mathbf{P}_i = \mathbf{I}$ and do not need to specify it in the code. The α_i is quantized and included in the code.

So, the information needed to fully define $\mathcal{T}$ for the decoder is $\{(\alpha_i, b_i, I_i, \mathbf{P}_i), i = 1, 2, \ldots, K\}$.

3.6 Generalized Model

In the previous section, we showed that typical fractal coders may be modeled by linear systems with dynamical equation of the form (3.13) and are fully described by $\mathbf{A}$ and $\mathbf{B}$. Given $\mathbf{A}$ and $\mathbf{B}$, the output is fully determined by the initial state and the input. Also, given the system, the steady-state (if it exists) is fully determined by the input and is independent of the initial state. The goal of the encoder is to send enough information to the decoder for it to construct the steady-state. This information consists of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{u}$. However, the matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{u}$ must be structured so that they can be represented by a very small numbers of bits compared to the final state. The fractal coder studied in the previous section is only one such structure. In the following section, we will analyze the problem for general $\mathbf{A}$ and $\mathbf{B}$s to gain insight from a more general point of view.

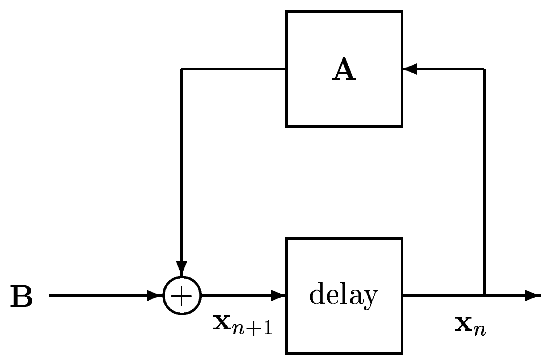


Figure 3.5 A block diagram representation of (3.17).

3.7 Finding the Steady-State

Now, we concentrate our attention on Equation (3.5), assume a constant input $\mathbf{u}_n$, and represent $\mathbf{B}\mathbf{u}_n$ simply by $\mathbf{B}$:

$$\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{B} \tag{3.17}$$

with the initial state $\mathbf{x}_0$. Figure 3.5 shows a block diagram representation of (3.17). This leads to the following series of equations.

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{A}\mathbf{x}_0 + \mathbf{B} && (3.18)\\
\mathbf{x}_2 &= \mathbf{A}\mathbf{x}_1 + \mathbf{B} = \mathbf{A}^2\mathbf{x}_0 + (\mathbf{A}+\mathbf{I})\mathbf{B}\\
\mathbf{x}_3 &= \mathbf{A}\mathbf{x}_2 + \mathbf{B} = \mathbf{A}^3\mathbf{x}_0 + (\mathbf{A}^2+\mathbf{A}+\mathbf{I})\mathbf{B}\\
&\vdots\\
\mathbf{x}_n &= \mathbf{A}\mathbf{x}_{n-1} + \mathbf{B} = \mathbf{A}^n\mathbf{x}_0 + (\mathbf{A}^{n-1}+\mathbf{A}^{n-2}+\cdots+\mathbf{I})\mathbf{B} && (3.19)\\
&\vdots
\end{aligned}$$

From digital control systems theory it is known that the sequence $(\mathbf{x}_n)$ converges, if and only if, the system described by (3.17) is *stable*. This happens when all the eigenvalues of $\mathbf{A}$ are within the unit circle. Then (3.19) may be written as

$$\mathbf{x}_n = \mathbf{A}^n\mathbf{x}_0 + (\mathbf{I}-\mathbf{A}^n)(\mathbf{I}-\mathbf{A})^{-1}\mathbf{B} \tag{3.20}$$

and by letting $n \to \infty$, we get

$$\begin{aligned}\mathbf{x}_\infty &= \mathbf{A}^\infty \mathbf{x}_0 + (\mathbf{I} - \mathbf{A}^\infty)(\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} \\ &= \mathbf{0} + (\mathbf{I} - \mathbf{0})(\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\end{aligned}$$

or

$$\mathbf{x}_\infty = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}. \tag{3.21}$$

Equation (3.21) may also be obtained by noting that $\mathbf{x}_\infty$ is the fixed point of $\mathcal{T}$ and we have

$$\mathbf{x}_\infty = \mathbf{A}\mathbf{x}_\infty + \mathbf{B}$$

which leads directly to (3.21).

From (3.21), it is clear that the steady-state of the system is independent of its initial state and is a linear function of $\mathbf{B}$. By substituting (3.21) in (3.20) we get

$$\mathbf{x}_n = \mathbf{A}^n \mathbf{x}_0 + (\mathbf{I} - \mathbf{A}^n)\mathbf{x}_\infty$$

which may be written as

$$\mathbf{x}_n - \mathbf{x}_\infty = \mathbf{A}^n(\mathbf{x}_0 - \mathbf{x}_\infty) \tag{3.22}$$

which shows an exponential convergence, if the state sequence $(\mathbf{x}_n)$ is convergent. Figure 3.6 provides a simple graph for visualizing (3.22) when $\mathbf{A}$ is 1×1 and also contractive. We note that for large n, (3.22) corresponds to the *power method* [145] for computing the eigenvalue of $\mathbf{A}$ with the largest magnitude. It can be proven that

$$\lim_{n \to \infty} (\mathbf{x}_n - \mathbf{x}_\infty) = \mathbf{v}_1,$$

where $\mathbf{v}_1$ is an eigenvector corresponding to the eigenvalue λ_1 of $\mathbf{A}$ with largest magnitude. However, this is true if $\mathbf{v}_1$ is real. If $\mathbf{A}$ has more than one eigenvalue with the same maximum value, then the limit will be a linear combination of the corresponding eigenvectors. In such a case, for large n,

$$\mathbf{x}_{n+1} - \mathbf{x}_n \approx \lambda_1(\mathbf{x}_n - \mathbf{x}_\infty)$$

or

$$\mathbf{x}_{n+1} \approx \lambda_1 \mathbf{x}_n + (1 - \lambda_1)\,\mathbf{x}_\infty.$$

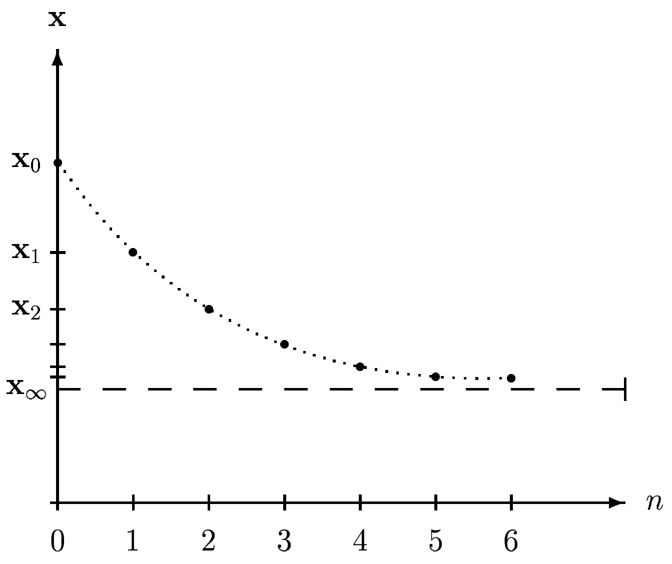


Figure 3.6 Exponential convergence of state vector to steady-state beginning from an arbitrary initial state.

3.8 Encoder and Decoder Errors

Let us assume that we apply the recursive formula of Equation (3.17) with $\mathbf{x}_0$ being the *original* signal (image) that is encoded. Using a notation similar to that of Section 2.1.2, we define

$$\mathbf{e}_E \overset{\text{def}}{=} \mathbf{x}_0 - \mathbf{x}_1 \tag{3.23}$$

$$\mathbf{e}_D \overset{\text{def}}{=} \mathbf{x}_0 - \mathbf{x}_\infty \tag{3.24}$$

and

$$\varepsilon_E \overset{\text{def}}{=} ||\mathbf{e}_E||$$

$$\varepsilon_D \overset{\text{def}}{=} ||\mathbf{e}_D||$$

and call ε_E the encoding error (or the collage error), and call ε_D the decoding error. These variables are shown in Figure 3.7 for a case of $\mathbf{A}$ being contractive.

Then, substituting $\mathbf{x}_0$ from (3.18) in (3.23) we get

$$\mathbf{e}_E = \mathbf{x}_0 - (\mathbf{A}\mathbf{x}_0 + \mathbf{B}) = (\mathbf{I} - \mathbf{A})\mathbf{x}_0 - \mathbf{B} \tag{3.25}$$

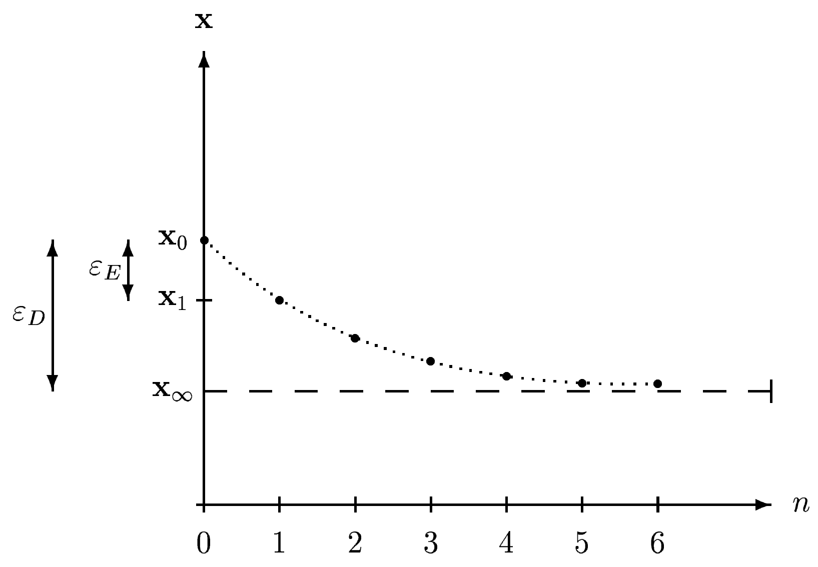


Figure 3.7 Convergence of state vector to steady-state when $\mathbf{x}_0$ is the original encoded signal (image).

and substituting $\mathbf{x}_\infty$ from (3.21) in (3.24) we get

$$\begin{aligned} \mathbf{e}_D &= \mathbf{x}_0 - (\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} \\ (\mathbf{I} - \mathbf{A})\mathbf{e}_D &= (\mathbf{I} - \mathbf{A})\mathbf{x}_0 - \mathbf{B}. \end{aligned} \tag{3.26}$$

Comparing (3.25) and (3.26) gives

$$\mathbf{e}_E = (\mathbf{I} - \mathbf{A})\mathbf{e}_D \tag{3.27}$$

or

$$\mathbf{e}_D = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{e}_E \tag{3.28}$$

which provides an explicit relationship between the encoding error vector $\mathbf{e}_E$ and the decoding error vector $\mathbf{e}_D$, in contrast to the inequality (2.8) of the collage theorem. Equation (3.27) may be used to obtain bounds for ε_D

$$\varepsilon_E = ||\mathbf{e}_E|| = ||(\mathbf{I} - \mathbf{A})\mathbf{e}_D|| \leq ||\mathbf{I} - \mathbf{A}||\,||\mathbf{e}_D|| \leq (||\mathbf{I}|| + ||\mathbf{A}||)\varepsilon_D.$$

Therefore,

$$\varepsilon_D \geq \frac{\varepsilon_E}{1 + ||\mathbf{A}||}. \tag{3.29}$$

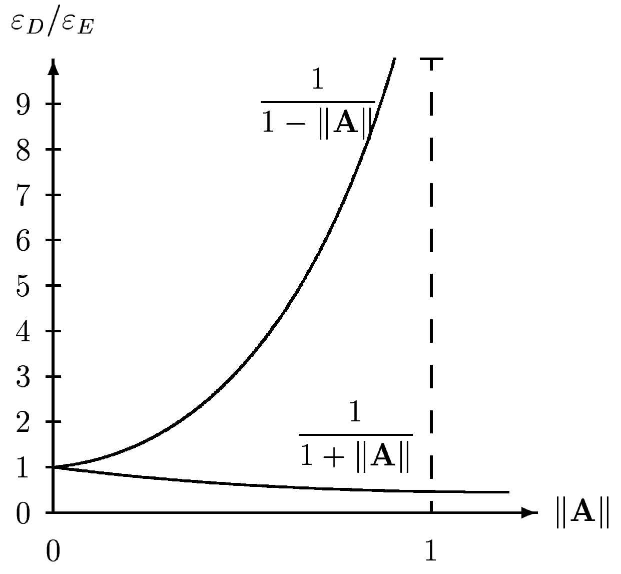


Figure 3.8 Bounds of $\varepsilon_D/\varepsilon_E$ vs. $\|\mathbf{A}\|$.

We can also rewrite (3.27) as

$$\begin{aligned}
\mathbf{e}_E &= \mathbf{e}_D - \mathbf{A}\mathbf{e}_D \\
\mathbf{e}_E + \mathbf{A}\mathbf{e}_D &= \mathbf{e}_D \\
||\mathbf{e}_E|| + ||\mathbf{A}||\,||\mathbf{e}_D|| &\geq ||\mathbf{e}_D|| \\
\varepsilon_E &\geq (1 - ||\mathbf{A}||)\varepsilon_D.
\end{aligned}$$

Then, if $\|\mathbf{A}\| < 1$,

$$\varepsilon_D \leq \frac{\varepsilon_E}{1 - ||\mathbf{A}||} \tag{3.30}$$

which is the collage theorem in the discrete case. Inequalities of (3.29) and (3.30) may be rewritten as

$$\frac{\varepsilon_E}{1 + ||\mathbf{A}||} \leq \varepsilon_D \leq \frac{\varepsilon_E}{1 - ||\mathbf{A}||}. \tag{3.31}$$

Figure 3.8 shows the bounds of $\varepsilon_D/\varepsilon_E$ vs. $\|\mathbf{A}\|$. If we use the notation

$$\mathbf{M} \overset{\text{def}}{=} (\mathbf{I} - \mathbf{A})^{-1} \tag{3.32}$$

then, we can rewrite (3.21) and (3.28) as

$$\mathbf{x}_\infty = \mathbf{MB} \tag{3.33}$$

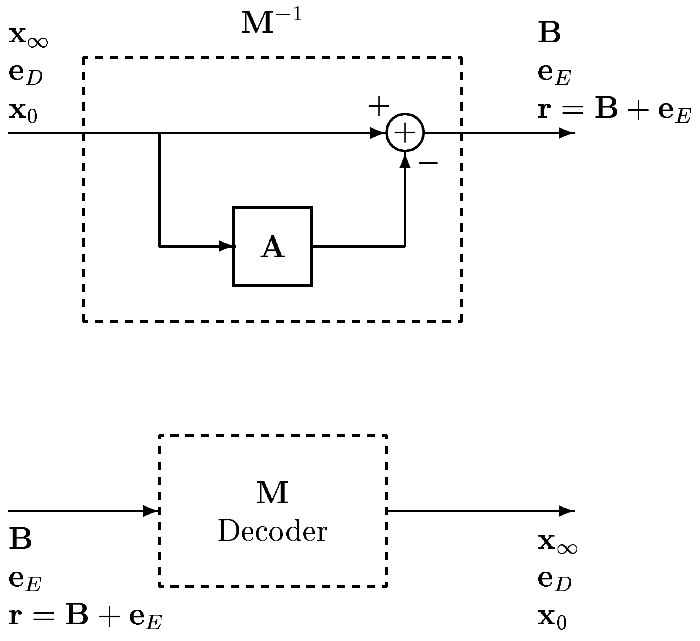


Figure 3.9 Block diagram representing the relation between $\mathbf{B}$, $\mathbf{x}_\infty$, $\mathbf{e}_E$, $\mathbf{e}_D$, $\mathbf{r}$, and $\mathbf{x}_0$.

$$\mathbf{e}_D = \mathbf{M}\mathbf{e}_E \tag{3.34}$$

which may be added to give

$$\begin{aligned}\mathbf{x}_\infty + \mathbf{e}_D &= \mathbf{M}(\mathbf{B} + \mathbf{e}_E) \\ \mathbf{x}_0 &= \mathbf{M}(\mathbf{B} + \mathbf{e}_E). \end{aligned} \tag{3.35}$$

Equations (3.33) and (3.34) clearly show that the relationship between $\mathbf{e}_D$ and $\mathbf{e}_E$ is exactly the same relationship that $\mathbf{x}_\infty$ has with $\mathbf{B}$. One may expect that selecting $\mathbf{B}$ close to $\mathbf{x}_\infty$ (or possibly to $\mathbf{x}_0$) will make $\mathbf{e}_D$ closer to $\mathbf{e}_E$. However, this may not be true in general.

Equations (3.33), (3.34) and (3.35) are summarized in Figure 3.9 and suggest that adding $\mathbf{e}_E$ to $\mathbf{B}$ changes the approximate output $\mathbf{x}_\infty$ to the exact output $\mathbf{x}_0$. This may be interpreted that both $\mathbf{B}$ and $\mathbf{e}_E$ represent some type of residuals in the encoding process, with the difference that $\mathbf{B}$ is transmitted to decoder, while $\mathbf{e}_E$ typically is not.

Let us denote $\mathbf{B} + \mathbf{e}_E$ by $\mathbf{r}$, i.e.,

$$\mathbf{r} \overset{\text{def}}{=} \mathbf{B} + \mathbf{e}_E.$$

Then, using (3.23) and (3.18), we have

$$\mathbf{x}_0 = \mathbf{A}\mathbf{x}_0 + \mathbf{r} \tag{3.36}$$

in contrast to

$$\mathbf{x}_0 \approx \mathbf{A}\mathbf{x}_0 + \mathbf{B}$$

from (3.14) and (3.16). $\mathbf{A}\mathbf{x}_0$ represents the *self-similar*ly encoded component of $\mathbf{x}_0$, and $\mathbf{r}$ represents the *residual* component of $\mathbf{x}_0$, and $\mathbf{B}$ is, in fact, the approximation of $\mathbf{r}$ which is sent to the decoder. If $\mathbf{r}$ was sent to the decoder without any error, then

$$\mathbf{r} = \mathbf{B} \;\Rightarrow\; \mathbf{e}_E = \mathbf{0} \;\Rightarrow\; \mathbf{e}_D = \mathbf{0} \;\Rightarrow\; \mathbf{x}_0 = \mathbf{x}_\infty$$

or using (3.32) and (3.36), we would have

$$\mathbf{x}_0 = \mathbf{M}\mathbf{r}.$$

This provides new insight into the encoding process:

(1) $\mathbf{e}_E$ is the error in transmitting the residual $\mathbf{r} = \mathbf{x}_0 - \mathbf{A}\mathbf{x}_0$ to the decoder. At the decoder, this error in the input causes an error $\mathbf{e}_D$ at the output, making the output $\mathbf{x}_\infty$ different from $\mathbf{x}_0$ by $\mathbf{e}_D$.

(2) By transmitting $\mathbf{B}$ to the decoder, the encoder is practically encoding the residual of the image after range blocks are approximated by domain blocks. Although this encoding is usually done by a simplistic method of sending the average value of the residual over each range block, more advanced techniques may be used for better encoding of this residual.

(3) In order to make a lossless coder/decoder, one obvious method is for the encoder to decode the image, find $\mathbf{e}_D$, and send it to the decoder in addition to the code for the fractal transform $\mathcal{T}$. However, (3.35) suggests the alternative approach of the encoder sending $\mathbf{e}_E$ rather than $\mathbf{e}_D$ to the decoder. The decoder may then add $\mathbf{e}_E$ to $\mathbf{B}$ and begin decoding. This removes the task of image decoding from the encoder and, thereby, reduces its complexity.

If we denote the eigenvalues of $\mathbf{A}$ by $\lambda_1, \lambda_2, \ldots, \lambda_N$, and its corresponding eigenvectors by $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$, then from (3.32), it is easy to show that the eigenvalues of $\mathbf{M}$ are $1/(1-\lambda_1), 1/(1-\lambda_2), \ldots, 1/(1-\lambda_N)$ with eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$. If any of the eigenvalues of $\mathbf{A}$ is close to 1, the magnitude of the corresponding eigenvalue of $\mathbf{M}$ becomes large and any component of $\mathbf{e}_E$, along its eigenvector, can cause a corresponding large component in $\mathbf{e}_D$ causing large decoding error.

3.9 Stability vs. Contractivity

3.9.1 Eventual contractivity

The proper operation of the attractor decoder requires the convergence of the sequence $(\mathbf{x}_n)$. There have been studies on this convergence from the operator point of view of Section 3.2. A sufficient condition for the convergence of $(\mathbf{x}_n)$ is for the operator $\mathcal{T}$ to be *contractive* (see Section 2.1.1). A more relaxed condition for convergence of $(\mathbf{x}_n)$, proposed by Fisher et al. [146, 147, 148], is *eventual contractivity.* An operator $\mathcal{T}$ is eventually contractive if there exists a positive integer N such that $\mathcal{T}^N$ is contractive. Obviously, all contractive operators are eventually contractive. It is known that eventual contractivity of $\mathcal{T}$ is a sufficient condition for $\mathcal{T}$ to have a unique fixed point $x^* \in X$, such that for any $x \in X$,

$$x^* = \mathcal{T}(x^*) = \lim_{n\to\infty} \mathcal{T}^n(x)$$

(see for example [148, page 36]). However, eventual contractivity is not a necessary condition. To show this, consider the function $f : R \mapsto R$, where $f(x) = x - 1$ if $x > 2$ and $f(x) = x/2$ if $x \leq 2$. It is easy to show that although $(f^n(x))$ is convergent to zero for any x, there is no N for which f^N is contractive in R.

The relation between contractivity of $\mathcal{T}$, eventual contractivity of $\mathcal{T}$, and the convergence of sequence (x_n) may be expressed as

$$\mathcal{T}\text{ contractive} \quad \underset{\not\Longleftarrow}{\Longrightarrow} \quad \mathcal{T}\text{ eventually contractive} \quad \underset{\not\Longleftarrow}{\Longrightarrow} \quad \forall x,\ (\mathcal{T}^n(x))\ \text{convergent to } x^*.$$

The following theorem provides a necessary condition for convergence which is similar, but not the same as, eventual contractivity.

Theorem: Let (X, d) be a complete metric space with $\mathcal{T} : X \mapsto X$. Also, assume that there is a point $x^* \in X$ such that for any point $x \in X$,

$$x^* = \lim_{n\to\infty} \mathcal{T}^n(x). \tag{3.37}$$

Then, given any $s \in R$, and for every $y_1, y_2 \in X$ there exists a K, such that for all $n > K$, $d(\mathcal{T}^n(y_1), \mathcal{T}^n(y_2)) < s\, d(y_1, y_2)$.

Proof: According to (3.37),

$$\forall \varepsilon > 0,\ \exists N > 0,\ \text{such that } n > N \ \Rightarrow\ d(\mathcal{T}^n(y_1), x^*) < \varepsilon \tag{3.38}$$

$$\forall \varepsilon > 0,\ \exists M > 0,\ \text{such that } n > M \ \Rightarrow\ d(\mathcal{T}^n(y_2), x^*) < \varepsilon \tag{3.39}$$

Now, let $\varepsilon = \frac{1}{2} s\, d(y_1, y_2)$ and let K be the maximum of M and N for this ε according to (3.38) and (3.39). Then

$$\forall n > K \quad , \quad d(\mathcal{T}^n(y_1), x^*) < \frac{1}{2} s\, d(y_1, y_2) \tag{3.40}$$

$$, \quad d(\mathcal{T}^n(y_2), x^*) < \frac{1}{2} s\, d(y_1, y_2). \tag{3.41}$$

Using the triangle inequality for metric d, we get

$$d(\mathcal{T}^n(y_1), \mathcal{T}^n(y_2)) \leq d(\mathcal{T}^n(y_1), x^*) + d(\mathcal{T}^n(y_2), x^*). \tag{3.42}$$

Combining (3.42) with (3.40) and (3.41) gives

$$d(\mathcal{T}^n(y_1), \mathcal{T}^n(y_2)) < \frac{1}{2} s\, d(y_1, y_2) + \frac{1}{2} s\, d(y_1, y_2), \tag{3.43}$$

and, therefore,

$$\forall n > K,\ \ d(\mathcal{T}^n(y_1), \mathcal{T}^n(y_2)) < s\, d(y_1, y_2). \ \ \square \tag{3.44}$$

3.9.2 Stability

The notions of contractivity and eventual contractivity are the main basis for analysis on convergence of attractor decoders in the literature on fractal coding. However, from the systems point of view, the convergence of the sequence (x_n) is extensively studied in terms of the *stability* of its generating system. In this section, we bring some definitions and results from the systems theory. For further details, the reader may see [149]. In what follows, $\mathbf{x}$ is assumed to be an N-dimensional vector and $\mathbf{A}$ is represented by an $N \times N$ matrix.

Theorem: The solutions to $\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t)$ form an N-dimensional vector space.

Definition: A *fundamental matrix* $\mathbf{\Psi}$ of $\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x}$ is an $N \times N$ matrix whose columns are linearly independent solutions of $\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}$. It is easy to show that $\dot{\mathbf{\Psi}} = \mathbf{A}(t)\mathbf{\Psi}$.

Definition: The matrix $\mathbf{\Phi}(t, t_0)$ defined as

$$\mathbf{\Phi}(t, t_0) \overset{\text{def}}{=} \mathbf{\Psi}(t)\mathbf{\Psi}^{-1}(t_0) \qquad \forall t, t_0 \in (-\infty, \infty)$$

is called the *state transition matrix* of $\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x}$, where $\mathbf{\Psi}$ is *any* fundamental matrix of $\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x}$.

It can be proven that $\mathbf{\Phi}(t, t_0)$ is independent from the choice of $\mathbf{\Psi}$. We denote the solution to (3.9) with the initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$ and input $\mathbf{u}_0$ (zero for $t < 0$) by $\boldsymbol{\phi}(t, t_0, \mathbf{x}_0, \mathbf{u}_0)$.

Theorem: For (3.10)

$$\begin{aligned} \mathbf{\Phi}(t, t_0) &= e^{\mathbf{A}(t-t_0)} = \mathbf{\Phi}(t - t_0) \\ \boldsymbol{\phi}(t, t_0, \mathbf{x}_0, \mathbf{u}) &= e^{\mathbf{A}(t-t_0)}\mathbf{x}_0 + \int_{t_0}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau. \end{aligned}$$

Definition: At time t_0 a system is called *relaxed* iff the output for $t \geq t_0$ is excited only by $\mathbf{u}(t)$ for $t \geq t_0$.

Definition: A system is called *bounded-input-bounded-output stable* or *BIBO stable* if its output to any bounded input is bounded.

Definition: A state $\mathbf{x}_e$ is called an *equilibrium state* iff

$$\forall t \geq t_0 \qquad \mathbf{x}_e = \boldsymbol{\phi}(t, t_0, \mathbf{x}_e, \mathbf{0}).$$

Definition: We call an equilibrium state $\mathbf{x}_e$ *stable in the sense of Lyapunov* at t_0 iff

$$\forall \varepsilon > 0 \;\; \exists \delta > 0, \qquad ||\mathbf{x}_0 - \mathbf{x}_e|| \leq \delta \;\; \Rightarrow \;\; ||\boldsymbol{\phi}(t, t_0, \mathbf{x}_0, \mathbf{0}) - \mathbf{x}_e|| \leq \varepsilon$$

where δ depends on ε and t_0.

An equilibrium state $\mathbf{x}_e$ is also called *uniformly stable i.s.L.* (in the sense of Lyapunov) for $t \geq t_0$ iff

$$\forall \varepsilon > 0 \ \exists \delta > 0, \qquad ||\mathbf{x}_0 - \mathbf{x}_e|| \leq \delta \ \Rightarrow \ ||\boldsymbol{\phi}(t, t_0, \mathbf{x}_0, \mathbf{0}) - \mathbf{x}_e|| \leq \varepsilon$$

where δ only depends on ε and not on t_0.

Definition: An equilibrium state $\mathbf{x}_e$ is called *asymptotically stable* at t_0 iff it is stable i.s.L. at t_0 and for any initial state $\mathbf{x}_0$ close to $\mathbf{x}_e$ we have $\lim_{t\to\infty} \mathbf{x} = \mathbf{x}_e$. More precisely, $\forall t_1$, $\exists \alpha > 0$ such that $||\mathbf{x}(t_1) - \mathbf{x}_e|| \leq \alpha$ implies

$$\forall \varepsilon > 0 \ \exists \, T > 0, \ \ \forall t \geq t_1 + T, \ \ ||\boldsymbol{\phi}(t, t_1, \mathbf{x}(t_1), \mathbf{0}) - \mathbf{x}_e|| \leq \varepsilon.$$

In the above definition, if $\mathbf{x}_e$ is uniformly stable i.s.L. for $t \geq t_0$ and T is independent of t_1, then $\mathbf{x}_e$ is called *uniformly* asymptotically stable for $t \geq t_0$.

Linear systems have the property that if they have one stable equilibrium state they will be *globally stable* or *stable in the large.*

Definition: A linear dynamical equation is called *totally stable* iff both the output and the state variables are bounded for any initial state and any bounded input.

Theorem: If a linear time-invariant dynamical equation is both controllable and observable,[2] then the following statements are equivalent:

(1) The system is totally stable.

(2) The system is BIBO stable.

(3) The system is asymptotically stable.

(4) The real component of all the eigenvalues of $\mathbf{A}$ are negative.

[2]For the definition of controllability and observability, the reader may refer to [149] or [150]. Here, we only mention that in (3.10) and (3.11), in the case of $\mathbf{B} = \mathbf{I}$ and $\mathbf{C} = \mathbf{I}$, the system is both controllable and observable.

So, the system described in (3.5) and (3.6) is stable iff the eigenvalues of $\mathbf{A}$ have negative real parts (i.e., are located in the left-hand side of the complex plane).

The concept of stability for continuous-time systems can be easily applied to discrete-time systems. However, the condition of stability for these systems is different. The system described in (3.5) and (3.6) is stable iff the eigenvalues of $\mathbf{A}$ have magnitudes less than 1, i.e., are located within the unit circle [149]. If $\mathbf{A}$ is an $N \times N$ matrix, and $\lambda_1, \lambda_2, \ldots, \lambda_N$ are its eigenvalues, then

$$\rho(\mathbf{A}) = \max_{1 \le i \le N} |\lambda_i|$$

is called the *spectral radius* of $\mathbf{A}$. Hence, the stability of the system of (3.5), (3.6) may be expressed as

$$\rho(\mathbf{A}) < 1.$$

3.10 Convergence

The whole operation of attractor coding is based on the convergence of the sequence generated by the decoder and makes the analysis of convergence of the decoder an important issue in attractor coding. However, the *rate* of convergence is also of great importance because

(1) it determines the speed of decoder, and

(2) it determines how much additional error is introduced into the decoded signal by the decoder, as can be seen from (3.31).

In the literature, the rate of convergence of attractor decoders is typically analyzed in terms of the contractivity factor of the operator $\mathcal{T}$, and sometimes in terms of its *Lipschitz constant.* For the linear time-invariant case, this is reduced to $\|\mathbf{A}\|$. However, in this work, we propose that using the eigenvalues of $\mathbf{A}$ provides a more powerful tool for analysis of convergence in attractor decoders. Eigenvalues of matrix $\mathbf{A}$ have also been used for analysis of convergence by Lundheim [151] and Hürtgen and Simon [152]. However, computing the norm of $\mathbf{A}$ for some norms is easier than computing the eigenvalues of $\mathbf{A}$. For $\|.\|_1$, $\|.\|_2$, and $\|.\|_\infty$, it can be shown that

$$\begin{aligned}
||\mathbf{A}||_1 &= \max_j \left(\sum_{i=1}^{N} |a_{i,j}| \right) \\
||\mathbf{A}||_2 &= \sqrt{\rho(\mathbf{A}^T\mathbf{A})} \\
||\mathbf{A}||_\infty &= \max_i \left(\sum_{j=1}^{N} |a_{i,j}| \right)
\end{aligned}$$

and for any norm,

$$\rho(\mathbf{A}) = \lim_{n\to\infty} ||\mathbf{A}^n||^{\frac{1}{n}} \leq \cdots \leq ||\mathbf{A}^n||^{\frac{1}{n}} \leq \cdots \leq ||\mathbf{A}^2||^{\frac{1}{2}} \leq ||\mathbf{A}||.$$

In the next section, we will investigate methods for computing the eigenvalues of $\mathbf{A}$ using flow graphs.

Convergence, just like eigenvalues, is independent of the norm being used. In terms of convergence, all norms are equivalent. There are $\mathbf{A}$'s for which some norms are greater than 1 and some are less than 1. Although $\|\mathbf{A}\| < 1$ is a sufficient condition for convergence, the reverse is not true. And, also, small $\|\mathbf{A}\|$ guarantees a fast convergence, but again, in general, the reverse is not true. On the contrary, $\rho(\mathbf{A}) < 1$ is both a necessary and sufficient condition for convergence and $\rho(\mathbf{A})$ directly dictates the speed of convergence in the long run as can be expected from (3.22). In fact, the long-term behavior of discrete-time systems is typically determined by the eigenvalue with the largest magnitude, and its corresponding eigenvector. This eigenvalue is called the *dominant* eigenvalue. In the case of continuous systems, the dominant eigenvalue is the one with the largest real part.

Regarding eventual contractivity, this author is not aware of any practical method for its computation based on its definition.

3.11 Graph Theoretical Approach

In the previous sections, we established the importance of eigenvalues of **A** in analysis of the attractor coder system described by dynamical Equations (3.5) and (3.6).

In fractal coding, image pixels are encoded by approximating them with other pixels in the same image. The structure of matrix **A** is determined by the pattern and level of interdependence of image pixels. However, each image pixel is typically dependent on only a small number of other pixels, and hence, the matrix **A** is very sparse. These dependencies may be better analyzed if represented by a flow graph. In such a representation, pixels are represented by vertices (nodes) and their dependencies by weighted arcs (arrows). The resulting flow graph also provides a representation of matrix **A**. Interestingly, many features of **A**, including its eigenvalues, have graph theoretical equivalents. In this section, we investigate these equivalencies and their implications on properties of the attractor coding system. This study sheds new light onto the difficult problem of analysis of convergence of attractor coders of images.

In Section 3.11.1, some basic terminology in graph theory is briefly introduced, which will help us in developing the concepts of the following sections. In Section 3.11.2, we will study different patterns of dependencies and their interpretations from the viewpoint of graph theory, structure of matrix **A**, eigenvalues of **A**, and properties of the attractor coder in terms of causality and stability. The two theorems presented in Sections 3.11.2.5 and 3.11.2.6 will cover the most general cases studied here and constitute the highlights of this section.

3.11.1 Concepts in graph theory

A *graph* is a pair of sets (V, E) where E is a set of unordered pairs of members of V. A graph is called *finite* if both V and E are finite. In this dissertation, we will only study finite graphs and refer to them simply as graphs. The elements of V are called *vertices* and V is called the *vertex-set* of the graph. Also, the elements of E are called *edges* and E is called the *edge-set* of the graph. If $e = \{a, b\} \in E$, then a and b are called the *end*

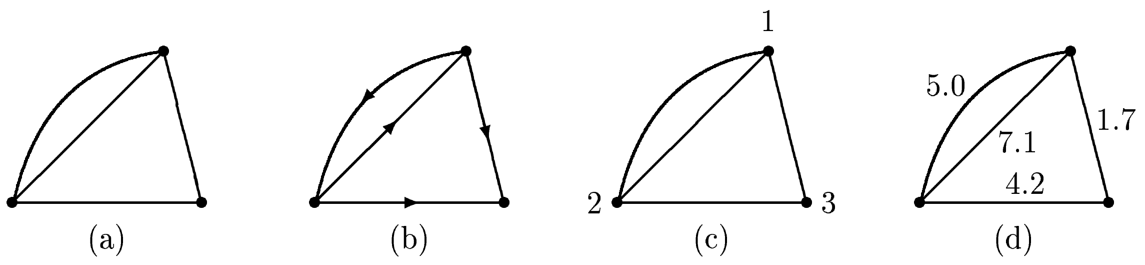


Figure 3.10 Representation of (a) a graph, (b) a digraph, (c) a labeled graph, and (d) a weighted graph.

points of e. We denote e by ab and write $e = ab$, and e is said to *join* vertices a and b and we call a and b *adjacent* vertices. In such case, a and b are also called *neighbors* of each other.

If E is a set of *ordered* pairs instead of unordered pairs, then the graph is called a *directed graph* or a *digraph* and the members of E are called *arcs*. If $e = (a, b) \in E$, we again denote e by ab and say e joins vertex a to vertex b.

Graphs may be represented by diagrams like Figure 3.10(a) where vertices are represented by points and edges are represented by curves connecting their end-point vertices. Similarly, digraphs may be represented as in Figure 3.10(b)

A *labeled graph* is a graph where the vertices are numbered 1 to N where N is the number of vertices in V (Figure 3.10(c)). If a number is assigned to each edge of a graph, the graph is called a *weighted graph* (Figure 3.10(c)). A labeled weighted digraph is called a *flow graph.* The *degree* of a vertex v in a graph, denoted by $d(v)$, is the number of edges that have v as an endpoint. The *in-degree* and the *out-degree* of a vertex v in a *digraph* G are the number of arcs $e = (u, v) \in E(G)$, and $e = (v, u) \in E(G)$, respectively, and are denoted by $d^-(v)$ and $d^+(v)$.

A graph G is called *regular* of degree K if

$$\forall v \in V(G), \quad d(v) = K.$$

A digraph G is called *regular* of degree K if

$$\forall v \in V(G), \quad d^+(v) = d^-(v) = K.$$

For a graph G, we may also denote the vertex-set by $V(G)$ and the edge-set by $E(G)$. A graph H is called a *subgraph* of a graph G if $V(H)$ and $E(H)$ are subsets of $V(G)$ and $E(G)$, respectively. In the special case of $V(G) = V(H)$, H is called a *spanning subgraph* of G. An *isolated vertex* is a vertex which is not adjacent to any other vertex in the graph.

A *path* is a graph (V, E) where V and E have the form

$$\begin{aligned} V &= \{v_1, v_2, \ldots, v_k\}, \\ E &= \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{K-1}, v_K\}\}. \end{aligned}$$

A path is fully described by its list of vertices or edges. Note that a vertex cannot be repeated in the list. A *cycle* is a path of the above form with the additional edge $\{v_K, v_1\}$. The *length* of a path or a cycle is the cardinal number of its edge-set.

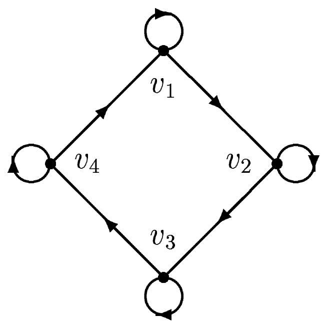
A *directed path* and a *directed cycle* are digraphs defined similarly to a path and a cycle, except that edges are ordered pairs instead of unordered pairs.

In a directed cycle C, the product of the weights of all of the arcs is called the *loop gain* of the directed cycle and is denoted by $\ell(C)$ or ℓ when C is specified. A *self-loop* is a cycle or a directed cycle of length 1. Two cycles or directed cycles are called *touching* if the intersection of their vertex-sets is not empty.

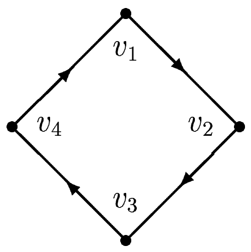
A digraph is called *acyclic* if it contains no directed cycles. A graph is called *connected* if every two distinct vertices in the graph are connected by a path.

A digraph is called *strongly connected* if every two distinct vertices u, w in the graph are connected by a directed path from u to w and by a directed path from w to u. A connected subgraph of a graph is called a *component* of the graph if it contains the maximal number of edges. For a digraph G, a strongly connected subgraph with the maximal number of arcs is called a *strong component* or *strongly-connected component* of G. Isolated vertices of the G are also called strong components of G.

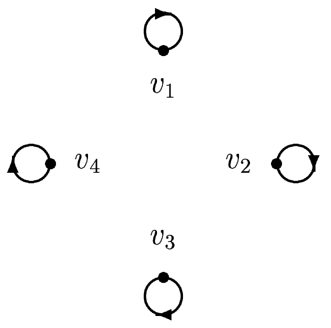
An *n-factor* of a digraph G is a spanning subgraph of G which is regular of degree n. For example, Figure 3.11(a) shows a digraph and Figures 3.11(b) and 3.11(c) show its 1-factors. Also, the graph of Figure 3.10 has no 1-factors.



(a)

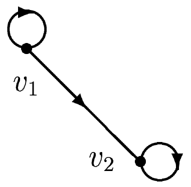


(b)

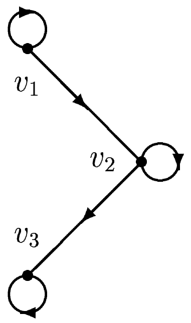


(c)

Figure 3.11 (a) A digraph, and (b), (c) its 1-factors.



(a)



(b)

Figure 3.12 Sectional subgraphs of Figure 3.11(a) with (a) $V_s = \{v_1, v_2\}$, and (b) $V_s = \{v_1, v_2, v_3\}$.

