Texture Segmentation-Based Image Coder Incorporating Properties of the Human Visual System

Jongwhan Jang

Center for Communications and Signal Processing
Department Electrical and Computer Engineering
North Carolina State University

TR-90/18
November 1990

Contents

List of Figures v

List of Tables ix

1 Introduction 1

2 An Overview of Image Compression 5

2.1 Introduction 5

2.2 Statistically-Based Image Compression Techniques 7

2.3 Symbolically-Based Image Compression Techniques 11

2.3.1 The Human Visual System (HVS) 13

2.3.2 Pyramidal Image Compression 17

2.3.3 Directional Decomposition Based Image Compression 19

2.3.4 Segmentation-Based Image Compression 20

2.3.5 Fractal Based Image Compression 24

2.4 Conclusions 26

3 Texture Analysis 27

3.1 Introduction 27

3.2 Fractal Geometry in Image Analysis 29

3.2.1 Fractal Dimension 31

3.2.2 Fractional Brownian Function 37

3.3 Conclusions 41

4 A New Approach for Segmentation-Based Image Coding 42

4.1 Introduction 42

4.2 The Codec 43

4.3 The Transmitter 44

4.3.1 Preprocessing 44

4.3.2 Image Segmentation 46
4.3.3 The Mixed Coder 47
4.4 The Receiver 48
4.5 The Basic Principle of the Mixed Coder 49

5 Image Segmentation Using Properties of the HVS and Fractals 52
5.1 Image Segmentation 57
5.2 Determination of the Block Size for Estimating the Fractal Dimension 60
5.2.1 Experimental Results for Determining the Best Block Size . . 61
5.3 Thresholds for the Fractal Dimension 68
5.3.1 The Experimental Results for Determining Thresholds D_1 and D_2 70
5.4 Selection of the Threshold for Regions Belonging to Perceived Constant Intensity 83
5.4.1 The Experimental Results 84
5.5 Conclusion 91

6 A Texture Segmentation-Based Image Coder 95
6.1 Introduction 95
6.2 The Mixed Encoder 96
6.2.1 The Boundary Coding 97
6.2.2 The Perceived Constant Regions 98
6.2.3 The Smooth and Rough Textural Regions 99
6.3 Nonoverlap and Overlap Segmentation Method 101
6.4 The Experimental Results for the Nonoverlap and Overlap Segmentation 103
6.4.1 The Pixel Percentage for Each Class 103
6.4.2 Variability of D_1 and D_2 111
6.4.3 The Boundary Coding 114
6.4.4 Coding of the Constant Regions 116
6.4.5 Coding of the Smooth and Rough Textural Regions 118
6.4.6 Bit Rate Computation 125
6.4.7 Performance Evaluation of the CODEC 128
6.5 Conclusion 145

7 Conclusions 146

8 Bibliography 148

9 Appendices **155**

9.1 Runlength Coding 155

9.2 Crack Coding 156

9.3 Arithmetic Coding 157

List of Figures

2.1 A general statistically-based image compression system 8
2.2 A general symbolically-based image compression system. 11
2.3 A simple contrast sensitivity measurement 14
2.4 Perspective drawing of the split-field experiment 15
2.5 A typical MTF curve 16
2.6 Block diagram of the pyramid coding method 18

3.1 A one-dimensional function is shown in (a) and its covering blanket for $\epsilon = 1, 2$ are shown in (b) and (c), respectively. The blanket areas are A(1) = 47 and A(2) = 78. The respective measured lengths are L(1) = 47/2 = 23.5 and L(2) = 78/4 = 19.5. 33
3.2 A natural image 36
3.3 The calculation of the fractal dimension of a natural image. The values of the estimated slope and estimated fractal dimension are -0.62729 and 2.62729, respectively. 36

4.1 The overall block diagram of the segmentation-based coder. 43
4.2 The block diagram of the transmitter characteristics. 45
4.3 The block diagram of the receiver characteristics. 48

5.1 Original test images. Each image is 256 × 256 pixels, with 256 gray levels. (a) Miss USA. (b) Lena. (c) House. 56
5.2 Centroid linkage window 58
5.3 Plot of fractal dimension versus block size in Miss USA. Miss USA with three subimages is given on the top. Three subimages on the top, middle, and bottom belong to perceived constant intensity, rough texture, and smooth texture respectively. A plot of fractal dimension versus block size is given on the bottom. The curves with a diamond symbol, a cross symbol, and a square symbol correspond to perceived constant intensity, rough texture, and smooth texture respectively. . . 64

5.4 Plot of fractal dimension versus block size in Lena. Lena with three subimages is given on the top. Three subimages on the top, the bottom and left corner, and the bottom and right corner belong to perceived constant intensity, rough texture, and smooth texture respectively. A plot of fractal dimension versus block size is given on the bottom. The curves with a diamond symbol, a cross symbol, and a square symbol correspond to perceived constant intensity, rough texture, and smooth texture respectively. 65

5.5 Plot of fractal dimension versus block size. House with three subimages is on the top. Subimages on the top and left corner, the top and right corner, and the bottom and right corner in the image belong to rough texture, perceived constant intensity, and smooth texture respectively. A plot of fractal dimension versus block size is given on the bottom. The curves with a diamond symbol, a cross symbol, and a square symbol correspond to perceived constant intensity, rough texture, and smooth texture respectively. 66

5.6 A typical MTF curve 69

5.7 Estimation of the fractal dimension for the perceived constant intensity blocks in Miss USA. Five 8×8 blocks are used. 73

5.8 Estimation of the fractal dimension for the smooth texture blocks in Miss USA. Five 8×8 blocks are used. 74

5.9 Estimation of the fractal dimension for the rough texture blocks in Miss USA. Five 8×8 blocks are used. 75

5.10 Estimation of the fractal dimension for the perceived constant intensity blocks in Lena. Five 8×8 blocks are used. 76

5.11 Estimation of the fractal dimension for the smooth texture blocks in Lena. Five 8×8 blocks are used. 77

5.12 Estimation of the fractal dimension for the rough texture blocks in Lena. Five 8×8 blocks are used. 78

5.13 Estimation of the fractal dimension for the perceived constant intensity blocks in House. Five 8×8 blocks are used. 79

5.14 Estimation of the fractal dimension for the smooth texture blocks in House. Five 8×8 blocks are used. 80

5.15 Estimation of the fractal dimension for the rough texture blocks in House. Five 8×8 blocks are used. 81

5.16 A plot of the fractal dimensions of the fifteen subimages for each class. The x-axis represents the fractal dimension and the y-axis the number of blocks at that fractal dimension. The curve with a diamond symbol corresponds to the perceive constant intensity, the curve with a cross symbol to the smooth texture, and the curve with the square symbol to the rough texture. 82
5.17 The mean of five subjects' JND measurements 87
5.18 The original JND curve and the approximated JND curve. The bold line corresponds to the approximated JND curve 87
5.19 The segmented images using $th_{con} = 5.5$. (a) Miss USA. (b) Lena. (c) House. 89
5.20 The segmented images using th_{JND}. (a) Miss USA. (b) Lena. (c) House. 91
5.21 The segmented images using th_{ap}. (a) Miss USA. (b) Lena. (c) House. 93

6.1 Comparison of the nonoverlap and overlap method. An image size and block size are 8×8 and 4×4 respectively. 102
6.2 The class type images of Miss USA. (a) 0 percent overlap. (b) 50 percent overlap. (c) 75 percent overlap. 106
6.3 The class type images of Lena. (a) 0 percent overlap. (b) 50 percent overlap. (c) 75 percent overlap. 108
6.4 The class type images of House. (a) 0 percent overlap. (b) 50 percent overlap. (c) 75 percent overlap. 110
6.5 Modeling a rough texture region using a 2-D polynomial. From left to to right are the 32×32 tree subimage in House; the original, the zero-order model, the first-order model, and the second-order model. . 120
6.6 Modeling a smooth texture region using a 2-D polynomial. From left to to right are the 32×32 chin subimage in Miss USA; the original, the zero-order model, the first-order model, and the second-order model. 121
6.7 Modeling a rough texture region using a 1-D polynomial. From left to to right are the 32×32 tree subimage in House; the original, the zero-order model, the first-order model, and the second-order model. . 122
6.8 Modeling a smooth texture region using a 1-D polynomial. From left to to right are the 32×32 chin subimage in Miss USA; the original, the zero-order model, the first-order model, and the second-order model. 123
6.9 The decoded images of the test images with $D_1 = 2.035$ and $D_2 = 2.363$. (a) The decoded image of Miss USA. (b) The decoded image of Lena. (c) The decoded image of House. 127
6.10 Plot of SNR versus rate, bit/pixel for Lena. D_1 is variable and D_2 is fixed to 2.363. 140

6.11 Plot of SNR versus rate, bit/pixel for House. D_1 is variable and D_2 is fixed to 2.363. 140
6.12 Plot of SNR versus rate, bit/pixel for House. D_1 is fixed 2.035 and D_2 is variable. 141
6.13 The reconstructed images for Miss USA. The images on the top and the bottom are coded at 1.0 bit/pixel with $D_1 = 2.001$ and $D_2 = 2.76$ and 0.2 bit/pixel with $D_1 = 2.005$ and $D_2 = 2.36$ respectively. 142
6.14 The reconstructed images for Lena. The images on the top and the bottom are coded at 1.0 bit/pixel with $D_1 = 2.005$ and $D_2 = 2.71$ and 0.2 bit/pixel with $D_1 = 2.050$ and $D_2 = 2.361$ respectively. 143
6.15 The reconstructed images for House. The images on the top and the bottom are coded at 1.0 bit/pixel with $D_1 = 2.005$ and $D_2 = 2.71$ and 0.2 bit/pixel with $D_1 = 2.06$ and $D_2 = 2.41$ respectively. 144

9.1 A crack code. (a) Set S; each point is labeled with a different latter. (b) Clockwise sequence of cracks around the border, beginning with the crack A_t at the top of A. The subscripts of t, r, b, l denote top, right, bottom, and left, respectively. 156

List of Tables

5.1 Statistics of fractal dimension in Miss USA 67
5.2 Statistics of fractal dimension in Lena 67
5.3 Statistics of fractal dimension in House 67
5.4 Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b). 73
5.5 Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b). 74
5.6 Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b). 75
5.7 Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b). 76
5.8 Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b). 77
5.9 Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b). 78

5.10 Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b). 79
5.11 Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b). 80
5.12 Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b). 81
5.13 The number of segments for test images using $th_{con} = 5.5$. 89
5.14 The number of segments for test images using th_{JND}. 91
5.15 The number of segments for test images using th_{ap}. 93

6.1 Pixel percentage of each class in Miss USA 106
6.2 Pixel percentage of each class in Lena 108
6.3 Pixel percentage of each class in House 110
6.4 Percentage of the pixels in each class in Miss USA with 50% overlap, D_1 variable, and $D_2 = 2.363$. 112
6.5 Percentage of the pixels in each class in Miss USA with 50% overlap, $D_1 = 2.035$, and D_2 variable. 112
6.6 Percentage of the pixels in each class in Lena with 50% overlap, D_1 variable, and $D_2 = 2.363$. 112
6.7 Percentage of the pixels in each class type in Lena with 50% overlap, $D_1 = 2.035$, and D_2 variable. 113
6.8 Percentage of the pixels in each class in House with 50% overlap, D_1 variable, and $D_2 = 2.363$. 113
6.9 Percentage of the pixels in each class in House with 50% overlap, $D_1 = 2.035$, and D_2 variable. 113
6.10 Summary of the numbers of the total segments and the boundary points using $D_1 = 2.035$ and $D_2 = 2.363$. 115
6.11 Summary of bits to represent the boundary using the three different coding. 115
6.12 Summary of the numbers of the total segments and the perceived constant regions using $D_1 = 2.035$ and $D_2 = 2.363$. 117
6.13 Summary of the number of bits to represent the constant region. . . . 117

6.14 The sum of squared error (SSE) values for each model. 120
6.15 The sum of squared error (SSE) values for each model. 121
6.16 The sum of squared error (SSE) values for each model. 122
6.17 The sum of squared error (SSE) values for each model. 123
6.18 Summary of the numbers of the segments in the smooth and rough textural regions and the number of bits to represent those regions using a 1-D polynomial. The polynomial coefficients were encoded using the arithmetic code. 124
6.19 Summary of coding information in Miss USA. D_1 is variable and D_2 is fixed to 2.363. 131
6.20 Summary of coding information in Lena. D_1 is variable and D_2 is fixed to 2.363. 132
6.21 Summary of coding information in House. D_1 is variable and D_2 is fixed to 2.363. 133
6.22 Summary of coding information in Miss USA. D_1 is fixed to 2.035 and D_2 is variable. 134
6.23 Summary of coding information in Lena. D_1 is fixed to 2.035 and D_2 is variable. 135
6.24 Summary of coding information in House. D_1 is fixed to 2.035 and D_2 is variable. 136
6.25 Summary of coding information in Miss USA. $D_1 = 2.035$, $D_2 = 2.363$, $TH_{er2} = 60$ and TH_{er1} is variable, where TH_{er1} and TH_{er2} are the thresholds of regions beloinging to the smooth and the rough textures respectively. 137
6.26 Summary of coding information in Lena. $D_1 = 2.035$, $D_2 = 2.363$, $TH_{er2} = 60$ and TH_{er1} is variable. 137
6.27 Summary of coding information in House. $D_1 = 2.035$, $D_2 = 2.363$, $TH_{er2} = 60$ and TH_{er1} is variable. 138
6.28 Summary of coding information in Miss USA. $D_1 = 2.035$, $D_2 = 2.363$, $TH_{er1} = 25$ and TH_{er2} is variable. 138
6.29 Summary of coding information in Lena. $D_1 = 2.035$, $D_2 = 2.363$, $TH_{er1} = 25$ and TH_{er2} is variable. 139
6.30 Summary of coding information in House. $D_1 = 2.035$, $D_2 = 2.363$, $TH_{er1} = 25$ and TH_{er2} is variable. 139

1

Introduction

The digital representation of an image requires a very large number of bits. For example, a 512×512 pixel, 256 gray level image requires over two million bits. This large number of bits is a substantial drawback when it is necessary to store or transmit a digital image. Prior to transmission or storage, one would like to have a system that reduces this number as much as possible, while keeping the degradation in the decoded image to a minimum. This is the goal of image compression, often referred to as image coding.

Early efforts in image compression, solely guided by information theory, led to a plethora of methods. The compression ratio, starting at one with the first digital picture in the early 1960's, appeared to have reached a saturation level around 10:1 in the early 1980's. This, however, did not mean that the upper bound given by the entropy of the source had also been reached. First, this entropy is not known and depends heavily on the model used for the source, i.e., the digital image. Second, information theory does not take into account what the human eye sees and how it sees. Recently, techniques attempting to overcome these limitations are incorporating properties of the human visual system (HVS) and tools of image analysis into image compression to achieve high compression ratios with small loss in visual quality. This

reasoning follows from the fact that in many compression applications, a human is the final observer of the image operated upon. Applications of various models of the HVS have in fact been empirically found to improve compression performance [25, 43, 57, 71, 83].

One such technique is segmentation-based image compression [7, 39, 43, 71]. In segmentation-based image compression, the image to be compressed is segmented, i.e. the pixels in the image are separated into mutually exclusive spatial regions based on some criteria. Once the image has been segmented, information is extracted describing the boundaries (shapes) and textures (interiors) of the image segments, and compression is achieved by efficiently encoding this information. Unfortunately, there are limitations with segmentation-based image compression. The main limitation is due to the fact that the image data have been segmented into regions of constant intensity. In complicated texture areas, a good representation of the texture requires many small segments. However, in order to get low bit rates, the number of segments must be limited and thus the quality is degraded.

We overcome the texture representation problem in the research described in this report by proposing a methodology for segmenting an image into texturally homogeneous regions with respect to the degree of roughness as perceived by the HVS. The segmented image information is then encoded for transmission. The proposed algorithm is applied to three different types of imagery. The first is a head and shoulder image with little texture variation. This image is typical of video teleconferencing applications and one which the previously proposed segmentation-based compression techniques are best suited. The second is a complex image with many edges and the third is a natural outdoor image with highly textured areas. The previous proposed segmentation-based compression techniques do not work well for the second

and third images. However, the proposed texture-based image compression technique works well for not only the first but also the second and the third type of image.

In the proposed texture-based image compression algorithm, the fractal dimension, the expected value, and the just noticeable difference (JND) are the measures used to characterize the texture information. The measured quantities are incorporated into a centroid-linkage region growing algorithm [32] which is used to segment each image into three texture classes. The region growing algorithm is directed by the texture feature distance between image blocks. After segmentation, the image can be viewed as being composed of region boundaries and texturally homogeneous regions. Since the decoded images will be viewed by humans, our prime motivation is the production of visually pleasing segmented images which can be encoded with high compression ratios.

The second aspect of this work is to propose appropriate compression techniques for the three textural classes and the region boundaries. The three classes, I, II, and III, are perceived constant intensity, smooth texture, and rough texture, respectively. A binary map representing the boundaries of the regions is encoded using a modified adaptive arithmetic coder [62, 73, 76]. In our work, the representation of the boundary information using blocks, not pixels, provides us with higher compression. Regions which belong to class I are modeled as flat planes, hence they only need to have their mean intensity value transmitted. The means are then encoded using a modified adaptive arithmetic code. The highest compression ratio is achieved for class I. Because of the sensitivity of the HVS to middle range spatial frequencies, regions belonging to class II require a more accurate representation. Regions belonging to class III contain the highest spatial frequencies. The HVS is less sensitive to the high spatial frequencies, thus these regions can be more highly compressed. The texture

information in class II and III are modeled by polynomial functions. Higher compression is achieved for class III by allowing the error between the original image and the modeled image to be greater than for class II. The result is a segmentation-based image coding system with high compression and a small loss in visual quality.