For a digraph G, with the vertex-set V, let $V_s \subseteq V$. The *sectional subgraph* of G associated with V_s, and denoted by $G[V_s]$, is the subgraph of G whose vertex-set is V_s and its edge-set is the set of all edges in G which connect any pair of vertices in V_s. Figure 3.12 shows two sectional subgraphs of the digraph of Figure 3.11(a). Given a square matrix $\mathbf{A}$, with elements $a_{i,j}$, the *Coats graph* [153] of $\mathbf{A}$ is a flow graph which has an arc with weight $a_{i,j}$ from vertex labeled j to vertex labeled i iff $a_{i,j} \neq 0$. If G is a Coates graph of $\mathbf{A}$, for simplicity we call G the graph of $\mathbf{A}$, and call $\mathbf{A}$ the matrix of G.

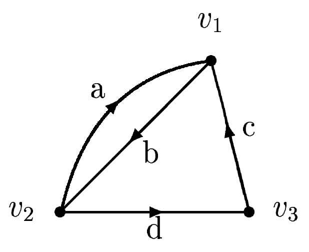


Figure 3.13 Coates graph of matrix $\mathbf{A}$ in (3.45).

Figure 3.13 shows the Coates graph of matrix

$$\mathbf{A} = \begin{bmatrix} 0 & a & c \\ b & 0 & 0 \\ 0 & d & 0 \end{bmatrix}. \tag{3.45}$$

The *Mason graph* [153] of a $\mathbf{A}$ is the Coates graph of $\mathbf{A}+\mathbf{I}$.

3.11.2 System analysis using flow graphs

In this section, we will analyze attractor coders with state equation of the form

$$\mathbf{x}_{n+1} = \mathbf{A}_{N\times N}\mathbf{x}_n + \mathbf{B}_{N\times 1} \tag{3.46}$$

using graph theory. In the following sections, we will refer to the system whose state equation is of the form (3.46) as the system of $\mathbf{A}$ or as the system of flow graph of $\mathbf{A}$. We also assume that all elements of $\mathbf{A}$ are real.

3.11.2.1 Single-path flow graph

Consider a flow graph made up of a single directed path of length $N-1$ as shown in Figure 3.14(a). The matrix of this flow graph is

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ a_{2,1} & 0 & \cdots & 0 & 0 \\ 0 & a_{3,2} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{N,N-1} & 0 \end{bmatrix}_{N\times N},$$

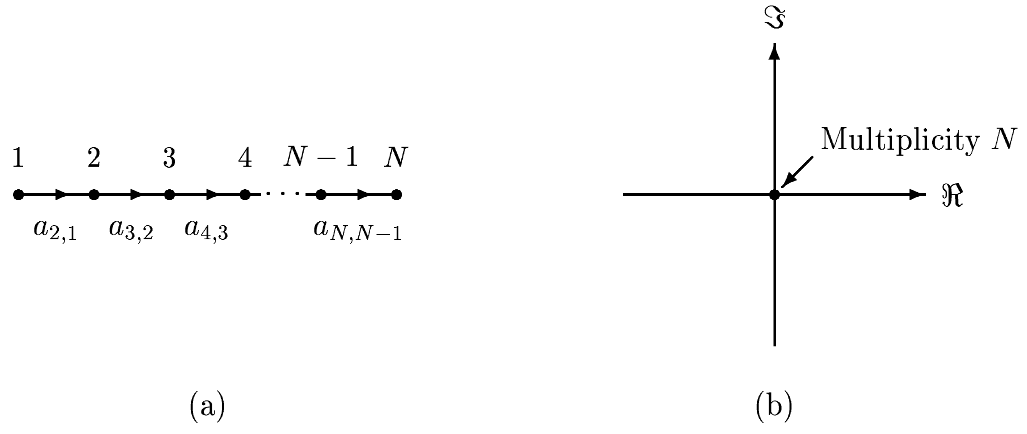


Figure 3.14 (a) A flow graph made up of a single directed path, and (b) location of eigenvalues of its matrix.

where all of the shown $a_{i,j}$ are nonzero. $\mathbf{A}$ has an eigenvalue of 0 with multiplicity N and $\rho(\mathbf{A}) = 0$; therefore, its system is stable. Furthermore, from the flow graph of $\mathbf{A}$, it is intuitionally clear that the system of $\mathbf{A}$ reaches its steady-state at most after N iterations. This can also be verified by showing that $\mathbf{A}^N = \mathbf{0}$ and using (3.22).

Note that any permutation[3] of $\mathbf{A}$ has the same eigenvalues and stability. Its graph will also be same as Figure 3.14(a), except for labeling. Also, the resulting system is not spatially causal, but it can be made causal by changing the order of the elements of $\mathbf{x}$ in (3.46), which is equivalent to a spatial reordering of the signal being coded.

Note that for such a system, during the decoding one needs not compute the value of all vertices at each iteration. The decoder may first compute only the value of state variable corresponding to vertex 1, then compute only vertex 2 from the value of vertex 1, and so on. Computationally, evaluation of the values of all vertices in this way has the same complexity as computing $\mathbf{A}\mathbf{x}_n + \mathbf{B}$ only once. This method of decoding causally encoded signals will be discussed further in later sections.

[3]A permutation of matrix $\mathbf{A}_{N\times N}$ is a matrix $\mathbf{PAP}^T$, where $\mathbf{P}$ is an $N \times N$ matrix obtained by reordering rows of $\mathbf{I}_{N\times N}$. Note that $\mathbf{A}$ and its permutation have the same eigenvalues, however, their eigenvectors are different only by row permutations.

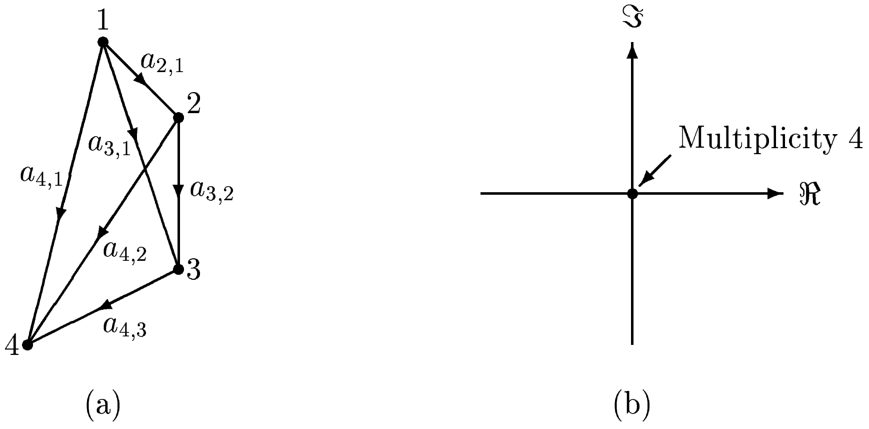


Figure 3.15 (a) An acyclic flow graph, and (b) location of eigenvalues of its matrix.

3.11.2.2 Acyclic flow graph

Consider an acyclic flow graph for which the vertices are labeled such that for each arc uw, the label of u is less than the label of w. Such a flow graph is shown in Figure 3.15(a). The matrix of such flow graph is of the form

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ a_{2,1} & 0 & 0 & \cdots & 0 & 0 \\ a_{3,1} & a_{3,2} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{N,1} & a_{N,2} & a_{N,3} & \cdots & a_{N,N-1} & 0 \end{bmatrix}_{N\times N}, \tag{3.47}$$

which is a lower triangular matrix with all diagonal elements being zero. Note that the signal-path graph of Section 3.11.2.1 is a special case of this graph. Let us denote the length of the directed path in G with the greatest length by L, where $L \leq N-1$. The system of this graph reaches its steady-state at most in $L+1$ iterations and

$$\mathbf{A}^{L+1} = \mathbf{0}$$

$$\rho(\mathbf{A}) = \lambda_1 = \lambda_2 = \cdots = \lambda_N = 0.$$

Any system whose state transition matrix $\mathbf{A}$ can be reduced to the form of (3.47) by a permutation, also has the same stability and convergence properties.

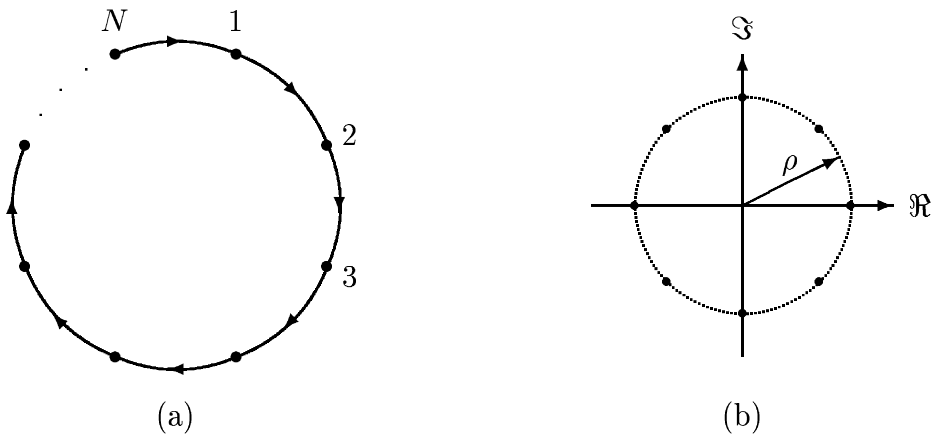


Figure 3.16 (a) A regular flow graph of degree one, and (b) location of eigenvalue of its matrix.

3.11.2.3 Single-cycle flow graph

Let us consider a regular flow graph of degree 1. Such a digraph consists of a single cycle. Figure 3.16(a) shows such a flow graph with more than one vertex and Figure 3.17(a) shows it with only one vertex. The matrix of the former case has the form

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & \cdots & 0 & a_{1,N} \\ a_{2,1} & 0 & \cdots & 0 & 0 \\ 0 & a_{3,2} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{N,N-1} & 0 \end{bmatrix}_{N\times N},$$

which is a circular matrix with one nonzero element per row. In the latter case, we have

$$\mathbf{A} = [a_{1,1}].$$

In either case, $\mathbf{A}$ represents a spatially-noncausal system. If we denote the loop gain of the directed cycle by ℓ, i.e.,

$$\ell \overset{\text{def}}{=} (a_{N,N-1})(a_{N-1,N-2}) \cdots (a_{3,2})(a_{2,1})(a_{1,N})$$

then, it is easy to show that

$$\mathbf{A}^N = \ell\, \mathbf{I}_{N\times N} \tag{3.48}$$

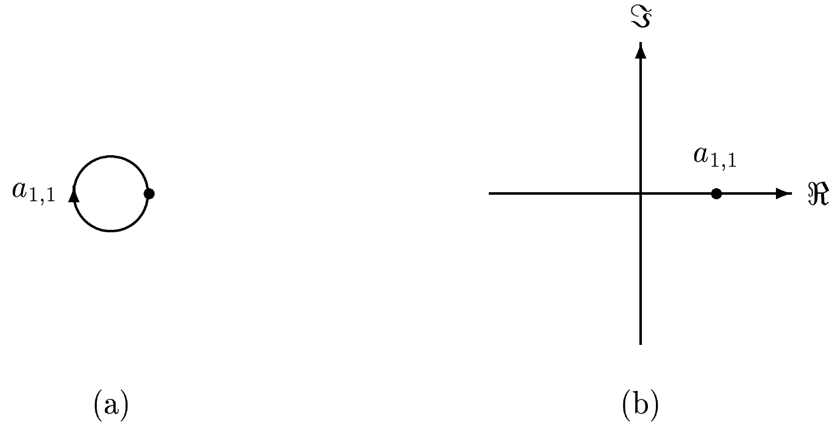


Figure 3.17 (a) A one-vertex regular flow graph of degree one, and (b) location of eigenvalues of its matrix.

and the characteristic equation of $\mathbf{A}$ is

$$\lambda^N - \ell = 0,$$

which has the solutions

$$\lambda_i = \sqrt[N]{|\ell|}\, e^{j\left(2\pi i + \angle \ell\right)/N} \qquad i = 0, 1, \ldots, N-1. \tag{3.49}$$

Hence, the eigenvalues of $\mathbf{A}$ are uniformly distributed on a circle of diameter $\sqrt[N]{|\ell|}$ with center $(0, 0)$ in the complex plane. Figures 3.16(b) and 3.17(b) show this for $\ell > 0$ with $N > 1$ and $N = 1$. Therefore, we have

$$\rho(\mathbf{A}) = \sqrt[N]{|\ell|}.$$

The system converges iff $\ell < 1$, in which case, the system is guaranteed to reach its steady-state only at $n = \infty$. Combining (3.48) and (3.22) shows that

$$\mathbf{x}_{kN} - \mathbf{x}_\infty = \ell^k(\mathbf{x}_0 - \mathbf{x}_\infty) \qquad k = 0, 1, 2, \ldots$$

i.e., after each N iterations the difference between each element of $\mathbf{x}_n$ and $\mathbf{x}_\infty$ is multiplied by ℓ.

3.11.2.4 Multiple-component flow graph

In this section, we analyze properties of a digraph G made up of K nonempty components. As there cannot be any arc between vertices of two different components of the flow graph, the matrix of G can always be made block diagonal in the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{2,2} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{3,3} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_{K,K} \end{bmatrix}_{N \times N},$$

by proper labeling of vertices in G. If we denote the collection of eigenvalues (including multiplicities) of any matrix $\mathbf{M}$ by $\Lambda(\mathbf{M})$, then it can be shown that

$$\Lambda(\mathbf{A}) = \bigcup_{i=1}^{K} \Lambda(\mathbf{A}_{i,i}). \tag{3.50}$$

and, therefore,

$$\rho(\mathbf{A}) = \max_{1 \le i \le K} \rho(\mathbf{A}_{i,i}). \tag{3.51}$$

Matrix $\mathbf{A}$ and its flow graph represent a system with isolated subsystems which do not have any interdependence while their internal structure is specified by $\mathbf{A}_{i,i}$s. Such a system is stable iff all of its subsystems are stable, as may be seen from (3.51).

3.11.2.5 Not-strongly-connected flow graph

We first introduce the concept of reducibility in matrices. A matrix $\mathbf{M}$ is called *reducible*[4] iff there exists some permutation matrix $\mathbf{P}$ such that

$$\mathbf{PMP}^T = \begin{bmatrix} \mathbf{M}_{1,1} & \mathbf{0} \\ \mathbf{M}_{1,2} & \mathbf{M}_{2,2} \end{bmatrix},$$

where $\mathbf{M}_{1,1}$ and $\mathbf{M}_{2,2}$ are square matrices. $\mathbf{M}$ is said to be *irreducible* if it is not reducible. A reducible matrix $\mathbf{M}$ represents a system which can be partitioned into two subsystems

[4]Some authors define reducibility only for nonnegative matrices [145, page 324], but here we do not impose nonnegativity.

with matrices $\mathbf{M}_{1,1}$ and $\mathbf{M}_{2,2}$, whose interrelation is only one way from system of $\mathbf{M}_{1,1}$ to system of $\mathbf{M}_{2,2}$ and is determined by $\mathbf{M}_{1,2}$.

We note that a flow graph is strongly connected iff its matrix is irreducible. On the other hand, if a flow graph G is not strongly connected, then it has $K \geq 2$ strong components. If $\mathbf{A}$ is the matrix of G, then it can be written (ignoring permutation) in a block lower triangular form

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{A}_{K-1,1} & \mathbf{A}_{K-1,2} & \cdots & \mathbf{A}_{K-1,K-1} & \mathbf{0} \\ \mathbf{A}_{K,1} & \mathbf{A}_{K,2} & \cdots & \mathbf{A}_{K,K-1} & \mathbf{A}_{K,K} \end{bmatrix}_{N\times N},$$

where $\mathbf{A}_{1,1}$, $\mathbf{A}_{2,2}, \ldots, \mathbf{A}_{K,K}$ are irreducible square matrices. Furthermore, $\mathbf{A}_{1,1}$, $\mathbf{A}_{2,2}, \ldots$, $\mathbf{A}_{K,K}$ are matrices of strong components of G. In this case, the eigenvalues of $\mathbf{A}$ and $\mathbf{A}_{1,1}$, $\mathbf{A}_{2,2}, \ldots, \mathbf{A}_{K,K}$ are again related by (3.50). These results may be summarized in the following theorem [153, page 208][154].

Theorem: If G is the Coates graph of a matrix $\mathbf{A}$, then the eigenvalues of $\mathbf{A}$ are the union (including multiplicities) of the eigenvalues of submatrices of $\mathbf{A}$ corresponding to strong components of G.

Here, matrix $\mathbf{A}$ and its flow graph G represent a system which can be partitioned into interconnected subsystems whose interdependence is spatially semicausal. This decomposition is maximal, i.e., the subsystems cannot be partitioned further this way. The submatrices $\mathbf{A}_{i,i}$ represent the internal structure of these subsytems, while $\mathbf{A}_{i,j}$, $i \neq j$ represent their interdependence. This system is again stable iff all of its subsytems are *internally* stable. The one-way interconnection of the subsytems does not affect the stability of neither the whole system nor its subsytems.

Due to the existence of directed cycles in the subsystems (as the subsystems have graphs which are strongly connected), the system is guaranteed to reach its steady-state only after infinite iterations. The only exception is when all the strong components of G are isolated vertices with no self-loops.

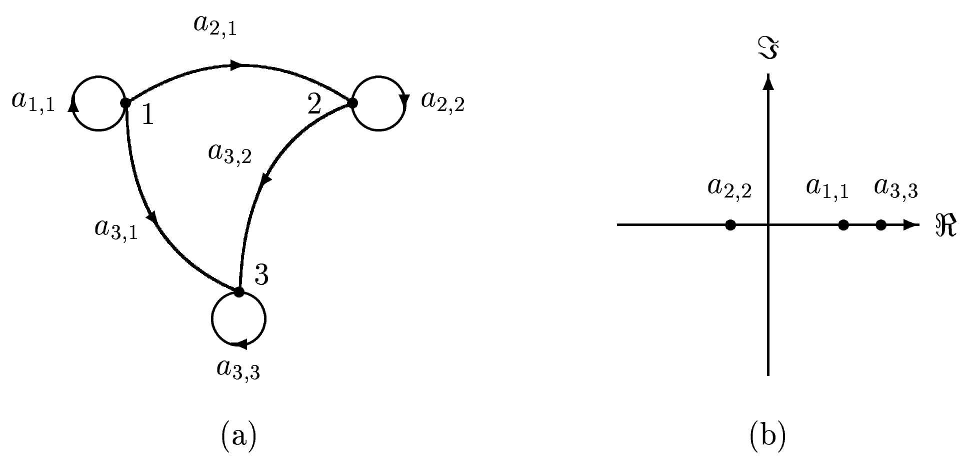


Figure 3.18 (a) Flow graph of a spatially semicausal system, and (b) the location of its eigenvalues in the complex plane.

Example: *Spatially semicausal system*

Consider a flow graph G of the form shown in Figure 3.18 (weights are not shown). This flow graph has the lower triangular matrix

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & 0 & 0 \\ a_{2,1} & a_{2,2} & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}.$$

This flow graph is not strongly connected but has three strongly connected components, which are the vertices and their self-loops. The eigenvalues of $\mathbf{A}$ are the eigenvalues of these strong components, namely $a_{1,1}$, $a_{2,2}$, and $a_{3,3}$. The system of G is stable iff $|a_{i,i}| < 1$, for $i = 1, 2, 3$.

Example: *Multiple isolated cycles with acyclic interdependencies*

Figure 3.19 shows a flow graph G. This flow graph is not strongly connected; for example, there is no directed path from node 9 to node 3. Flow graph G is made up of three strong components, each of which is a directed cycle. Note that, in general, these strong

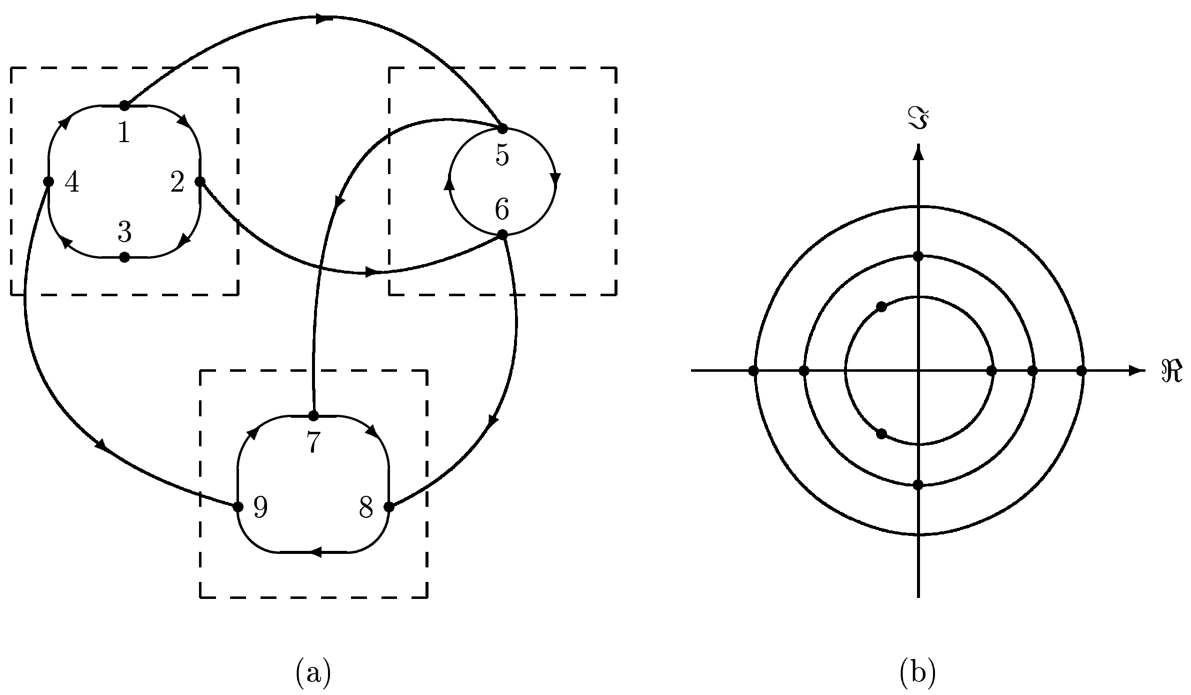


Figure 3.19 (a) A flow graph which is not strongly connected made up of three strong components, and (b) location of its eigenvalues on the complex plane.

components need not be only directed cycles. The matrix **A** of this flow graph is

$$
\mathbf{A} = \left[\begin{array}{cccc|cc|ccc}
0 & 0 & 0 & a_{1,4} & 0 & 0 & 0 & 0 & 0 \\
a_{2,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & a_{3,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & a_{4,3} & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
a_{5,1} & 0 & 0 & 0 & 0 & a_{5,6} & 0 & 0 & 0 \\
0 & a_{6,2} & 0 & 0 & a_{6,5} & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & a_{7,5} & 0 & 0 & 0 & a_{7,9} \\
0 & 0 & 0 & 0 & 0 & a_{8,6} & a_{8,7} & 0 & 0 \\
0 & 0 & 0 & a_{9,4} & 0 & 0 & 0 & a_{9,8} & 0
\end{array}\right]_{9\times 9},
$$

which has the form

$$
\mathbf{A} = \begin{bmatrix}
\mathbf{A}_{1,1} & \mathbf{0} & \mathbf{0} \\
\mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \mathbf{0} \\
\mathbf{A}_{3,1} & \mathbf{A}_{3,2} & \mathbf{A}_{3,3}
\end{bmatrix}_{9\times 9},
$$

where $\mathbf{A}_{1,1}$, $\mathbf{A}_{2,2}$, and $\mathbf{A}_{3,3}$ are matrices representing the three strong component of G, each of which is a directed cycle. $\mathbf{A}_{2,1}$, $\mathbf{A}_{3,1}$, and $\mathbf{A}_{3,2}$ are rectangular matrices representing the interconnection between the strong components or subsystems. Although the subsystems of this system are not spatially causal, their interdependence is semicausal.

According to the above theorem, the collection of eigenvalues of $\mathbf{A}$ is the union of eigenvalues of $\mathbf{A}_{1,1}$, $\mathbf{A}_{2,2}$, and $\mathbf{A}_{3,3}$. On the other hand, the eigenvalues of $\mathbf{A}_{1,1}$, $\mathbf{A}_{2,2}$, and $\mathbf{A}_{3,3}$ can easily be found from (3.49). Hence, the system is stable iff the magnitude of the loop gain of the directed cycle in each of the three subsystems is less than 1.

3.11.2.6 Touching cycles

The previous sections covered analyses of several types of matrices, their flow graphs, and their systems. Any flow graph G that contains no touching directed cycles can be analyzed, as in Section 3.11.2.3, if G contains a single, directed cycle, and as in Section 3.11.2.5, if it contains multiple directed cycles. On the other hand, touching cycles significantly complicate the eigenvalue problem. Yet, there are cases where the eigenvalues may still be found. We have already seen systems made up of subsytems that are interconnected causally. Here, we study subsystems that are interconnected in the form of a directed cycle to make a larger system.

Let $\mathbf{A}$ be a block circular matrix of the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{A}_{1,K} \\ \mathbf{A}_{2,1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{3,2} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_{K,K-1} & \mathbf{0} \end{bmatrix}_{N\times N},$$

where the shown zero diagonal submatrices are square. In other words, the number of columns in $\mathbf{A}_{i,j}$ is the same as the number of rows in $\mathbf{A}_{j,k}$. Figure 3.20 shows the flow graph of such a system. Note that $\mathbf{A}$ represents a spatially noncausal system.

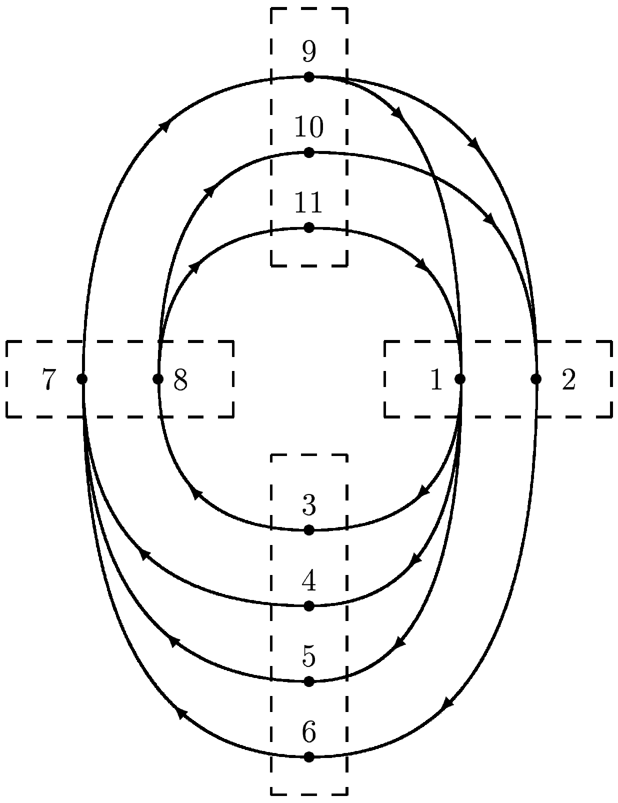


Figure 3.20 Flow graph of four subsystems with cyclic interdependence.

It can be shown that $\mathbf{A}^K$ is a block diagonal matrix of the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{R}_1 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_2 & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R}_{K-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{R}_K \end{bmatrix}_{N\times N}, \tag{3.52}$$

where

$$\begin{aligned} \mathbf{R}_1 &= \mathbf{A}_{1,K} \quad \mathbf{A}_{K,K-1} \quad \mathbf{A}_{K-1,K-2} \cdots \mathbf{A}_{3,2}\mathbf{A}_{2,1}, \\ \mathbf{R}_2 &= \mathbf{A}_{2,1} \quad \mathbf{A}_{1,K} \quad \mathbf{A}_{K,K-1} \quad \cdots \mathbf{A}_{4,3}\mathbf{A}_{3,2}, \\ &\vdots \\ \mathbf{R}_K &= \mathbf{A}_{K,K-1}\mathbf{A}_{K-1,K-2}\mathbf{A}_{K-2,K-3} \cdots \mathbf{A}_{2,1}\mathbf{A}_{1,K}. \end{aligned}$$

Note that $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_K$ are all square matrices with possibly different dimensions. The system of $\mathbf{A}^K$ is made up of K isolated subsystems and (3.50) may be used to compute the eigenvalues of $\mathbf{R}_K$.

Note that (3.52) shows that after each K iteration, the state of the subsystems is dependent only on their own values before the K iterations and not the other subsystems. The square matrices $\mathbf{R}_i$, $i = 1, 2, \ldots, K$ represent internal dependencies of the subsystems after the K iterations.

Lemma: Let $\mathbf{P} = \mathbf{M}_1\mathbf{M}_2$ and $\mathbf{Q} = \mathbf{M}_2\mathbf{M}_1$ be square matrices. Then, any nonzero eigenvalue of $\mathbf{P}$ is also an eigenvalue of $\mathbf{Q}$ (and vice versa).

Proof: Let λ be a nonzero eigenvalue of $\mathbf{P}$, and let $\mathbf{x}$ be its corresponding eigenvector, i.e.,

$$\mathbf{P}\mathbf{x} = \lambda\mathbf{x}, \tag{3.53}$$

where

$$\begin{aligned} \mathbf{x} &\neq \mathbf{0}, \\ \lambda &\neq 0. \end{aligned}$$

Equation (3.53) may be rewritten as

$$\mathbf{M}_1\mathbf{M}_2\mathbf{x} = \lambda\mathbf{x}.$$

Multiplying both sides by $\mathbf{M}_2$ from the left-hand side gives

$$\begin{aligned} \mathbf{M}_2\mathbf{M}_1\mathbf{M}_2\mathbf{x} &= \lambda\mathbf{M}_2\mathbf{x} \qquad (3.54) \\ \mathbf{Q}(\mathbf{M}_2\mathbf{x}) &= \lambda(\mathbf{M}_2\mathbf{x}), \end{aligned}$$

which shows that λ is an eigenvalue of $\mathbf{Q}$ with eigenvector $\mathbf{M}_2\mathbf{x}$ if $\mathbf{M}_2\mathbf{x} \neq \mathbf{0}$. We prove $\mathbf{M}_2\mathbf{x} \neq \mathbf{0}$ by contradiction. If $\mathbf{M}_2\mathbf{x} = \mathbf{0}$, then we multiply both sides of (3.54) by $\mathbf{M}_1$

$$\begin{aligned} \mathbf{M}_1\mathbf{M}_2\mathbf{M}_1\mathbf{M}_2\mathbf{x} &= \mathbf{0} \\ \mathbf{P}^2\mathbf{x} &= \mathbf{0} \\ \lambda^2\mathbf{x} &= \mathbf{0}, \end{aligned}$$

which cannot be true as neither λ nor $\mathbf{x}$ is zero. $\square$

Using this lemma, it is easy to show that the nonzero eigenvalues of $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_K$ are the same. Let J be the index of $\mathbf{R}_i$ with the smallest dimension, i.e.,

$$\dim(\mathbf{R}_J) = \min_{1\leq i\leq K} \dim(\mathbf{R}_i),$$

and let

$$D = \dim(\mathbf{R}_J).$$

D is the dimension of the smallest subsystem of the K subsystems. Then, $\mathbf{A}^K$ has, at most, D distinct nonzero eigenvalues and the rest of its eigenvalues are zero. Therefore, $\mathbf{A}$ has, at most, KD distinct eigenvalues located on J circles centered at the origin in the complex plane. For example, in the flow graph of Figure 3.20, just by looking at the flow graph, one can say $K = 4$, $D = 2$ and, that $\mathbf{A}$ has, at most, $KD = 8$ nonzero eigenvalues located on D circles centered at the origin. In general, in terms of the spectral radius of $\mathbf{A}$ we have

$$\rho(\mathbf{A}) = \sqrt[K]{\rho(\mathbf{R}_J)}.$$

The stability of this system can be determined from interdependence of the state elements of the smallest subsystem after K iterations. Using an argument similar to that of Section 3.11.2.1, it can be shown that it is possible to apply K iterations of $\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{B}$ in a manner that is computationally equivalent to computing $\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{B}$ only once.

3.11.2.7 The general case

In terms of stability, there are methods for determining the stability of a system from the coefficients of its characteristic equation without solving the characteristic equation (e.g., Section 8-6 in [149] or Section 6-3 in [155]). On the other hand, the coefficients of the characteristic equation have graph theoretical interpretations in terms of directed cycles of the Coates graph of matrix **A**. However, combining the two does not seem to give any simple and intuitional graph theoretical interpretation of the concept of stability.

Here, we bring only a theorem describing the relationship between coefficients of the characteristic equation of **A** and directed cycles in its Coates graph [153, pages 206–210]. **Theorem**: Let **A** be an $N \times N$ matrix and G be its Coates graph. Let $S_{i,1}, S_{i,2}, \ldots, S_{i,U_i}$ for $1 \leq i \leq N$ be the i-vertex sectional subgraphs of G. Let $f_{i,j,1}, f_{i,j,2}, \ldots, f_{i,j,F_{i,j}}$ for $1 \leq i \leq N$, $1 \leq j \leq U_i$ be the 1-factors of $S_{i,j}$. For any f, let $c(f)$ denote the number of cycles in f, and let $p(f)$ denote the product of the weights of all the arcs in f. Then,

$$|\lambda \mathbf{I} - \mathbf{A}| = \lambda^N + \sum_{i=1}^{N} \lambda^{N-i} \left(\sum_{j=1}^{U_i} \sum_{k=1}^{F_{i,j}} (-1)^{c(f_{i,j,k})} p(f_{i,j,k}) \right). \tag{3.55}$$

Example: Let

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

The Coates graph of **A**, its i-index sectional subgraphs, and the 1-factors of these sectional subgraphs are shown in Figure 3.21. Using (3.55) and with the help of Figure 3.21 we can write

$$\begin{aligned} |\lambda \mathbf{I} - \mathbf{A}| &= \lambda^2 + \lambda^1[(-1)a + (-1)d] + \lambda^0[(-1)^2 ad + (-1)^1 bc] \\ &= \lambda^2 - (a+d)\lambda + (ad - bc). \end{aligned}$$

3.11.3 Spatial causality

In this section, we study properties that are related to spatial causality of the system and may refer to spatial causality simply as causality. In this dissertation, we call a system spatially causal iff some permutation of **A** has zero elements, on and above the

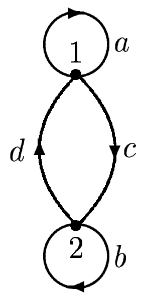
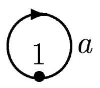
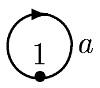
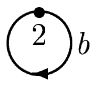
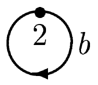
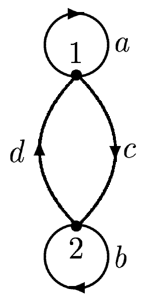
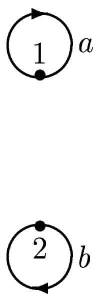
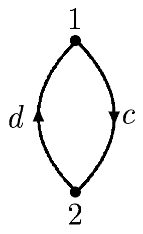
Graph of **A**		i-vertex sectional subgraphs of **A**		1-factors of i-vertex sectional subgraphs of **A**			
	$i = 1$	$j = 1$		$k = 1$		$c = 1$	$p = a$
		$j = 2$		$k = 1$		$c = 1$	$p = d$
	$i = 2$	$j = 1$		$k = 1$		$c = 2$	$p = ad$
				$k = 2$		$c = 1$	$p = bc$

Figure 3.21 A flow graph, its i-index sectional subgraphs, and the 1-factors of these sectional subgraphs for evaluating (3.55).

diagonal, i.e., is lower triangular with zero diagonal elements, or equivalently, if the flow graph of $\mathbf{A}$ is acyclic.

We also call a system spatially semicausal if some permutation of $\mathbf{A}$ is lower triangular. This happens iff the only cycles in the flow graph of $\mathbf{A}$ are self-loops.

In Section 3.11.2.2, we saw that for any spatially causal system with the longest directed path of length $L \leq N-1$, the system reaches its steady-state in $L+1$ iterations. In other sections, we also saw that for some cyclic graphs the system is guaranteed to reach its steady-state only after infinitely many iterations. This brings up the question of whether only spatially-causal systems may reach their steady-state in a finite number of iterations. In other words, we know that spatial-causality is a sufficient condition for finite-iteration convergence, but is this a necessary condition? To answer this question, we first review some concepts from matrix theory.

We know that for any $N \times N$ matrix $\mathbf{A}$, there are linearly independent vectors $\mathbf{v}_1$, $\mathbf{v}_2, \ldots, \mathbf{v}_N$ (which are called generalized eigenvectors of $\mathbf{A}$), such that for

$$\mathbf{Q} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N],$$

we have

$$\mathbf{Q}^{-1}\mathbf{A}\mathbf{Q} = \hat{\mathbf{A}},$$

where $\hat{\mathbf{A}}$ has a *Jordan canonical form*, has eigenvalues of $\mathbf{A}$ on diagonal, and has 0 or 1 on the superdiagonal [149, Section 2–6]. Matrix $\hat{\mathbf{A}}$ is block diagonal with each block being a *Jordan block* of the form

$$\begin{bmatrix} \lambda & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \lambda & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \lambda & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \lambda & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \lambda \end{bmatrix}_{m \times m} . \tag{3.56}$$

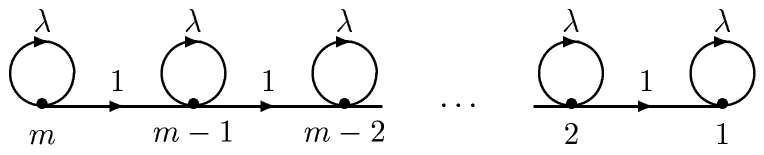


Figure 3.22 Coates graph of a Jordan block.

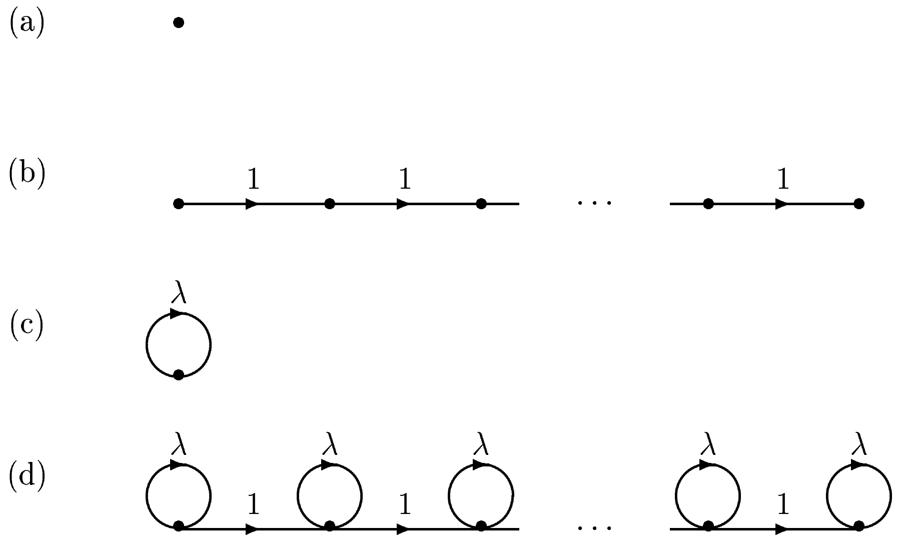


Figure 3.23 The form of components of the Coates graph of a matrix with Jordan canonical form.

If the eigenvalues of $\mathbf{A}$ are all distinct, then $\hat{\mathbf{A}}$ is diagonal and $\mathbf{v}_1$, $\mathbf{v}_2, \ldots, \mathbf{v}_N$ are the eigenvectors of $\mathbf{A}$. Note that (3.56) represents a flow graph of the form shown in Figure 3.22. Hence, we may conclude that the graph of $\hat{\mathbf{A}}$ is only made up of components of the form shown in Figure 3.23. The forms (a) and (b) happen only for zero eigenvalues and the forms (b) and (c) happen only for some repeated eigenvalues. This shows that any system can be transformed into a semicausal system by a proper change of basis. If all of the eigenvalues are zero, then it can be transformed into a causal system made up of only single-path components. If we further require that the basis be orthonormal in C^N, then we may use *Schur's Theorem* [145].

Schur's Theorem: For any $N \times N$ matrix $\mathbf{A}$, there is a unitary matrix $\mathbf{U}$ such that $\mathbf{U}^H\mathbf{A}\mathbf{U}$ is upper triangular.

In this case, the diagonal elements of $\mathbf{U}^H\mathbf{A}\mathbf{U}$ are the eigenvalues of $\mathbf{A}$ and $\mathbf{U}^H\mathbf{A}\mathbf{U}$ represent a general semicausal system. Again, $\mathbf{U}^H\mathbf{A}\mathbf{U}$ will be causal iff all eigenvalues of $\mathbf{A}$ are zero. The answer to the question proposed earlier is given by the following theorem.