In summary, the main contributions of this report are:

- a new technique for segmenting an image into texturally consistent regions;
- use of the fractal dimension for relating the spatial frequency of textures to the spatial frequency response of the human visual system;
- a new algorithm for encoding the segmented image information; and
- good quality compressed images at 0.2 to 0.4 bpp for higher textural images.

In Chapter 2, the prerequisite background material, emphasizing work which is pertinent to the methods used in this research is covered, as are the properties of the HVS. In Chapter 3, fractal models in texture analysis are covered. The fractal dimension and the power spectral density (PSD) of the fractional Brownian function (FBF) are derived and discussed. In Chapter 4, a complete description of the new image compression system is given. In Chapter 5, image segmentation is developed and evaluated using properties of the HVS and the fractal dimension. In Chapter 6, the mixed coding scheme is described and the performance of the new image compression system is evaluated using computer simulated data of actual images. Finally, conclusions and further research are provided in Chapter 7.

2

An Overview of Image Compression

2.1 Introduction

Many data processing applications involve storage of large volumes of data. For example, to represent a 512×512 pixel, 256 gray level digital image, over two million bits are required. In addition, the number of data processing applications such as in the areas of meteorology, military reconnaissance, medicine, and electronic publishing is increasing rapidly. At the same time, there has been a proliferation of computer communication networks and teleprocessing applications, which involve massive transfers of data over long-distance communication links. For example, to transmit an uncompressed 512×512 pixel, 256 gray level digital image over a 64Kbit/s channel requires more than thirty seconds. The requirements are even higher for a color image of the same size. To reduce the data storage requirements and the data communication costs, there is a need to reduce the redundancy in the data representation. Image compression techniques have attempted to reduce the amount of data needed to transmit or store digital images, while keeping the degradation in the quality of the decoded image to a minimum.

In this chapter, we briefly review the recent advances in image compression techniques. In general, any image compression method can be broadly classified as being

either statistically-based (algebraic) or symbolically-based (structural). The statistical approaches to image compression are based on information theoretic principles and the methods used usually involve very localized, pixel-oriented features of the image. A summary of these techniques is presented in Section 2.2. However, due to the limitations of the statistical approaches, researchers were interested in finding a new approach to image compression for very low bit rate applications. Many of the new approaches are known as symbolically-based (second generation [43]) image compression. Symbolically-based image compression methods employ tools of image analysis and properties of the human visual system (HVS) to achieve good image quality at very low data rates. In symbolically-based compression, the geometric structure of the image scene is emphasized, as opposed to the algebraic structure of the pixels used by statistically-based compression methods. In Section 2.3 we summarize the work in the developing area of symbolically-based image compression.

Image compression methods can be further classified beyond the two main categories mentioned above. For example, the classification can be based on the techniques the compression method employs and the distortion the compression method introduces in the image. One possible classification of compression methods is as adaptive or non-adaptive. In a typical image, the statistical characteristics of an image differ considerably from one region to another. For example, walls and skies have approximately uniform background intensities, whereas faces and trees have large, detailed variations in intensities. To compensate for this, parameters of the coder are adapted to variations in the local statistics of the image, such as local image contrast. A coder that employs such parameter variation techniques is classified as adaptive. If this type of variation is not used, the compression technique is non-adaptive. Some examples of adaptive image compression techniques are adaptive differential pulse code modu-

lation, adaptive delta modulation, and adaptive transform compression [27, 28, 90].

Another classification describes whether the method is distortionless or non distortionless. If a compression method is distortionless then the decoded image is perfect recreation of the original image. Nearly all distortionless techniques are based on information theoretic approaches and usually attain data rates in the neighborhood of two or four bits per pixel (bpp) for an original 8 bpp image [40]. Non-distortionless compression methods introduce differences between the decoded image and the original image, but they allow lower data rates. However, the decoded image must be kept as close to the original as possible.

We briefly introduce the basic concept of the statistically-based image compression techniques and then describe in more detail the symbolically-based image compression techniques.

2.2 Statistically-Based Image Compression Techniques

Most of the compression techniques developed from the early 1960's to the present fit into the category of the statistically-based image compression techniques. A block diagram of the general statistical image compression system is shown in Figure 2.1. The statistically-based image compression techniques address the image compression problem from an information theory viewpoint, with the focus on eliminating the statistical redundancy among the pixels in the image.

Ideally, the most useful preprocessor, as shown in Figure 2.1, is a transformation of the image to the most suitable domain for coding. The best one can do is find a preprocessor that maps the data into uncorrelated spatial-domain data or a set of independent transform-domain coefficients. For example, the mapping might remove

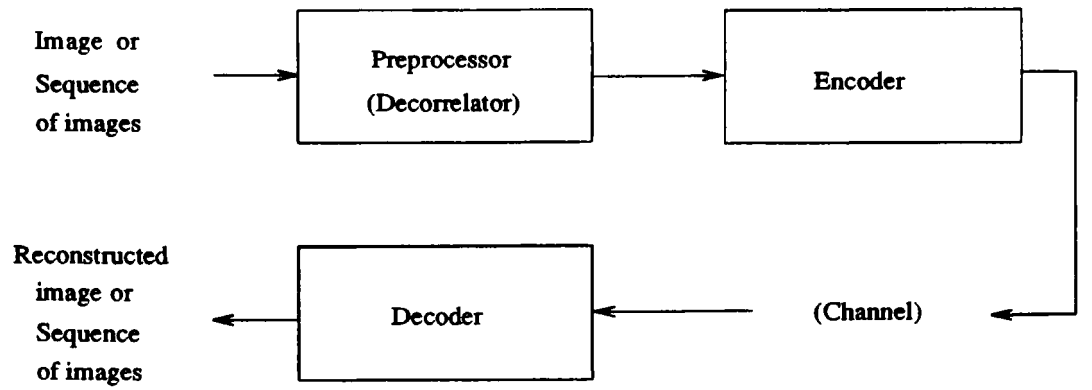


Figure 2.1: A general statistically-based image compression system

the mutual redundancy between successive pixels or take the discrete Fourier transform of the image pixels. The desire for the pixels to be independent is based on rate-distortion theory. Rate distortion theory defines the optimum coder to be the coder that attains the best possible signal fidelity for a given date rate, or the coder that attains the best possible data for a given signal fidelity [26].

Shannon has shown that for any data source, better rates can be achieved by coding blocks of data, rather than individual data points. In fact, the optimal coder is achieved as $N \rightarrow \infty$, where N is the length of the block of data being coded [81]. An example of such a block coder is a vector quantizer [26]. Obviously, a coder with infinite block length is impossible, and even a coder with a reasonably long block length is difficult to design and implement. However, it has been shown that N coders of block length one are nearly as good as (within about 0.45 bits/sample) as one coder of block length N, for the squared error distortion measure [33]. Thus, if the data samples can be transformed so that they are statistically independent, then

nearly optimum coder performance can be achieved with a coder of length one, i.e. a simple quantizer. This fact forms the basis for statistically-based image coding.

Many excellent reviews of statistical image compression techniques exist in the literature. In 1966, Schreiber wrote an interesting review of the early years of image compression [79]. Pratt [67] presented an overall summary of the state of image compression in 1979. Netravali and Limb wrote an informative review of image compression techniques in 1980 [56], as did Jain in 1981 [36]. In addition, Jain [36] contains an extensive bibliography of publications in image compression and related areas. Musmann, *et al.* [54] presented a review of the advances made in image compression techniques since 1981, with special emphasis placed on advances in the coding of color television and video-conference signals. In addition to these review papers, there are many books and special issues of professional journals which deal exclusively with image compression [19, 20, 29, 80].

In general, statistically-based image compression techniques can be categorized into five classes: predictive coding, transform coding, hybrid coding, interpolative and extrapolative coding, and a miscellaneous category [56].

Predictive image compression operates directly on the pixel intensity values in an image. The objective is to generate an error signal by subtracting a predicted pixel value from the actual pixel value. The predicted pixel value is a weighted average (adaptive or nonadaptive) of spatially and/or temporally adjacent pixels. The only information that needs to be transmitted is the error signal.

Transform image compression maps an image into a domain where a large amount of the image information is packed into a small fraction of the transform coefficients. Compression is achieved by encoding only a fraction of the transform coefficients.