Theorem: If the system $\mathbf{x}_{n+1} = \mathbf{A}_{N\times N}\mathbf{x}_n + \mathbf{B}_{N\times 1}$ reaches its steady-state in M iterations for any initial state $\mathbf{x}_0$, then

(1) all the eigenvalues of $\mathbf{A}$ are zero,

(2) the system also reaches its steady-state in exactly N iterations (nontrivial when $N < M$),

(3) there is a basis in $\mathcal{C}^N$ such that the representation of $\mathbf{A}$ in that basis has a graph made of only directed path(s) with weights 1, and

(4) there is an orthonormal basis in $\mathcal{C}^N$ such that the representation of $\mathbf{A}$ in that basis has an acyclic graph.

Proof: As the system reaches its steady-state in M iterations

$$\mathbf{x}_M = \mathbf{x}_\infty.$$

Using (3.22) we get

$$\forall \mathbf{x}_0, \quad \mathbf{A}^M(\mathbf{x}_0 - \mathbf{x}_\infty) = \mathbf{0},$$

or

$$\forall \mathbf{x}, \quad \mathbf{A}^M\mathbf{x} = \mathbf{0},$$

and, therefore, $\mathbf{A}^M = \mathbf{0}$. If λ is an eigenvalue of $\mathbf{A}$, then $\exists \mathbf{v} \neq \mathbf{0}$ such that

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}.$$

Therefore,

$$\begin{aligned} \mathbf{A}^M\mathbf{v} &= \lambda^M\mathbf{v} \\ \mathbf{0} &= \lambda^M\mathbf{v} \\ \lambda &= 0. \end{aligned}$$

Hence, all of the eigenvalues of $\mathbf{A}$ are zero and the characteristic equation of $\mathbf{A}$ is

$$\lambda^N = 0. \tag{3.57}$$

Then, according to the *Cayley-Hamilton theorem* [149], $\mathbf{A}$ satisfies (3.57), too, i.e.,

$$\mathbf{A}^N = \mathbf{0}, \tag{3.58}$$

which may be combined with (3.22) to give

$$\begin{aligned} \mathbf{x}_N - \mathbf{x}_\infty &= \mathbf{0}, \\ \mathbf{x}_N &= \mathbf{x}_\infty. \end{aligned}$$

For the third part of the theorem, this basis is the same basis that converts $\mathbf{A}$ into its Jordan canonical form, and as all of the eigenvalues of $\mathbf{A}$ are zero, the Jordan form has diagonal elements 0 and superdiagonal elements 1 resulting in components that are only of the forms of Figures 3.23(a) and 3.23(b).

For the fourth part, this basis forms the columns of the unitary matrix of the Schur's theorem. □

As there are matrices like

$$\begin{bmatrix} a & a \\ -a & -a \end{bmatrix}, \tag{3.59}$$

whose all eigenvalues are zero but do not represent a spatially-causal system, spatial causality is not a necessary condition for finite-iteration convergence. However, the above theorem shows that there is a change of basis that converts such systems to spatially causal systems with acyclic graphs. If the eigenvalues are not zero, then the system may only be transformed to a semicausal system.

CHAPTER 4

MATCHING PURSUIT

Approximation of the members of a vector space by a linear combination of a small number of members in a possibly large set (dictionary) of redundant vectors in that space has been of interest in different areas of science. More specifically, one may be interested in finding the smallest number of vectors in the dictionary whose linear combination approximates the given vector within a given error threshold. In the general case, this is a rather difficult optimization problem.

More formally, given a vector $\mathbf{x} \in R^M$, a scalar error threshold ε, and a set

$$U = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_P\} \subset R^M,$$

the problem is to find a subset

$$U^* = \{\mathbf{u}_{J_1}, \mathbf{u}_{J_2}, \ldots, \mathbf{u}_{J_S}\} \subset U, \quad 1 \leq J_1, J_2, \ldots, J_S \leq P$$

with smallest S, and a corresponding collection $\{\alpha_1, \alpha_2, \ldots, \alpha_S\}$, such that

$$\|(\alpha_1 \mathbf{u}_{J_1} + \alpha_2 \mathbf{u}_{J_2} + \cdots + \alpha_S \mathbf{u}_{J_S}) - \mathbf{x}\| \leq \varepsilon. \tag{4.1}$$

If we use the notation

$$\begin{aligned}
\mathbf{U}^* &= \begin{bmatrix} \mathbf{u}_{J_1} & \mathbf{u}_{J_2} & \cdots & \mathbf{u}_{J_S} \end{bmatrix}_{M \times S}, \\
\boldsymbol{\alpha} &= \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_S \end{bmatrix}, \\
\hat{\mathbf{x}} &\overset{\text{def}}{=} \alpha_1 \mathbf{u}_{J_1} + \alpha_2 \mathbf{u}_{J_2} + \cdots + \alpha_S \mathbf{u}_{J_S} = \mathbf{U}^* \boldsymbol{\alpha},
\end{aligned}$$

then (4.1) may be written as

$$\|\mathbf{x} - \hat{\mathbf{x}}\| \leq \varepsilon.$$

If the number S and the set U^* are found, the corresponding $\{\alpha_1, \alpha_2, \ldots, \alpha_S\}$ can simply be computed by least squares methods. A similar and equally difficult problem is that if a positive integer L is given, find the L vectors in the dictionary whose linear combination can best approximate a given vector in the vector space.

These problems are difficult combinatorial optimization problems. In fact, it has recently been proven that, in the general case, finding the optimal solution is NP-hard [156, 157]. However, an efficient suboptimal greedy solution to this problem has been discovered by different researchers in different contexts but with basically the same underlying mathematics.

In statistics, this greedy algorithm was found and named *projection pursuit* [158]. It was used for the computation of conditional expectation of random variables. In control theory, such a method was developed for nonlinear system identification [159]. In the context of time-frequency decomposition, it was named *matching pursuit* [12] and was used in signal analysis for extraction of patterns from noisy signals. In the context of image coding, it was developed for a generalized image coding method unifying transform coding, vector quantization, and fractal coding [8]. In this chapter, we will refer to this method as matching pursuit.

Matching pursuit has recently been used for video coding. Vetterli and Kalker used a rate-distortion optimized version of matching pursuit for motion compensated video coding [15]. Neff and Zakhor used matching pursuit for coding motion residual images of video sequences [14, 160].

The orthogonal matching pursuit, described in Section 4.3, was developed independently by Chen et al. [159], Pati et al. [161], Davis et al. [13], and Gharavi-Alkhansari and Huang [9, 162]. The rate-distortion optimized matching pursuit, described in Section 4.4, was proposed in [10], [11], and [15].

For a detailed mathematical description of matching pursuit, the reader may refer to the above references. In this chapter, we only review the basic principles of matching pursuit and the motivation behind its design.

4.1 A Simplistic Approach

The first approach that may come to mind for solving the problem mentioned earlier is to first find the $\mathbf{u}$ in U that best resembles or approximates $\mathbf{x}$, call it $\mathbf{u}_{J_1}$, and include it in $\mathbf{U}^*$, i.e., set $\mathbf{U}^* = [\mathbf{u}_{J_1}]$. Using least squares, find the best approximation of $\mathbf{x}$ that can be obtained by a factor of $\mathbf{u}_{J_1}$. If the norm of the residual error is below ε, we stop the procedure. Otherwise, we select the next best member of U in terms of similarity with $\mathbf{x}$, include it in $\mathbf{U}^*$, i.e., $\mathbf{U}^* = [\mathbf{u}_{J_1}\ \mathbf{u}_{J_2}]$, and repeat the process. This approach is summarized in Figure 4.1. If the members of U are orthogonal, then this approach leads to the optimal solution. However, when the members of U are not orthogonal and possibly not even linearly independent with $P > M$, then the solution could be far from optimal. A major problem with this approach is that it happens quite frequently in practice that there is a cluster of vectors in U that are very similar to each other and to $\mathbf{x}$, and selecting only one of them is nearly as good as selecting all of them. If one of the members of this cluster is selected at any stage, there is a good chance that the other members of the cluster will be selected in the following iterations, even though all of the contributions they can make to coding $\mathbf{x}$ have already been achieved by selecting the first member. This situation is especially serious when the angle between members of the cluster is very small.

4.2 Standard (Nonorthogonal) Matching Pursuit

To avoid this problem in matching pursuit, when a member $\mathbf{u}_{J_1}$ of U is selected, $\mathbf{x}$ is orthogonolized with respect to $\mathbf{u}_{J_1}$. The essence of matching pursuit is that for a given vector $\mathbf{x}$ to be approximated, first choose the vector from the dictionary which has

```
$T := \{1, 2, \ldots, P\}$
$\mathbf{U}^* := [\ ]$
$\mathbf{r} := \mathbf{x}$
$k := 0$
while $\|\mathbf{r}\| > \varepsilon$
{
        $k := k + 1$
        $c_i := \mathbf{x} \cdot \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|}$        $\forall i \in T$
        $J_k := \arg(\max_{i \in T} c_i)$
        $T := T - \{J_k\}$
        $\mathbf{U}^* := \left[ \begin{array}{cc} \mathbf{U}^* & \mathbf{u}_{J_k} \end{array} \right]$
        $\hat{\mathbf{x}} := \mathbf{U}^* (\mathbf{U}^{*T} \mathbf{U}^*)^{-1} \mathbf{U}^{*T} \mathbf{x}$
        $\mathbf{r} := \mathbf{x} - \hat{\mathbf{x}}$
}
$S := k$
$\boldsymbol{\alpha} := (\mathbf{U}^{*T} \mathbf{U}^*)^{-1} \mathbf{U}^{*T} \mathbf{x}$
```

Figure 4.1 Algorithm of the simplistic approach.

the strongest correlation coefficient (highest absolute value) with $\mathbf{x}$. Then, remove any component of its form from $\mathbf{x}$, i.e., orthogonalize $\mathbf{x}$ with respect to the selected dictionary vector, and obtain the residual of $\mathbf{x}$. The selected dictionary vector is, in fact, the one that results in the residual of $\mathbf{x}$ with the smallest energy. Repeat this process for the residual of $\mathbf{x}$ with the rest of the dictionary vectors until the residual becomes smaller than a threshold or until no other dictionary vector has significant correlation with the residual. This algorithm is shown in Figure 4.2. In this recursive procedure, the norm of $\mathbf{r}$ reduces at each iteration. In the matching pursuit, the vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_P$ in the dictionary U represent different features and structures that may be present in $\mathbf{x}$. The structures typically have overlaps, i.e., some features may be present in different structures. P is typically larger than M, and, in such case, $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_P$ cannot all be orthogonal.

The motivation for this algorithm is that at each iteration k, the algorithm finds the vector $\mathbf{u}_{J_k}$ in the dictionary which is most similar to the residual $\mathbf{r}$. Then, it finds the optimal approximation of $\mathbf{r}$ which can be obtained by multiplying a scalar c_{J_k} to $\mathbf{u}_{J_k}$. The best c_{J_k} in the least squares sense is the one that is obtained by projecting $\mathbf{r}$ onto normalized $\mathbf{u}_{J_k}$, i.e.,

$$c_{J_k} = \mathbf{r} \cdot \frac{\mathbf{u}_{J_k}}{\|\mathbf{u}_{J_k}\|}.$$

Removing the component $c_{J_k} \cdot \frac{\mathbf{u}_{J_k}}{\|\mathbf{u}_{J_k}\|}$ from $\mathbf{r}$ generates the new residual with the smallest possible norm, and the new residual is orthogonal to $\mathbf{u}_{J_i}$, $1 \leq i \leq k$. By removing $c_{J_k} \cdot \frac{\mathbf{u}_{J_k}}{\|\mathbf{u}_{J_k}\|}$ from $\mathbf{r}$, in a sense, $\mathbf{r}$ is exhausted from the feature represented by $\mathbf{u}_{J_k}$ and the residual has no component of its form. Then, $\mathbf{r}$ is replaced by the new residual and the process repeats. Structures $\mathbf{u}_i$, which were strongly similar to $\mathbf{x}$ in the first iteration but were not selected then because they were not as good a match as $\mathbf{u}_1$ in that iteration, do not necessarily have strong similarity in this next iteration. Their new similarity, measured by inner products, is only affected by new features they have that were not represented or coded by $\mathbf{u}_{J_1}$. In other words, $\mathbf{x}$ does not have any component of the form $\mathbf{u}_{J_1}$ anymore, and if some $\mathbf{u}_i$ were similar to $\mathbf{x}$, mainly due to the $\mathbf{u}_{J_1}$ component, then there is no point in using them anymore because that component has already been

```
$T := \{1, 2, \ldots, P\}$
$\mathbf{U}^* := [\ ]$
$\mathbf{r} := \mathbf{x}$
$k := 0$
while $\|\mathbf{r}\| > \varepsilon$
{
        $k := k + 1$
        $c_i := \mathbf{r} \cdot \dfrac{\mathbf{u}_i}{\|\mathbf{u}_i\|}$        $\forall i \in T$
        $J_k := \arg(\max_{i \in T} c_i)$
        $T := T - \{J_k\}$
        $\mathbf{U}^* := \begin{bmatrix} \mathbf{U}^* & \mathbf{u}_{J_k} \end{bmatrix}$
        $\mathbf{r} := \mathbf{r} - c_{J_k} \dfrac{\mathbf{u}_{J_k}}{\|\mathbf{u}_{J_k}\|}$
}
$S := k$
$\boldsymbol{\alpha} := \left(\mathbf{U}^{*T}\mathbf{U}^*\right)^{-1}\mathbf{U}^{*T}\mathbf{x}$
or
$\boldsymbol{\alpha} := \begin{bmatrix} \dfrac{c_{J_1}}{\|\mathbf{u}_{J_1}\|} & \dfrac{c_{J_2}}{\|\mathbf{u}_{J_2}\|} & \cdots & \dfrac{c_{J_S}}{\|\mathbf{u}_{J_S}\|} \end{bmatrix}$
```

Figure 4.2 Algorithm of the basic standard matching pursuit.

coded. However, if they have some other features in addition to $\mathbf{u}_{J_1}$ that are still present in $\mathbf{x}$, they may be selected in this next iteration due only to these features.

4.3 Orthogonal Matching Pursuit

In the standard matching pursuit, when a member $\mathbf{u}_{J_k}$ of the dictionary is selected it is used to completely exhaust $\mathbf{r}$ from any component of its form. So, in the following iterations, in choosing $\mathbf{u}_{J_{k+1}}$, among $\mathbf{u}_i$s, any component of the form $\mathbf{u}_{J_k}$ present in $\mathbf{u}_i$s should not influence their choice. However, in the standard matching pursuit, orthogonolization of $\mathbf{r}$, with respect to $\mathbf{u}_{J_k}$, in the following iterations favors selection of the remaining $\mathbf{u}_i$s, which have little projection on $\mathbf{u}_{J_k}$. In other words, the components of $\mathbf{u}_{J_k}$ present in $\mathbf{u}_i$s mislead the following selections. This problem can be solved by orthogonolizing $\mathbf{u}_i$s with respect to $\mathbf{u}_{J_k}$. This leads to a more powerful algorithm, which is called orthogonal matching pursuit, and its basics are shown in Figure 4.3. The residual is guaranteed to reach zero if the procedure is repeated M times.

So, in orthogonal matching pursuit, after a vector in the dictionary is selected, one removes any component of its form not only from $\mathbf{r}$, but also from all other dictionary vectors before repeating the process. *Orthogonal matching pursuit* is computationally more expensive than the nonorthogonal version, but typically gives significantly better results in the context of coding. However, if all the dictionary vectors are orthogonal, then the results for both the orthogonal and the standard matching pursuit are the same.

4.3.1 Finding the coefficients of the selected vectors

For each vector, after the selection is made among the library vectors, we use least squares [145] to find the coefficients of the selected library vectors. However, in the orthogonal matching pursuit, the computations done for *selecting* the vectors from the library include most of the computations that are necessary for *finding the coefficients* of the selected vectors. In this case, the least squares can be done by a QR decomposition

```
$T := \{1, 2, \ldots, P\}$
$\mathbf{U}^* := [\,]$
$\mathbf{r} := \mathbf{x}$
$k := 0$
while $\|\mathbf{r}\| > \varepsilon$
{
        $k := k + 1$
        $c_i := \mathbf{r} \cdot \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|} \qquad \forall i \in T$
        $J_k := \arg(\max_{i \in T} c_i)$
        $T := T - \{J_k\}$
        $\mathbf{U}^* := \left[\begin{array}{cc} \mathbf{U}^* & \mathbf{u}_{J_k} \end{array}\right]$
        $\mathbf{r} := \mathbf{r} - c_{J_k} \frac{\mathbf{u}_{J_k}}{\|\mathbf{u}_{J_k}\|}$
        $\mathbf{u}_i := \mathbf{u}_i - \frac{\mathbf{u}_{J_k} \mathbf{u}_{J_k}^T}{\|\mathbf{u}_{J_k}\|^2} \mathbf{u}_i \qquad \forall i \in T$
}
$S := k$
$\boldsymbol{\alpha} := \left(\mathbf{U}^{*T} \mathbf{U}^*\right)^{-1} \mathbf{U}^{*T} \mathbf{x}$
```

Figure 4.3 Algorithm of the basic orthogonal matching pursuit.

[145] of the matrix $\mathbf{U}^*$, i.e., by writing

$$\mathbf{U}^*{}_{M\times S} = \mathbf{Q}_{M\times S}\mathbf{R}_{S\times S},$$

where the columns of $\mathbf{Q}$ are orthonormal and $\mathbf{R}$ is an upper-triangular matrix. Then, $\boldsymbol{\alpha}$ can be found from[1]

$$\mathbf{R}\boldsymbol{\alpha} = \mathbf{Q}^T\mathbf{x}.$$

The upper triangular matrix $\mathbf{R}$ and the vector $\mathbf{Q}^T\mathbf{x}$ are computed directly in the process of selecting the library vectors. Therefore, $\boldsymbol{\alpha}$ can be found with a small number of computations after the library vectors are selected.

4.4 Rate-Distortion Optimized Matching Pursuit

The standard matching pursuit or its orthogonal version tries to find the smallest number of vectors in the dictionary that can approximate a given vector within a given error threshold. In the context of coding, after the selection process is done, the coefficients of these dictionary vectors need to be quantized and entropy coded along with their indices, and the number of selections made. Different dictionary vectors have different costs in terms of bitrate depending on the probabilistic model used for their entropy coding. Hence, a better performance is expected if, instead of selecting the smallest number of vectors from the dictionary, one selects vectors that need the shortest code collectively.

More specifically, for each $i = 1, 2, \ldots, K$, we would like to approximate each vector $\mathbf{x}_i$ by vectors selected from a dictionary U_i of vectors. After this selection is made, each of the coefficients of the selected vectors are quantized and the number of selections, indices, and quantized coefficients are entropy coded. The goal is to do this using the shortest possible code.

In other words, we want to encode vectors $\mathbf{x}_i$, $i = 1, \ldots, K$ by entropy coding of the numbers S_i, $i = 1, \ldots, K$, the indices $J_{i,k}$, $i = 1, \ldots, K$, $k = 1, \ldots, S_i$, and the quantized

[1]For a proof, see for example [145, page 238].

coefficients $\widehat{\alpha}_{i,k}$, $i = 1, \ldots, K$, $k = 1, \ldots, S_i$. Also, let us denote the distortion (error energy) of each of the vectors $\mathbf{x}_i$ by D_i, i.e.,

$$D_i = \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2, \qquad i = 1, \ldots, K,$$

denote the rate (number of bits) spent on coding $\mathbf{x}_i$ by R_i, and use the notation

$$D \stackrel{\text{def}}{=} \sum_{i=1}^{K} D_i.$$

Assuming a target rate of R for $\sum_{i=1}^{K} R_i$, the coding problem is that of minimizing

$$D = \sum_{i=1}^{K} D_i(R_i)$$

subject to the constraint

$$g \stackrel{\text{def}}{=} \sum_{i=1}^{K} R_i - R = 0.$$

This constrained minimization problem may be solved using *Lagrange multipliers* [163] by minimizing

$$\begin{aligned} h &= D + \lambda g && (4.2) \\ &= \sum_{i=1}^{K} D_i(R_i) + \lambda \left(\sum_{i=1}^{K} R_i - R \right) \\ &= \left(\sum_{i=1}^{K} \left(D_i(R_i) + \lambda R_i \right) \right) - \lambda R, && (4.3) \end{aligned}$$

where h is called the *Lagrangian.* If we assume that D_i is a continuous function of R_i, we may take partial derivatives with respect to R_i, $i = 1, 2, \ldots, K$ and λ, and set them to zero. This gives

$$\begin{aligned} \frac{\partial h}{\partial R_i} &= \frac{\partial D_i}{\partial R_i} + \lambda = 0, \qquad i = 1, 2, \ldots, K, && (4.4) \\ \frac{\partial h}{\partial \lambda} &= \sum_{i'=1}^{K} R_i - R = 0. \end{aligned}$$

Equation (4.4) corresponds to minimizing each term of the sum in (4.3) individually. So, the minimum can be found by computing the minimum of $D_i + \lambda R_i$ for each $\mathbf{x}_i$. Equation (4.4) may be rewritten as

$$\frac{\partial D_1}{\partial R_1} = \frac{\partial D_2}{\partial R_2} = \cdots = \frac{\partial D_K}{\partial R_K} = -\lambda. \qquad (4.5)$$

Replacing partial derivative with δ in Equation (4.5) gives

$$\frac{\delta D_1}{\delta R_1} = \frac{\delta D_2}{\delta R_2} = \cdots = \frac{\delta D_K}{\delta R_K} = -\lambda.$$

Regarding the selection process in the matching pursuit, we note that for each vector $\mathbf{x}_i$,

(1) selection of each dictionary vector is based on δD_i, i.e., on how much it reduces the energy (distortion) of $\mathbf{x}_i$, and

(2) the selection process stops when the distortion D_i of the $\mathbf{x}_i$ goes below a threshold.

Therefore, the *selection criterion* and the *stopping criterion* are both based on distortion of the residual of $\mathbf{x}_i$ and do not take into account the bitrate cost of the selections. This can be improved by changing both the selection and the stopping criteria.

Selection Criterion. Due to the entropy coding stage, the number of bits required to encode $\mathbf{x}_i$ is not exactly proportional to the number of vectors used for coding it. In other words, the number of bits required for representing an index or a quantized coefficient depends on the frequency of selection of the dictionary vector or the quantized coefficient. Therefore, the best dictionary vector to be selected at each stage is not the one which gives the greatest reduction in distortion D_i, but the one which gives the greatest reduction in $D_i + \lambda R_i$. In other words, the best vector to select is the one with the smallest $\delta D_i + \lambda \delta R_i$, where δD_i is the change in the energy of the residual (distortion) of $\mathbf{x}_i$ and δR_i is the number of bits spent on coding the index, and the coefficient of the newly selected vector plus any change in the cost of previously selected vectors. So, rate-distortion optimized matching pursuit uses this improved selection criterion.

Stopping Criterion. In the standard and orthogonal matching pursuit, the encoder tries to encode all of the $\mathbf{x}_i$s with approximately the same distortion energy ε^2 using a minimum number of dictionary vectors. However, this is not optimal because by doing this, some $\mathbf{x}_i$s select many dictionary vectors with each one contributing slightly to the reduction of the error energy. Many of the bits used for coding these vectors may be used to make a larger reduction in D if they are used for coding other vectors.

This suggests that an optimal solution is obtained when $\delta D_i/\delta R_i$, rather than D_i, is kept the same for all $\mathbf{x}_i$s. This modifies the stopping criterion for the selection process. In rate-distortion optimized matching pursuit, the goal is to minimize $D_i + \lambda R_i$. So, the selection process stops when $D_i + \lambda R_i$ cannot be decreased any further, which is equivalent to $\delta D_i + \lambda \delta R_i$ becoming greater than zero. For $\delta R_i > 0$, this means that $-\delta D_i/\delta R_i$ becomes smaller than the threshold λ.

D_i is typically a decreasing function of R_i and, therefore, $\delta D_i/\delta R_i$ is usually negative and $-\delta D_i/\delta R_i$ represents how much the distortion D_i (and, therefore, D) is reduced for every bit of R_i spent for this reduction. Threshold λ determines the trade-off between δD_i and δR_i. Requiring $-\delta D_i/\delta R_i \geq \lambda$ means that for every bit spent, D_i must decrease by at least λ. Note that rate-distortion optimality can be applied to both the standard and the orthogonal matching pursuit.

CHAPTER 5

THE NEW METHOD

In this chapter, a new block-based image coding method is proposed. This method, which was originally aimed at solving some major shortcomings of fractal coders, results in a very general framework for image and video coding.

5.1 Motivation

In block transform coding methods like JPEG, each block in an image is encoded by approximating it with a linear combination of fixed basis blocks as shown in Figure 5.1. In these methods, the set of basis blocks is the same for all of the blocks being encoded and is also independent of the image. These methods are usually not adaptive in the sense that they cannot take into account image structures like repetitions or similarities between blocks that are located at different parts of the image. The basis blocks cannot change from one image to another or from one block to another.

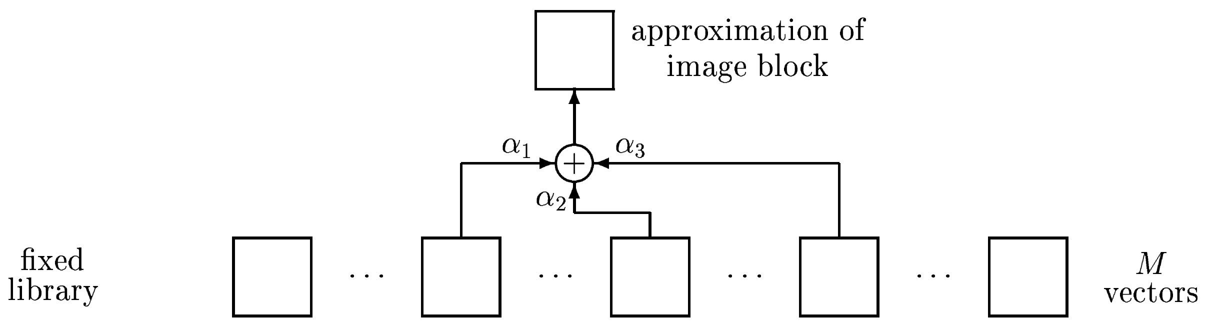


Figure 5.1 Selection of vectors from library in block transform methods.

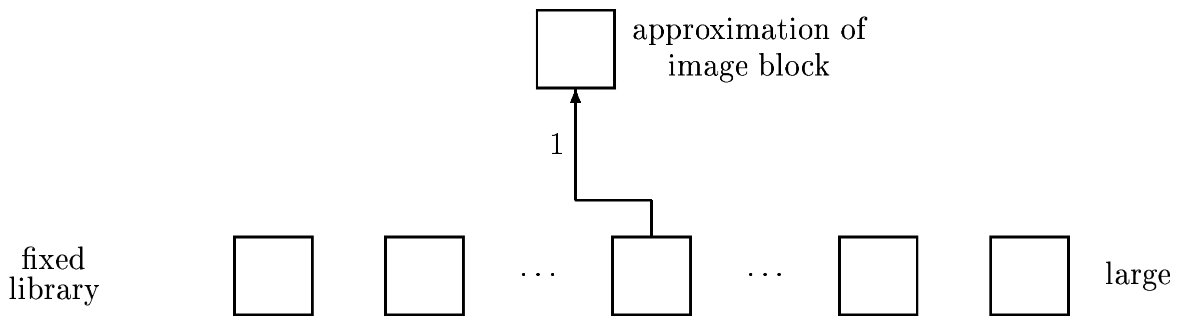


Figure 5.2 Selection of vectors from library in VQ methods.

On the other hand, vector quantization (VQ) methods are designed for a class of images. They have codebooks that can suitably approximate any block in the class of images for which they are designed. In vector quantization, each block of an image is approximated by a single block from the codebook (Figure 5.2).

So, in contrast to block transform coding, which uses a linear combination of (typically orthogonal) blocks to approximate a given block, VQ uses a single block, which is, of course, chosen from a larger set of blocks. Although in standard VQ the codebook is designed for a class of images, it is typically not adaptive to single images and blocks because it is not efficient to send a new codebook for every image or every block in an image.

Fractal-based methods may be viewed as block coding methods that approximate a range block by a linear combination of up to a few fixed blocks (typically for dc and low frequency components) [4, 111, 132], and a single other block made by applying some contractive transformation on a domain block in the same image (for encoding details of the range block) as shown in Figures 5.3 and 5.4. So, it is similar to VQ[1] in the sense that it uses a single block from a large library of blocks to approximate the details of the block. However, fractal-based methods are adaptive because the codebook can be different from one image to another and, in some methods, different from one range block to another range block without the need for sending the codebook each time. This adaptivity has more potential for image compression. In fact, because fractal-based

[1]For a comparison between VQ and Jacquin's method, see either of the following references [3, 4, 72].

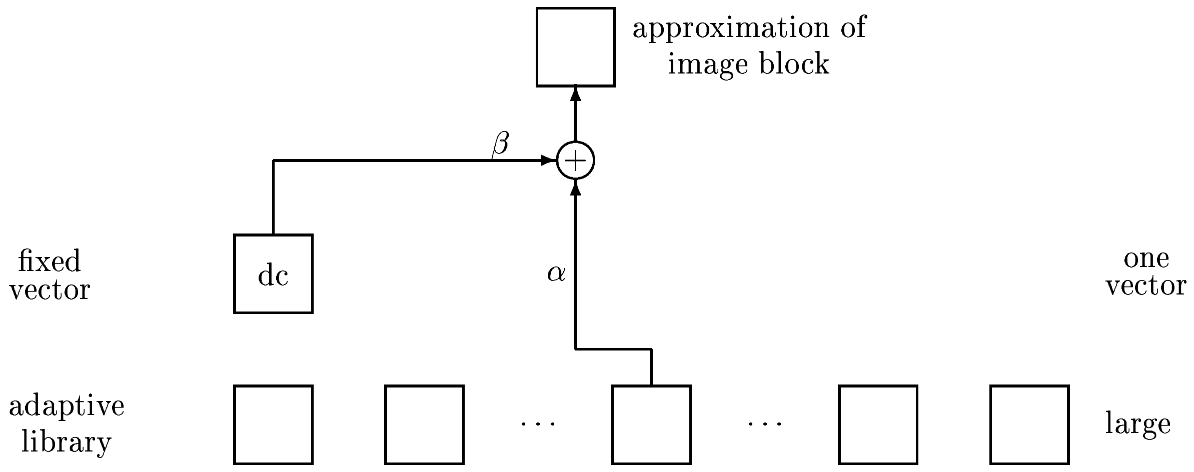


Figure 5.3 Selection of vectors from library in basic fractal methods.

methods approximate each part of an image by another part, they have the potential to exploit global relationships better than VQ or block transform coding (e.g., DCT).

However, fractal-based methods have the problem that the codebook, although very adaptive, is not very controllable. The codebook is limited to blocks that exist in the image (possibly limited to the neighborhood of the range block) or simple transformed versions of them. This pool is not always capable of providing a good approximation to highly detailed or unique range blocks in the image. In fact, due to this limitation, very detailed image blocks cannot be encoded very well. And, more generally, fractal methods usually cannot efficiently encode images at arbitrarily high PSNRs. Fractal methods are also crucially dependent on a strong dc component being present in image blocks and put special emphasis on encoding dc components of image blocks and are not suited for data types with small dc components. Also, when an image block is similar to several other domain blocks in the image, fractal coders always use the best match only, and cannot take advantage of the similarity of other blocks. Although fractal coders exploit self-similarity of images at different scales (inter-scale self-similarity), they do not take advantage of self-similarity of images at the *same* scale (intra-scale self-similarity). Many researchers have unsuccessfully tried using such self-similarities in fractal coders.

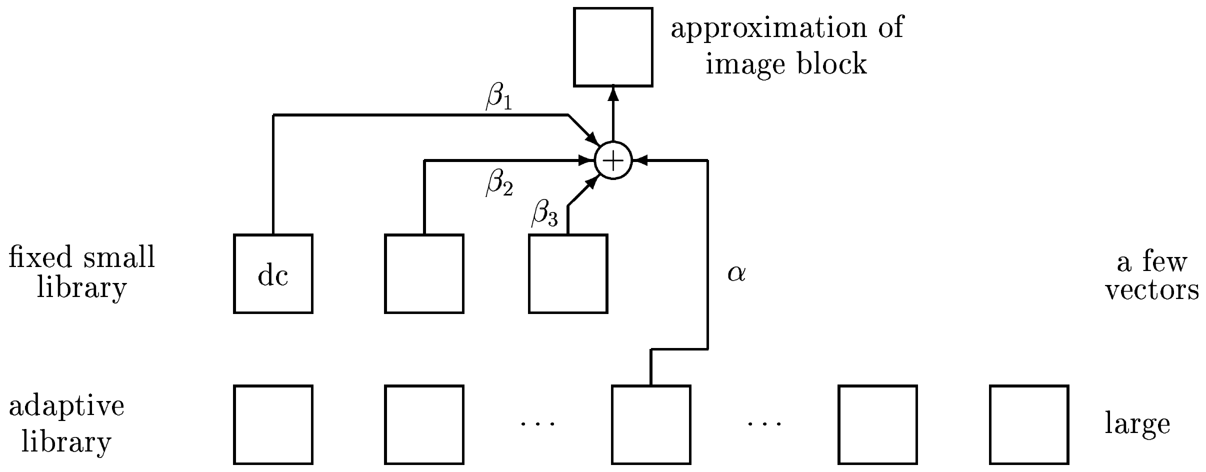


Figure 5.4 Selection of vectors from library in generalized fractal methods.

In this chapter, we propose a method that efficiently solves these shortcomings of fractal coders while keeping their advantages. This work begins by adding a series of major revisions in the way fractal coding is done and results in a very general algorithm that unifies the three methods of block-based transform coding, VQ, and most of the earlier fractal image coders. The utilization of intra-scale image self-similarities in the case of video coding makes the block prediction methods, like DPCM, adaptive block prediction methods like block motion compensation, and hybrid coding methods like motion compensation combined with transform coding of residual errors, special cases of the proposed coding method.

However, the new coding algorithm requires the solution to the important and challenging mathematical problem addressed in Chapter 4. The matching pursuit solution allows the encoder to automatically detect the type of redundancy present in the image block and the order of approximation that makes the most efficient code for the block.

Figure 5.5 shows the structure of a general compression system. Different components of this system are described in this chapter for the proposed method.

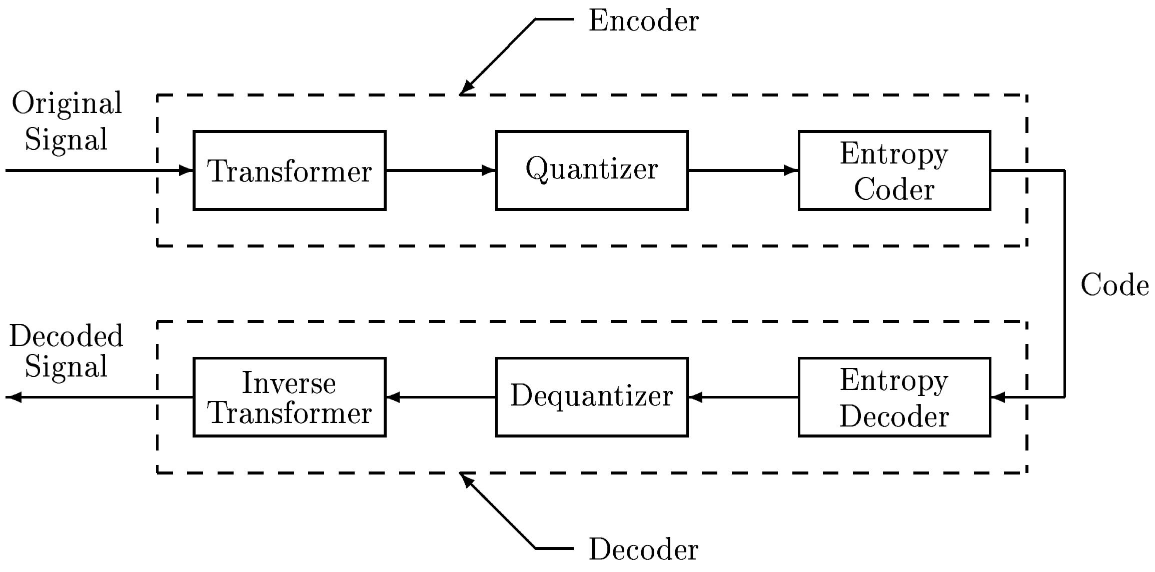


Figure 5.5 Structure of a general compression system.

5.2 The Transform

The transformation used in the proposed method is an extension of the basic form described in Section 3.5. In this transform, an $N_1 \times N_2$ image is first partitioned into nonoverlapping range blocks of size $M_1 \times M_2$, where $N_1 = K_1M_1$ and $N_2 = K_2M_2$. Each range block is considered as an $M = M_1M_2$ dimensional vector. There are $K = K_1K_2$ range blocks denoted by $\mathbf{x}_i$, $i = 1, 2, \ldots, K$. For each range block, a library U_i of blocks of size $M_1 \times M_2$ is made and a small number of them are chosen such that their linear combination gives a good approximation of the range block (Figure 5.6). The number and the index of these selected library blocks and their corresponding coefficients constitute the code for the range block. It is notable that the library can be different from one range block to another and its adaptive components are neither necessary nor sent for the decoding process and are not part of the code.

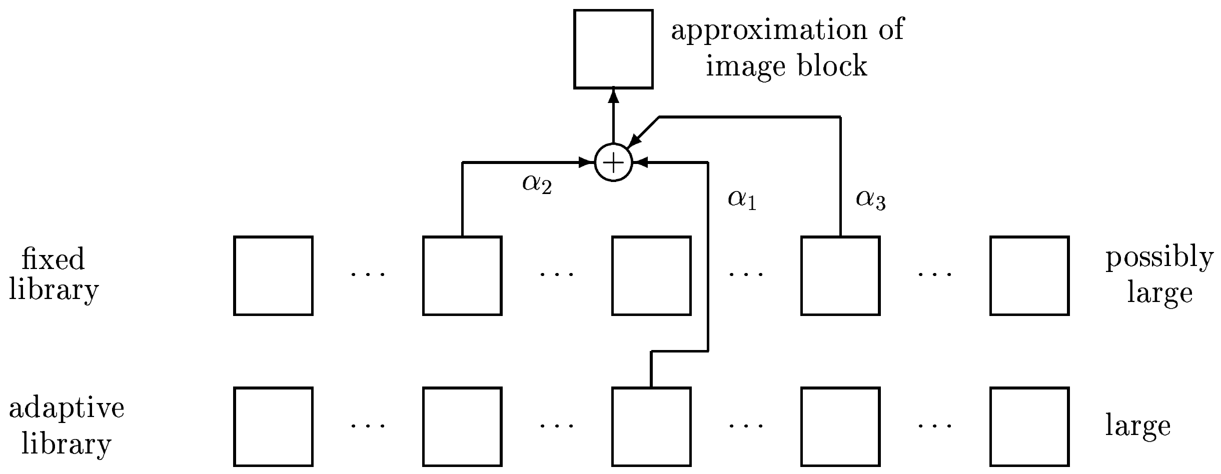


Figure 5.6 Selection of vectors from library in the proposed method.

5.2.1 Making the library of blocks

For each range block $\mathbf{x}_i$, $i = 1, 2, \ldots, K$, the library U_i is made up of two disjoint subsets.

5.2.1.1 Fixed library blocks (FLB)

This set is denoted by F and contains a series of blocks $\mathbf{f}_j$, $j = 1, 2, \ldots, P_F$ which are independent of the range block being encoded. This set is designed by the encoder designer. For example, it can be the M orthonormal M-dimensional basis blocks used in DCT, the set of blocks in the codebook of a vector quantizer designed for the class of images being encoded, or merely a very small set of orthonormal blocks that the range block has strong components of their form.[2]

This set of FLBs is the same for all of the range blocks in the image being encoded and must be sent to the encoder offline. The presence of these blocks in the library can serve the following purposes:

- The encoder can use shorter codes for the indices of some of these blocks that are commonly present in the range blocks in the form of a strong component.

[2]As will be mentioned later, depending on the type of FLBs, a limit may be put on the maximum number of FLBs used, e.g., this may be 1 if a VQ codebook is used.

- Enable the decoder to encode range blocks more accurately which are very different from other blocks in the image and, therefore, cannot be well encoded by other subsets of the library.

5.2.1.2 Adaptive library blocks (ALB)

This set is denoted by A_i and contains library blocks $\mathbf{a}_{i,j}$, $j = 1, 2, \ldots, P_{A_i}$ that are generated by applying some transformation $\mathcal{T}_i$ on domain blocks in a neighborhood of the range block. This set is range-block dependent, i.e., it may be different from one range block to another. There are many ways of making these library blocks using the guideline that it must be possible to represent $\mathcal{T}$ with a short code and $\mathcal{T}$ can generate blocks that are very similar to the block being encoded. Two such sets of ALBs are described below and will be used for obtaining experimental results. There are several other possibilities that are not explored in this dissertation.

(1) *Higher Scale Library Blocks (HSLB)*

These are blocks that are generated by lowpass filtering and subsampling (shrinking) domain blocks of size $LM_1 \times LM_2$ of the image ($L > 1$) in both directions by the factor L to give $M_1 \times M_2$ library blocks. The filtering is typically a simple averaging.[3] The set may also be complemented by adding rotated or reflected versions of the above blocks, which are more generally called pixel shuffling or isometric transformation.