Hybrid coding refers to methods which utilize a combination of predictive and transform domain information. Predictive and transform coding techniques each have some attractive characteristics and limitations. The combination of these two techniques has the capability of achieving higher compression than either of the two coders individually and has the advantages of hardware simplicity of predictive coders and high performance of transform coders. For more details, see [36, 51, 56, 75].

Interpolative and extrapolative methods extract a subset of the pixels in an image by subsampling. This subset is then transmitted, and the decoder interpolates or extrapolates to fill in the missing pixels. The subsampling of the image is done in the spatial and/or temporal domains. Simple interpolative coding consists of the following steps: 1) choose certain pixels for transmission, 2) construct an interpolation of the nontransmitted pixels, and 3) evaluate the interpolation error. The interpolation function can be zero, first-order, or higher order polynomials. It has been shown that interpolation using straight lines is quite effective and not much is gained by interpolation using polynomials of higher degree [7]. If higher order polynomials are used in the interpolation, it may be necessary to transmit polynomial coefficients, and the subset of image pixels. In addition, the computation time involved in the interpolation process grows rapidly with the degree of the fitting polynomial. For more details, see papers [15, 21, 45, 55].

Examples of some important statistically-based techniques that do not fit into any of the above categories include bit-plane coding, curve fitting methods, and run-length coding [21, 31]. Some of these methods are simply one-dimensional compression methods applied to two-dimensional image signals.

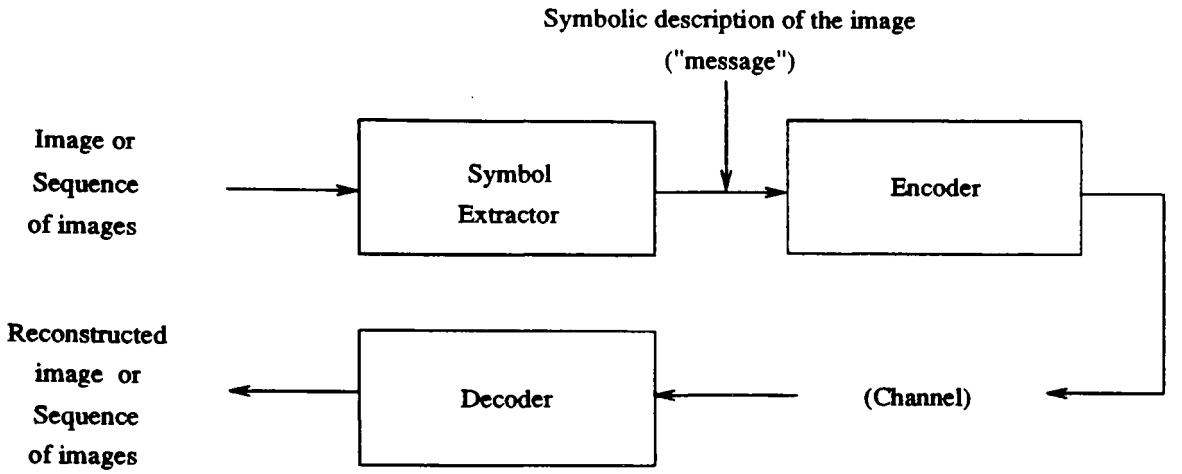


Figure 2.2: A general symbolically-based image compression system.

2.3 Symbolically-Based Image Compression Techniques

The statistically-based image compression techniques, solely guided by information theory, led to a plethora of methods. The compression ratio appeared to have reached a saturation level around one bpp [43] in the early 1980's. For many applications like video telephony, entertainment video, and image transmission for meteorology, lower bit rates were desirable. A new approach to image compression was necessary if high compression ratios were to be attained. A new approach was introduced and has been known as symbolically-based or second generation image compression techniques. A block diagram of a general symbolic image compression system is shown in Figure 2.2

There are two main limitations in the statistically-based image compression techniques. First, since the entropy of the digital image is normally not known, the upper bound based on an estimate of first-order entropy cannot be expected to work well. Second, information theory does not take into account what a human, the

final observer of the image information, sees and how it sees. Symbolically-based image compression techniques have attempted to overcome these limitations combining properties of the HVS and tools of image analysis to get high compression ratios with small loss in visual quality. Global, rather than local pixel-oriented features of the image are emphasized. Examples of such global features include the size, shape, or orientation of objects in the image scene. Extracting the types of features that can be used to provide a symbolic description of the image scene is the ultimate goal of the message extractor in a symbolic image compression scheme. This symbolic description might take the form of a list of scene attributes, for example "there is a chair in the upper left corner of the scene," or "man in a red shirt is running from left to right in the scene while turning his head and looking at the camera." Notice that these are high level descriptions of the scene and do not deal with actual image pixel values, but with the scene content. The encoder then efficiently encodes these scene descriptions or "messages".

Since the symbolically-based image compression techniques are fairly new, there have not been many general reviews of these types of compression methods published yet. There are, however, several review papers of the second generation compression techniques in the literature [43]. In addition to this paper, there is mention of some second generation compression techniques in [54, 56].

To better appreciate the symbolically-based image compression techniques, the relevant properties of the HVS are first described in this section. Following that is a discussion of the major symbolically-based image compression techniques; pyramidal image compression [9], directional decomposition-based compression [43], segmentation-based compression [7, 38], and fractal-based compression [5, 35, 89].

2.3.1 The Human Visual System (HVS)

In many applications like video phone, teleconferencing, TV, and medical imaging, the final observer of the image data is a human. Thus it is very important that an image coder be designed to meet the needs of the human observer. Ideally, no bits should be required to encode the information in an image that is not important to the human viewer and all the bits should be used to encode the information that is important to human perception. For this reason, the more that is known about the requirements of the HVS, the better the coding method can be designed. However, the HVS is very complex and not completely understood, therefore, making image coding a difficult problem.

Despite the complexity of the HVS, a great deal of research has been done in an effort to determine some of its basic properties. This research is generally based on experiments with human subjects, so the results are necessarily subjective. Discussions of some of the basic techniques and significant results in the area of HVS research can be found in [43, 56, 79]. The books by Cornsweet [14] and Marr [52] are useful references on human vision. Here we will briefly summarize some of the most well established properties of the HVS [79] for image coding applications.

A property of the HVS that has been studied extensively is contrast sensitivity. Contrast sensitivity is measured by showing a subject a test pattern, and varying the intensity of neighboring regions in the test pattern until the difference in intensity is just noticeable. There are many ways to measure the contrast sensitivity [14]. For example, consider a patch of intensity $I + \Delta I$ surrounded by a background of intensity I, as shown in Figure 2.3a, The just noticeable difference (JND) ΔI is to determined as a function of I. The fraction $\Delta I/I$, called the Weber fraction, is plotted as a

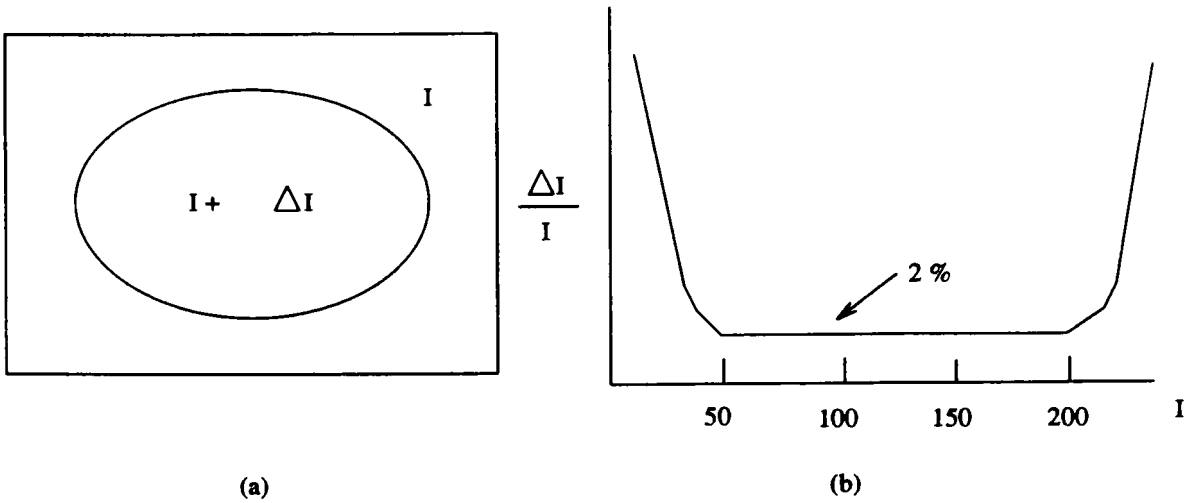


Figure 2.3: A simple contrast sensitivity measurement

function of I in Figure 2.3b.