(2) *Same Scale Library Blocks (SSLB)*

These are blocks that are taken directly from the image (with no shrinking) and are restricted to be in parts of the image that are already encoded. In other words, these blocks are selected causally. This subset may again be complemented by adding rotated or reflected versions of the SSLBs.

[3]The issue of optimal choice of L is studied by some investigators [4, 164], but the optimal choice of filter is still an open problem.

The set of ALBs introduces adaptiveness to the encoding algorithm as each range block can use a different set of ALBs. This can potentially increase the efficiency of the encoding algorithm.

The HSLBs exploit fractal properties of the image by using its inter-scale similarities. However, the assumption of self-similarities of image at different scales typically implies self-similarities of the image at the *same scale*. In other words, when one larger part of an image is similar to smaller parts of the image, smaller parts are also similar to each other. These same scale, self-similarities of images are what is exploited by the SSLBs. In fact, some experimental results suggest that self-similarities of natural images at the same scale are typically stronger than self-similarities of images at different scales when simple shrinking transformations are used [6], i.e., for a range block, the best match is more likely to be found among SSLBs rather than HSLBs. The SSLBs actually allow the process to use the code of parts of the image that are already encoded to be used for parts that are not encoded yet. The concept of encoding data by copying parts of the data that are already encoded was introduced for lossless compression of 1-D data first by Ziv and Lempel [165]. It was extended to lossy compression of 2-D grayscale images by Saito et al. [166]. For video sequences, this method has proven to be very effective and forms the basis for all motion compensation methods.

5.2.2 The selection process

Using a notation similar to that of Chapter 4, we denote the members of U_i by $\mathbf{u}_{i,j}$, $j = 1, 2, \ldots, P_i$, i.e.,

$$U_i = \{\mathbf{u}_{i,1}, \mathbf{u}_{i,2}, \ldots, \mathbf{u}_{i,P_i}\} = \{\mathbf{f}_{i,1}, \mathbf{f}_{i,2}, \ldots, \mathbf{f}_{i,P_F}, \mathbf{a}_{i,1}, \mathbf{a}_{i,2}, \ldots, \mathbf{a}_{i,P_{A_i}}\},$$

$$P_i = P_F + P_{A_i}.$$

This dictionary is typically overcomplete ($P_i > M$) and the problem of making the optimum selection among the dictionary blocks is an important part of the transform. This problem is the problem that is addressed by matching pursuit algorithms as discussed in Chapter 4. In this system, we use the rate-distortion optimized standard/orthogonal matching pursuit.

5.2.3 Substitution of the range block with its approximation

After finding the coefficients of the selected library blocks for the current range block, the encoding process may be continued in either of two ways.

(1) *Encoding without Range Block Substitution*
Begin encoding the next range block using ALBs constructed from the original image and the FLBs.

(2) *Encoding with Range Block Substitution*
Make the approximation of the current range block by multiplying the computed coefficients by their corresponding library blocks and adding them together. Then, update the image by substituting the current range block in the image by its approximation. By doing so, the ALBs for the following range blocks will be constructed from the updated image instead of the original image.

The motivation for the second method is as follows. At the decoder side, at the iterations following the convergence of the reconstructed image (attractor), the ALBs for each range block will be constructed from the converged image and *not* from the original image. So, at the encoder side, it is reasonable to try to make the best approximation of the range block by using the ALBs from the best guess that we can have from the attractor. After encoding each range block at the encoder, substituting the range block in the image with the approximated range block, moves the image closer to the attractor and, hence, gives a more accurate set of ALBs for approximating the following range blocks.

All of the fractal-based compression methods published earlier use the method of encoding without range block substitution. However, our experiments have shown that encoding with range block substitution significantly improves the quality of the decoded image without affecting the performance of the encoder.

This substitution method is applicable to all fractal/attractor coding methods and corresponds to different matrices $\mathbf{A}$ and $\mathbf{B}$ in (3.17). Using the notation of Section 3.7, we define a new transformation

$$\widehat{\mathcal{T}}_i(\mathbf{x}) = \mathbf{H}_i \mathcal{T}_i(\mathbf{d}_i) + \mathbf{E}_i \mathbf{x},$$

where

$$\mathbf{E}_i = \begin{bmatrix} \mathbf{I}_{(i-1)M \times (i-1)M} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}_{M \times M} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{(K-i)M \times (K-i)M} \end{bmatrix}.$$

Then,

$$\widehat{\mathcal{T}}_i = (\alpha_i \mathbf{H}_i \mathbf{P}_i \mathbf{G} \mathbf{K}_i + \mathbf{E}_i)\, \mathbf{x} + \mathbf{H}_i \mathbf{B}_i$$

and

$$\mathcal{T}(\mathbf{x}) = \widehat{\mathcal{T}}_K \widehat{\mathcal{T}}_{K-1} \dots \widehat{\mathcal{T}}_2 \widehat{\mathcal{T}}_1 \mathbf{x}.$$

$\mathbf{A}$ may be computed as

$$\mathbf{A} = \prod_{i=1}^{K} (\alpha_i \mathbf{H}_i \mathbf{P}_i \mathbf{G} \mathbf{K}_i + \mathbf{E}_i),$$

and for $\mathbf{B}$ we have

$$\mathbf{B} = \sum_{i=1}^{K-1} \left(\prod_{j=i+1}^{K} (\alpha_j \mathbf{H}_j \mathbf{P}_j \mathbf{G} \mathbf{K}_j + \mathbf{E}_j) \right) \mathbf{H}_i \mathbf{B}_i + \mathbf{H}_K \mathbf{B}_K.$$

5.2.4 Comparison with other methods

We make the following observations on our proposed encoding process.

- Most other fractal-based image coding methods are close to special cases of this method where maximum number of FLBs is set to 1 (for dc component) or a small number and no SSLBs are allowed.

- If the maximum number of ALBs is set to 0, the maximum number of FLBs is set to M, and the FLBs are selected orthogonal, this method reduces to a block transform coding.

- On the other hand, if the maximum number of ALBs is again set to 0 and the maximum allowed number of FLBs is set to 1, then this method reduces to a vector quantization method.

The resulting fractal coding method has the following advantages over most fractal coding methods:

- It typically gives better compression performance due to a better and less constrained method of selection of blocks from the dictionary.
- Because the dictionary provides an over-complete set of vectors for approximation of each range block, this method provides a method for arbitrarily high PSNR and even lossless image coding.
- This method performs relatively well even when the image blocks do not have a strong dc component, e.g., in residual images or some nonphotographic 2-D signals.
- The compression time is only moderately increased with the increase of PSNR of the decoded image.

5.2.5 Structure of the code

To construct the code for each range block, the values $\alpha_1, \alpha_2, \ldots, \alpha_S$ obtained in the previous section will be quantized using uniform quantizers. We denote these quantized values by $\widehat{\alpha}_1, \widehat{\alpha}_2, \ldots, \widehat{\alpha}_S$. The number of selected library blocks S, their indices $J_1, J_2, \ldots, J_S$, and their corresponding quantized coefficients $\widehat{\alpha}_1, \widehat{\alpha}_2, \ldots, \widehat{\alpha}_S$ represent each range block $\mathbf{x}_i$, i.e.,

$$\mathbf{x}_i \xrightarrow{\text{code}} S_i, (\widehat{\alpha}_{i,1}, J_{i,1}), \ldots, (\widehat{\alpha}_{i,S_i}, J_{i,S_i}).$$

5.3 Quantizer/Dequantizer

Among the code parameters generated by the transform, only the coefficients $\alpha_{i,j}$ have continuous values and need to be quantized. A uniform quantizer is used which takes a continuous value α and generates an index $I(\alpha)$ to a quantization bin. If we denote the quantization bin size (step size) by Δ, then

$$I(\alpha) = \text{round}\left(\frac{\alpha}{\Delta}\right).$$

Under mild conditions, it is proven that uniform quantizers are optimal in a rate-distortion sense, when the quantized values are entropy coded [167, Section 9.9].

Note that the quantization error can also be included in δD of Section 4.4 to improve the quality of decisions made by the rate-distortion optimized matching pursuit algorithm.

The task of the dequantizer is to convert bin numbers to quantized values, i.e.,

$$\widehat{\alpha} = \Delta\ I(\alpha).$$

Selection of Δ affects both the rate R and the distortion D. To select Δ optimally in a rate-distortion sense, it must be selected so that it minimizes the Lagrangian $D + \lambda R$. Consider the linear expansion

$$\hat{\mathbf{x}} = \sum_{i=1}^{S} \alpha_i \mathbf{u}_i,$$

where $\mathbf{u}_i$, $i = 1, 2, \ldots, S$ are orthonormal. The average additional distortion generated by quantizing each α_i is

$$D = \frac{1}{12}\Delta^2.$$

On the other hand, the dependence of R on Δ may be formulated as [167, Section 9.9]

$$R = -\int_{-\infty}^{+\infty} f(\alpha_i) \log_2 f(\alpha_i) d\alpha_i - \log_2 \Delta,$$

where f is the probability distribution function of α_i. Minimizing $D + \lambda R$, with respect to Δ, gives

$$\begin{aligned}\frac{\partial D}{\partial \Delta} + \lambda \frac{\partial R}{\partial \Delta} &= 0 \\ \frac{1}{6}\Delta - \lambda \frac{1}{\Delta \ln 2} &= 0 \\ \Delta &= \sqrt{\frac{6}{\ln 2}\lambda} \approx 2.94\sqrt{\lambda}.\end{aligned}$$

5.4 Entropy Coding

For the coding method to be efficient, the code represented in Section 5.2.5 should be entropy coded to construct the final code for the image. All entropy coders use some underlying probabilistic model of the source signal, either explicitly or implicitly. Given this probabilistic model, entropy coders differ on how well they can compress the source signal. However, there is an upper bound for the performance of these coders called the *Entropy*, which is denoted by H, and is a function of the source model.

5.4.1 The coder

The two general, most commonly used coding methods are *Huffman coding* [168] and *arithmetic coding* [169]. Huffman code can perform close to optimal if the probabilities of the alphabet of the source are close to powers of 2. Arithmetic coders, on the other hand, can perform close to optimal for any arbitrary alphabet probabilities. Due to this advantage, an arithmetic coder was used in the system implemented for this work. The implemented arithmetic coder is a version of the one proposed by Bell et al. [169] and Witten et al. [170]. In this algorithm, the source model and the coder are made quite separate, which makes it easily adaptable to different source models, some with complex structures. The precision to which symbol frequencies may be held in the program provided for this algorithm is 14 bits. In another version of this program implemented by Neal [171], this precision is increased to 27 bits. This later version was modified by the author to allow more flexible models for the source and was used for entropy coding in the encoder.

5.4.2 The source model

The implemented arithmetic coder uses three different models for S_i, $J_{i,j}$, and $\widehat{\alpha}_{i,j}$. Each of these models will be described in the following sections. However, in the implementation, each of them have the following features:

(1) Each model may be initialized with some initial statistics (histograms). These statistics should either be known to the decoder or should be sent to it through a header in the code. Otherwise, all symbols are given the same probabilities.

(2) The model may be updated during the encoding (and, therefore, also during decoding), i.e., by encoding each new symbol and the frequency of the symbol that is being tracked by the encoder is incremented. These frequencies must also be computed adaptively at the decoder. Otherwise, the probabilities are always based on initial statistics.

5.4.3 Model for the number of selected blocks S_i

S_is were modeled in the simplest way and their model uses the frequency of each S_i (including initial statistics). S_i can have values ranging from 0 to M. In other words, this model is based on $P(S_i)$ approximated by

$$\begin{aligned} P(S_i = K) &= \frac{n(k)}{\sum_{j=0}^{M} n(j)}, \\ i &= 1, 2, \ldots, K, \\ k &= 0, 1, 2, \ldots, M, \end{aligned}$$

where $n(x)$ represents the number of times x blocks were selected for approximating a range block.

5.4.4 Model for the indices of selected blocks $J_{i,j}$

As mentioned before, the selection process of matching pursuit is an iterative process and is done in iterations or stages. The library blocks for range block $\mathbf{x}_i$ are selected in S_i

stages $j = 1, 2, \ldots, S_j$, resulting in indices $J_{i,j}$ and coefficient $\alpha_{i,j}$. Experimental results show that the probability of a specific library block being selected is highly dependent on the stage number j at which it is selected. Therefore, the indices $J_{i,j}$ are modeled using the conditional probability $P(J_{i,j}|j)$. At each stage j, the model estimates the probability of occurrence of a specific index by computing

$$P(J_{i,j} = k) = \frac{n(k|j)}{\sum_{l=0}^{M} n(l|j)},$$

where $n(x|y)$ denotes the number of times a library block with index x was selected at stage y.

5.4.5 Model for the quantized coefficients $\widehat{\alpha}_{i,j}$

The probabilities of $\widehat{\alpha}_{i,j}$ are conditioned on $J_{i,j}$, as they are highly correlated, i.e.,

$$P(\widehat{\alpha}_{i,j} = x) = \frac{n(x|J_{i,j} = y)}{\sum_z n(z|J_{i,j} = y)},$$

where $n(u|J_{i,j} = y)$ is the number of times the quantized coefficient u has been selected for library block number y.

5.5 The Inverse Transform

The decoding algorithm, in the most general case, is similar to the one proposed by Jacquin [3, 4, 71, 72] and is based on the Contraction Mapping Theorem. Using the notation of Chapter 3, we begin with any initial image $\mathbf{x}$ and for each range block $\mathbf{x}_i$ bring the blocks from the image that have the same address as the selected domain blocks and apply the corresponding transformations $\mathcal{T}_i$ on these blocks to make an approximation of the selected ALBs. Then, these approximated library blocks and the selected FLBs are multiplied by their corresponding coefficients and are added together to make an approximation of the range block. This process is repeated for all range blocks until the resulting image does not change significantly with further iteration.

The convergence of the decoding process is proven only for the case where the maximum allowed number of ALBs is 1 and their combination coefficients are less than 1 [4]. However, experimental results suggest that the decoder converges even when the maximum number of allowed ALBs is greater than 1 [6].

5.5.1 Immediate vs. delayed substitution

There are two methods for updating the image at each iteration in the decoding process. These methods differ in the time at which the computed range block at each iteration substitutes the old range block in the image so they can be used in the construction of the ALBs for the following range blocks.

(1) *Decoding with Delayed Substitution*

The ALBs needed for making the range blocks in image at each iteration are made from the image in the previous iteration. In other words, the approximated range block computed from the image at iteration n is used as part of the image in iteration $n + 1$. The following range blocks in iteration n do not use the new approximated block as part of the image.

(2) *Decoding with Immediate Substitution*

At each iteration, immediately after a range block is computed it substitutes the corresponding part of the image and the following range blocks in the same iteration will use the updated image for making their ALBs.

Jacquin's method and nearly all of its following fractal-based methods use the first method. The second method was proposed by Kaouri in 1991 [172] and has a much faster convergence.

So, the performance of the decoder is affected by the way the approximated range blocks are used both in encoder and decoder. Encoding with range block substitution improves the quality of the reconstructed image in decoder and decoding with immediate substitution improves the rate of convergence in the decoder.

5.5.2 Noniterative decoding by causal encoding

As mentioned before, the SSLBs are restricted to be chosen causally from the image, but the HSLBs are not. If we restrict the HSLBs to be also chosen causally and use encoding with range block substitution and decoding with immediate substitution, then the whole encoding system becomes causal and the decoding process needs only a single iteration to converge (i.e., it becomes noniterative) and there are no restrictions on the coefficients of the library blocks (except due to possible numerical stability issues). This issue was studied in Chapter 3 and is further described below in the context of the new coding method.

If the HSLBs are selected causally in the encoder, then during the encoding of each range block, the approximation of the range block is made from FLBs and ALBs that are constructed from already encoded parts of the image and the result is put back in the image.

At the decoder, the approximated range block is made from FLBs and ALBs that are constructed from already decoded parts of the image and, then, the result is put back in the image.

It is clear that at each step of encoding, the encoder is using informations that are all available to decoder at the similar step and, therefore, the approximation of the range block that the decoder reconstructs at the first run through the image is *exactly* the same as the approximation that the encoder makes.

So, the decoder not only reaches the attractor in one iteration, but, also, the attractor is exactly the approximation of the image made at the encoder and no additional error is introduced into the image during the decoding. In fact, no restriction of contractivity is put on the transformations that are used by the encoder.

In this case, using the notation used in Section 2.1, we have $A = W(B)$ giving

$$h(B, A) = h(B, W(B))$$

in contrast to the error limit given by the Collage Theorem

$$h(B, A) \leq \frac{h(B, W(B))}{1 - s}.$$

In fact, this method can be classified as an adaptive block prediction coding where the prediction parameters are being sent to the decoder.

However, selecting domain blocks noncausally, as it is done in almost all fractal-based methods, provides a larger variety of ALBs for encoding a range block and also makes the encoding quality of all the range blocks uniform, (i.e., there is no difference in the ALBs available for encoding a range block at the beginning of the image and one located at the end). Although imposing causality on the HSLBs can cause a major problem for the fractal-based coders that used a single ALBs and one or a few FLBs, it does not seriously affect the performance of the generalized coding method proposed here, when the number of FLBs is large enough.

In summary, selecting the HSLBs noncausally makes the encoding of the range blocks more uniform, which is important in some cases. On the other hand, the price of using HSLBs noncausally at the encoder is that for the image to be reconstructible at the decoder side, the following is required:

(1) The decoding process needs to be iterative.

(2) Some additional error is introduced during the decoding process. (An upper limit to this error is given by the Collage Theorem.)

(3) $\mathcal{T}$ needs to be contractive.

CHAPTER 6

COMPRESSION OF STILL IMAGES

In this chapter, we present the results of applying the algorithm of Chapter 5 to natural still images.

6.1 Settings of the Experiments

The parameter settings listed below are used in these experiments.

- The rate-distortion optimized orthogonal matching pursuit is used for the selection of dictionary vectors.
- The histograms are trained using eight training images: “Airplane,” “Baboon,” “Peppers,” “Sailboat,” “Splash,” “Aerial,” “Stream,” and “Bank.” They are used three times and the histograms are accumulated. These histograms are then used for encoding the actual test image.
- For FLBs, the M DCT basis blocks are used.
- 128 SSLBs and 64 HSLBs are used. The SSLBs are selected from the causal neighborhood of each range block, i.e., they are taken from a square area in the image with the range block at its center, while avoiding parts of this square that are not encoded yet as shown in Figure 6.1. The HSLBs are selected from the (noncausal) neighborhood of range blocks. The size of the neighborhood is not the same for the SSLBs and the HSLBs and, in either case, any parts of the square that fall out of the image are not used.

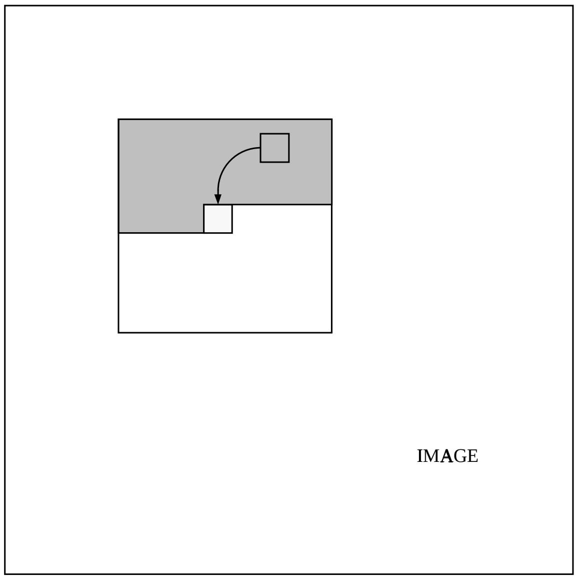


Figure 6.1 The neighborhood of a range block with the shaded region representing the causal part.

- In practice, it was found that using pixel shufflings, such as rotation by multiples of 90 degrees or reflections, only slightly affect the performance of the coding system and largely increase the computation time, and, therefore, were not used.

- The method of encoding with range block substitution (see Section 5.2.3) was used for its smaller decoding error.

- In practice, it was found that in nearly all range blocks, the constant DCT basis block, corresponding to zero vertical and horizontal frequencies, was selected as the first match in the matching pursuit process. This is due to the presence of a strong dc component in blocks of natural images and the low cost of these blocks in terms of bitrate due to their frequent use. Therefore, in the selection algorithm, the constant DCT basis block was forced to be selected as the first selection for all range blocks. The coefficient of this basis block is also predicted from the quantized coefficients from the top and the left side range block approximations.

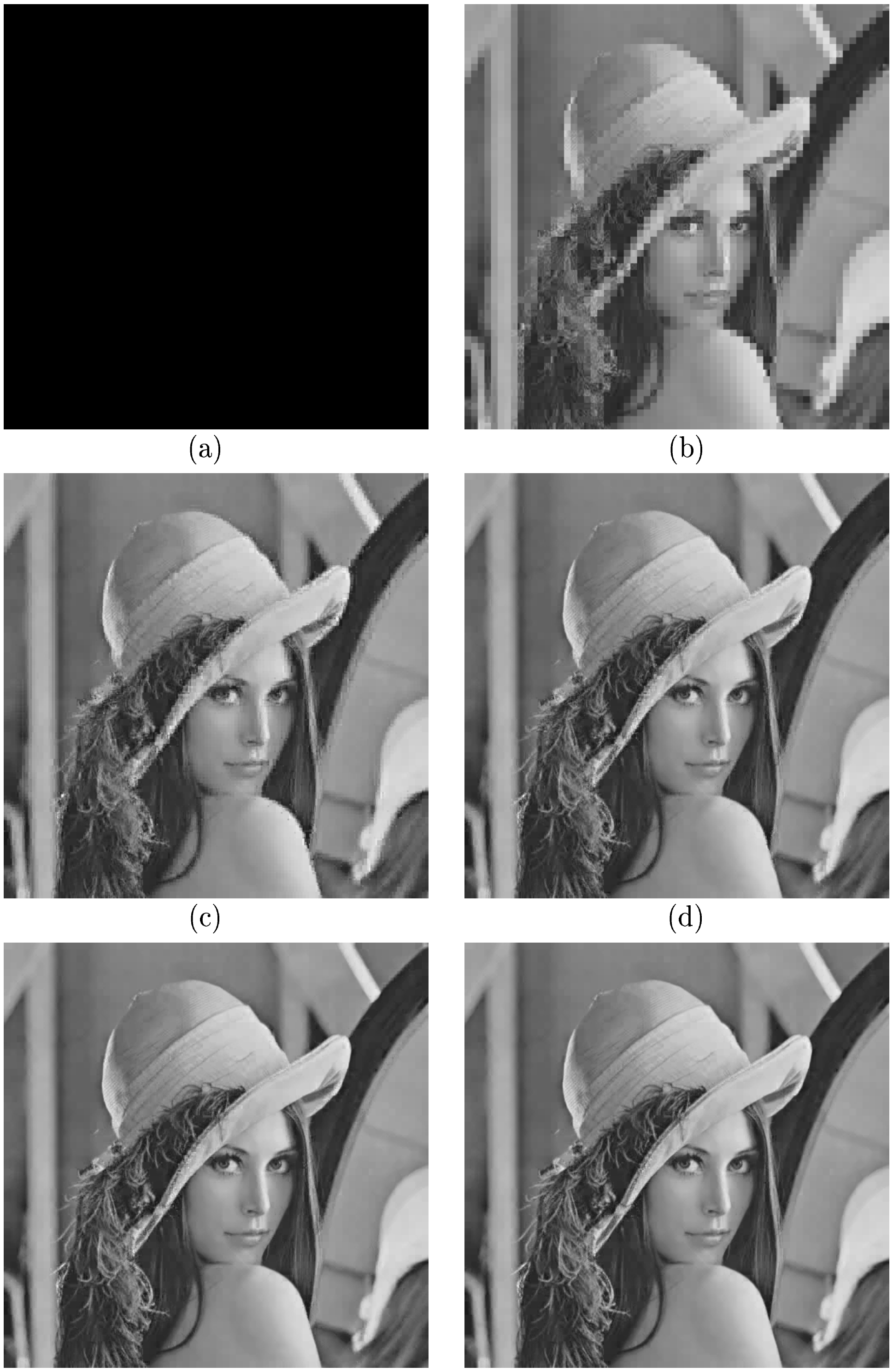


Figure 6.2 Five iterations of the decoding process.

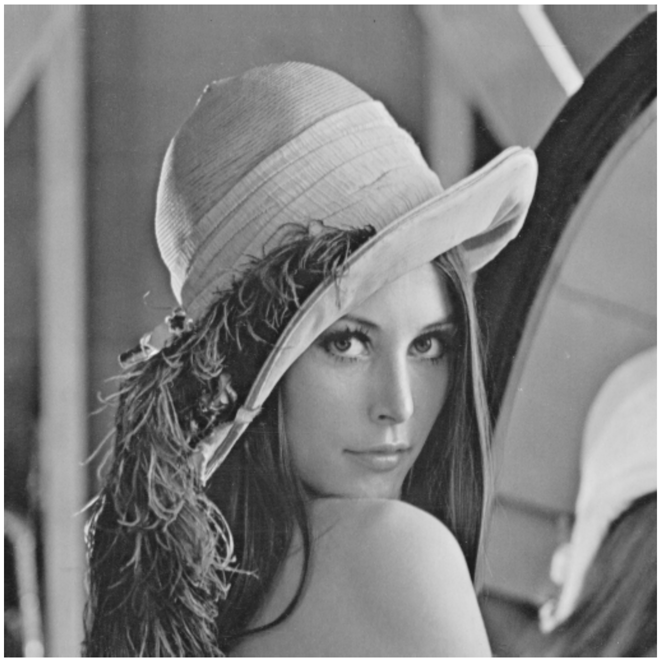


Figure 6.3 Original 512 × 512, 8 bits/pixel Lena image.

6.2 Test Results

Figure 6.2 shows the iterative decoding of the 512×512 test image Lena[1]. This image was compressed to 0.43 bits/pixel using 8×8 range blocks. Beginning with an initial 512×512 black (all zero) image, the decoder generates a sequence of images that converge to the decoded Lena image. Figure 6.2 shows images resulting from the first five iterations having PSNRs of (b) 24.22 dB, (c) 27.74 dB, (d) 30.42 dB, (e) 32.21 dB and (f) 33.24 dB. After about 10 iterations, at a PSNR of 34.50 dB, the change in the image becomes negligible. The original Lena image is shown in Figure 6.3 and the final decoded image is shown in Figure 6.4. Figures 6.5 and 6.6 show the Lena image compressed at 0.22 bpp (31.2 dB PSNR) and 0.15 bpp (29.2 dB PSNR).

Figure 6.7 shows PSNR vs. bitrate for compression of the 512×512 Lena image with the method of this chapter (curves u,v, and w) and other fractal-based methods (see Table 2.1 for references). In this figure, the curves "u," "v," and "w" correspond to compression using range blocks of size 16×16, 8×8, and 4×4, respectively. Figure

[1]Available from "ftp://ipl.rpi.edu/pub/image/still/usc/gray/" at the time of this writing.

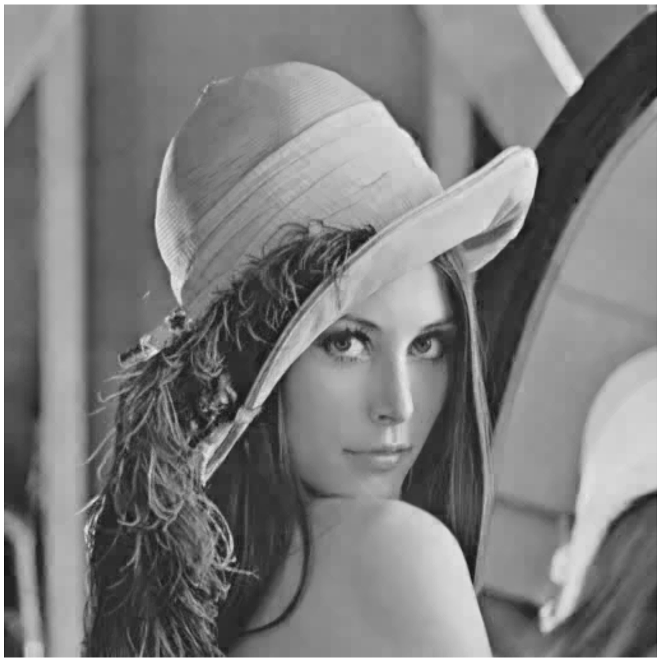


Figure 6.4 Decoded Lena image: 0.43 bits/pixel at 34.5 dB PSNR.

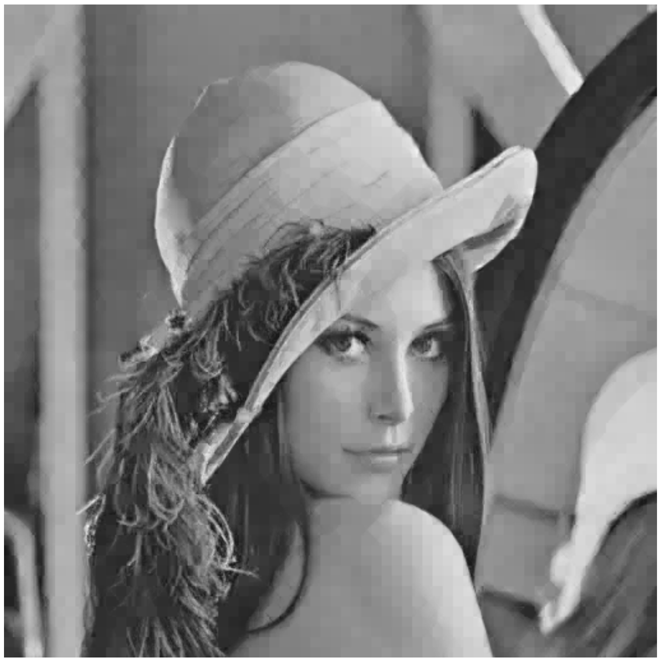


Figure 6.5 Decoded Lena image: 0.22 bits/pixel at 31.2 dB PSNR.

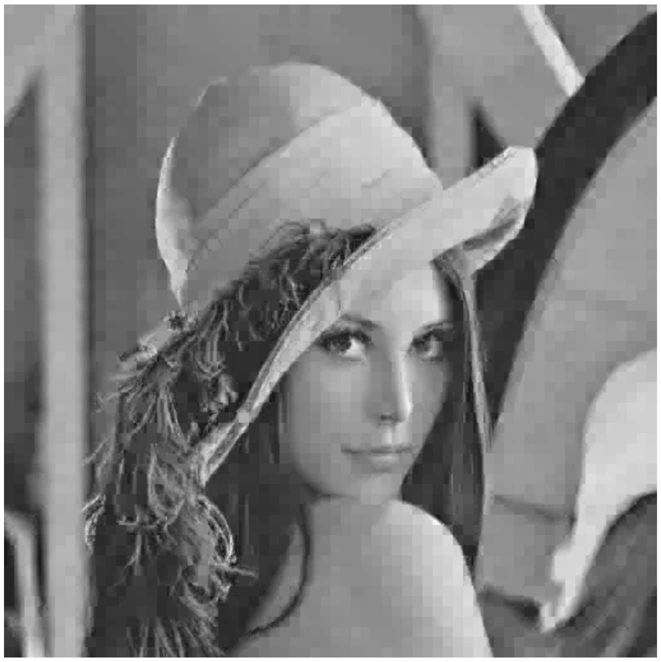


Figure 6.6 Decoded Lena image: 0.15 bits/pixel at 29.2 dB PSNR.

6.8 shows a similar comparison with some nonfractal methods (DCT-based and wavelet based). The curve "o" corresponds to an optimized version of baseline JPEG coding method [128]. See Table 2.1 for other references.

At bitrates below 0.2 bpp, using 16×16 range blocks provides a better performance compared to using 8×8 blocks. Figures 6.9, 6.10, and 6.11 show decoded images using 16×16 range blocks at bitrates of 0.088, 0.038, and 0.019 bpp, and PSNRs of 28.6, 25.9, and 23.8 dB. The encoding compression times depends on the value λ and is typically about one minute on a Hewlett-Packard Apollo Series 735 workstation computer. The decoding time typically takes a few seconds if the encoding is done noncausally, and less than a second if it is done causally.

6.3 Comparison

These results suggest that in terms of compression ratio and PSNR,

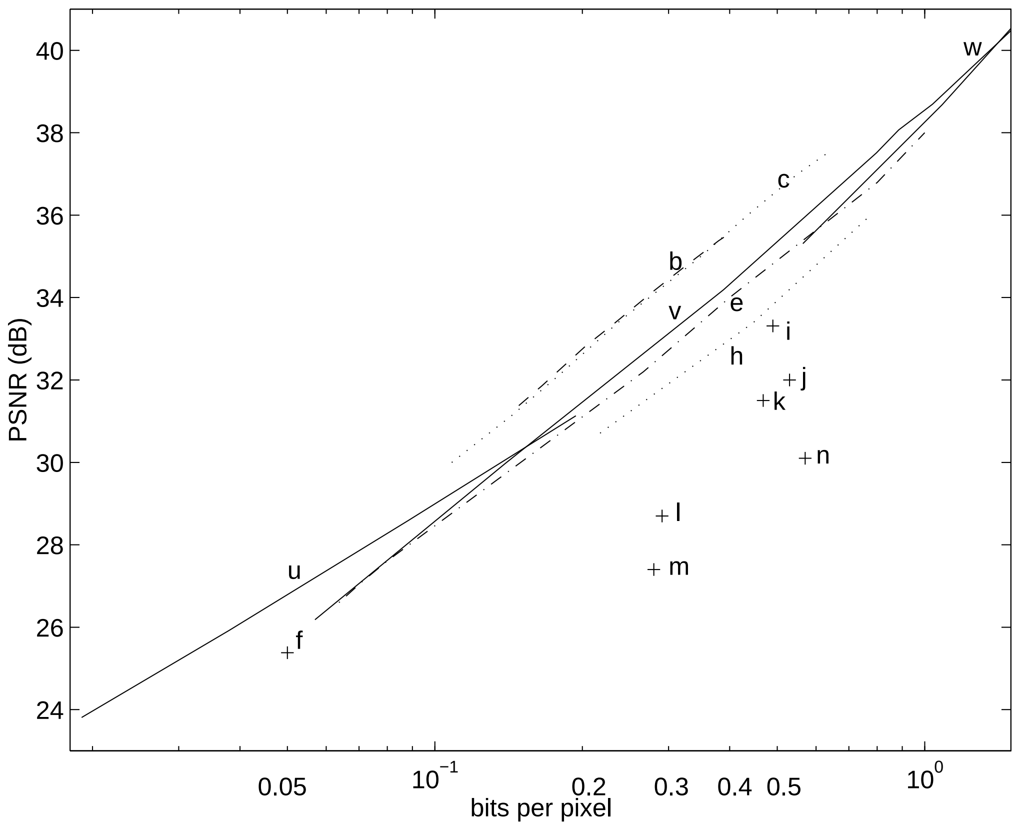


Figure 6.7 PSNR vs. bitrate for compression of the 512 × 512 Lena image with the the method of this chapter (u,v,w), and other fractal-based methods (see Table 2.1 for references).

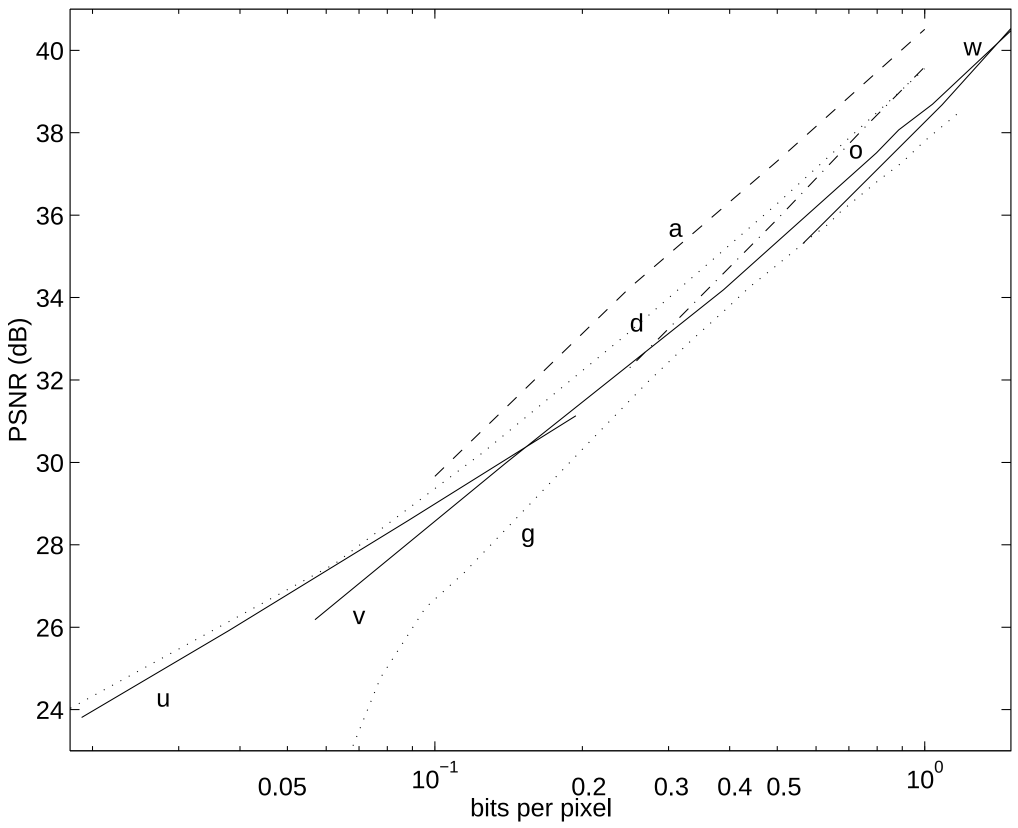


Figure 6.8 PSNR vs. bitrate for compression of the 512 × 512 Lena image with the the method of this chapter (u,v,w), and some nonfractal methods (see Table 2.1 for references).

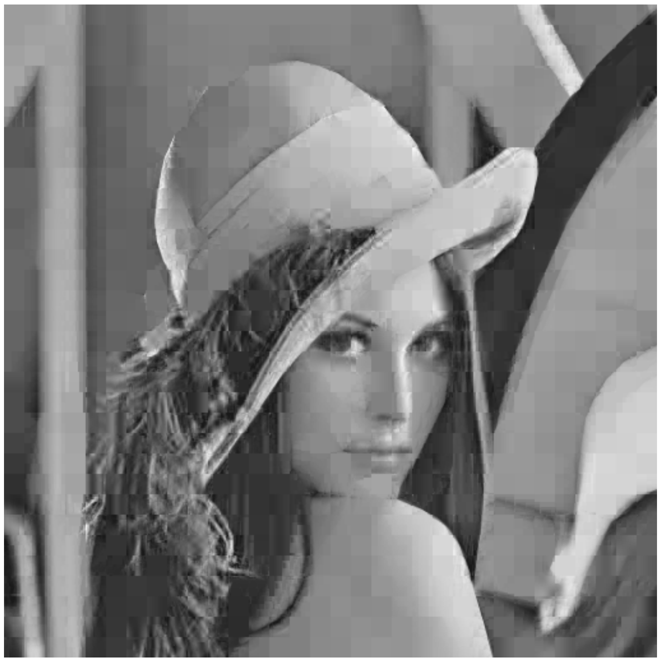


Figure 6.9 Decoded Lena image: 0.088 bits/pixel at 28.6 dB PSNR.

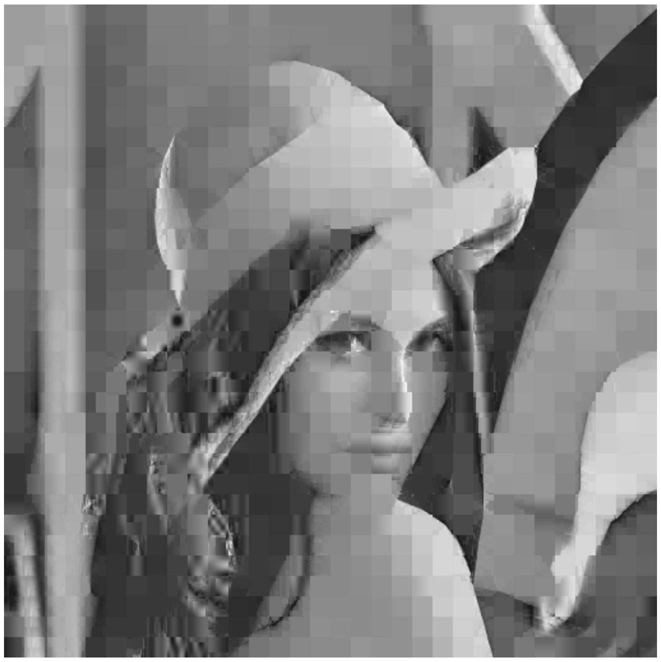


Figure 6.10 Decoded Lena image: 0.038 bits/pixel at 25.9 dB PSNR.

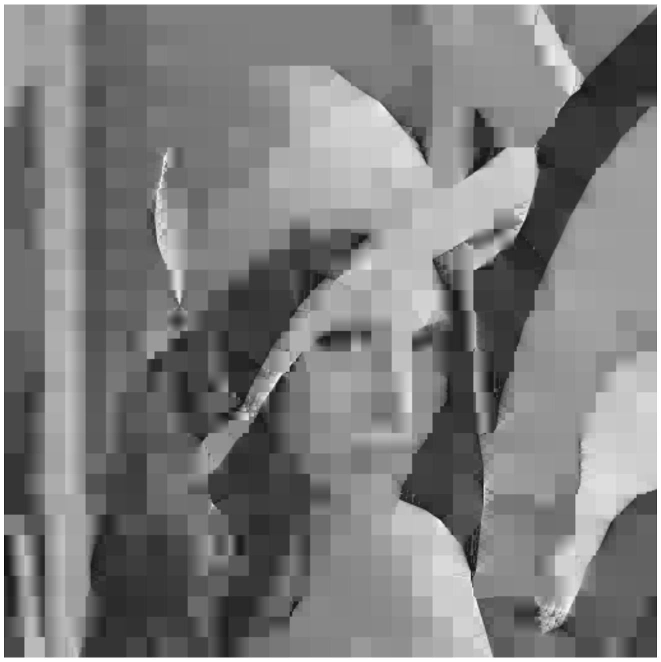


Figure 6.11 Decoded Lena image: 0.019 bits/pixel at 23.8 dB PSNR.

- The performance of the proposed method, is far better than the original Jacquin's method and many of its variations, and better than most other published fractal methods. The only two other published fractal-based methods that are reported to perform better are those by Barthel et al. [119], and Rinaldo and Calvagno [47]. The former uses a quadtree partitioning, which is not used in the method proposed in this work, and seems to be able to improve the performance of the system. The latter uses matching of range and domain blocks in the wavelet domain and takes advantage of better compression properties of this domain.

- The proposed method performs better than typical implementations of JPEG. At very low bitrates, for the same level of image quality, the method can compress more than 3 times the JPEG implementation. Compared to an optimized version of JPEG [128], the performance of the system is the same at around 0.3 bpp, but somewhat behind at higher bitrates. At very low bitrates, the performance of the proposed system is very close to the wavelet-based zerotree methods and does a much better job of preserving sharp edges. However, the blockiness of the images is disturbing at very low bitrates.

- In contrast to most published fractal image coders, the proposed method can approximate an image with arbitrarily small distortion and the compression time only moderately increases with the increase of PSNR of the decoded image.

6.4 Conclusions

In this chapter, sample experimental results for the algorithm of Chapter 5 were provided. These results show that the proposed method is a highly efficient fractal coder and gives a compression ratio 4 to 5 times better than the original Jacquin's method, and is among the top three fractal coding methods ever reported. The coder works very well in coding sharp large edges, even at very low bitrates.

CHAPTER 7

COMPRESSION OF VIDEO SEQUENCES

In this chapter, we present results of applying the algorithm of Chapter 5 to natural video sequences.

7.1 The Approach

The method of Chapter 5 can be easily extended to encoding of image sequences. The extension can be done in either of the following two ways:

(1) By considering a sequence of images as a three dimensional set and use 3-D blocks instead of 2-D blocks. In this case, 3-D transform coding becomes a special case of the proposed method when only FLBs are allowed for encoding image 3-D blocks.

(2) By considering a sequence of images as a series of two dimensional images and for the range blocks of any image, allow the domain blocks to be chosen from the neighborhood of the range block which includes parts (or possibly all) of neighboring frames.

In fact, when only one SSLB (with its coefficient restricted to 1) and no HSLBs are allowed, block prediction methods like DPCM, and adaptive block prediction methods like block motion compensation [173], become special cases of this algorithm. Also, hybrid coding methods [173] like motion compensation combined with transform coding of residual errors (which is equivalent to DPCM of transform coefficients of blocks shifted according to motion vectors) is equivalent to using the basis blocks of the transform coding as FLBs and allowing one SSLB.

The second method is used for the test results provided in this chapter. It treats the temporal direction differently from the spatial directions, as they are very different in nature, and it is very close to the well established method of motion compensation which has proven to be a very powerful method of coding for video sequences.

The above method is applied to natural image sequences, some examples of which are brought in this chapter. All of the experiments reported in this section are done under the following settings:

- The rate distortion optimized orthogonal matching pursuit is used for the selection of dictionary vectors.
- For the initial statistics of the entropy coders, no prior training is done. The encoder/decoder gathered these statistics as they progressed, i.e., an adaptive entropy coder is used.
- No pixel shuffling (rotation/reflection) of domain blocks is done.
- The first frame is coded only using FLBs.
- Each of the following frames extract their ALBs from the previous frame. For both SSLBs and HSLBs, a spiral pattern is traversed around the position of the range block in the previous frame. As the selections are made from previous frame, the coding is being done causally, and neither SSLBs nor HSLBs are restricted to be only on top or left of the range block.
- Due to a causal encoding process, the decoding process is noniterative and the decoder reaches its steady-state in only one iteration (see Sections 5.2.3 and 5.5.1). In other words, because the adaptive part of the dictionary for each block is made from the previous frame, the encoder is causal. This makes a noniterative decoder possible as the decoder is aware of the dictionary used by the encoder at all times.

Higher-scale dictionary blocks are taken from the neighborhood of the range block in the filtered, subsampled previous frame. These blocks are introduced to let the encoder

Table 7.1 Parameters of the test sequence Miss America

Number of frames	150
Format	grayscale CIF/PAL
Number of grayscale levels	256 (8 bpp)
Width of frames	360 pixel
Height of frames	288 pixel
Frame Rate	25 Hz

take advantage of the inter-scale redundancies in the sequences and make the fractal element of the coding. Same-scale dictionary blocks are taken directly from the neighborhood of the range block in the previous frame. These blocks are introduced to let the coder take advantage of the intra-scale redundancies in the image, similar to motion compensation methods.

7.2 Experimental Results

The standard Miss America test sequence[1] was used obtained for the experimental results of this chapter. Table 7.1 shows the parameters of this sequence.

7.2.1 Experiment 1

Figure 7.1 shows three original (a), (c), (e) and decoded (b), (d), (f) frames of the 12.5 Hz (25 Hz subsampled by 2) Miss America video sequence. The test sequence was coded at 80 Kbits/sec with an average PSNR of 36–37 dB. For these results, 8×8 range blocks were used. The 64 DCT basis blocks were used as the fixed dictionary blocks. For this experiment, 128 causal same-scale and 64 higher-scale blocks were used for the adaptive part of the dictionary. The higher-scale dictionary blocks were obtained from lowpass filtering followed by subsampling of 16×16 domain blocks. No rotation or reflection of domain blocks was used.

[1]Available from "ftp://ipl.rpi.edu/pub/image/sequence/missa/" at the time of this writing.

Table 7.2 General coding parameters

M_1 and M_2	8
L	2
FLB type	2-D DCT basis vectors
P_F	64
P_{A_i}	200
Number of SSLBs	100
Number of HSLBs	100

Table 7.3 Coding parameters for Section 7.2.2

Number of coded frames	150
Frame step	1
λ	41
Step size in fetching SSLBs and HSLBs	2
Method of gathering statistics	each frame individually
Initial statistics for each frame	statistics of previous frame
Average bit/pixel	0.124
Average bit/frame	12.8 Kbit/frame
Bitrate	321 Kbit/sec
Average number of library blocks used per image block	1.52
Average PSNR	38.0 (dB)
Average CPU time for encoding per frame	23.0 sec
Average CPU time for decoding per frame	0.45 sec

As mentioned earlier, both the encoder and the decoder use the previous decoded frame for making the adaptive part of the dictionary for each range block in the current frame of the sequence.

7.2.2 Experiment 2

Table 7.2 provides the parameter setting common to this experiment and the following experiments.

Table 7.3 gives the additional settings specific to the experiment of this section and the main coding results. In this test, all the 150 frames of the sequence were coded at 321 Kbit/sec with an average PSNR of 38.0 dB. Figures 7.2 and 7.3 show frame number

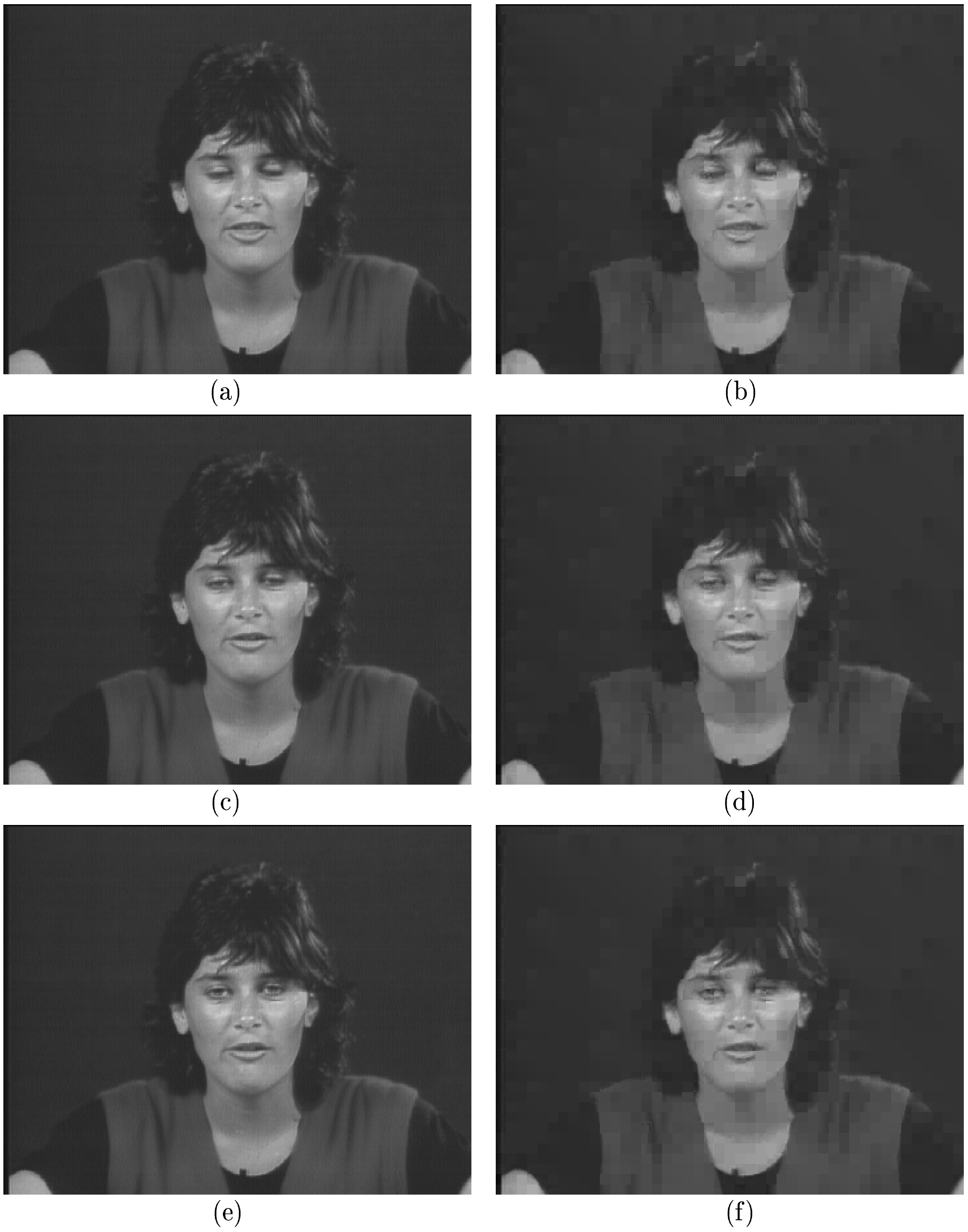


Figure 7.1 Three original (a, c, e) and decoded (b, d, f) frames of the Miss America video sequence.

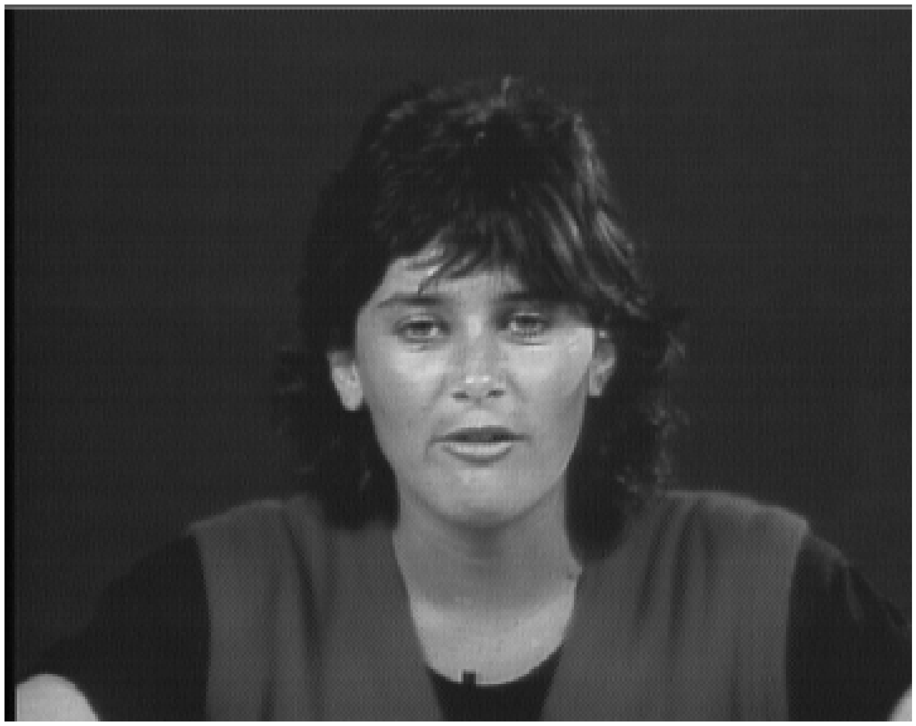


Figure 7.2 Frame number 11 of the original Miss America sequence.

Table 7.4 The histogram of number of blocks used for encoding image blocks of the last frame in Section 7.2.2

# of selections	0	1	2	3	4	5	6	7	8	9	10	11	12	13	rest
count	0	1038	442	101	10	24	0	0	0	3	0	1	0	1	0

11 of the original and the decoded sequence. The difference between these two is shown in Figure 7.4.

Figures 7.5 and 7.6 show the plot of bit per pixel and PSNR vs. frame number for the coded sequence. The high rate and low quality of the first frame is in part due to the fact that it does not use any ALBs and, in part, because it does not use any initial statistics for its entropy coding. Figure 7.7 shows the average number of library blocks used per image block in the coded sequence. Except for the blocks of the first frame, which use an average of more than 2 vectors from the dictionary, the following frames on the average use between 1.4 to 1.6 selections per range block. Table 7.4 shows the histogram of the number of blocks used for encoding image blocks of the last frame.

Table 7.5 Coding parameters for Section 7.2.3

Number of coded frames	50
Frame step	1
λ	41
Step size in fetching SSLBs and HSLBs	1
Method of gathering statistics	cumulative
Initial statistics for each frame	cumulative statistics of all previous frames
Average bit/pixel	0.10
Average bit/frame	10.4 Kbit/frame
Bitrate	261 Kbit/sec
Average number of library blocks used per image block	1.08
Average PSNR	37.4 (dB)
Average CPU time for encoding per frame	22.6 sec
Average CPU time for decoding per frame	0.43 sec

7.2.3 Experiment 3

In this experiment, only the first 50 frames of the sequence were coded. The step size in fetching SSLBs and HSLBs is set to 1 and the cumulative statistics of all previous frames (except the first frame) is used for the entropy coder. The complete setting of parameters of this experiment and the coding results are provided in Table 7.5. In Figure 7.8, the decoded frame number 11 of the sequence is shown. Figures 7.9, 7.10, and 7.11 show the bit per pixel, PSNR, and the average number of selections per range block for each coded frame. Table 7.6 presents the histogram of the number of blocks used for encoding image blocks of the last 48 frames of the sequence. Changing the search resolution and using the cumulative histogram instead of the histogram of the previous frame, has significantly reduced the average number of blocks used for each range block and has reduced the bitrate compared to the first 50 frames of experiment of Section 7.2.2. The histograms of the FLBs, the SSLBs, and the HSLBs are shown in Figures 7.12, 7.13, and 7.14, respectively. These histograms clearly show that the selection process favors the SSLBs far more than the HSLBs and the FLBs. Even among the SSLBs, the first few SSLBs are by far the best choices for finding matches for the range blocks.

Table 7.6 The histogram of number of blocks used for encoding image blocks of the last 48 frames of the sequence in Section 7.2.3

number of selections	0	1	2	3	4	5	6	7	8	9	10	rest
count	0	77562	0	1309	174	310	0	19	3	0	3	0

Table 7.7 Coding parameters for Section 7.2.4

Number of coded frames	50
Frame step	3
λ	576
Step size in fetching SSLBs and HSLBs	1
Method of gathering statistics	cumulative
Initial statistics for each frame	cumulative statistics of all previous frames
Average bit/pixel	0.076
Average bit/frame	7.8 Kbit/frame
Bitrate	65.2 Kbit/sec
Average number of library blocks used per image block	1.02
Average PSNR	33.5 dB
Average CPU time for encoding per frame	25.8 sec
Average CPU time for decoding per frame	0.49 sec

7.2.4 Experiment 4

This experiment provides an example of coding at lower bitrates. The frame step is changed from 1 to 3 (skips 2 frames) and the value of λ is increased to change the trade-off between rate and distortion in favor of lower rates. The setting and the coding results of this experiment are shown in Table 7.7. Figure 7.15 shows the decoded frame number 10 of the sequence (the fourth decoded frame). Some blockiness has appeared in the decoded frames. Figures 7.16, 7.17, and 7.18 show the bit per pixel, PSNR, and the average number of library blocks used per image block as a function of the coded frame number. Table 7.8 shows the histogram of the number of blocks used for encoding image blocks of the last 48 frames of the sequence. The cumulative histograms of the usage of FLBs, SSLBs, and HSLBs for the last 48 frames are shown in Figures 7.19, 7.20, and 7.21. Although the SSLBs, especially the first few ones, are still the dominant part of the

Table 7.8 The histogram of the number of blocks used for encoding image blocks of the last 48 frames of the sequence in Section 7.2.4

number of selections	0	1	2	3	4	5	6	7	rest
count	0	78938	0	418	0	22	1	1	0

Table 7.9 Bitrate and PSNR for the compression of the first 50 frames of the Miss America sequence, obtained from the proposed method, the H.261, and the H.263.

	Bitrate	Average PSNR
Proposed Method	261 Kbits/sec	37.4 dB
H.261	257 Kbits/sec	36.9 dB
H.263	282 Kbits/sec	32.6 dB

library, the usage of the FLBs and HSLBs has increased significantly. This is because the frame step size has increased from 1 to 3, which makes the subsequent decoded frames less similar, and the SSLBs which typically represent motion in the scene do not give as good a match as before.

7.3 Comparison with H.261 and H.263 Standards

In this section, we compare the performance of the proposed video compression system with that of coders based on the H.261 [174] and the H.263 [175] international standards. For H.261, the PVRG implementation [176] is used. For H.263, the Telenor R&D implementation [177] is used. The results of experiment 3 (Section 7.2.3) are used for this comparison. Table 7.9 shows the results of compression of the first 50 frames of the Miss America sequence using these three methods[2].

The original frame number 11 of this sequence, and its decompressed versions from the above three methods are shown in Figures 7.22, 7.23, 7.24, and 7.25.

[2]For the PVRG, the command "p64 -CIF -a 0 -b 49 -f 25 -x 520000 input" is used. For the Telenor software, the command "tmn -a 0 -b 49 -x 3 -S 0 -Z 25.0 -R 25.0 -q 9 -i input -o output" is used. In both cases, all the color components of the input sequence are set to 0.

From these results, it can be seen that, for this sequence, and for close bitrates, the proposed method, and the H.261 perform significantly better than the H.263 method in terms of PSNR. Visually, the quality of of the decompressed images using the proposed method is clearly better than the other two methods. While the H.261 decoded images suffer from significant blockiness at this bitrate, the H.263 result shows noise-like distortion. However, the encoding time for both the H.261 and the H.263 algorithms are much lees than that of the proposed method.

7.4 Conclusions

This chapter provided a new approach to generalized fractal coding of video sequences which seamlessly combines the motion compensation techniques with fractal techniques. It uses the efficient selection process of matching pursuit to make the optimal selection of FLBs, SSLBs, and HSLBs. In the competition of these subsets of the dictionary, experiments show that the SSLBs provide the best selection in the majority of cases. However, as the frame rate decreases and the quality of the matches found by motion compensation decrease, the system automatically selects more FLBs and HSLBs.

In comparison to other fractal video coding techniques, the approach of this chapter provides an efficient algorithm, which is among the best in the literature (see [131]–[133], [138], [139], [178]).

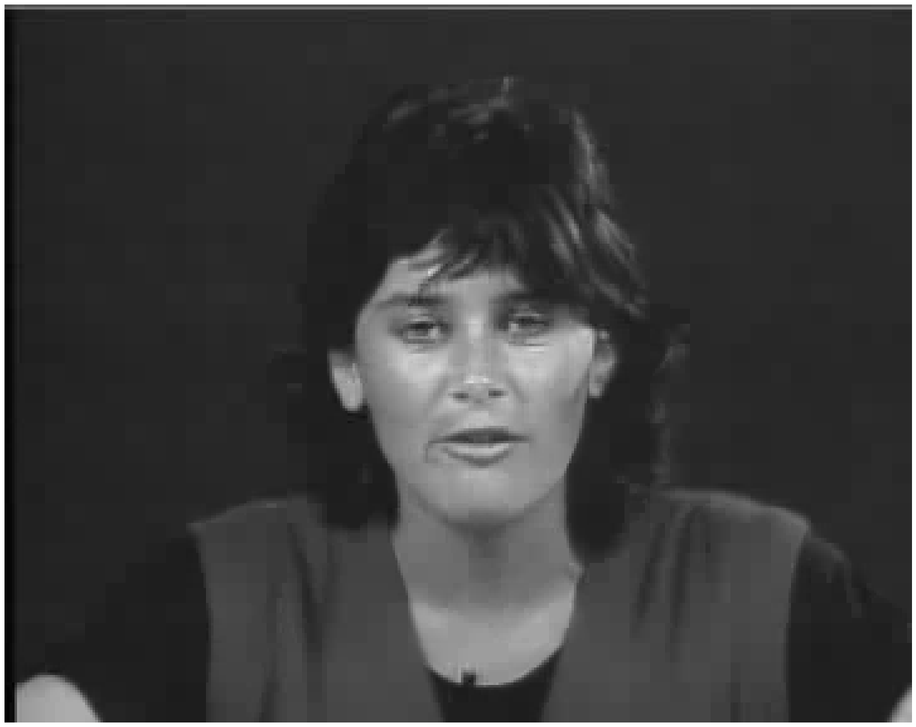


Figure 7.3 Decoded frame number 11 of the sequence in Section 7.2.2.

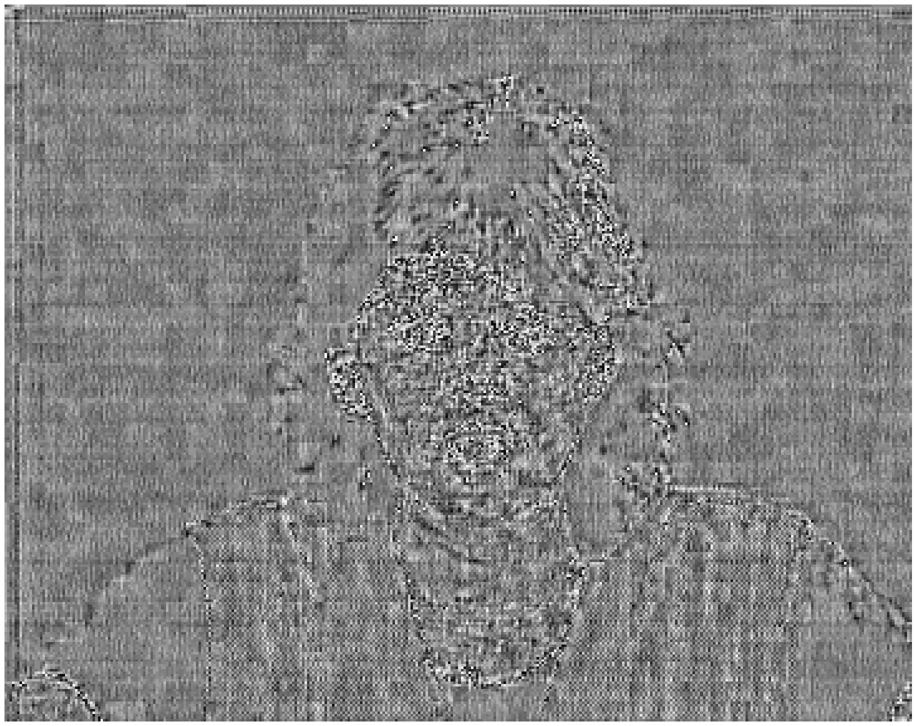


Figure 7.4 Error between frame 11 of the original sequence and its decoded version in Section 7.2.2.

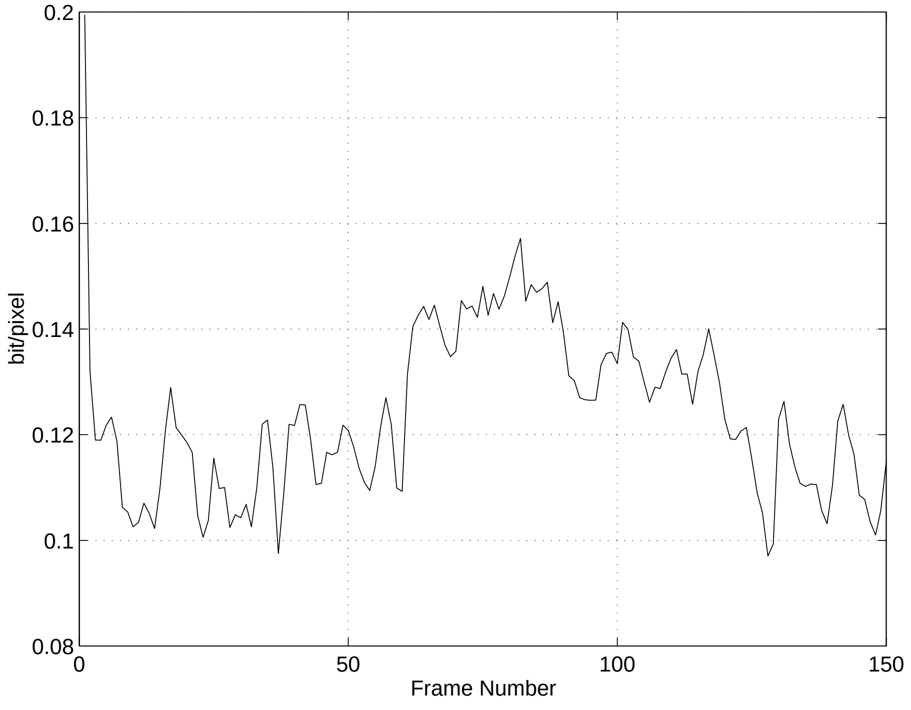


Figure 7.5 Bit/pixel for the Miss America sequence coded in Section 7.2.2.

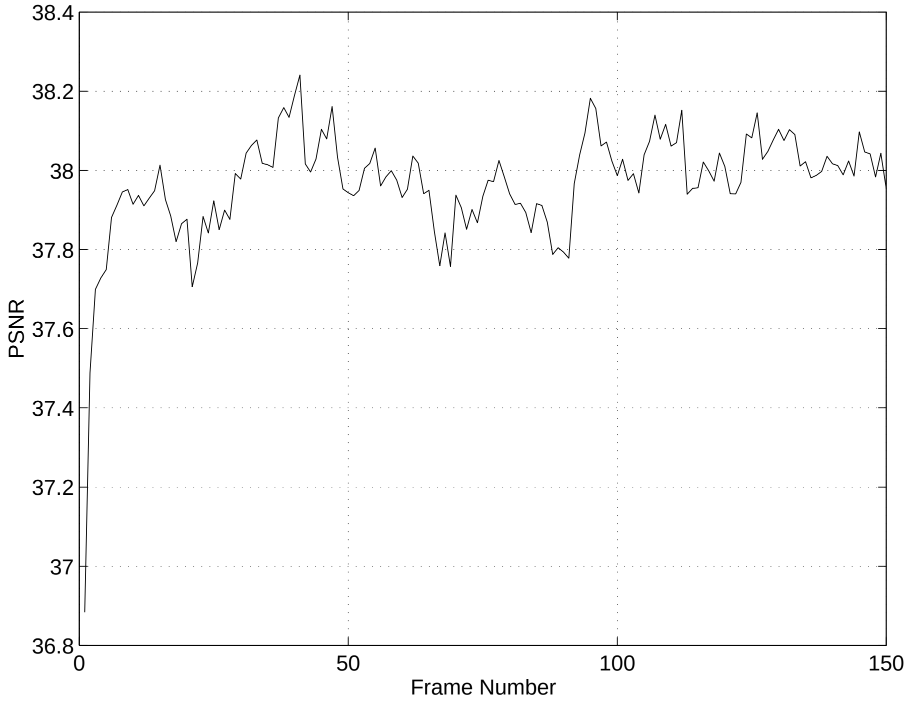


Figure 7.6 PSNR (in dB) for the Miss America sequence coded in Section 7.2.2.

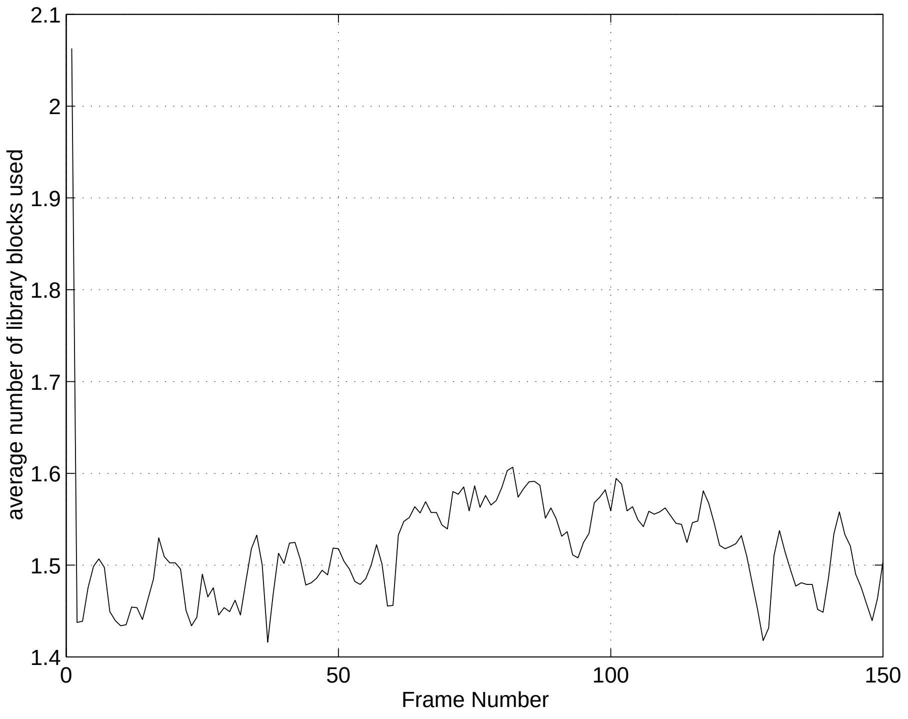


Figure 7.7 Average number of library blocks used per image block for coding of Miss America sequence coded in Section 7.2.2.

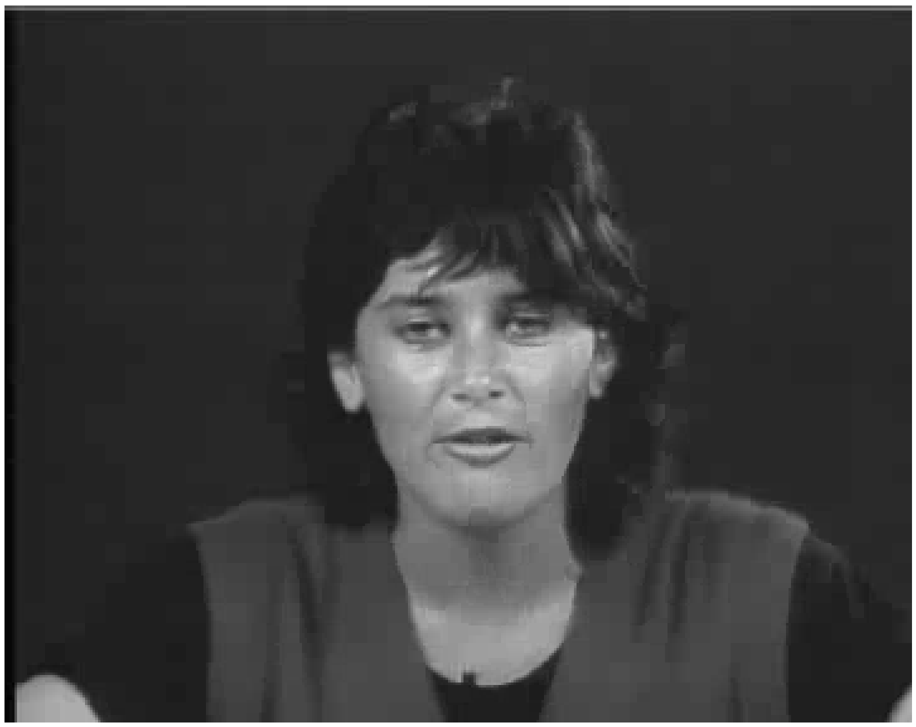


Figure 7.8 Decoded frame number 11 of the sequence in Section 7.2.3.

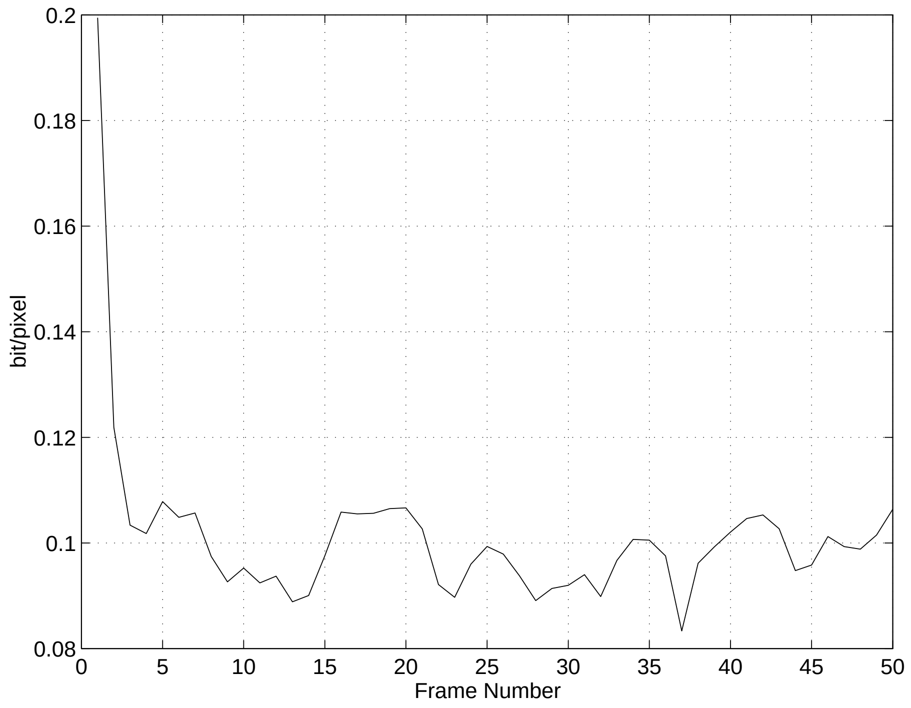


Figure 7.9 Bit/pixel for the Miss America sequence coded in Section 7.2.3.

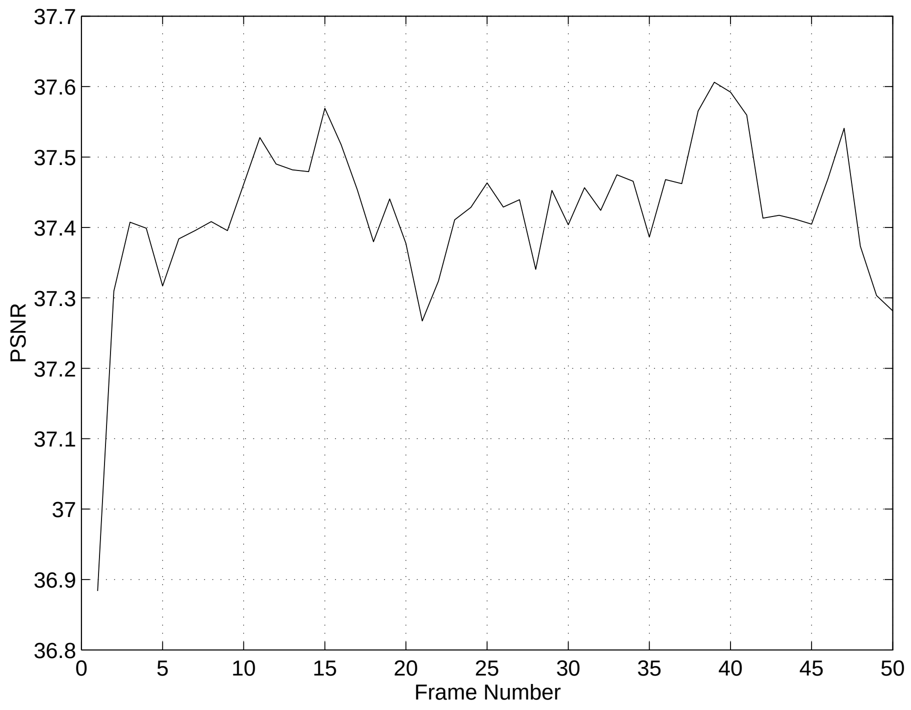


Figure 7.10 PSNR (in dB) for the Miss America sequence coded in Section 7.2.3.

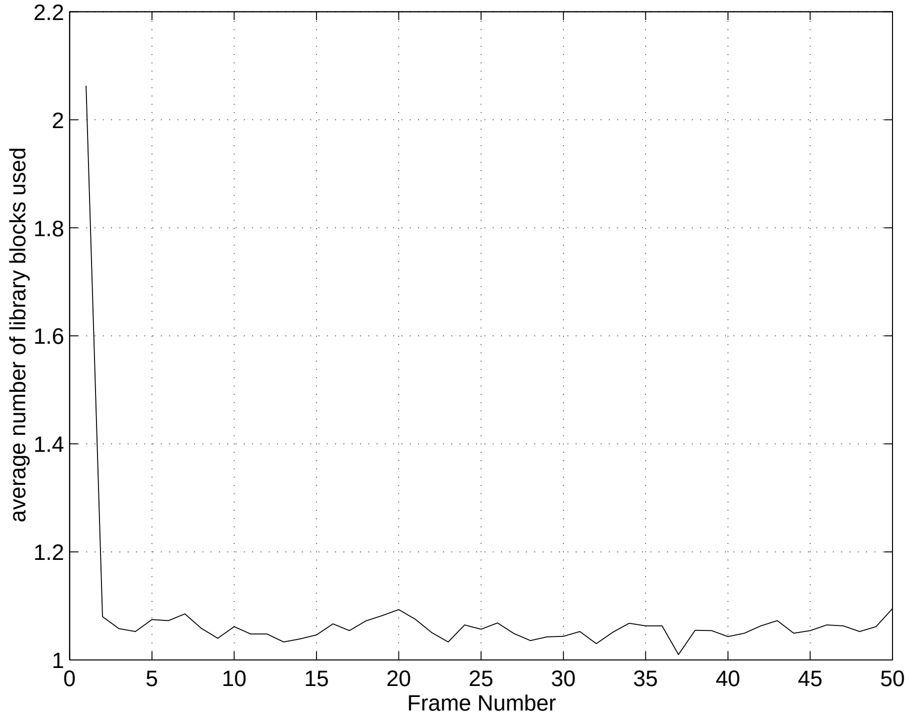


Figure 7.11 Average number of library blocks used per image block for coding of Miss America sequence coded in Section 7.2.3.