Figure 2.3b shows that the HVS has greatly reduced contrast sensitivity in very bright or very dark intensity regions of an image. However, this experiment is time-consuming. An alternative is a method introduced by Hamilton [30]. It is referred to as a split-field technique, measures the JND quickly and reliably. Here, the display is divided down the middle into two equal-size fields, see Figure 2.4. The left field is a constant intensity reference field and the right field begins at the top with the reference intensity and increases linearly up to 40 steps above the reference with each level presented as a band 20 pixels in height. To perform the test, an observer simply clicks the mouse at the point where the difference between the left and right fields is no longer discernible. This point is the JND between the reference intensity on the left and the test intensity on the right. To improve the results of the test, many measurements are taken for many subjects and the results averaged.

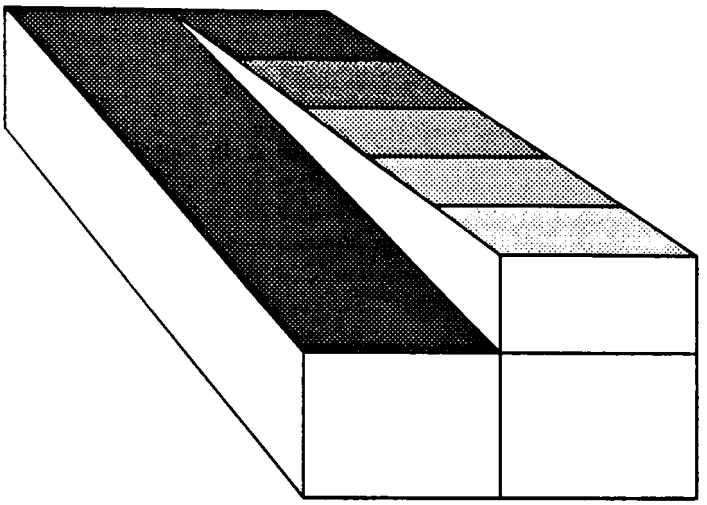


Figure 2.4: Perspective drawing of the split-field experiment

The contrast sensitivity of a human can be used for designing quantizers, as a threshold for the split-merge condition in segmentation-based compression, or for human vision based image distortion measurements. More discussion of the use of this technique and how we incorporate JND measurements into the proposed codec will be given in Section 5.4.

A second important property of the HVS is the modulation transfer function (MTF). The MTF is the response measured by an observer who was shown two sine wave grating transparencies, a reference grating of constant contrast and spatial frequency, and a variable-contrast test grating whose spatial frequency is set at some value different from that of the reference [66]. The contrast of the test grating is varied until the brightness of the bright and dark regions of the two transparencies appear identical. The typical curve of the MTF is shown in Figure 2.5.

The shape of the MTF curve is similar to a band-pass filter and suggests that

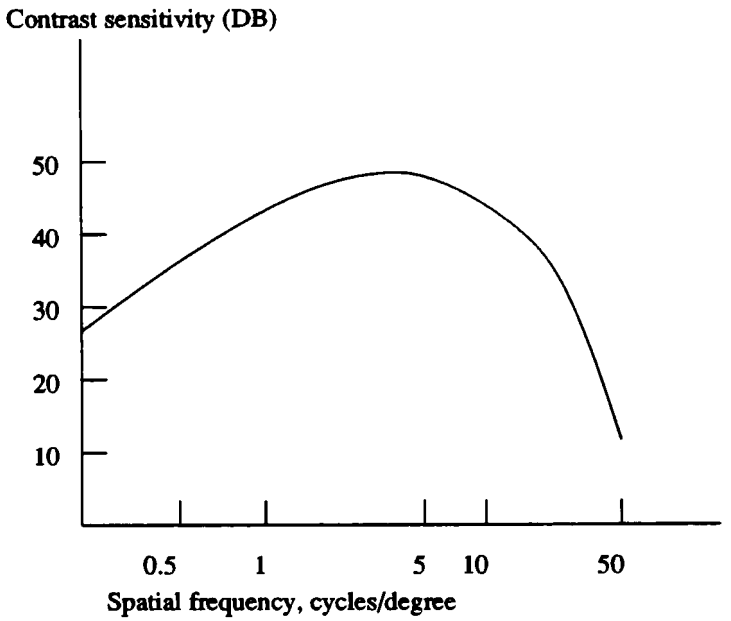


Figure 2.5: A typical MTF curve

the HVS is more sensitive to middle spatial frequencies and less sensitive to low and high spatial frequencies. This implies that the middle spatial frequencies play a more important role in perceived image quality than other frequencies. This property is very important in segmentation-based image compression [38]. For example, if an image is segmented into regions with respect to the information content at the different frequencies, the image coder should require more bits to encode regions which contain middle spatial frequencies to maintain quality, and use very few bits to encode regions which contain low and high frequencies which the HVS is less sensitive to.

A third property of the HVS is saturation effect. The contrast sensitivity of the eye is known to decrease as the intensity of the visual stimulus moves away from the middle range of intensity values [14]. That is, the eye has reduced sensitivity to differences at very high gray levels and differences at very low gray levels. This phenomenon can be used to reduce the dynamic range of the image data.

Each of the above properties help to characterize the aspects of the HVS that are most important in the development of image compression techniques. We now proceed to present the symbolically-based image compression techniques.

2.3.2 Pyramidal Image Compression

Pyramidal image compression [9] features a hierarchical representation for the image. The hierarchical structure is similar to that of the nervous system and it uses functions similar to those in the HVS. The representation is generated using an iterative application of low-pass filtering. A block diagram of this system is shown in Figure 2.6.

Starting with the original image $x(m,n)$, a low-pass version $x_1(m,n)$ is computed using local averaging with a unimodal Gaussian-like two-dimensional impulse response. The low-pass image, with a cutoff frequency of f_1, can be viewed as a prediction of $x(m,n)$. The prediction error $\epsilon_1(m,n)$ is the difference between the original image and the low-pass filtered image.

$$\epsilon_1(m,n) = x(m,n) - x_1(m,n) \tag{2.1}$$

Clearly, if one coded the low-pass image and the prediction error this would be equivalent to directly coding the original image. Compression can be achieved with this representation in two ways: (1) Since the error image is high-pass and the HVS has