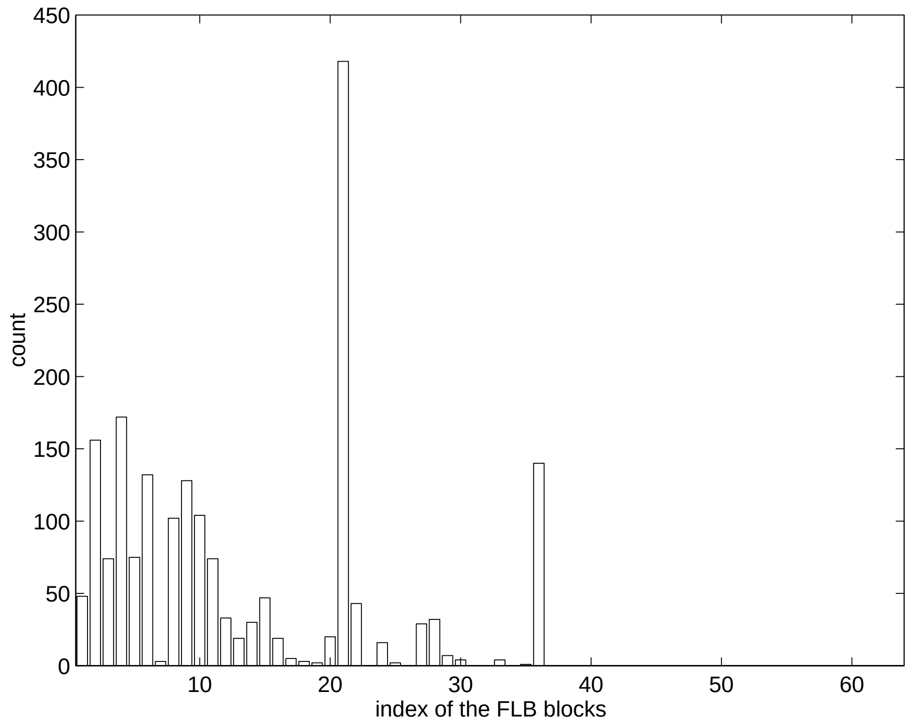


Figure 7.12 Cumulative histogram of FLBs used in the last 48 frames of the experiment of Section 7.2.3 (total count = 1942). Indices increase along a zigzag pattern in the 8×8 2-D DCT domain.

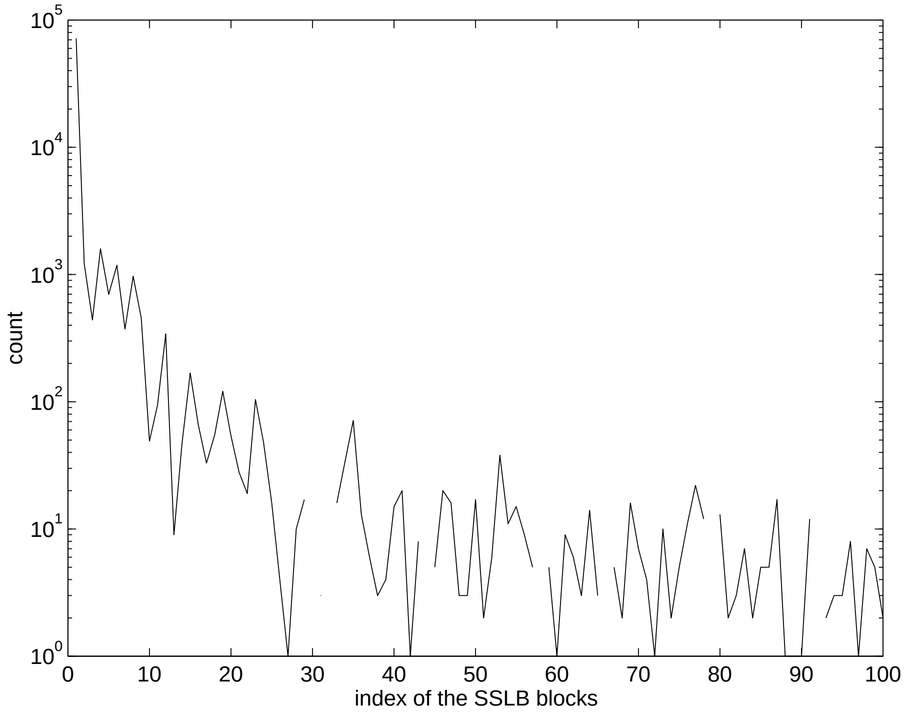


Figure 7.13 Cumulative histogram of SSLBs used in the last 48 frames of the experiment of Section 7.2.3 (total count = 80782). Indices increase along a spiral pattern in the previous frame centered at the location of the range block. Missing points represent zero occurrences.

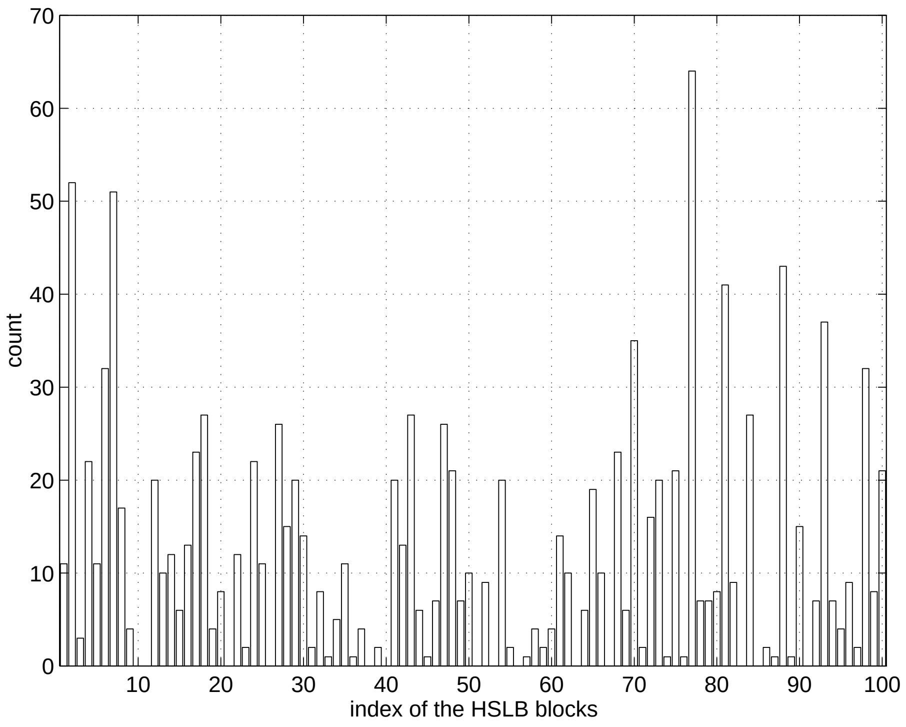


Figure 7.14 Cumulative histogram of HSLBs used in the last 48 frames of the experiment of Section 7.2.3 (total count = 1198). Indices increase along a spiral pattern in the previous frame centered at the location of the range block. Missing points represent zero occurrences.

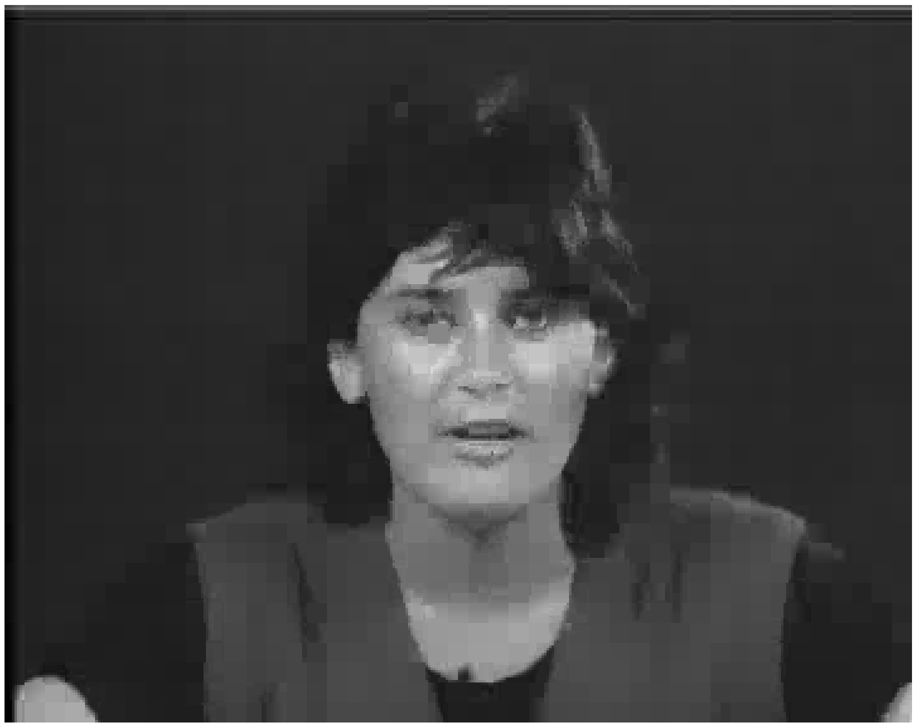


Figure 7.15 Decoded frame number 10 of the sequence in experiment in Section 7.2.4.

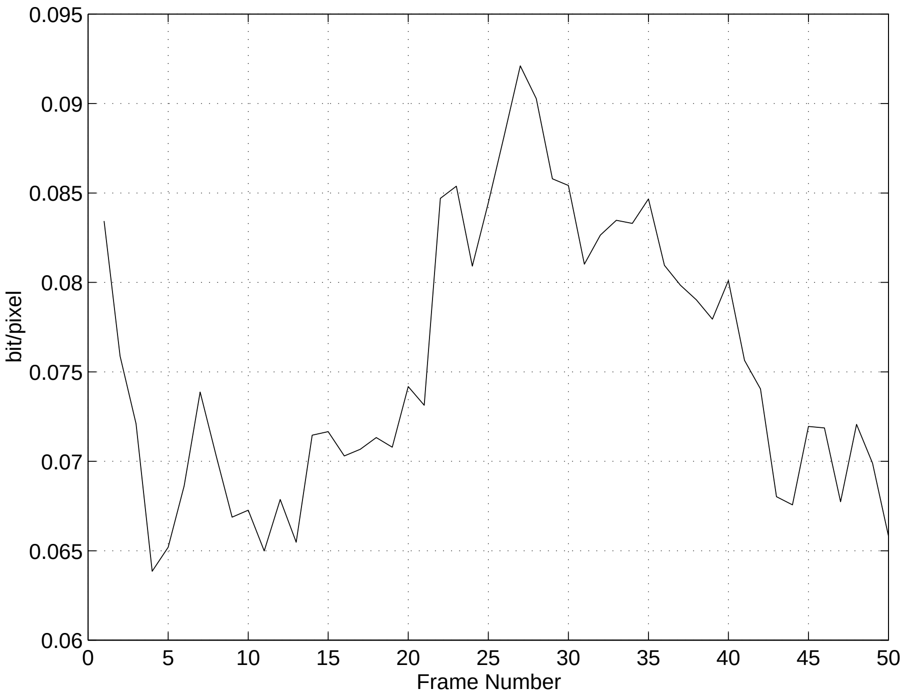


Figure 7.16 Bit/pixel for the Miss America sequence coded in Section 7.2.4.

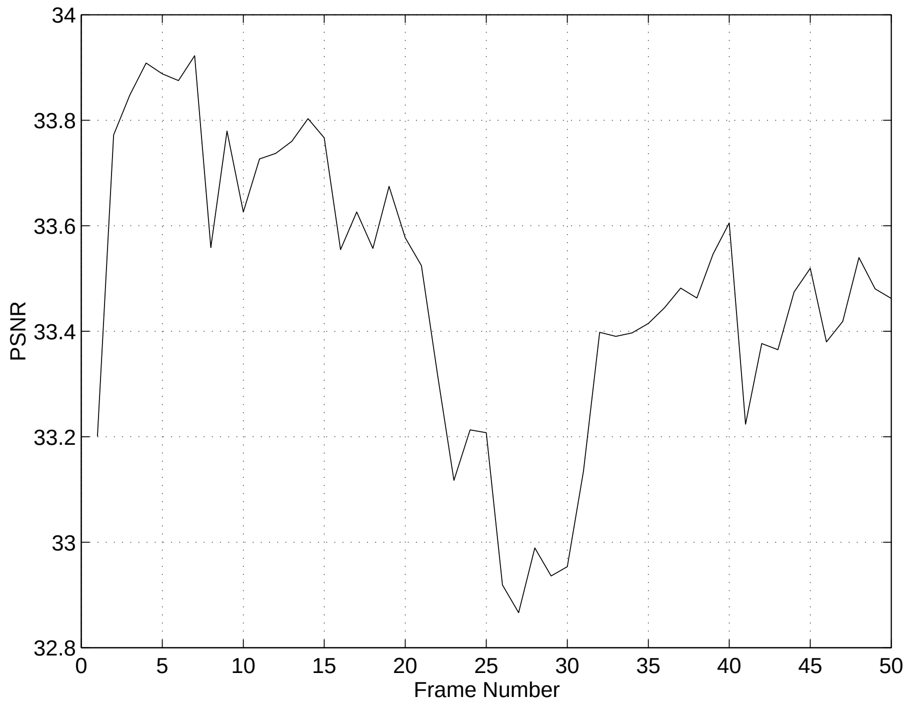


Figure 7.17 PSNR (in dB) for the Miss America sequence coded in Section 7.2.4.

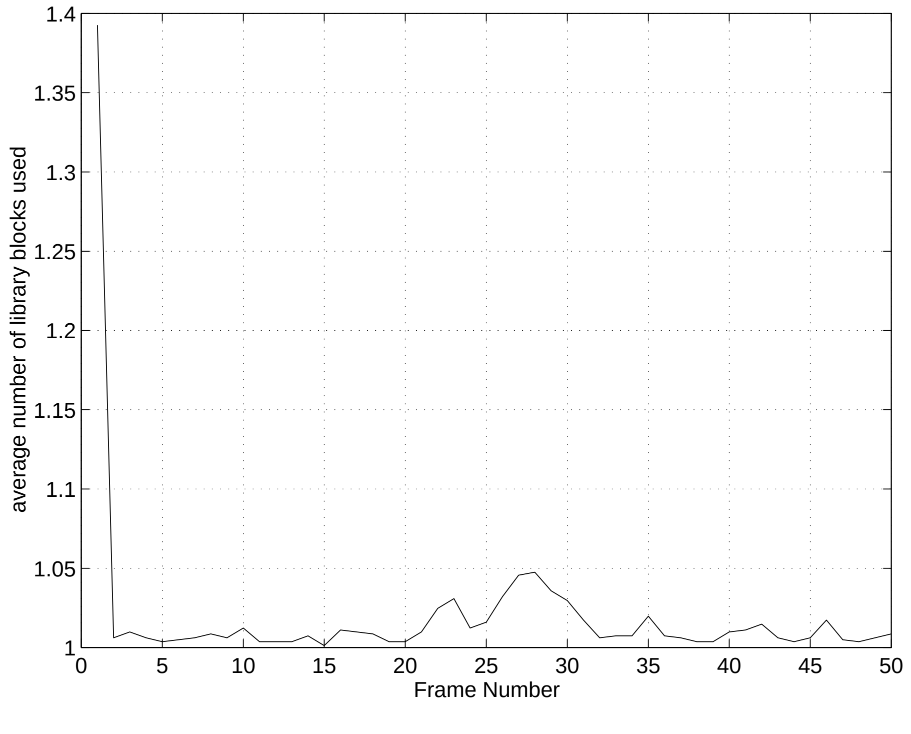


Figure 7.18 Average number of library blocks used per image block for coding of Miss America sequence coded in Section 7.2.4.

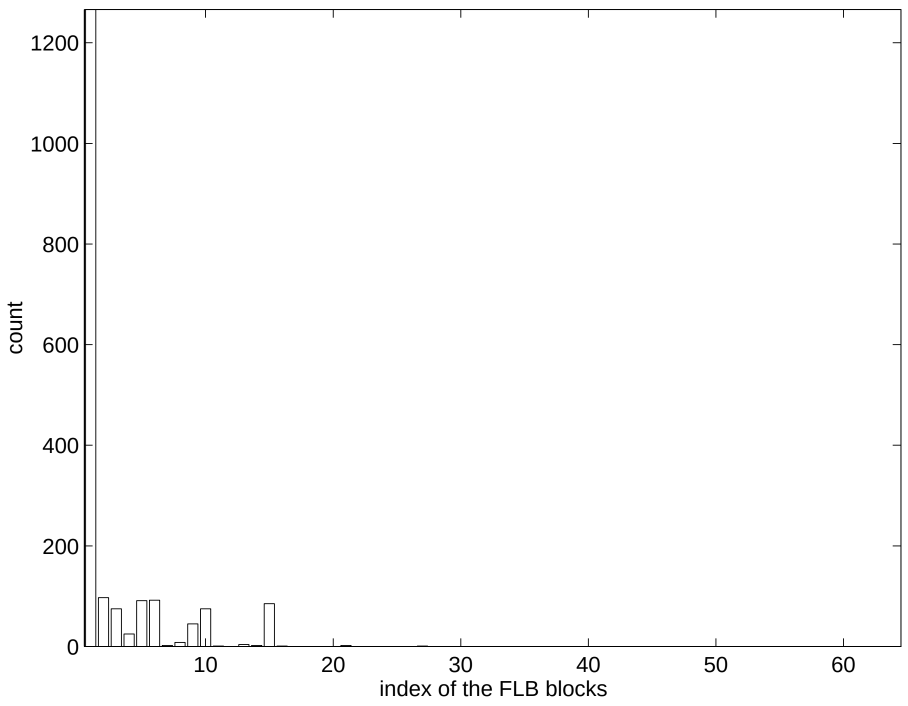


Figure 7.19 Cumulative histogram of FLBs used in the last 48 frames of the experiment of Section 7.2.4 (total count = 1872). Indices increase along a zigzag pattern in the 8×8 2-D DCT domain.

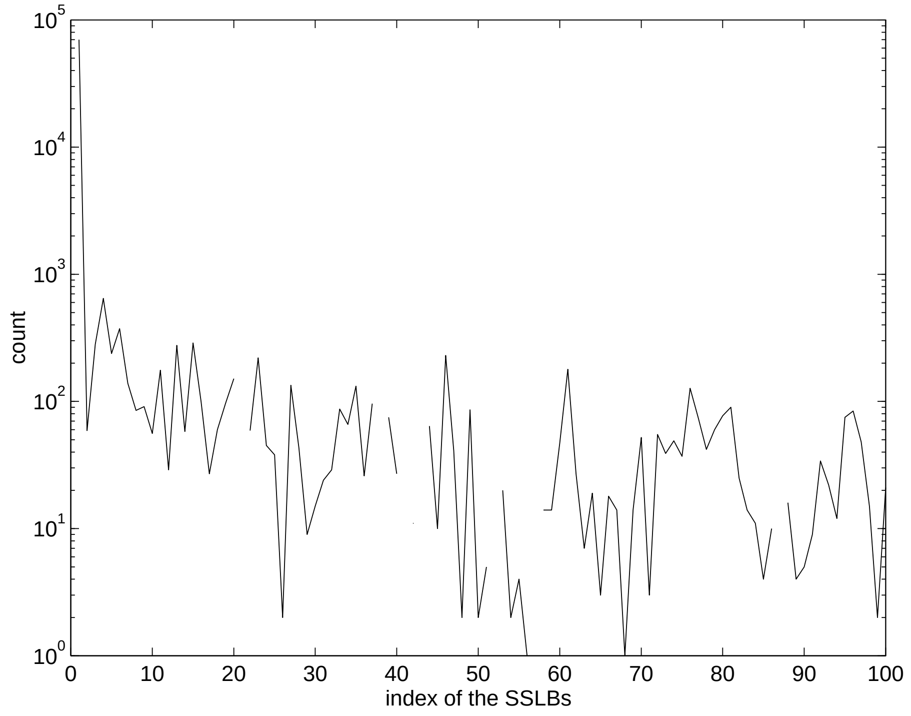


Figure 7.20 Cumulative histogram of SSLBs used in the last 48 frames of the experiment of Section 7.2.4 (total count = 76372). Indices increase along a spiral pattern in the previous frame centered at the location of the range block. Missing points represent zero occurrences.

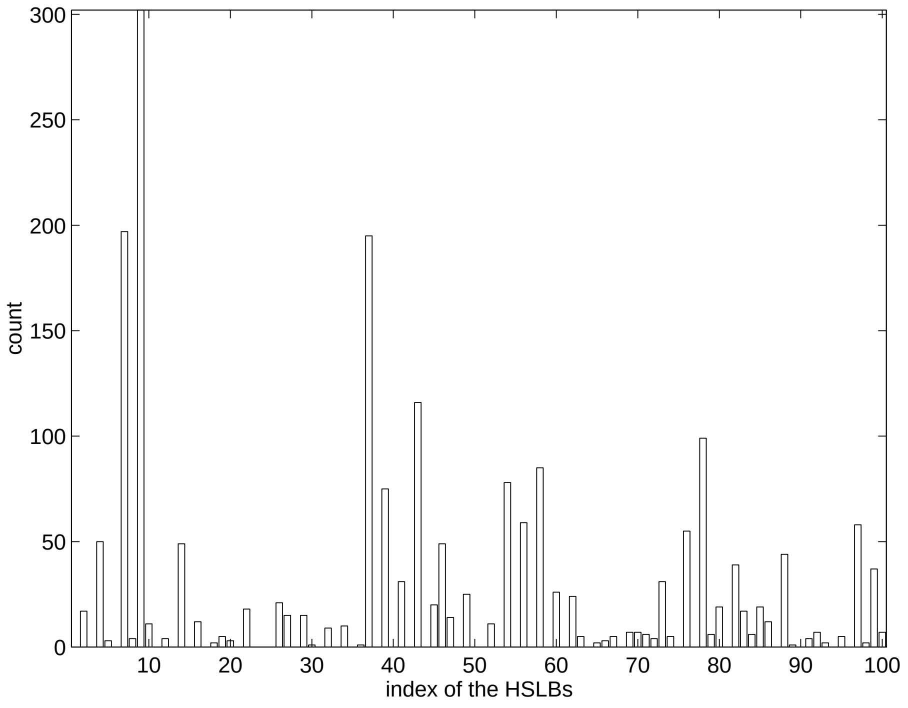


Figure 7.21 Cumulative histogram of HSLBs used in the last 48 frames of the experiment of Section 7.2.4 (total count = 2071). Indices increase along a spiral pattern in the previous frame centered at the location of the range block. Missing points represent zero occurrences.

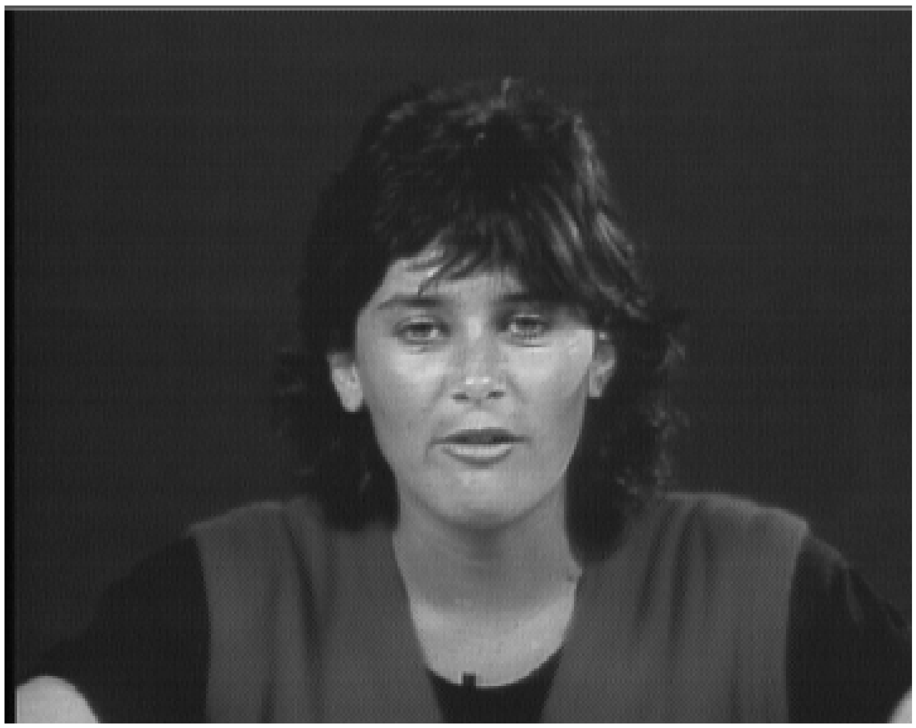


Figure 7.22 Frame number 11 of the original Miss America sequence.

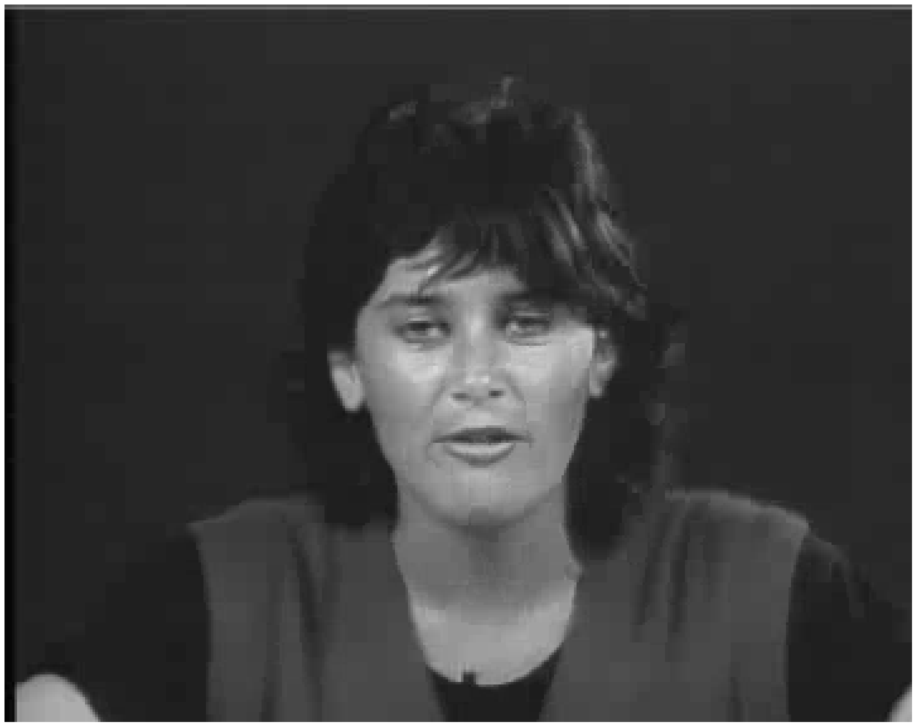


Figure 7.23 Decoded frame number 11 of the sequence in Test 3.

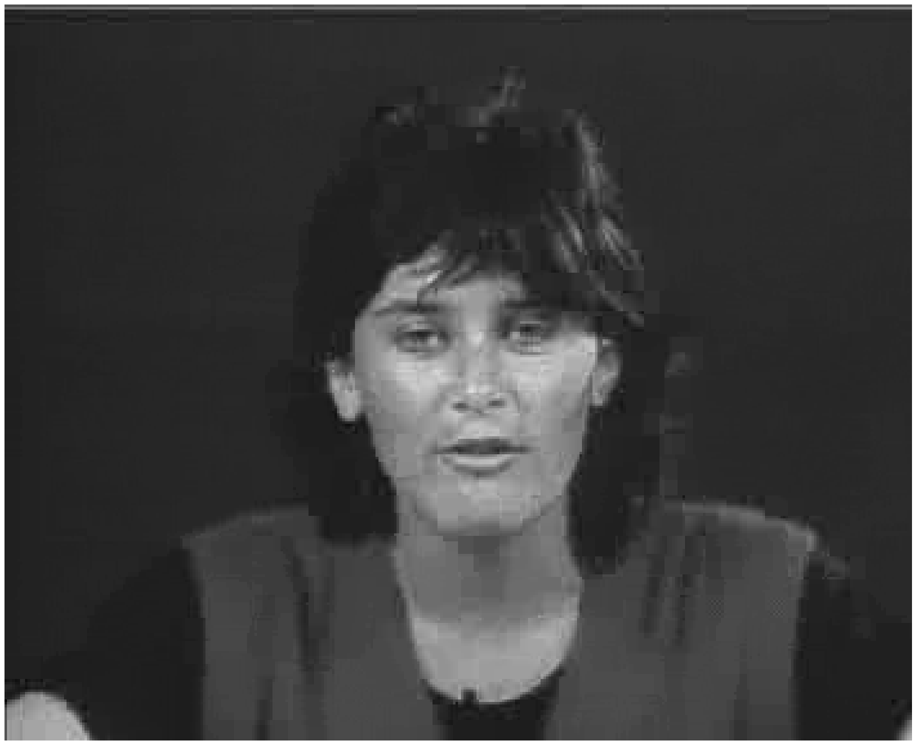


Figure 7.24 Decoded frame number 11 of the sequence by H.261.

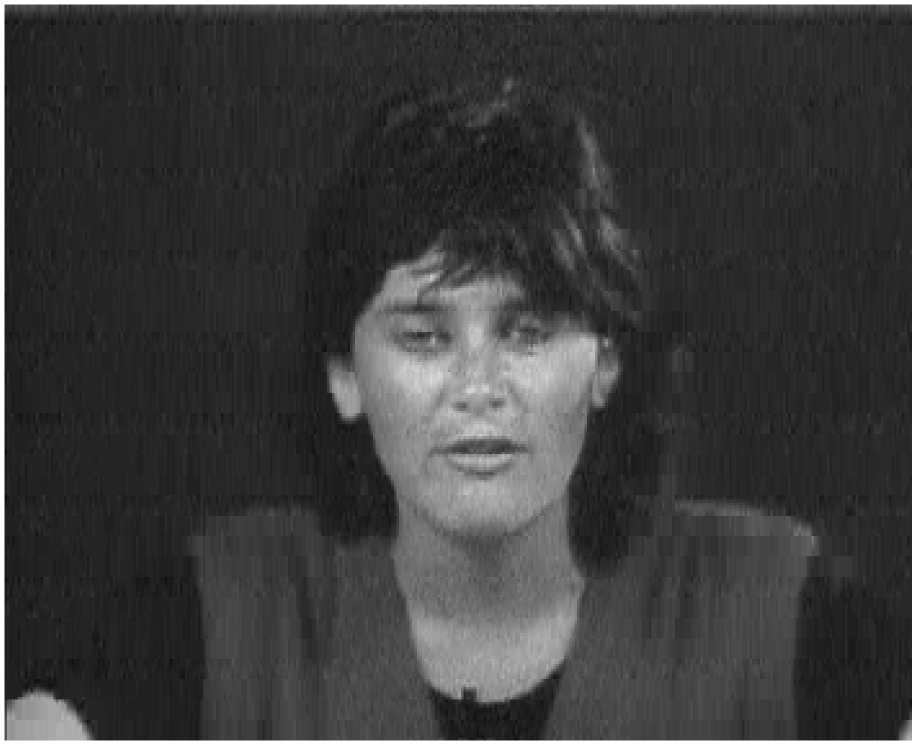


Figure 7.25 Decoded frame number 11 of the sequence by H.263.

CHAPTER 8

RESOLUTION ENHANCEMENT

8.1 Resolution Enhancement

The code generated by fractal coding of a digital image provides a resolution independent representation of the image as this code can be decoded to generate a digital image at any resolution. When the image is decoded at a size larger than the original encoded image, image details beyond the resolution of the original image are predicted by assuming local self-similarity in image at different scales. In this chapter, we (1) present a formulation of how decoding may be done at a higher resolution, (2) evaluate the accuracy of the predicted details using a frequency analysis of fractally enlarged test images, and (3) propose a method for fractal resolution enhancement without the low-frequency loss of information due to fractal coding.

8.2 Classical Image Interpolation

Image Interpolation is an important operation in image processing and is used for image enlargement. One classical method for image interpolation is based on sampling theory and is done by using the frequency components of the smaller image for the low frequency components of the enlarged image, and by assuming that the high-frequency components of the enlarged image are all zero [179]. This principle is shown in Figure 8.1. In this approach, it is basically assumed that the enlarged image is bandlimited. The high-frequency components of the enlarged image are assumed to be zero because these information of the original image have been lost in the sampling process. The

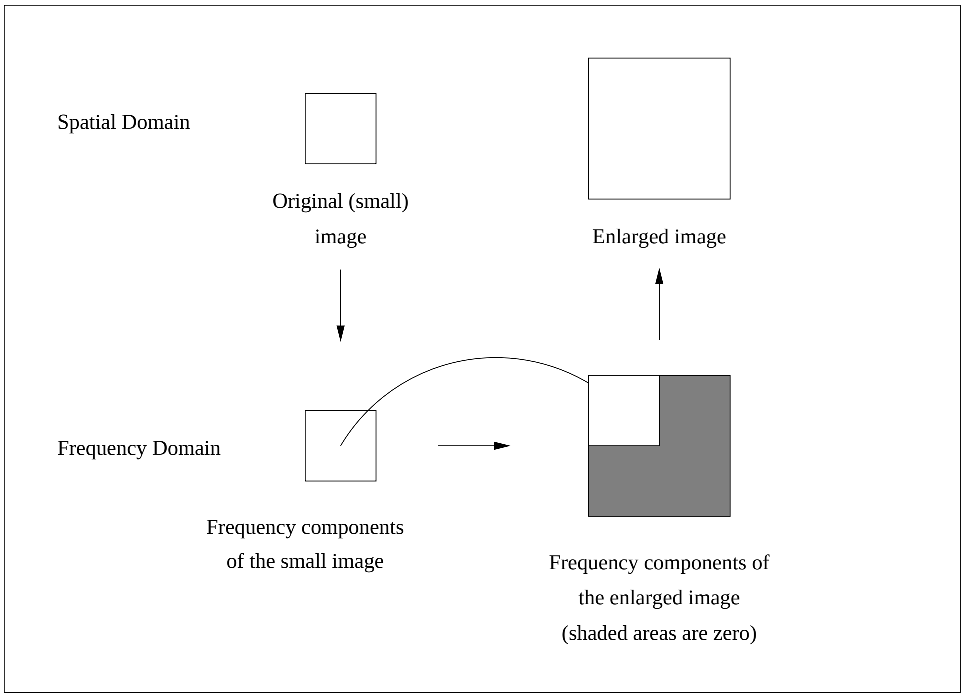

Spatial Domain
Original (small) image
Enlarged image
Frequency Domain
Frequency components of the small image
Frequency components of the enlarged image (shaded areas are zero)

Figure 8.1 Classical image interpolation.

bandlimitedness is usually assumed in the Fourier or DCT domain, however, it may also be done in the wavelet domain. For example, interpolation assuming bandlimitedness in the Haar wavelet domain is equivalent to image enlargement by pixel repetition.

The bandlimitedness assumption, however, is not generally true, and if the enlarged image had been obtained by a direct sampling of a continuous image it would have had nonzero high-frequency components.

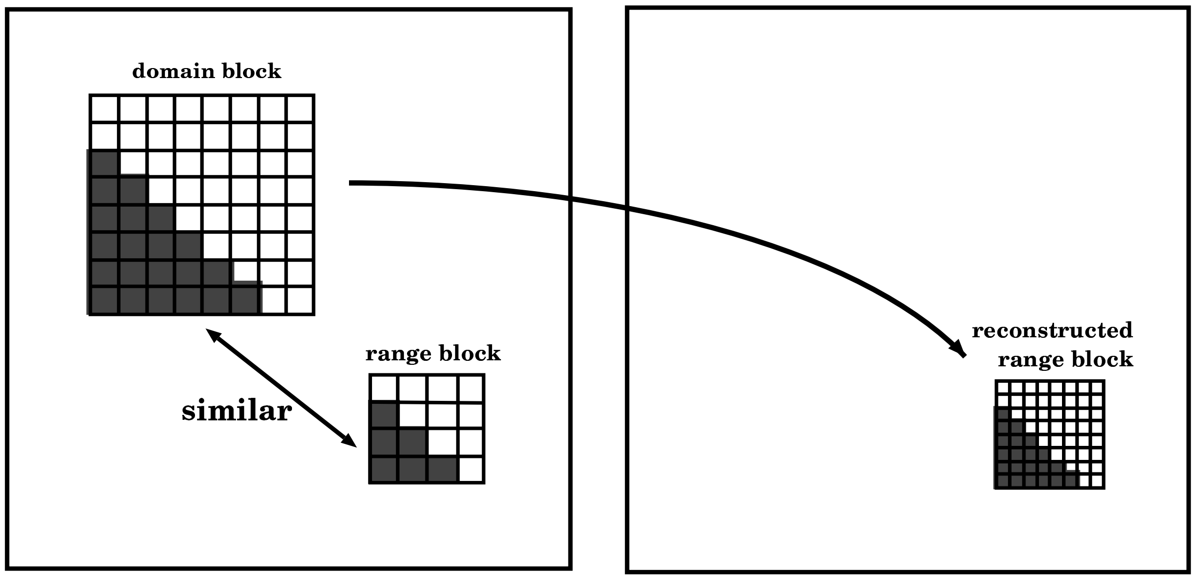
8.3 Prediction of Higher Resolution Information from Lower Resolution

Multiresolution analysis has many applications in image coding and analysis. Some wavelet-based multiresolution image compression methods, like zerotree [121], do some type of prediction of higher resolution information from lower resolution information. Fractal coders also do this type of prediction, though in a different way [47, 49].

The fractal code generated by fractal encoding of a digital image describes relationships (in the form of affine functions) between various segments of the image and is independent of the resolution of the original image. In other words, the fractal code is a *resolution independent* representation of the image and theoretically represents an approximation of the original image in the continuous-space domain. A decoder may decode this code to generate a digital image at any resolution. The resolution of the decoded image may be higher than the resolution of the original image. This increase of resolution is sometimes referred to as *fractal zoom*. This concept must be distinguished from the *fractal interpolation* that is studied in the mathematical literature.

The higher resolution obtained is not created by a simplistic technique such as repeating the pixels of the image, but by actually generating more detail. In fact, the additional higher resolution information is generated using information from the lower resolution image. When an image is reconstructed at the same resolution as the originally encoded image, the domain blocks of the image in the decoding process are shrunk (lowpass filtering followed by subsampling), which eliminates some of the details of the domain blocks. However, if the image is reconstructed at a higher resolution, in the shrinking of the domain block, the details of the domain block are only shrunk to generate the extra resolution in the range block. In fact, details of the domain blocks are used for missing details of the range block. The details in the domain block are also generated to some extent from details of other domain blocks, used for encoding each part of it. In other words, it is implicitly assumed that if the range block is similar to its corresponding domain block, then the details of the range block (which are *beyond* the resolution of the originally encoded image) are also similar to the details of the domain block (which are *within* the resolution of the encoded image). This principle is demonstrated in Figure 8.2. This assumption is a typical property of self-similarity of fractal sets at different scales, and the resolution independence is a property of the code generated by fractal-based methods.