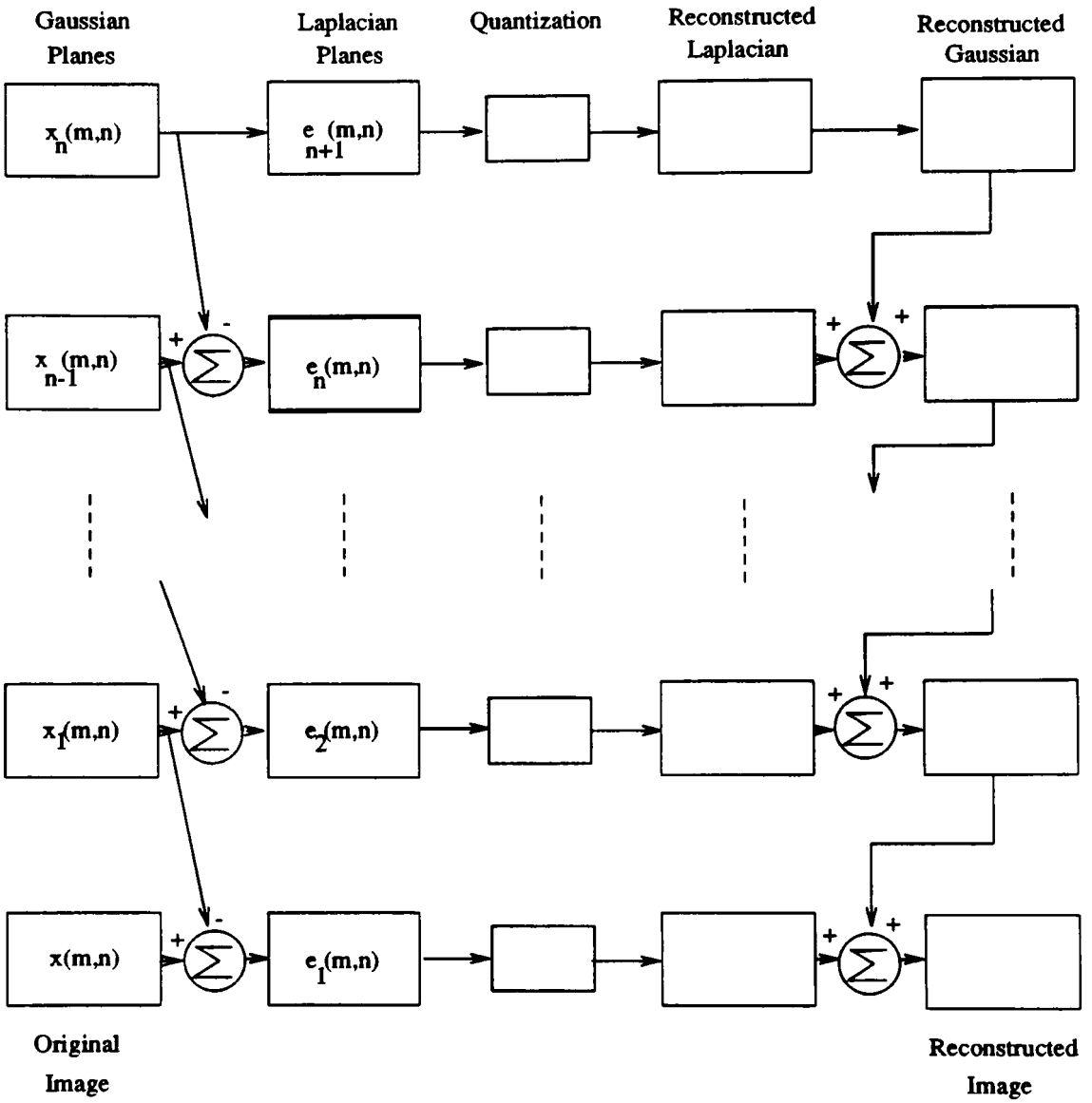


Figure 2.6: Block diagram of the pyramid coding method

less sensitivity at high frequencies, the error image can be coded with fewer bits than the original image. (2) By the two-dimensional sampling theorem, the low-pass filtered image can be represented with fewer samples than the original image.

An advantage of pyramidal coding is that the procedure described above can be applied iteratively. Specifically, the low-pass filtered image $x_1(m,n)$ can be filtered a second time, at a lower cut-off frequency f_2 (typically half the frequency of the first filtering operation). This twice-filtered image $x_2(m,n)$ is now a prediction for $x_1(m,n)$, and the error for this prediction is

$$\epsilon_2(m,n) = x_1(m,n) - x_2(m,n) \tag{2.2}$$

After n iterations, a series of prediction error images $\epsilon_1(m,n),\ldots,\epsilon_n(m,n)$ are obtained. If these images are viewed as stacked one above the other, the result is a pyramidal data structure. At each iteration the dimension of the error image is reduced (through spatial decimation) by a factor equal to the ratio of the cutoff frequencies used in that iteration and the previous iteration (typically a factor of two). The resulting error images are quantized and transmitted to the receiver.

To reconstruct the received image data, interpolation filters are used to reconstruct the error images from their decimated versions. A pixel-by-pixel sum of the reconstructed error images yields the decoded image. A nice feature of this system is that the quality of the decoded picture can be improved as desired at the expense of a lower compression ratio. Good quality images can be obtained around 0.8 bpp.

2.3.3 Directional Decomposition Based Image Compression

The motivation of directional decomposition image compression [43] is largely due to the existence of directionally-sensitive neurons in the HVS. In this method, the original image is decomposed into a series of images using filtering operations employing Gaussian windows. The entire spatial frequency plane is covered with one low-pass filter, plus a set of high-pass, directional filters. The purpose of each directional filter

is to extract edges in the image with a particular spatial orientation. The filtered versions of the original image are coded to form the compressed image.

The messages to be coded are the low-pass image and the directionally-filtered images. The low-frequency component is suitable for transform coding. Each of the directionally-filtered images is spatially decimated and then represented by coding the positions and magnitudes of the edges in the decimated image. The edge positions are coded using a run-length Huffman code, and the magnitudes of the edges are quantized and coded using 3 bit codewords. This coarse quantization is possible due to the reduced contrast sensitivity of the HVS at high frequencies.

To reconstruct the original image, the low-frequency component is obtained by inverse transforming the coded coefficients and then the high-frequency directional edge images are reconstructed by decoding the edge information and interpolating. Once all the filtered images have been reconstructed, they are summed to form the final decoded image. This method can achieve compression ratios around 0.2 to 0.5 bpp.

2.3.4 Segmentation-Based Image Compression

For segmentation-based image compression techniques [7, 38], the image to be compressed is first segmented. In image segmentation, the pixels in an image are divided into mutually exclusive spatial regions based on some criteria. Alternatively, the criteria used could be as simple as the similarity of the pixel gray levels (yielding flat image segments) [12, 72]. The criteria could be more complex, such as how well the pixels fit a given planar model (facet-based segmentation) [80], a two-dimensional polynomial model [7], a statistical model (texture-based segmentation) [69, 82, 88] or a fractal model [38, 63]. Properties of the HVS can also be incorporated into the cri-

teria to obtain a reconstructed image with a small visual loss. For example, contrast sensitivity and the MTF, can be combined with classical segmentation algorithms. In general, segmentation is carried out in three steps: preprocessing, region growing, and elimination of artifacts.

The purpose of the preprocessing is to reduce the local granularity of original image without affecting its contours, so that very small-sized regions are not obtained after region growing. A key problem in preprocessing is the reconciliation of two apparently contradictory goals; namely, granularity removal and edge preservation. Most of the granularity removal filters have low-pass characteristics and therefore smooth the edges as well. An inverse gradient filter [86] may be a solution of the problem. This filter behaves likes a low-pass filter in areas free of contours and like an all-pass filter in highly contrasted areas.

The mechanism of region growing is the following. Regions to be extracted must be characterized with some property in the first step. The property might be, for example, the gray level of a pixel, the variation of the gray level, or the energy within a given frequency band. The selection of this property plays a very important role in the complexity of the method and in the exactness of the contours obtained after segmentation. Then, starting with a given pixels in the picture, its neighboring pixels are examined to see whether they share the same property. If this is the case, the pixel is included in the region, and in turn, its neighboring pixels are examined, and so on. When there are no more pixels left, connected to the region and sharing the same property, the procedure stops and restarts at any other pixel which is not included in the first region. The segmentation is complete when all the pixels of the picture are assigned to some region. The above procedure can also apply to the block-based region growing algorithm if a feature set based on blocks of an image is defined.

After region growing, there are artifacts such as false contours, which do not correspond to real objects in the original image, such as small regions generated by noise. The number of these contours is much higher than that of the objects in the original image. Two ways are available to remove this problem: elimination of the small regions and merging weakly contrasted adjacent regions. If it is assumed that regions containing a number of pixels less than a threshold are not significant, their elimination drastically decreases the number of small regions. To avoid the creation of holes in the image, these regions are included in one of their adjacent regions. To minimize the corresponding distortion, the enclosing region is chosen as the adjacent region whose mean gray level is closest to that of the small region to be included. The second possibility to decrease the number of regions is to merge adjacent regions whose contrast is below a certain level. The contrast between adjacent regions is defined as the mean gray level difference calculated along their common border.

After the segmentation is complete, the image consists of a set of disjoint regions separated by contours. Both the contour (boundary) information and the region information must be encoded. The contours may be approximated with straight lines and circle segments and then the information describing this approximation is encoded [43]. Alternatively, a binary image describing where segment contours are located in the image may be encoded [72]. The interior of a segment is represented by encoding, for example, the coefficients in a polynomial models describing each segment, or for flat segments, the average gray level of the pixels in each segment.

Kunt, *et al.* [43] divided an image into segments using a region growing technique based on information of intensity value. Contour coding is carried out in a three-mode procedure: 1) approximation by line segments, 2) approximation by circle segments, and 3) without approximation. The cost, associated with each mode, in terms of

number of bits for coding, is evaluated and the cheapest mode is chosen. Texture coding is used to encode the missing part of the messages with a two-dimensional polynomial function. An underlying assumption is that within each region there is no longer any sharp discontinuity. The order of the polynomial is determined as a function of the approximation error and of the cost involved in coding polynomial coefficients. The approximation criterion used is the mean squared error (MSE) which is minimized over each region for polynomials of order 0, 1, and 2. The granularity removed with preprocessing is added back in the form of a pseudo-random noise to render the image more natural. The MSE between the original image and the image reconstructed with a polynomial function is computed in each region. The error is used to control the variance of a zero-mean Gaussian pseudo-random signal added as microtexture. This method achieved a good reconstructed image with the compression ratio around 50:1.

Biggar, *et al.* [7] made a performance comparison between segmentation-based coding and transform coding techniques. In the segmentation-based method, a criterion which minimizes the sum squared error (SSE) between the segmentation image and the original image was used. The results of the comparison show that, in terms of the objective SSE measure, the segmentation-based coder performs better than the transform coder at low bit rates (below about 1 bpp) and favorably over the entire useful range of rates. Furthermore, he extended his segmentation-based schemes for video coding by applying segmentation to the frame difference signal [6]. When the frame difference is segmented, a spatially dependent weighting function is combined to encourage region boundaries near those in the last frame. The results suggest that effective low rate video coding is achievable. Rajala, *et al.* [74] discussed aspects of a segmentation-based image coding in a packet-switched network environment.

In this environment, an image coder and the network must be treated as a whole. They suggested a set of requirements that need to be considered when designing a codec. Segmentation-based compression methods typically achieve a compression ratio around 0.2 to 0.7 bpp.

2.3.5 Fractal Based Image Compression

Fractal-based compression is largely motivated by computer-generated fractal images. Mandelbrot [50], followed by Voss [85] showed that computer-generated fractals provided dramatically natural images such as clouds, trees, continents, planets and so on. A distinctive feature of such fractal images is self-similarity on many different scales; when magnified, a small portion of the image resembles some the larger part, it comes from either exactly or very closely. Once written to produce the detail on scale, much the same software can be reused in a loop to repeat the image on successively larger (or smaller) scales. Thus remarkably complex fractal images blossom from a small, simple piece of programs.

The self-similarity property of computer-generated fractal images intrigued Barnsley. In the early 1980's Barnsley set out trying to use it to compress the data needed to re-create an image. At a time when most work in fractals focused on producing complex and realistic images from fairly compact computer programs, Barnsley was attempting the opposite. Starting with a complex image, he attempted to find a set of fractals that would produce an image, or at least a close copy of it.

In early 1984 Barnsley, *et al.* [2] developed an iterated function system (IFS) to reduce an image to a set of fractals. Their system is described as follows. An image is divided into segments which can be generated by fractals. The IFS matches each of the corresponding segments with IFS code that represents a fractal image close in

appearance to the segments. The IFS hunts for a similar-looking fractal image by using a scale called the Hausdorff metric, which measures how similar to two images are in terms of their spectral and spatial characteristics. The codes that produce the fractal images are called iterated function system (IFS) codes. They can be used to re-create the original image, and are stored in place of the pixel information that made up the original image. Therefore, very high compression can be obtained.

For example, an image of rain falling on a seashore might be broken down into rain, rocks in the water near the shore, foam in the water near the rocks, the water itself, birds in the sky, clouds, the sky itself, a strip of beach, and some grass near the beach. The images are first divided into segments. The IFS system then matches each of the segments with code that represents a fractal image close in appearance to the segment using the Hausdorff metric. Jacquin [35] proposed a fractal-based block compression technique. The main characteristic of his technique is that image blocks rather than an image are reduced to a set of fractals. Furthermore, his system can be faster because it can be implemented in parallel.

Although these methods can achieve the high compression ratios, there are some limitations. One limitation is that this method may work well only for images which have characteristics of self-similarity. Another limitation is that it is computation-intensive in both the encoding and decoding phases because this method uses many iterations to generate the fractal images. For example, the IFS system carried out on Masscomp 5600 workstations with Aurora graphics took about 100 hours to compress a 780×1024 pixel, 256 gray level digital image and 30 minutes to decode on the Masscomp.

2.4 Conclusions

In this chapter we provided an brief review of the previous approaches to the image compression problem which are called as the statistically-based image compression techniques. The statistically-based image compression techniques have a main limitation that the data rates may have reached a saturation level around 1 bpp. We then examined a new approach to image compression which is called as the symbolically-based image compression techniques combining properties of the HVS and tools of image analysis. The symbolically-based image compression techniques can achieve lower data rates (below about 0.2 to 0.7 bpp) than the statistically-based image compression techniques.

3

Texture Analysis

3.1 Introduction

Among the characteristics of images, texture has been recognized as one of the most important. It is important because pixels can be grouped into relatively large, homogeneous regions and provide the essential structure information in an image. For example, the grass, the sky, and a tree will define relatively large homogeneous regions each with its own textural structure. When a relatively large region has a single texture, the large amount of redundancy can be removed. A good representation of texture information in an image is necessary for developing a system with high compression.

Texture may be classified as being artificial or natural. Artificial textures consist of arrangements of symbols placed against a neutral background. These symbols may be line segments, dots, stars, or alphanumeric characteristics. Natural textures, as the name implies, are images of natural scenes containing semi-repetitive arrangements of pixels. Examples include photographs of brick walls, terrazo tile, sand, grass, tree, etc. Brodatz [8] has published an album of naturally occurring textures. A general overview of texture analysis can be found in Lipkin and Rosenfeld [46].

Of particular interest to this research are measures for discriminating different

textures. Haralick, *et al.* [31] proposed co-occurrence statistics as a texture distance measure. It is based on the estimation of the second-order conditional probability density functions corresponding to a pair of pixels separated by a distance in relative orientation. Texture features can be extracted using these density functions. However, this method suffers from several problems; 1) co-occurrence statistics obtained from different spatial dependence of a pair of pixels may provide the different structure information of an image and 2) a large number of computations and sometimes excessive memory requirements are needed. Zucker, *et al.* [78] developed an algorithm to find spatial relations that best capture the structure of textures when the co-occurrence matrix representation is used. Conners, *et al.* [13] proposed a compressed structural description of Haralick's method. Unser [84] describes an alternative to the co-occurrence method which is nearly as efficient, while requiring substantially less memory. Laws [44] used texture energy as a texture distance measure. The texture energy measure is computed using filters of dimension 3×3 or 5×5 pixels which match local features like edge, spot, line, ripple, etc. Although good results have been obtained, this approach remains heuristic, the filter set is incomplete, and its elements are not mutually orthogonal. Other approaches to texture analysis include autocorrelation functions [16, 68], gradient vector histograms [70] and resolution-dependence [88].

Most proposed texture analysis techniques have been used for the classification and segmentation of textures regions; few have been used for texture coding. Kocher and Kunt [41] proposed a segmentation-based compression system by contour-texture modeling. Image compression is achieved by approximating the contour information and the texture information in each region. The contour information is given by the location of the boundaries of each region and the texture information by means of

2-D polynomial functions. It was assumed that the texture within the region does not contain any sharp discontinuity, thus a 2-D polynomial adequately models the texture content. Unfortunately, in images containing complex textures, e.g., trees and bushes having many sharp discontinuities, their method does not work well. However, this method can achieve a compression ratio around 0.2 to 0.6 bpp for images with low texture content, like a head and shoulder image.

Most of the models discussed above are two-dimensional, not three-dimensional. Use of two-dimensional models leads to difficulty if one wants to describe the three-dimensional information and then relate that to the human perception of texture. The fractal model developed by Mandelbrot [49] offers the potential of unifying and simplifying these various two-dimensional texture descriptions, as well as the possibility of interpreting them in terms of the three-dimensional structure of the image. The principal advantage of describing textures in terms of fractals, rather than in any of these other methods, is that it allows us to capture a simple physical relationship that underlines the texture structure. A relationship that allows us to interpret the two-dimensional texture measurements in terms of the three-dimensional world. The fact that this physical interpretation can be lost with most two-dimensional characterizations of texture makes it advantageous to characterize texture problems in terms of the three-dimensional fractal surface model. Therefore, a promising approach to texture modeling for image coding is to use fractals. Some important properties of fractals in terms of image coding are discussed in later sections.

3.2 Fractal Geometry in Image Analysis

If we regard the pixel intensity in an image as the height above a plane, then the intensity surface of a texture image can be viewed as a rugged surface. The fractal model provides an excellent explanation of the ruggedness of natural surfaces. An application of the fractal model has been used in the graphical simulation of natural phenomena like mountains, clouds, trees, human faces, and animals [1, 3, 18, 23]. The fractal model has been applied to texture image analysis [60, 63, 64], as well as image coding [4, 35].

One important characteristic of a fractal is the fractal dimension D, which is related to the metric properties, length and surface of a curve. D provides a good measure of perceptual roughness of the curve or surface, with increasing values in D representing perceptually rougher curves and surfaces [63].

There are a number of different fractal models [50] available to describe non-random and random fractal objects. An typical example of non-random fractal objects is the Koch curve [50] which a mathematically iterative program models by superimposing smaller and smaller triangles. Other examples are a Cantor set, a Sierpinski triangle and so on. These non-random fractal objects have exact scale invariance, i.e., the shapes are invariant under magnification. However, most objects like coast-lines, trees, mountains and etc. are only statistically scale invariant, since they are only invariant in an average sense. For example, magnification of coast-lines are qualitatively identical, not quantitatively. These statistically scale invariant objects are called as the random fractal objects. Many random fractal objects have been by the iteration of complex functions (M set and Julia set curves) and the fractal Brownian function (FBF) [50, 59]. The most useful fractal model has been the Brow-

nian function (FBF) $I(x, y)$ [47, 49] since it produce surfaces that closely resemble natural surfaces. The FBF model belongs to the class of statistically self-affine fractals [63]. The FBF model regards naturally occurring rough surfaces as the end result of random walks. Such random walks are basic physical processes in our universe. An intensity surface of a texture image can also be viewed as the end result of a random walk, so the FBF can be used for the analysis of image texture.