For fractal-based resolution enhancement, an $N_1 \times N_2$ image is first coded using the fractal method to generate a code. Then, the decoder generates an $sN_1 \times sN_2$ image using



(a) Low resolution image (b) High resolution image

Figure 8.2 A demonstration of a simplified version of the resolution enhancement method: (a) approximate each range block in the original image with a decimated domain block in the same image, and (b) use the domain block in place of the range block for a higher resolution.

$$N_1 \times N_2 \text{ image} \longrightarrow \boxed{\text{Fractal Coder}} \xrightarrow{\text{Fractal Code}} \boxed{\text{Fractal Decoder/Magnifier}} \longrightarrow sN_1 \times sN_2 \text{ image}$$

Figure 8.3 Block-diagram of a fractal-based resolution enhancer.

this code. This process is shown in Figure 8.3. In this system, the encoder part is that of a fractal image compression system. However, the decoder needs special provisions. The decoder has a priori knowledge of $\mathbf{G}, \mathbf{H}_i$, $i = 1, 2, \ldots, K$, and the code is made up of information that specifies $\{\alpha_i, b_i, I_i, \mathbf{P}_i,\ i = 1,\ 2, \ldots, K\}$. This information specifies the transformation $\mathcal{T} : R^N \mapsto R^N$. The decoder uses the information to construct another transformation $\bar{\mathcal{T}} : R^{sN} \mapsto R^{sN}$, where $\bar{\mathcal{T}}$ can be written similar to (3.15) and (3.16) as

$$\begin{aligned} \bar{\mathcal{T}} &= \bar{\mathbf{A}}_{sN\times sN}\mathbf{x} + \bar{\mathbf{B}}_{sN\times 1} \\ &= \left(\sum_{i=1}^{K} \bar{\alpha}_i \bar{\mathbf{H}}_i \bar{\mathbf{P}}_i \bar{\mathbf{G}} \bar{\mathbf{K}}_i\right)\mathbf{x} + \left(\sum_{i=1}^{K} \bar{\mathbf{H}}_i \bar{\mathbf{B}}_i\right), \end{aligned} \tag{8.1}$$

where

$$
\begin{aligned}
\bar{\alpha}_i &= \alpha_i \\
\bar{\mathbf{H}}_i &= \begin{bmatrix} \mathbf{0}_{s(i-1)M\times sM} \\ \mathbf{I}_{sM\times sM} \\ \mathbf{0}_{s(K-i)M\times sM} \end{bmatrix}_{sN\times sM}, \\
\bar{\mathbf{G}} &= \begin{bmatrix} \bar{\mathbf{w}} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{w}} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \bar{\mathbf{w}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \bar{\mathbf{w}} \end{bmatrix}_{sM\times sLM}, \\
\bar{\mathbf{w}} &= \begin{bmatrix} \frac{1}{sL} & \frac{1}{sL} & \cdots & \frac{1}{sL} \end{bmatrix}_{1\times sL}, \\
\bar{I}_i &= s(I_i - 1) + 1, \\
\bar{\mathbf{K}}_i &= \begin{bmatrix} \mathbf{0}_{sLM\times(\bar{I}_i-1)} & \mathbf{I}_{sLM\times sLM} & \mathbf{0}_{sLM\times(sN-(\bar{I}_i-1)-sLM)} \end{bmatrix}_{sLM\times sN}, \\
\bar{\mathbf{B}}_i &= \begin{bmatrix} b_i \\ b_i \\ \vdots \\ b_i \end{bmatrix}_{sM\times 1}.
\end{aligned}
$$

Regarding $\bar{\mathbf{P}}_i$, it must represent the same operation that $\mathbf{P}_i$ does, but for a $sM \times sM$ block. In the 1-D case, for example,

$$\mathbf{P}_i = \mathbf{I}_{M\times M}$$

gives

$$\bar{\mathbf{P}}_i = \mathbf{I}_{sM\times sM},$$

and

$$\mathbf{P}_i = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}_{M \times M}$$

gives

$$\bar{\mathbf{P}}_i = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}_{sM \times sM} .$$

It can be shown that decimating $\bar{\mathbf{x}}_\infty$ by applying an averaging filter of length s followed by subsampling converts $\bar{\mathbf{x}}_\infty$ to exactly $\mathbf{x}_\infty$. The operator for this decimation has the form

$$\begin{aligned} \mathbf{V} &= \begin{bmatrix} \mathbf{v} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{v} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{v} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{v} \end{bmatrix}_{N \times sN} , \\ \mathbf{v} &= \begin{bmatrix} \frac{1}{s} & \frac{1}{s} & \cdots & \frac{1}{s} \end{bmatrix}_{1 \times s} . \end{aligned}$$

It can be shown that

$$\mathbf{V}\bar{\mathbf{x}}_\infty = \mathbf{x}_\infty .$$

8.4 Proposed Method

Although there have been many publications on fractal image coding, there have not been any published studies on the resolution enhancement feature of the fractal coders. Given a digital image, in order to be able to evaluate the quality of an interpolated version of this image, we need to know what the larger image would be if it was obtained directly by sampling of a continuous-space image at the higher rate of the larger image. Therefore, we first take an image O_n of size $n \times n$, assuming it is the real larger image. We then decimate (shrink) it using the classical method of lowpass filtering (to avoid aliasing) followed by subsampling to obtain the smaller $m \times m$ image. Note that the decimation process removes the high-frequency components of the larger image. This decimation may be done using ideal lowpass filters, for example, in the DCT domain or in the Haar domain. In the DCT case, we denote the resulting image by $C_{m \leftarrow n} O_n$, and in the Haar case by $H_{m \leftarrow n} O_n$. We then take $C_{m \leftarrow n} O_n$ or $H_{m \leftarrow n} O_n$ as the initial image, encode it using a fractal coder to obtain a fractal code, and decode the code using the fractal decoder/magnifier back to size $n \times n$. We denote the enlarged images by $F_{n \leftarrow m} C_{m \leftarrow n} O_n$ and $F_{n \leftarrow m} H_{m \leftarrow n} O_n$. We can also enlarge $C_{m \leftarrow n} O_n$ and $H_{m \leftarrow n} O_n$ by classical interpolation where we denote the results by $C_{n \leftarrow m} C_{m \leftarrow n} O_n$ and $H_{n \leftarrow m} H_{m \leftarrow n} O_n$. Note that $C_{n \leftarrow m} C_{m \leftarrow n} O_n$ is an ideally lowpassed version of O_n in the DCT domain and $H_{n \leftarrow m} H_{m \leftarrow n} O_n$ is an ideally lowpassed version of O_n in the Haar domain. We may then compare O_n with $F_{n \leftarrow m} C_{m \leftarrow n} O_n$ and $C_{n \leftarrow m} C_{m \leftarrow n} O_n$ at different DCT frequency bands, or compare O_n with $F_{n \leftarrow m} H_{m \leftarrow n} O_n$ and $H_{n \leftarrow m} H_{m \leftarrow n} O_n$ at different Haar wavelet bands. Note that $C_{n \leftarrow m} C_{m \leftarrow n} O_n$ and O_n are exactly the same at low DCT frequencies, and at high frequency bands $C_{n \leftarrow m} C_{m \leftarrow n} O_n$ is all zero, and similarly for the Haar case. However, this is not true between $F_{n \leftarrow m} C_{m \leftarrow n} O_n$ and O_n or between $F_{n \leftarrow m} H_{m \leftarrow n} O_n$ and O_n.

With a coding application in mind, the final result of the final interpolation is $F_{n \leftarrow m} C_{m \leftarrow n} O_n$ or $F_{n \leftarrow m} H_{m \leftarrow n} O_n$. The fractal coding is typically lossy, which means that in contrast to $C_{n \leftarrow m} C_{m \leftarrow n} O_n$ and $H_{n \leftarrow m} H_{m \leftarrow n} O_n$, which have exactly the same low frequency components as in O_n, the low-frequency components of the $F_{n \leftarrow m} C_{m \leftarrow n} O_n$ and $F_{n \leftarrow m} H_{m \leftarrow n} O_n$ are different from that of O_n.[1] This means that decimation of $F_{n \leftarrow m} C_{m \leftarrow n} O_n$ or $F_{n \leftarrow m} H_{m \leftarrow n} O_n$ does not result in $C_{m \leftarrow n} O_n$ and $H_{m \leftarrow n} O_n$.

[1]However, for the typical fractal coders, the Haar transform has the property that low-frequency components of $F_{n \leftarrow m} H_{m \leftarrow n} O_n$ are the same as the whole frequency components of $F_{m \leftarrow m} H_{m \leftarrow n} O_n$, but this is not true in DCT domain.

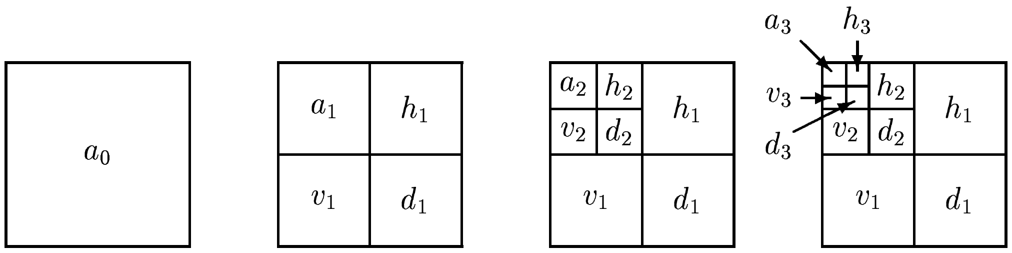


Figure 8.4 Frequency bands of Tables 8.1, 8.2, 8.3, and 8.4.

However, when an interpolation-only application is in mind, the fractal coder may still be used without loss in low-frequency components by replacing the low-frequency components of $F_{n\leftarrow m}C_{m\leftarrow n}O_n$ or $F_{n\leftarrow m}H_{m\leftarrow n}O_n$ with frequency components of $C_{m\leftarrow n}O_n$ and $H_{m\leftarrow n}O_n$, which is the frequency decomposition of the smaller image that is known by the interpolator. However, this method cannot be used in coding applications as the $C_{m\leftarrow n}O_n$ and $H_{m\leftarrow n}O_n$ are not available to the decoder.

8.5 Experimental Results

The 512×512 test image Lena was used as O_n with $n = 512$. Then, $C_{m\leftarrow n}O_n$, $H_{m\leftarrow n}O_n$, $C_{n\leftarrow m}C_{m\leftarrow n}O_n$, and $H_{n\leftarrow m}H_{m\leftarrow n}O_n$ were computed for $m = 256$ and $m = 128$. Three different fractal coding algorithms [180]–[182] were used and for each of them $F_{n\leftarrow m}C_{m\leftarrow n}O_n$ and $F_{n\leftarrow m}H_{m\leftarrow n}O_n$ were computed. The resulting images were then transformed into frequency domain (DCT domain for C cases and Haar wavelet domain for H cases). The rms error at different frequency bands between the results and O_n were computed. These results are shown in Tables 8.1, 8.2, 8.3, and 8.4.[2] The frequency bands a_i, v_i, h_i, and d_i are shown in Figure 8.4.

In these tables, the rms under "original image" are rms of the original image (not the difference), while the data under the name of the methods is for the difference with the original image. Fisher refers to [180] (pp stands for with postprocessing), N&G refers to [181], and ISI refers to [182]. The original Lena image is shown in Figure 8.5 and images

[2]The experimental results for Fisher's program are obtained by Robert DeNardo. The results for N&G and the ISI are obtained by Yoichi Tenda.

Table 8.1 Rms of the Original 512×512 image and the rms error of its approximations obtained by interpolation from the 256×256 image $H_{256 \leftarrow 512} O_{512}$ at different frequency bands (Haar).

	original	classical	Fisher	Fisher pp	N&G	ISI
a_0	133.0	6.7	6.9	6.7	8.2	6.3
a_1	265.6	0.0	7.5	7.5	11.1	7.5
v_1	10.6	10.6	8.7	8.2	9.2	7.7
h_1	7.3	7.3	6.3	5.9	6.3	5.3
d_1	4.0	4.0	4.5	4.4	4.4	3.8
a_2	529.9	0.0	6.7	7.3	11.9	10.6
v_2	29.6	0.0	8.4	8.2	12.4	7.7
h_2	19.1	0.0	7.6	7.2	10.5	5.0
d_2	12.1	0.0	7.3	7.0	9.5	5.1
a_3	1055.2	0.0	3.4	6.0	14.3	16.6
v_3	77.7	0.0	7.3	7.8	10.3	8.7
h_3	48.2	0.0	7.3	7.3	11.3	6.0
d_3	32.9	0.0	7.8	8.0	11.5	7.8

corresponding to classical and Fisher's method are shown in Figures 8.6-8.23 at the end of this chapter.

The results show that the prediction of the high-frequency bands by the $F_{512 \leftarrow m} H_{m \leftarrow 512} O_{512}$ is slightly better than $H_{512 \leftarrow m} H_{m \leftarrow 512} O_{512}$ for the case of $m = 256$, but worse for $m = 128$. In the case of DCT domain, $F_{512 \leftarrow m} C_{m \leftarrow 512} O_{512}$ is slightly worse than $C_{512 \leftarrow m} C_{m \leftarrow 512} O_{512}$ for both $m = 256$, but clearly worse for $m = 128$.

In the Haar domain, the results suggest that prediction of high frequency bands from one band lower in frequency is much better than prediction from several bands lower. This implies that blocks in frequency bands are more similar in closer bands and suggests that a better fractal resolution enhancer may be obtained if blocks in high-frequency bands are approximated by blocks in only one band above them. This is in contrast to matching blocks of different sizes in spatial domain, which corresponds to matching trees of blocks in the Haar domain.

Table 8.2 Rms of the Original 512×512 image and the rms error of its approximations obtained by interpolation from the 128×128 image $H_{128 \leftarrow 512} O_{512}$ at different frequency bands (Haar).

	original	classical	Fisher	Fisher pp	N&G	ISI
a_0	133.0	11.5	12.1	11.8	13.7	9.6
a_1	265.6	18.6	19.8	19.2	23.5	15.2
v_1	10.6	10.6	10.6	10.3	11.0	8.9
h_1	7.3	7.3	7.7	7.5	7.5	6.3
d_1	4.0	4.0	4.8	4.8	4.6	4.2
a_2	529.9	0.0	21.5	21.2	31.9	8.1
v_2	29.6	29.6	25.3	24.3	26.9	22.4
h_2	19.1	19.1	17.1	16.4	17.4	14.7
d_2	12.1	12.1	13.0	12.8	13.1	11.6
a_3	1055.2	0.0	19.5	19.6	33.9	12.7
v_3	77.7	0.0	24.4	23.9	35.4	5.9
h_3	48.2	0.0	21.0	20.4	30.1	5.8
d_3	32.9	0.0	21.0	20.7	27.8	5.5

Table 8.3 Rms of the Original 512×512 image and the rms error of its approximations obtained by interpolation from the 256×256 image $C_{256 \leftarrow 512} O_{512}$ at different frequency bands (DCT).

	original	classical	Fisher	N&G	ISI
a_0	133.0	4.0	7.4	8.4	5.9
a_1	265.8	0.0	8.9	12.4	8.2
v_1	3.9	3.9	6.3	6.1	4.3
h_1	6.4	6.4	9.0	8.8	6.8
d_1	3.0	3.0	4.4	4.2	3.4
a_2	530.8	0.0	8.0	14.2	10.6
v_2	13.2	0.0	8.6	10.9	5.5
h_2	22.5	0.0	10.2	13.6	9.2
d_2	11.4	0.0	8.5	10.2	6.6
a_3	1058.6	0.0	5.4	16.9	17.0
v_3	34.8	0.0	8.1	12.4	5.1
h_3	63.7	0.0	8.8	13.0	8.3
d_3	35.8	0.0	9.2	14.1	8.1

Table 8.4 Rms of the Original 512×512 image and the rms error of its approximations obtained by interpolation from the 128×128 image $C_{128 \leftarrow 512} O_{512}$ at different frequency bands (DCT).

	original	classical	Fisher	N&G	ISI
a_0	133.0	8.2	13.0	14.3	10.1
a_1	265.8	14.2	22.3	25.7	17.6
v_1	3.9	3.9	7.2	6.6	4.9
h_1	6.4	6.4	10.2	10.0	7.2
d_1	3.0	3.0	4.7	4.2	4.6
a_2	530.8	0.0	26.3	36.6	10.4
v_2	13.2	13.2	17.6	17.3	16.3
h_2	22.5	22.5	27.8	28.4	26.7
d_2	11.4	11.4	14.5	14.0	12.3
a_3	1058.6	0.0	23.5	40.3	13.3
v_3	34.8	0.0	23.7	31.2	6.7
h_3	63.7	0.0	31.7	41.2	9.5
d_3	35.8	0.0	25.5	32.8	10.9

8.6 Conclusions

The resolution enhancement feature of fractal coders is analyzed and evaluated against the classical interpolation method using three different fractal coding methods. In the Haar wavelet domain, the fractal enhancement shows slightly better results when change of scale is by a factor of 2. However, in other cases, the classical results perform better. Our study suggests a new type of fractal coder operating directly in the frequency domain.

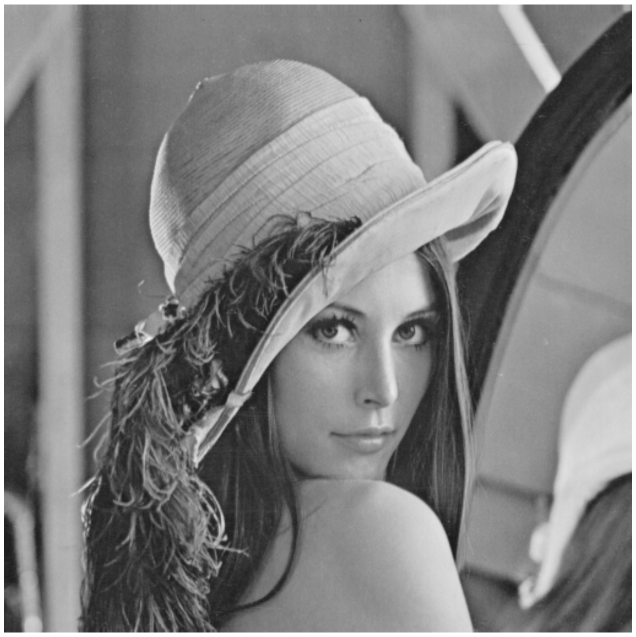


Figure 8.5 O_{512}: Original 512×512 Lena image.

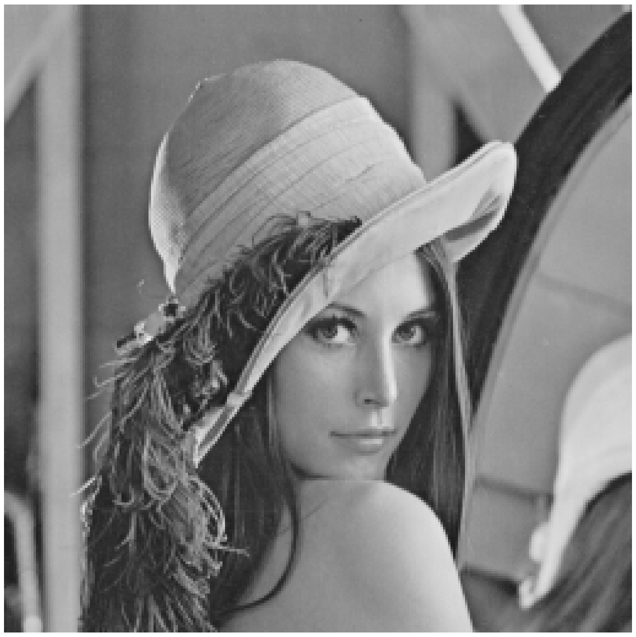


Figure 8.6 $H_{256 \leftarrow 512} O_{512}$.

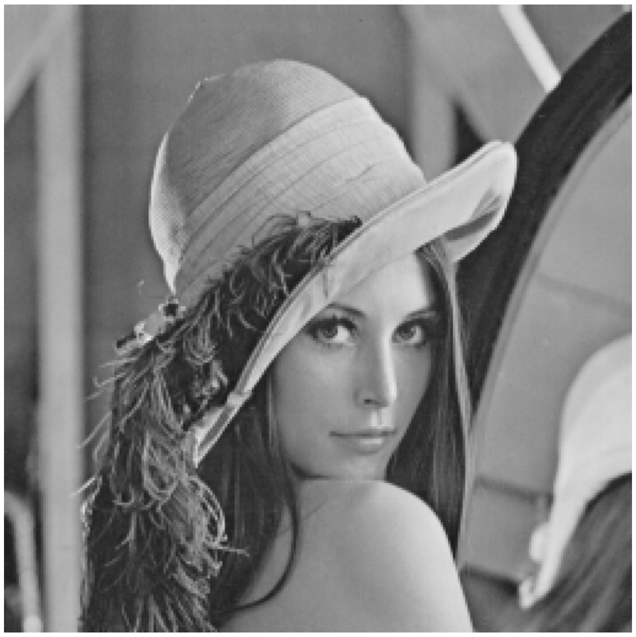


Figure 8.7 $H_{512\leftarrow 256}H_{256\leftarrow 512}O_{512}$.

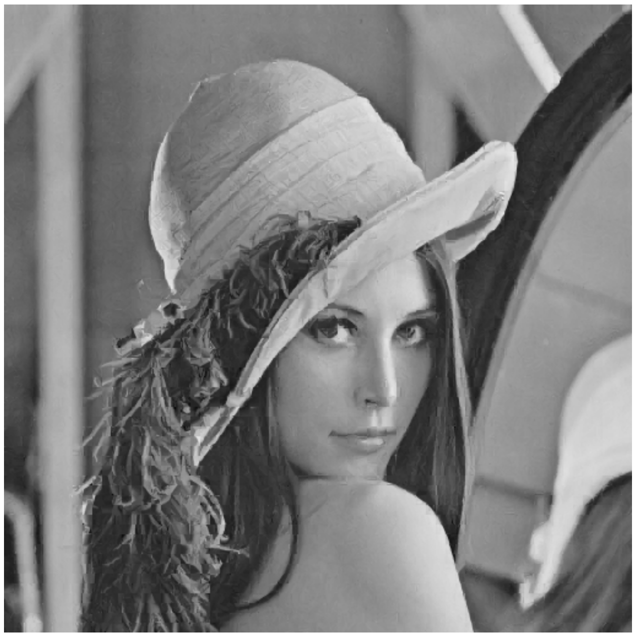


Figure 8.8 $F_{512\leftarrow 256}H_{256\leftarrow 512}O_{512}$ (Fisher).

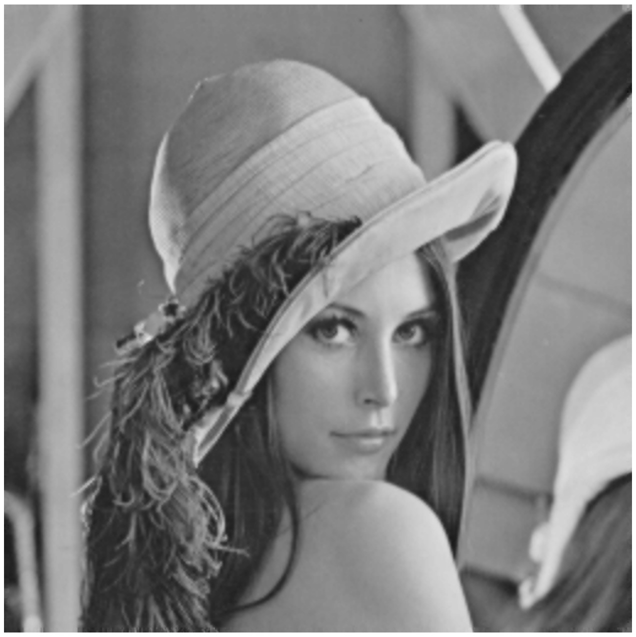


Figure 8.9 $F_{512\leftarrow 256}H_{256\leftarrow 512}O_{512}$ (ISI).

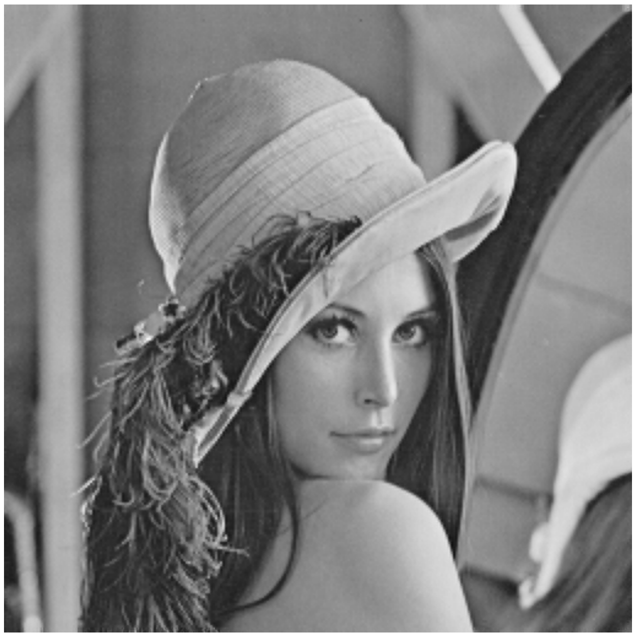


Figure 8.10 $F_{512\leftarrow 256}H_{256\leftarrow 512}O_{512}$ (ISI with subband replacement).

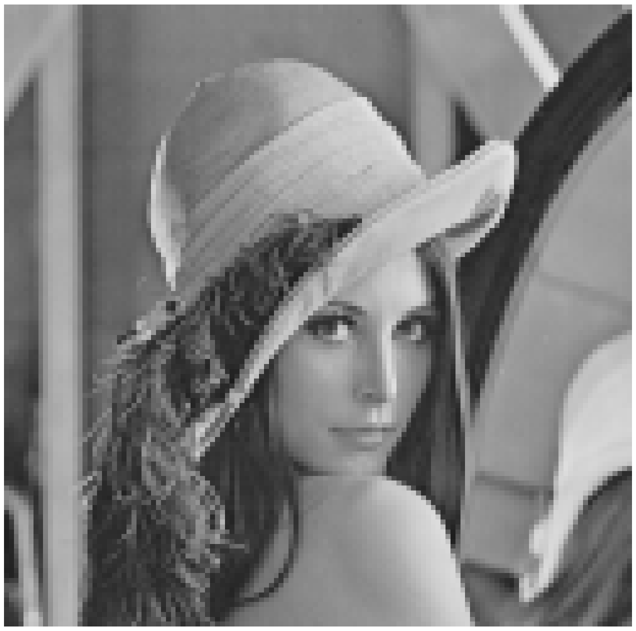


Figure 8.11 $H_{128\leftarrow512}O_{512}$.

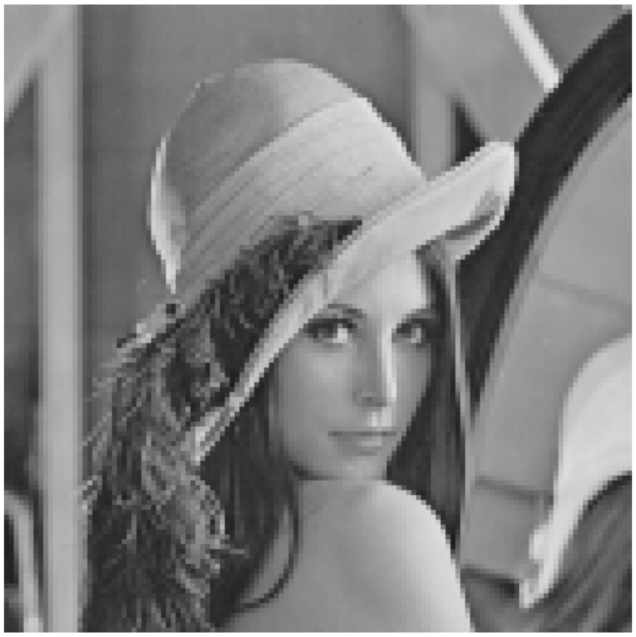


Figure 8.12 $H_{512\leftarrow128}H_{128\leftarrow512}O_{512}$.

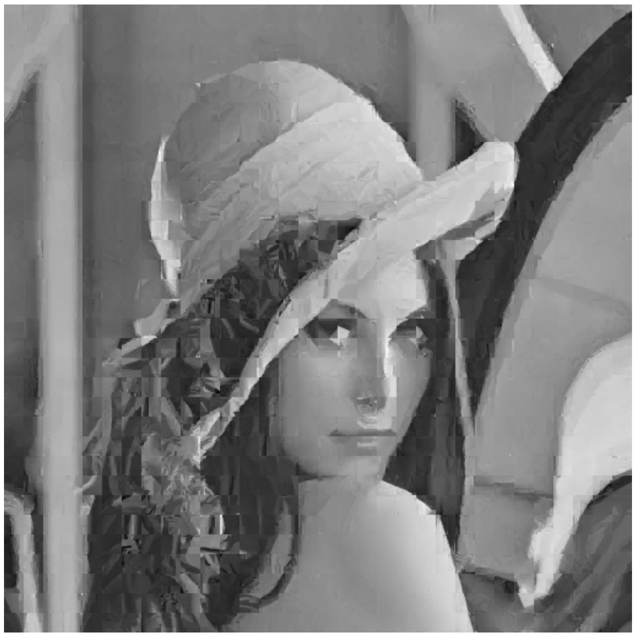


Figure 8.13 $F_{512\leftarrow 128}H_{128\leftarrow 512}O_{512}$ (Fisher).

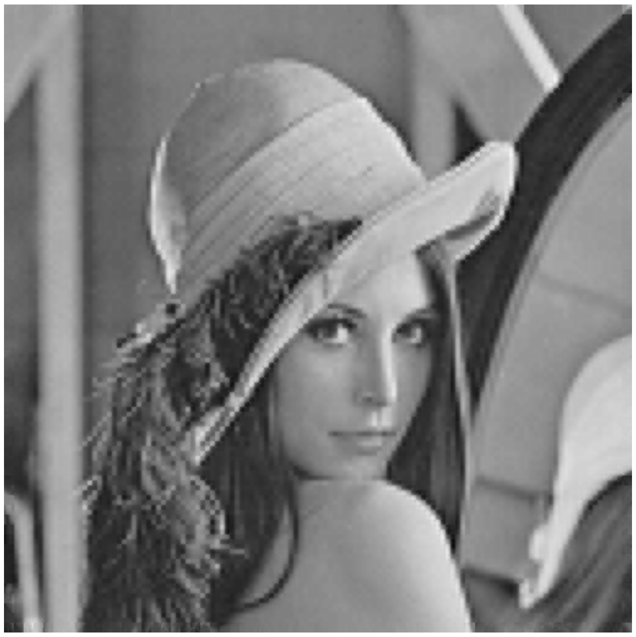


Figure 8.14 $F_{512\leftarrow 128}H_{128\leftarrow 512}O_{512}$ (ISI).

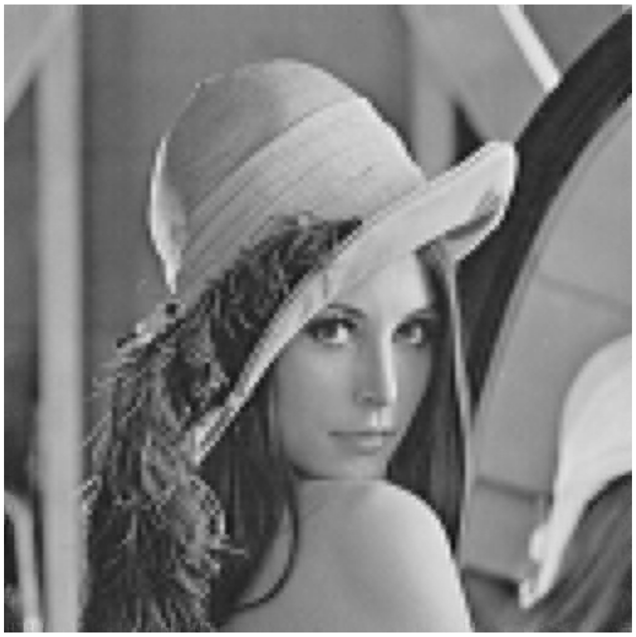


Figure 8.15 $F_{512\leftarrow128}H_{128\leftarrow512}O_{512}$ (ISI with subband replacement).

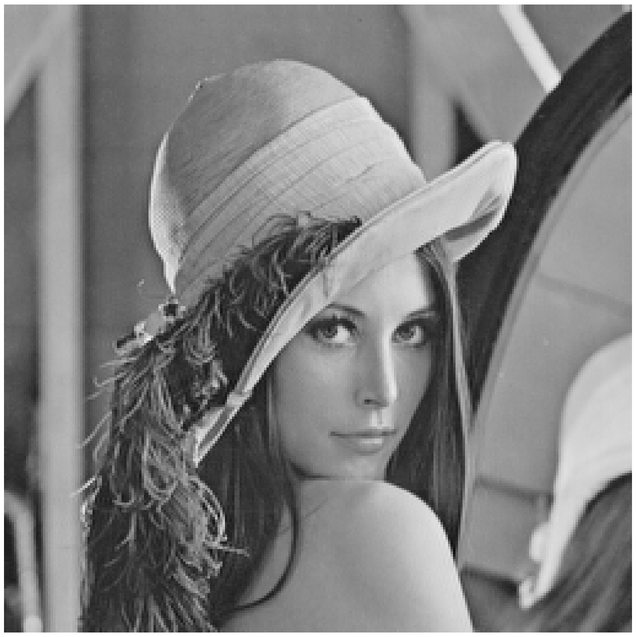


Figure 8.16 $C_{256\leftarrow512}O_{512}$.

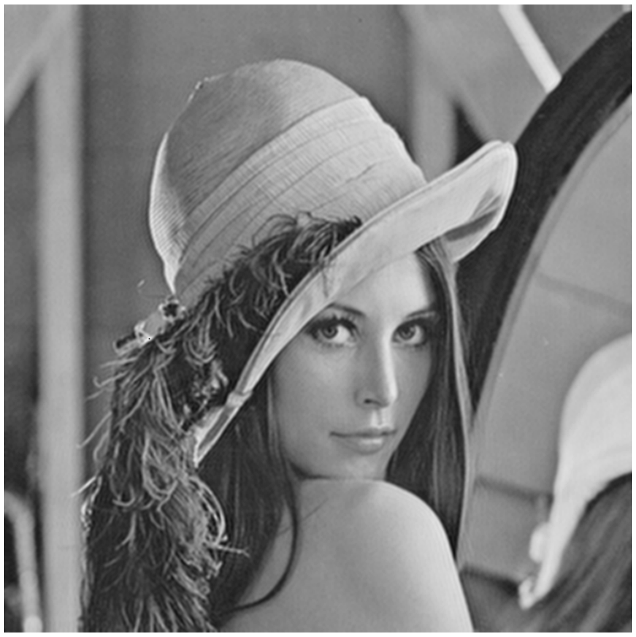


Figure 8.17 $C_{512\leftarrow 256}C_{256\leftarrow 512}O_{512}$.

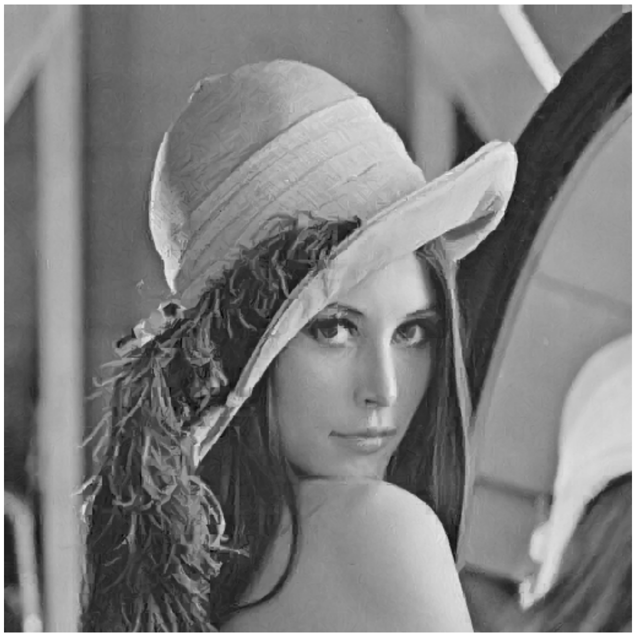


Figure 8.18 $F_{512\leftarrow 256}C_{256\leftarrow 512}O_{512}$ (Fisher).

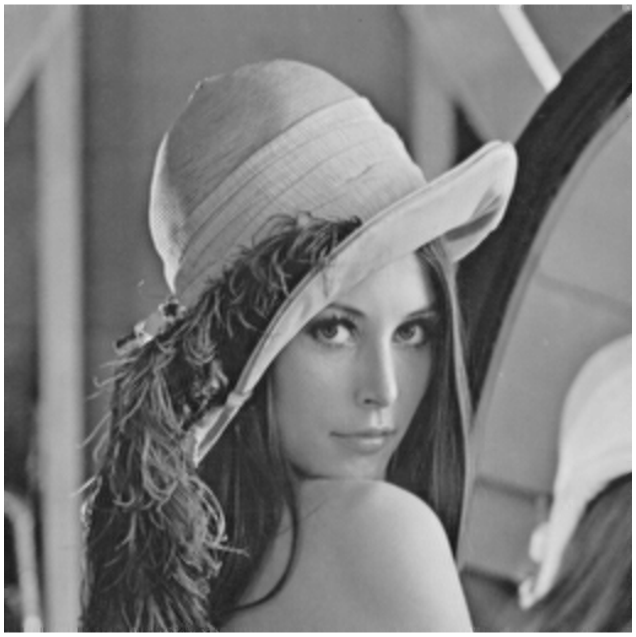


Figure 8.19 $F_{512\leftarrow 256}C_{256\leftarrow 512}O_{512}$ (ISI).

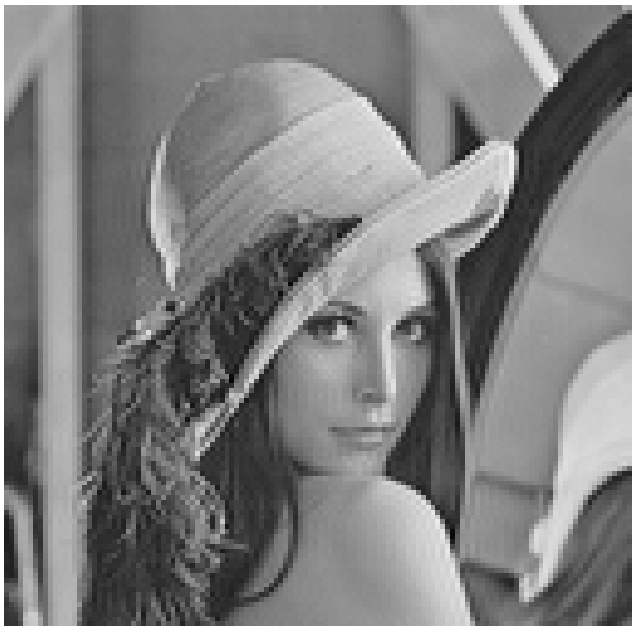


Figure 8.20 $C_{128\leftarrow 512}O_{512}$.

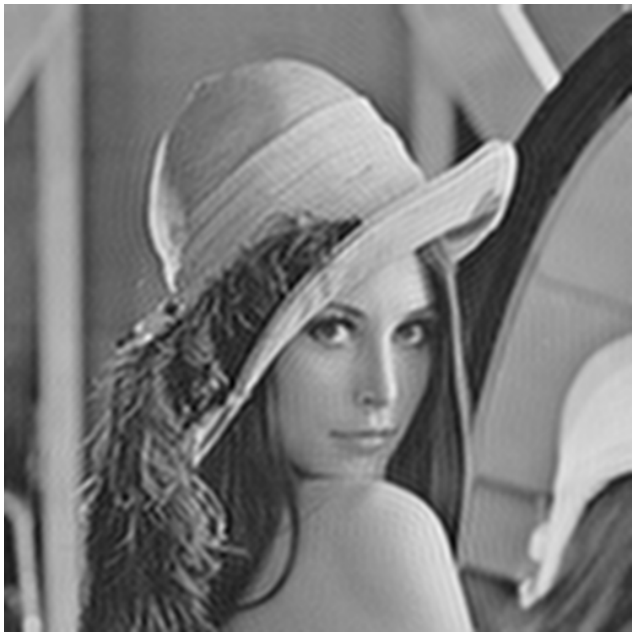


Figure 8.21 $C_{512\leftarrow128}C_{128\leftarrow512}O_{512}$.

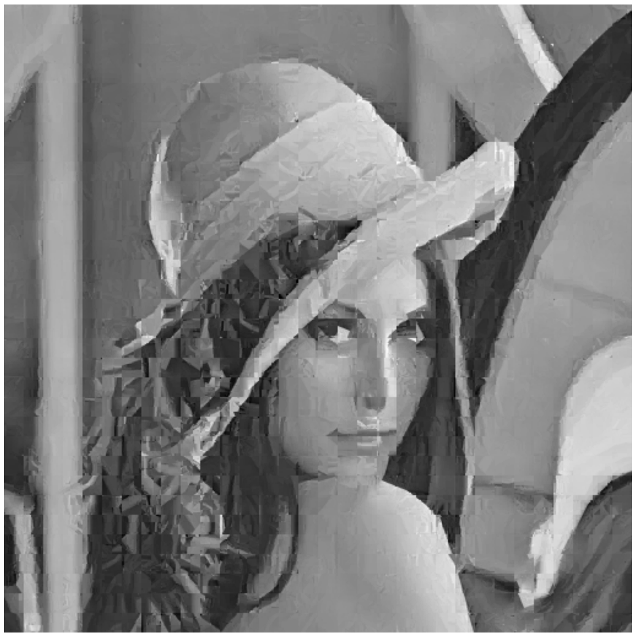


Figure 8.22 $F_{512\leftarrow128}C_{128\leftarrow512}O_{512}$ (Fisher).

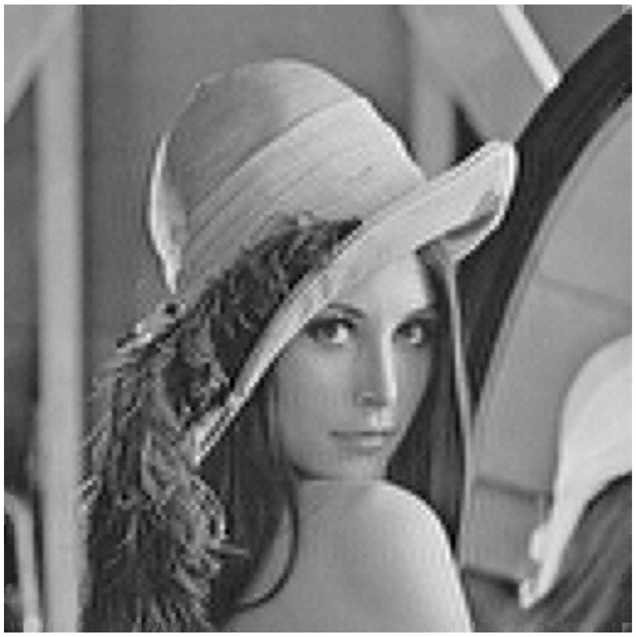


Figure 8.23 $F_{512\leftarrow128}C_{128\leftarrow512}O_{512}$ (ISI).

CHAPTER 9

CONCLUSIONS

In this dissertation, the theory and implementation of fractal and attractor image and video coding methods have been investigated. Fractal and attractor coders propose new paradigms in image coding. Fractal coding proposes exploiting repetition of patterns at different scales, and this work has extended this to exploiting the repetition of patterns at the same and at different scales. Natural images are not random, and it is intuitively clear that most parts of these images are repetition of similar patterns, which may vary greatly from one image to another, and parts of an image may be encoded by giving reference to another part of the image with some deformation parameters.

On the other hand, attractor coders propose the paradigm of encoding an image by specifying the state transition operator of a dynamical system whose steady-state is close to a desired image. This is based on the notion that visually complex and sometimes natural looking images can be generated by rather simple dynamical systems. These new paradigms present new and difficult questions to the researchers in this field, and even some of the very fundamental questions in them are not yet answered.

The class of coders proposed by generalized fractal coders and attractor coders is quite large, and current coders based on these methods are only concentrated around the base provided by Jacquin's method, whose practicality is established, and its mathematics are somewhat better understood. However, research by investigations in this field, and specifically this dissertation, work on expanding the radius of work around this basic approach and expanding its horizons.

Although the field of fractal-based image coding is relatively young and its methods are different from other image coding methods, the performance of these methods has been comparable to those of the state-of-the-art image compression methods in terms of combination of image quality and compression ratio. However, the complexity of the encoder for fractal-based image coders is typically high.

The nature of fractal coding is that of basic research. Currently, fractal coders cannot typically perform better than the state-of-the-art image coders in terms of compression ratio and PSNR, and in terms of complexity of encoder, they typically require much more compression time. However, the performance of these coders has been greatly improved since the work of Jacquin [4], and in terms of compression ratio, they perform 4 to 5 times better than Jacquin's original method.

In this dissertation, we began with the basic fractal coding method of Jacquin. His work introduced new concepts into the field of coding. However, despite its novelty, this method, in its basic form, had several shortcomings. Part of the work of this dissertation was to pinpoint its strength and weaknesses. The coder works well in coding edges, however, it falls short in coding textures. It relies heavily on the fact that natural images have strong dc components and performs poorly for nonphotographic 2-D signals without strong dc components. It relies on exploiting only one type of redundancy present in an image, namely, similarity of blocks at different scales, and does not take advantage of other types of redundancies present in images, and has no integration with the state-of-the-art compression techniques. Unique blocks in the image cannot be coded well, and for those image blocks for whom several similar blocks are present in the image, at most, only one can be used. The coder also cannot encode images with arbitrarily high quality. It also does not use powerful entropy coders for the coding of its parameters and does not have any statistical model for these parameters. There is also an exponential relation between PSNR of the decoded image and compression time.

To overcome these problems, a novel coding approach was designed. We developed theory and a new and general implementation that addresses and removes these shortcomings. Fractal image compression takes advantage of similarities present in images at different scales. These types of similarities do not seem to be enough to be the sole basis of image compression algorithms superior to other state-of-the-art image compression methods and are one of several types of redundancies present in images, e.g, intra-scale. However, we showed that they may be used in combination with other compression methods to significantly enhance their performance in terms of image quality and bitrate. In fact, during this process, we devised a very general compression method that takes advantage of inter-scale similarities, intra-scale similarities, and compaction properties of transform coding methods. In the context of still image coding, the resulting method has the block transform coding, standard VQ, and most of the earlier fractal compression methods as special cases. For video coding, block prediction methods like DPCM, adaptive block prediction methods like block motion compensation, and hybrid coding methods like motion compensation combined with transform coding of residual errors become special and restricted cases of this algorithm. In fact, this algorithm describes a wide spectrum of image coding methods in which the above methods are near extreme cases. It automatically falls back into special cases depending on the situation.

Fractal image coders are typically computationally expensive. The added image quality and reduced bitrate obtained from integrating them with other methods comes with this cost. This makes it more suitable for applications in telecommunications and storage, where the compression need not be done live or in real time. In fact, this is the reason why these methods have found their greatest success in the market of software and CD ROM developers.

In this dissertation, we also found general improvements that are applicable to all types of fractal coders. We introduced the concept of substitution of the range block with its approximation in the encoder for reducing the decoding error (Section 5.2.3) and discovered (simultaneously with several other researchers) the concept of decoding with immediate substitution for faster decoder convergence (Section 5.5.1). We also established the relation between the causality of encoder and the iterativeness of the decoder, and showed that a causal encoder with substitution of the range block with its approximation in the encoder, and decoding with immediate substitution, results in a fractal coding system whose decoder is noniterative and whose fractal transform need not be contractive. This reduces the complexity of the decoder by one order of magnitude. By these observations, this work does the following:

- For the first time, establishes the usefulness of using intra-scale similarities in natural images, in addition to intra-scale similarities used by fractal coders. This was previously believed to be useless because it caused very large decoder errors. This was solved by the observation made in the causality of the encoder.

- Clarifies the fact that the iterativeness of the decoder of the fractal coders is not inherent of fractal coders and is due to the noncausality of the fractal coders in use.

The coding method that was designed in this research needed an algorithm to find the solution to a combinatorial optimization problem. To solve this problem, we independently discovered a method that is currently known as matching pursuit, its orthogonal version and its rate-distortion optimized version. We were the first to use these methods for image compression and among the first to use it for video compression. This method has applications in statistics, control theory, and signal analysis.

The new image coding algorithm proposed in this dissertation gives a compression ratio 4 to 5 times greater than the original Jacquin method and is among the top three fractal coding methods ever reported.

In this dissertation, we also proposed one of the most efficient fractal video coders, which seamlessly combines the motion compensation techniques with fractal techniques. This dissertation also seems to be the first work to investigate the resolution enhancement feature of fractal coders and compare it with other interpolation methods.

The analysis of fractal image coding systems is a rather difficult problem. In Chapter 3, we provided a new analysis of fractal coders using control systems theory and graph theory. The system theoretical approach does not seem to have ever been addressed by other researchers, and this work is the most comprehensive graph-theoretical analysis of fractal coders. The analysis of attractor/fractal coders of Chapter 3 provides links between the young field of attractor coding and the well-established fields of systems theory and graph theory. Common attractor decoders are modeled as linear systems whose stability is both necessary and sufficient for convergence of the decoder. This stability is dictated by the location of the eigenvalues of the sparse state transition matrix of the system. The relation between these eigenvalues, spatial causality of the system, and the patterns of interdependency between signal elements (or image pixels) is investigated for several cases using concepts from graph and matrix theory.

REFERENCES

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

[2] M. F. Barnsley and A. D. Sloan, "A better way to compress images," *BYTE*, pp. 215–223, Jan. 1988.

[3] A. E. Jacquin, "A novel fractal block-coding technique for digital images," in *Proceedings of IEEE ICASSP-90*, Albuquerque, NM, Apr. 3–6, 1990, pp. 2225–2228.

[4] A. E. Jacquin, "A fractal theory of iterated Markov operators with applications to digital image coding," PhD dissertation, Georgia Institute of Technology, Atlanta, GA, 1989.

[5] M. Gharavi-Alkhansari and T. S. Huang, "Fractal-based image and video coding," in *Video Coding: The Second Generation Approach*, L. Torres and M. Kunt, Eds., Boston, MA: Kluwer Academic Publishers, 1996, pp. 265–303.

[6] M. Gharavi-Alkhansari and T. S. Huang, "A fractal-based image block-coding algorithm," in *Proceedings of IEEE ICASSP-93*, vol. V, Minneapolis, MN, Apr. 27–30, 1993, pp. 345–348.

[7] M. Gharavi-Alkhansari and T. S. Huang, "A fractal-based image block-coding algorithm," in *Proceedings of Picture Coding Symposium*, Lausanne, Switzerland, Mar. 17–19, 1993, p. 1.7.

[8] M. Gharavi-Alkhansari and T. S. Huang, "Fractal-based techniques for a generalized image coding method," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, Austin, TX, Nov. 13–16, 1994, pp. 122–126.

[9] M. Gharavi-Alkhansari and T. S. Huang, "Generalized image coding using fractal-based methods," in *Proceedings of the International Picture Coding Symposium*, Sacramento, CA, Sept. 21–23, 1994, pp. 440–443.

[10] M. Gharavi-Alkhansari and T. S. Huang, "Fractal image coding using rate-distortion optimized matching pursuit," in *Proceedings of the SPIE, Visual Communications and Image Processing*, vol. 2727, Orlando, FL, Mar. 17–20, 1996, pp. 1386–1393.

[11] M. Gharavi-Alkhansari and T. S. Huang, "Fractal video coding by matching pursuit," in *Proceedings of IEEE International Conference on Image Processing*, Lausanne, Switzerland, Sept. 16–19, 1996.

[12] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3397–3415, Dec. 1993.

[13] G. Davis, S. Mallat, and Z. Zhang, "Adaptive time-frequency decompositions," *Optical Engineering*, vol. 33, pp. 2183–2191, July 1994.

[14] R. Neff and A. Zakhor, "Very low bit rate video coding using matching pursuits," in *Proceedings of the SPIE, Visual Communications and Image Processing I*, vol. 2308, Chicago, IL, Sept. 25–28, 1994, pp. 47–60.

[15] M. Vetterli and T. Kalker, "Matching pursuit for compression and application to motion compensated video coding," in *Proceedings of IEEE International Conference on Image Processing*, vol. 1, Austin, TX, Nov. 13–16, 1994, pp. 725–729.

[16] D. W. Lin, "Fractal image coding as generalized predictive coding," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, Austin, TX, Nov. 13–16, 1994, pp. 117–121.

[17] B. B. Mandelbrot, *Les Objets Fractals: Forme, Hasard et Dimension.* Paris, France: Flammarion, 1975. (In French).

[18] B. B. Mandelbrot and R. F. Voss, *Fractals: Form, Chance and Dimension.* San Francisco, CA: Freeman, 1977.

[19] M. F. Barnsley, *Fractals Everywhere.* San Diego, CA: Academic Press, Inc., 1988.

[20] G. A. Edgar, *Measure, Topology, and Fractal Geometry.* Undergraduate Texts in Mathematics, New York: Springer-Verlag, 1990.

[21] K. J. Falconer, *Fractal Geometry: Mathematical Foundations and Applications.* Chichester: John Wiley and Sons, 1990.

[22] A. P. Pentland, "Fractal-based description of natural scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, pp. 661–674, Nov. 1984.

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

[24] A. P. Pentland, "Fractal surface models for communications about terrain," in *Proceedings of the SPIE, Visual Communications and Image Processing II*, vol. 845, Oct. 1987, pp. 301–306.

[25] A. Ait-Kheddache and S. A. Rajala, "Texture classification based on higher-order fractals," in *Proceedings of IEEE ICASSP-88*, New York City, NY, Apr. 11–14, 1988, pp. 1112–1115.

[26] F. Arduini, C. Dambra, S. Dellepiane, S. B. Serpico, G. Vernazza, and R. Viviani, "Fractal dimension estimation by adaptive mask selection," in *Proceedings of IEEE ICASSP-88*, New York City, NY, Apr. 11–14, 1988, pp. 1116–1119.

[27] F. Arduini, S. Fioravanti, and D. D. Giusto, "A multifractal-based approach to natural scene analysis," in *Proceedings of IEEE ICASSP-91*, Toronto, Canada, May 14–17, 1991, pp. 2681–2684.

[28] J. M. Keller, R. M. Crownover, and R. Y. Chen, "Characteristics of natural scenes related to the fractal dimension," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, pp. 621–627, Sept. 1987.

[29] S. Peleg, J. Naor, R. Hartley, and D. Avnir, "Multiple resolution texture analysis and classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, pp. 518–523, July 1984.

[30] T. Peli, V. Tom, and B. Lee, "Multi-scale fractal and correlation signatures for image screening and natural clutter suppression," in *Proceedings of the SPIE, Visual Communications and Image Processing IV*, vol. 1199, Nov. 8–10, 1989, pp. 402–415.

[31] T. Peli, "Multiscale fractal theory and object characterization," *Journal of Optical Society of America A*, vol. 7, pp. 1101–1112, 1990.

[32] M. Stein, "Fractal image models and object detection," in *Proceedings of the SPIE, Visual Communications and Image Processing II*, vol. 845, Oct. 1987, pp. 293–306.

[33] A. M. Vepsäläinen and J. Ma, "Estimating of fractal and correlation dimension from 2D and 3D images," in *Proceedings of the SPIE, Visual Communications and Image Processing IV*, vol. 1199, Nov. 8–10, 1989, pp. 431–438.

[34] R. J. Stevens, A. F. Lehar, and F. H. Preston, "Manipulation and presentation of multidimensional image data using Peano scan," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, pp. 520–526, Sept. 1983.

[35] K.-M. Yang, L. Wu, and M. Mills, "Fractal based image coding scheme using Peano scan," in *Proceedings of IEEE International Symposium on Circuits and Systems*, Espoo, Finland, June 7–9, 1988, pp. 2301–2304.

[36] I. Gerner and Y. Y. Zeevi, "Generalized scanning and multiresolution image compression," in *DCC'91: Data Compression Conference*, J. A. Storer and J. H. Reif, Eds., Snowbird, UT, IEEE Computer Society Press, Apr. 8–11, 1991, p. 434.

[37] K. S. Thyagarajan and S. Chatterjee, "Fractal scanning for images compression," in *Proceedings of Twenty-Fifth Asilomar Conference on Signals, Systems and Computers*, Nov. 4–6, 1991.

[38] E. Walach and E. Karnin, "A fractal-based approach to image compression," in *Proceedings of IEEE ICASSP-86*, Tokyo, Japan, Apr. 7–11, 1986, pp. 529–532.

[39] N. Zhang and H. Yan, "Hybrid image compression method based on fractal geometry," *Electronics Letters*, vol. 27, pp. 406–408, Feb. 28, 1991.

[40] M. Temerinac, A. Kozarev, Z. Trpovski, and B. Simsic, "An efficient image compression algorithm based on filter bank analysis and fractal geometry," in *Proceedings of Signal Processing VI: Theories and Applications*, J. Vandewalle, R. Boite, M. Moonen, and A. Oosterlinck, Eds., New York, Elsevier Science Publishers, 1992, p. 1373.

[41] C. Chang and S. Chatterjee, "Fractal based approach to shape description, reconstruction and classification," in *Proceedings of Twenty-Third Asilomar Conference on Signals, Systems and Computers*, Oct. 30–Nov. 1, 1989, pp. 172–176.

[42] B. D. Goel and S. C. Kwatra, "A data compression algorithm for color images based on run-length coding and fractal geometry," in *IEEE International Conference on Communications'88*, June 12–15, 1988, pp. 1253–1256.

[43] J. Jang and S. Rajala, "Segmentation based image coding using fractals and the human visual system," in *Proceedings of IEEE ICASSP-90*, Albuquerque, NM, Apr. 3–6, 1990, pp. 1957–1960.

[44] J. Jang and S. A. Rajala, "Texture segmentation-based image coder incorporating properties of the human visual system," in *Proceedings of IEEE ICASSP-91*, Toronto, Canada, May 14–17, 1991, pp. 2753–2756.

[45] C. K. Cheong, K. Aizawa, T. Saito, and M. Hatori, “Structural edge detection based on fractal analysis for image compression,” in *IEEE International Symposium on Circuits and Systems*, San Diego, CA, May 10–13, 1992, pp. 2461–2464.

[46] A. Pentland and B. Horowitz, “A practical approach to fractal-based image compression,” in *DCC'91: Data Compression Conference*, J. A. Storer and J. H. Reif, Eds., Snowbird, UT, IEEE Computer Society Press, Apr. 8–11, 1991, pp. 176–185.

[47] R. Rinaldo and G. Calvagno, “An image coding scheme using block prediction of the pyramid subband decomposition,” in *Proceedings of IEEE International Conference on Image Processing*, vol. 2, Austin, TX, Nov. 13–16, 1994, pp. 878–882.

[48] R. Rinaldo and G. Calvagno, “Image coding by block prediction of multiresolution subimages,” *IEEE Transactions on Image Processing*, vol. 4, pp. 909–920, July 1995.

[49] G. Davis, “Self-quantized wavelet subtrees: A wavelet-based theory for fractal image compression,” in *DCC'95: Data Compression Conference*, Snowbird, UT, Mar. 28–30, 1995.

[50] M. F. Barnsley and S. Demko, “Iterated function systems and the global construction of fractals,” *Proceedings of the Royal Society of London*, vol. A399, pp. 243–275, 1985.

[51] M. F. Barnsley, “Fractal functions and interpolation,” *Constructive Approximation*, vol. 2, pp. 303–329, 1986.

[52] M. F. Barnsley, “Making chaotic dynamical systems to order,” in *Chaotic Dynamics and Fractals*, M. F. Barnsley and S. G. Demko, Eds., New York: Academic Press, Inc., 1986, pp. 53–68. also in *Proceedings of Conference on Chaotic Dynamics*, Atlanta, GA, March 25–29, 1985, pp. 53–68.

[53] M. F. Barnsley, V. Ervin, D. Hardin, and J. Lancester, "Solution of an inverse problem for fractals and other sets," *Proceedings of the National Academy of Sciences USA*, vol. 83, pp. 1975–1977, Apr. 1986.

[54] M. F. Barnsley and A. D. Sloan, "Chaotic compression," *Computer Graphics World*, pp. 107–108, Nov. 1987.

[55] M. F. Barnsley and A. E. Jacquin, "Application of recurrent iterated function systems to images," in *Proceedings of the SPIE, Visual Communications and Image Processing*, vol. 1001, 1988, pp. 122–131.

[56] M. F. Barnsley, A. E. Jacquin, F. Malassenet, L. Reuter, and A. Sloan, "Harnessing chaos for image synthesis," in *Computer Graphics Conference Proceedings*, vol. 22, Aug. 1988, SIGGRAPH, pp. 131–140.

[57] M. F. Barnsley and J. Elton, "A new class of Markov processes for image encoding," *Advances in Applied Probability*, vol. 20, pp. 14–32, 1988.

[58] M. F. Barnsley, "Fractal modeling of real world images," in *The Science of Fractal Images*, H. O. Peitgen and D. Saupe, Eds., New York: Springer-Verlag, 1988, pp. 219–242. Based on Lecture Notes for Fractals: Introductions, Basics and Perspectives, in SIGGRAPH'87 (Anaheim, California).

[59] M. F. Barnsley, S. Demko, J. Elton, and J. Geronimo, "Invariant measures for Markov processes arising from function iteration with place-dependent probabilities," *Annales de l'Institut Henry Poincare: Probabilites et statiques*, vol. 24, no. 3, pp. 367–394, 1988.

[60] M. F. Barnsley and A. D. Sloan, "Fractal image compression," in *Proceedings of the Scientific Data Compression Workshop*, H. K. Ramapriyan, Ed., Snowbird, UT, NASA Godard Space Flight Center, May 3–5, 1988, pp. 351–365. NASA conference publication 3025.

[61] M. F. Barnsley, *Constructive Approximation*, vol. 5. New York: Springer-Verlag, 1989. Special issue on fractal approximation.

[62] M. F. Barnsley, J. H. Elton, and D. P. Hardin, "Recurrent iterated function systems," *Constructive Approximation*, vol. 5, no. 1, pp. 3–31, 1989.

[63] M. F. Barnsley, "Iterated function systems," in *Chaos and Fractals: The Mathematics Behind the Computer Graphics*, R. L. Devaney, L. Keen, K. T. Alligood, J. A. Yorke, M. F. Barnsley, B. Branner, J. Harrison, and P. J. Holmes, Eds., Providence, RI: American Mathematical Society, 1989, pp. 127–144.

[64] M. F. Barnsley, "Methods and apparatus for image compression by iterated function systems." United States Patent Number 4,941,193, 1990.

[65] M. F. Barnsley and A. D. Sloan, "Method and apparatus for processing digital data." United States Patent Number 5,065,447, 1991.

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

[67] M. F. Barnsley and H. Rising III, *Fractals Everywhere*. 2nd ed., Boston, MA: Academic Press Professional, 1993.

[68] M. F. Barnsley and L. Anson, *The Fractal Transform*. Boston, MA: Jones and Bartlett Publishers, Apr. 1993.

[69] H. Krupnik, D. Malah, and E. Karnin, "Fractal representation of images via the discrete wavelet transform," in *IEEE 18th Conv. of EE in Israel*, Tel-Aviv, Israel, Mar. 7–8, 1995.

[70] J. E. Hutchinson, "Fractals and self-similarity," *Indiana University Mathematics Journal*, vol. 30, pp. 713–747, Sept.–Oct. 1981.

[71] A. E. Jacquin, “Fractal image coding based on a theory of iterated contractive image transformations,” in *Proceedings of the SPIE, Visual Communications and Image Processing*, vol. 1360, Oct. 1–4, 1990, pp. 227–239.

[72] A. E. Jacquin, “Image coding based on a fractal theory of iterated contractive image transformations,” *IEEE Transactions on Image Processing*, vol. 1, pp. 18–30, Jan. 1992.

[73] D. C. Knill, D. Field, and D. Kersten, “Human discrimination of fractal images,” *Journal of Optical Society of America. A, Optics and Image Science*, vol. 7, pp. 1113–1123, June 1990.

[74] F. Arduini, S. Fioravanti, and D. D. Giusto, “On computing multifractality for texture discrimination,” in *Signal Processing VI: Theories and Applications. Proceedings of the Sixth European Signal Processing Conference (EUSIPCO-92)*, Aug. 24–27, 1992, pp. 1457–1460.

[75] S. K. Rogers et al., “Synthetic aperture radar segmentation using wavelets and fractals,” in *Proceedings of the IEEE International Conference on Systems Enginering*, 1991, pp. 21–24.

[76] C. V. Stewart, B. Moghaddam, K. J. Hintz, and L. M. Novak, “Fractaional Brownian motion models for synthetic aperture radar imagery,” *Proceedings of the IEEE*, vol. 81, pp. 1511–1522, Oct. 1993.

[77] W. S. Kuklinski, “Utilization of fractal image models in medical image processing,” in *Proceedings of Fractals in Engineering'94*, Montreal, Canada, June 1–4, 1994, pp. 180–186.

[78] J. Lévy Véhel and P. Mignot, “Multifractal segmentation of images,” in *Proceedings of Fractals in Engineering'94*, Montreal, Canada, June 1–4, 1994, pp. 187–193.

[79] J. Theiler, “Estimating fractal dimension,” *Journal of Optical Society of America. A, Optics and Image Science*, vol. 7, pp. 1055–1073, June 1990.

[80] B. Moghaddam, K. J. Hintz, and C. V. Stewart, "A comparison of local fractal dimension estimation methods," *Pattern Recognition and Image Analysis*, vol. 2, pp. 93–96, Mar. 1992.

[81] E. W. Jacobs, R. D. Boss, and Y. Fisher, "Fractal-based image compression, II," Naval Ocean Systems Center, San Diego, CA, Tech. Rep. 1362, June 1990.

[82] L. Thomas and F. Deravi, "Pruning of the transform space in block-based fractal image compression," in *Proceedings of IEEE ICASSP-93*, vol. V, Minneapolis, MN, Apr. 27–30, 1993, pp. 341–344.

[83] L. Thomas and F. Deravi, "Region-based fractal image compression using heuristic search," *IEEE Transactions on Image Processing*, vol. 4, pp. 832–838, June 1995.

[84] R. E. H. Franich, R. L. Lagendijk, and J. Biemind, "Fractal coding in an object-based system," in *Proceedings of IEEE International Conference on Image Processing*, vol. 2, Austin, TX, Nov. 13–16, 1994, pp. 405–408.

[85] J. H. Elton and Z. Yan, "Approximation of measure by Markov processes and homogeneous affine iterated function systems," *Constructive Approximation*, vol. 5, pp. 69–87, 1989.

[86] G. C. Freeland and T. S. Durrani, "IFS fractals and the wavelet transform," in *Proceedings of IEEE ICASSP-90*, Albuquerque, NM, Apr. 3–6, 1990, pp. 2345–2348.

[87] A. Arneodo, E. Bacry, and J. F. Muzy, "Solving the inverse fractal problem from wavelet analysis," *Europhysics Letters*, vol. 25, pp. 479–484, Mar. 1, 1994.

[88] S. Abenda and G. Turchetti, "Inverse problem for fractal sets on the real line via the moment method," *Il Nuovo Cimento*, vol. 104B, pp. 213–227, Aug. 1989.

[89] S. Abenda, "Inverse problem for one-dimensional fractal measures via iterated function systems and the moment method," *Inverse Problems*, vol. 6, pp. 885–896, Dec. 1990.

[90] S. Abenda, S. Demko, and G. Turchetti, "Local moments and inverse problem for fractal measures," *Inverse Problems*, vol. 8, pp. 739–750, Oct. 1992.

[91] D. Bessis and S. Demko, "Stable recovery of fractal measures by polynomial sampling," *Physica D*, vol. 47, pp. 427–438, 1991.

[92] C. Cabrelli, U. Molter, and E. R. Vrscay, "Recurrent iterated function systems: Invariant measures, a collage theorem and moment relations," in *Fractals in the Fundamental and Applied Sciences*, H.-O. Peitgen, J. M. Henriques, and L. F. Penedo, Eds., New York: Elsevier Science Publishers, 1991, pp. 71–80.

[93] B. Forte and E. R. Vrscay, "Solving the inverse problem for measures using iterated function systems," *Advances in Applied Probability*, submitted in 1993.

[94] C. R. Handy and G. Mantica, "Inverse problems in fractal construction: Moment method solution," *Physica D*, vol. 43, pp. 17–36, May 1990.

[95] G. Mantica, "Techniques for solving inverse fractal problems," in *Fractals in the Fundamental and Applied Sciences*, H.-O. Peitgen, J. M. Henriques, and L. F. Penedo, Eds., New York: Elsevier Science Publishers, 1991, pp. 255–268.

[96] D. van Schooneveld, "The moment method for invariant measure approximation," Department of Mathematics and Computer Science, Delft University of Technology, Internal Report, 1990.

[97] E. R. Vrscay and C. J. Roehrig, "Iterated function systems and the inverse problem of fractal construction using moments," in *Computers and Mathematics*, E. Kaltofen and S. M. Watt, Eds., Berlin, Germany: Springer-Verlag, 1989, pp. 250–259.

[98] E. R. Vrscay, "Moment and collage methods for the inverse problem of fractal construction with iterated function systems," in *Fractals in the Fundamental and Applied Sciences*, H.-O. Peitgen, J. M. Henriques, and L. F. Penedo, Eds., New

York: Elsevier Science Publishers, 1991, pp. 443–461. Proceedings of a conference held in June 6–8, 1990.

[99] E. R. Vrscay and D. Weil, "Missing moment and perturbative methods for polynomial iterated function systems," *Physica D*, vol. 50, pp. 478–492, July 1991.

[100] B. Forte and E. R. Vrscay, "Solving the inverse problem for function/image approximation using iterated function systems, I. Theoretical basis," in *Proceedings of Fractals in Engineering'94*, Montreal, Canada, June 1–4, 1994, pp. 143–152.

[101] B. Forte and E. R. Vrscay, "Solving the inverse problem for function/image approximation using iterated function systems, II. Algorithm and computations," in *Proceedings of Fractals in Engineering'94*, Montreal, Canada, June 1–4, 1994, pp. 153–164.

[102] G. Mantica and A. Sloan, "Chaotic optimization and the construction of fractals: Solution of an inverse problem," *Complex Systems*, vol. 3, pp. 37–62, Feb. 1989.

[103] R. Shonkwiler, F. Mendivil, and A. Deliu, "Genetic algorithms for the 1-D fractal inverse problem," in *4th International Conference on Genetic Algorithms (ICGA 91)*, San Diego, CA, July 13–16, 1991, pp. 495–501.

[104] R. Rinaldo and A. Zakhor, "Inverse problem for two-dimensional fractal sets using the wavelet transform and the moment method," in *Proceedings of IEEE ICASSP-92*, vol. IV, San Francisco, CA, Mar. 23–26, 1992, pp. 665–668.

[105] R. Rinaldo and A. Zakhor, "Fractal approximation of images," in *DCC'93: Data Compression Conference*, Snowbird, UT, Mar.–Apr. 1993, p. 451.

[106] R. Rinaldo and A. Zakhor, "Inverse and approximation problem for two-dimensional fractal sets," *IEEE Transactions on Image Processing*, vol. 3, pp. 802–820, Nov. 1994.

[107] C. A. Cabrelli, B. Forte, U. M. Molter, and E. R. Vrscay, "Iterated fuzzy set systems: A new approach to the inverse problem for fractals and other sets," *Journal of Mathematical Analysis and Applications*, vol. 171, pp. 79–100, Nov. 15, 1992.

[108] D. J. Nettleton and R. Garigliano, "Evolutionary algorithms and a fractal inverse problem," *Biosystems*, vol. 33, no. 3, pp. 221–231, 1994.

[109] J. H. Chen and J. D. Kalbfleisch, "Inverse problems in fractal construction: Hellinger distance method," *Journal of the Royal Statistical Society, Series B Methodological*, vol. 56, no. 4, pp. 687–700, 1994.

[110] A. E. Jacquin, "Fractal image coding: A review," *Proceedings of the IEEE*, vol. 81, pp. 1451–1465, Oct. 1993.

[111] G. E. Øien, S. Lepsøy, and T. A. Ramstad, "An inner product space approach to image coding by contractive transformations," in *Proceedings of IEEE ICASSP-91*, Toronto, Canada, May 14–17, 1991, pp. 2773–2776.

[112] D. M. Monro and F. Dudbridge, "Fractal approximation of image blocks," in *Proceedings of IEEE ICASSP-92*, vol. III, San Francisco, CA, Mar. 23–26, 1992, pp. 485–488.

[113] D. M. Monro and F. Dudbridge, "Fractal block coding of images," *Electronics Letters*, vol. 28, pp. 1053–1055, May 21, 1992.

[114] Y. Fisher, "A discussion of fractal image compression," in *Chaos and Fractals: New Frontiers of Science*, H.-O. Peitgen, H. Jürgens, and D. Saupe, Eds., Springer-Verlag, 1992, pp. 903–919.

[115] Y. Fisher, "Fractal image compression," in *SIGGRAPH '92 Course Notes: From Folk Art to Hyperreality*, P. Prusinkiewicz, Ed., 1992, pp. 1–21.

[116] 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, Dec. 1992.

[117] S. Lepsøy, G. Øien, and T. A. Ramstad, "Attractor image compression with a fast non-iterative decoding algorithm," in *Proceedings of IEEE ICASSP-93*, vol. V, Minneapolis, MN, Apr. 27–30 1993, pp. 337–340.

[118] G. Vines and M. H. Hayes, III, "Adaptive IFS image coding with proximity maps," in *Proceedings of IEEE ICASSP-93*, vol. V, Minneapolis, MN, Apr. 27–30, 1993, pp. 349–352.

[119] K. U. Barthel, J. Schüttemeyer, T. Voyé, and P. Noll, "A new image coding technique unifying fractal and transform coding," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, Austin, TX, Nov. 13–16, 1994, pp. 112–116.

[120] Z. Xiong, K. Ramchandran, M. T. Orchard, and K. Asai, "Wavelet packets-based image coding using joint space-frequency quantization," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, Austin, TX, Nov. 13–16, 1994, pp. 324–328.

[121] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.

[122] Y. Fisher and S. Menlove, "Fractal encoding with HV partitions," in *Fractal Image Compression: Theory and Application*, Y. Fisher, Ed., New York: Springer-Verlag, 1995, pp. 119–136.

[123] K. Culik II and J. Kari, "Inference algorithms for WFA and image compression," in *Fractal Image Compression: Theory and Application*, Y. Fisher, Ed., New York: Springer-Verlag, 1995, pp. 243–258.

[124] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, pp. 30–44, Apr. 1991.

[125] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard.* New York: Van Nostrand Reinhold, 1993.

[126] I. K. Kim and R.-H. Park, "Image coding based on fractal approximation and vector quantization," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, Austin, TX, Nov. 13–16, 1994, pp. 132–136.

[127] G. Lu and T. L. Yew, "Image compression using quadtree partitioned iterated function systems," *Electronics Letters*, vol. 30, pp. 23–24, Jan. 6, 1994.

[128] M. Crouse and K. Ramchandran, "Joint thresholding and quantizer selection for decoder-compatible baseline JPEG," in *Proceedings of IEEE ICASSP-95*, Detroit, MA, May 9–12, 1995.

[129] J. M. Beaumont, "Image data compression using fractal techniques," *British Telecommunications Technical Journal*, vol. 9, pp. 93–109, Oct. 1991.

[130] L. P. Hurd, M. A. Gustavus, and M. F. Barnsley, "Fractal video compression," in *Digest of Papers. Thirty-Seventh IEEE Computer Society International Conference (COMPCON)*, Feb. 24–28, 1992, pp. 41–42.

[131] B. Hürtgen and P. Büttgen, "Fractal approach to low rate video coding," in *Proceedings of the SPIE, Visual Communications and Image Processing*, vol. 2094, Cambridge, MA, Nov. 8–11, 1993, pp. 120–131.

[132] H. Li, M. Novak, and R. Forchheimer, "Fractal-based image sequence compression scheme," *Optical Engineering*, vol. 32, pp. 1588–1595, July 1993.

[133] M. S. Lazar and L. T. Bruton, "Fractal coding of digital video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, pp. 297–308, June 1994.

[134] E. Reusens, "Sequence coding based on the fractal theory of iterated transformations systems," in *Proceedings of the SPIE, Visual Communications and Image Processing*, vol. 2094, Cambridge, MA, Nov. 8–11, 1993, pp. 132–140.

[135] O. Kiselyov and P. Fisher, "Self-similarity of the multiresolutional image/video decomposition: Smart expansion as compression of still and moving pictures," in

DCC'94: Data Compression Conference, Snowbird, UT, IEEE Computer Society Press, Mar. 29–31, 1994, p. 514.

[136] D. L. Wilson, J. A. Nicholls, and D. M. Monro, "Rate buffered fractal video," in *Proceedings of IEEE ICASSP-94*, vol. 5, Adelaide, Australia, Apr. 19–22, 1994, pp. 505–508.

[137] D. M. Monro and J. A. Nicholls, "Real time fractal video for personal communications," in *Proceedings of Fractals in Engineering'94*, Montreal, Canada, June 1–4, 1994, pp. 206–209.

[138] Y. Fisher, D. Rogovin, and T. P. Shen, "Fractal (self-VQ) encoding of video sequences," in *Proceedings of the SPIE, Visual Communications and Image Processing*, Chicago, IL, Sept. 25–28, 1994.

[139] A. Bogdan, "Multiscale (inter/intra-frame) fractal video coding," in *Proceedings of IEEE International Conference on Image Processing*, vol. 1, Austin, TX, Nov. 13–16, 1994, pp. 760–764.

[140] B.-B. Paul and M. H. Hayes, "Fractal-based compression of motion video sequences," in *Proceedings of IEEE International Conference on Image Processing*, vol. 1, Austin, TX, Nov. 13–16, 1994, pp. 755–759.

[141] B.-B. Paul and M. H. Hayes III, "Video coding based on iterated function systems," in *Proceedings of IEEE ICASSP-95*, vol. 4, Detroit, MI, May 9–12, 1995, pp. 2269–2272.

[142] D. Saupe, "Breaking the time complexity of fractal image compression," Institut für Informatik, Universität Freiburg, Tech. Rep. 53, 1994.

[143] D. Saupe and R. Hamzaoui, "Complexity reduction methods for fractal image compression," in *I.M.A. Conf. Proc. on Image Processing; Mathematical Methods and Applications*, J. M. Blackledge, Ed., New York, Oxford University Press, Sept. 1994.

[144] D. Saupe, "Accelerating fractal image compression by multi-dimensional nearest neighbor search," in *DCC'95: Data Compression Conference*, J. A. Storer and M. Cohn, Eds., Snowbird, UT, Mar. 28–30, 1995, pp. 222–231.

[145] S. J. Leon, *Linear Algebra with Applications.* 3rd ed., New York: Macmillan Publishing Company, 1990.

[146] Y. Fisher and A. Lawrence, "Fractal image encoding," Netrologic, Inc., Tech. Rep., Nov. 17, 1990. SBIR Phase I, Final Report.

[147] Y. Fisher, E. W. Jacobs, and R. D. Boss, "Iterated transform image compression," Naval Ocean Systems Center, San Diego, CA, Tech. Rep. 1408, Apr. 1991. Final Report. Sep 89–Oct 90.

[148] Y. Fisher, *Fractal Image Compression: Theory and Application.* New York: Springer-Verlag, 1995.

[149] C.-T. Chen, *Linear System Theory and Design.* New York: Holt, Rinehart and Winston, Inc., 1984.

[150] D. G. Luenberger, *Introduction to Dynamic Systems. Theory, Models, and Applications.* New York: John Wiley & Sons, 1979.

[151] L. M. Lundheim, "Fractal signal modelling for source coding," PhD dissertation, The Norwegian Institute of Technology, Trondheim, Norway, 1992.

[152] B. Hürtgen and S. F. Simon, "On the problem of convergence in fractal coding schemes," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, Austin, TX, Nov. 13–16, 1994, pp. 103–106.

[153] W.-K. Chen, *Applied Graph Theory.* New York: North-Holland Publishing Company, 1976.

[154] F. Harry, "A graph theoretic method for the complete reduction of a matrix with a view toward finding its eigenvalues," *Journal of Mathematics and Physics*, vol. 38, pp. 104–111, Apr. 1959.

[155] C. H. Houpis and G. B. Lamont, *Digital Control Systems: Theory, Hardware, Software.* New York: McGraw-Hill Book Company, 1985.

[156] G. Davis, "Adaptive nonlinear approximations," PhD dissertation, New York University, New York, NY, 1994.

[157] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive nonlinear approxiamtions," *Journal of Constructive Approximation*, submitted in 1995.

[158] J. H. Friedman and W. Stuetzle, "Projection pursuit regression," *Journal of the American Statistical Association*, vol. 76, pp. 817–823, Dec. 1981.

[159] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," *International Journal of Control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[160] R. Neff and A. Zakhor, "Matching pursuit video coding at very low bit rates," in *DCC'95: Data Compression Conference*, Snowbird, UT, Mar. 28–30, 1995.

[161] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal projection pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proceedings of Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, vol. 1, Pacific Grove, CA, Nov. 1–3, 1993, pp. 40–44.

[162] M. Gharavi-Alkhansari and T. S. Huang, "A generalized image coding method using fractal-based techniques," in *Proceedings of SBT/IEEE International Telecommunications Symposium*, Rio de Janeiro, Brazil, Aug. 22–26, 1994.

[163] D. G. Luenberger, *Linear and Nonlinear Programming.* 2nd ed., New York: Addison Wesley Publishing Company, 1989.

[164] G. E. Øien, "l_2-optimal attractor image coding with fast decoder convergence," PhD dissertation, The Norwegian Institute of Technology, Trondheim, Norway, 1993.

[165] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. IT-23, pp. 337–343, May 1977.

[166] T. Saito, Hideaki, R. Abe, T. Komatsu, and H. Harashima, "Self-organizing pattern-matching coding for picture signals," in *Proceedings of IEEE ICASSP-89*, Glasgow, Scotland, May 23–26, 1989, pp. 1671–1674.

[167] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Boston, MA: Kluwer Academic Publishers, 1992.

[168] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proceedings of the Institute of Radio Engineers (IRE)*, vol. 40, pp. 1098–1101, 1952.

[169] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression.* Prentice Hall Advanced Reference Series, Computer Science, Englewood Cliffs, NJ: Prentice Hall, 1990.

[170] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, pp. 520–606, June 1987.

[171] R. M. Neal, "Low-precision arithmetic coding implementation," May 19, 1993. ANSI C source code available by anonymous ftp from ftp.cs.toronto.edu, directory /pub/radford/.

[172] H. A. Kaouri, "Fractal coding of still images," in *IEE Sixth International Conference on Digital Processing of Signals in Communications (Conf. Pub. No. 340)*, Loughborough, UK, Sept. 2–6, 1991, pp. 235–239.

[173] H. G. Musmann, P. Pirsch, and H.-J. Grallert, "Advances in picture coding," *Proceedings of the IEEE*, vol. 73, pp. 523–548, Apr. 1985.

[174] International Telecommunication Union (ITU), *Video Codec for Audiovisiual Services at* $p \times 64$ *kb/s.* Geneva, Switzerland, Mar. 1993. Recomendation H.261.

[175] International Telecommunication Union (ITU), *Video Coding for Low Bit Rate Communication.* Geneva, Switzerland, Mar. 1996. Recomendation H.263.

[176] Portable Research Video Group, *PVRG-P64 Software Codec, Version 1.2*, 1993. Available from ftp://havefun.stanford.edu/pub/p64/.

[177] Telenor R&D, *Telenor's H.263 Software, Version 2.0.* Oslo, Norway, 1996. Available from http://www.nta.no/brukere/DVC/h263_software/.

[178] M. Ali and T. G. Clarkson, "Using linear fractal interpolation functions to compress video images," in *Proceedings of Fractals in Engineering'94*, Montreal, Quebec, Canada, June 1–4, 1994, pp. 232–236.

[179] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* Englewood Cliffs, NJ: Prentice Hall, 1989.

[180] Y. Fisher, "Fractal image compression with quadtrees," in *Fractal Image Compression: Theory and Application*, Y. Fisher, Ed., New York: Springer-Verlag, 1995, pp. 55–77.

[181] M. Nelson and J.-L. Gailly, "Fractal image compression," in *The Data Compression Book*, New York: M&T Books, 2nd ed., 1996, pp. 457–510.

[182] Iterated Systems, Inc., *Fractal Imager.* Iterated Systems, Inc., Norcross, GA. Coding software available from http://www.iterated.com.

VITA

Mohammad Gharavi-Alkhansari was born in 1963 in Tehran, Iran. He received the B.S. degree from the University of Tehran, Tehran, Iran, in September 1987, and the M.S. degree from Iowa State University, Ames, Iowa, in December 1990; both degrees in Electrical Engineering. Since June of 1991, he has been working towards the Ph.D. degree at the Electrical and Computer Engineering Department at the University of Illinois at Urbana-Champaign. From June 1989 to May 1991, he was a research assistant at the Department of Electrical Engineering and Computer Engineering at Iowa State University, and from June 1991 to the present at the Beckman Institute for Advanced Science and Technology at the University of Illinois at Urbana-Champaign. His research interests include computer vision, and digital signal processing and coding applied to images, video and biomedical data. He is a member of Tau Beta Pi and Sigma Xi.