3.2.1 Fractal Dimension

The definition of the fractal dimension is a set for which the Hausdorff-Besicovich dimension is strictly greater than the topological dimension. We consider object X in an E-dimensional space. $N(\epsilon)$ is the number of E-dimensional spheres of diameter ϵ needed to cover X, where E is an integer and the E-dimensional space is the minimum integer dimensional space among all possible integer dimensional spaces which can envelop X. Thus, if $N(\epsilon)$ is given by

$$N(\epsilon) = K(\frac{1}{\epsilon})^D, \qquad as\ \epsilon \to 0, \tag{3.1}$$

where K is a constant. X has Hausdorff dimension D. If D is fractional, D is also called the fractal dimension. For fractal objects, D is independent of ϵ.

If the fractal dimension is to be used to characterize the texture in an image, we need a method for estimating the fractal dimension from the given dataset. Many different estimators have been proposed: box counting [1], yardstick [17], maximum likelihood [48], and blanket [61], variance [63], power spectrum [63], probability density function [85]. In our case, a blanket method is adopted since it is computationally efficient.

The blanket method is described as follows. A one-dimensional object such as a coastline is given. All points with distances to the coastline of no more than ϵ are considered. These points form a strip of width 2ϵ, and the suggested length $L(\epsilon)$ of the coast is the area $A(\epsilon)$ of the strip divided by 2ϵ. As ϵ decreases, $L(\epsilon)$ increases. Using Eq. (3.1), the length of coastline is given for each ϵ by

$$\text{Length} = \frac{A(\epsilon)}{2\epsilon} = \epsilon \cdot N(\epsilon) = K\,\epsilon^{1-D} \tag{3.2}$$

where K is a constant. A one-dimensional illustration of a curve is shown in Figure 3.1. In extending Eq. (3.2) to surfaces, all points in the three-dimensional space at distance ϵ from the surface are considered, covering the surface with a blanket of thickness 2ϵ. The surface area is then the volume occupied by the blanket divided by 2ϵ. The covering blanket is defined by its upper surface U_ϵ and its lower surface B_ϵ. Initially, given the gray level function $g(i,j)$, $U_0(i,j) = B_0(i,j) = g(i,j)$. For $\epsilon = 1, 2, 3, \ldots$, two blanket surfaces are defined as follows:

$$U_\epsilon(i,j) = max\,[\,U_{\epsilon-1}(i,j) + 1,\ \ max\,[\,U_{\epsilon-1}(m,n)\ \ for\ S]\,] \tag{3.3}$$

$$B_\epsilon(i,j) = min\,[\,B_{\epsilon-1}(i,j) - 1,\ \ min\,[\,B_{\epsilon-1}(m,n)\ \ for\ S]\,] \tag{3.4}$$

where $S = [\,(m,n) : |(m,n) - (i,j)| \leq 1]$.

The image points (m,n) with distance equal to or less than one from (i,j) are the four closest neighbors of (i,j). Similar expressions exist when the eight-neighborhood is desired. The blanket definition uses the fact that the blanket of the surface for radius ϵ includes all the points of the blanket for radius $\epsilon - 1$, together with all the points within radius 1 from the surfaces of that blanket. Eq. (3.3), for example, ensures that the new upper surface U_ϵ is higher by at least 1 from $U_{\epsilon-1}$, and also at a distance at least 1 from $U_{\epsilon-1}$ in the horizontal and vertical directions.

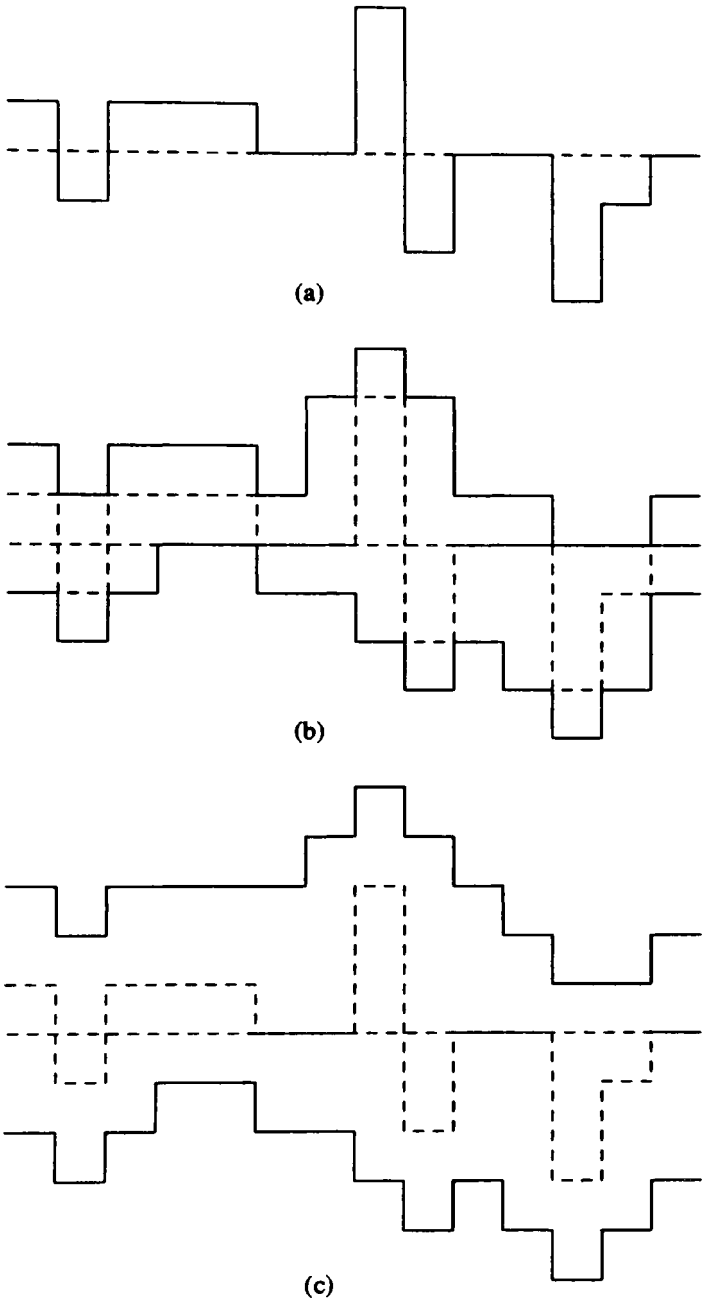


Figure 3.1: A one-dimensional function is shown in (a) and its covering blanket for ϵ = 1, 2 are shown in (b) and (c), respectively. The blanket areas are A(1) = 47 and A(2) = 78. The respective measured lengths are L(1) = 47/2 = 23.5 and L(2) = 78/4 = 19.5.

The volume of the blanket at scale ϵ is computed from U_ϵ and B_ϵ by

$$V(\epsilon) = \sum_{i,j}(U_\epsilon(i,j) - B_\epsilon(i,j)) \tag{3.5}$$

The volume can now be used to obtain an estimate of the surface area which leads us directly to an estimate of the fractal dimension. The area $A(\epsilon)$ is given by

$$A(\epsilon) = \frac{V_\epsilon}{2\epsilon} = K\epsilon^{2-D} \tag{3.6}$$

where K is a constant.

From a theoretical viewpoint, if a surface is a perfect fractal surface, then the fractal dimension will remain constant over all ranges of scales ϵ. In practice, there are scale range limitations of fractal dimensions due to limitations in textural images. For example, the resolution limit of the image system sets a lower limit on the fractal scaling behavior. An upper limit may be set by the structure being examined. Thus, a real surface will be fractal over some range of scales rather than over all scales. These limiting scales can be expressed as upper (ϵ_{max}) and lower cutoff (ϵ_{min}) scales.

To compute the fractal dimension, we apply the log function to both sides of Eq.(3.6). Then,

$$\log A(\epsilon) = (2 - D)\log(\epsilon) \; + \; K \tag{3.7}$$

Using Eq. (3.7), we can define an algorithm for estimating the fractal dimension of an image surface. First, calculate the volume $V(\epsilon)$ using Eq. (3.5). Second, calculate the surface area $A(\epsilon)$ for various scales ϵ using Eq. (3.6). Third, use Eq. (3.7) to plot $\log A(\epsilon)$ versus $\log(\epsilon)$. Fourth, choose ϵ_{max} and ϵ_{min}. ϵ_{max} and ϵ_{min} are found by the experiment. Fifth, use a least squares linear regression to fit a straight line to the plot of $\log A(\epsilon)$ vs. $\log(\epsilon)$. Sixth, the fractal dimension D is equal to 2 minus the slope of

the straight line. Specifically, the algorithm for estimating the fractal dimension of an image surface is as follows.

Algorithm I: Estimating the fractal dimension for an image surface.

Step 1) Calculate the volume $V(\epsilon)$,

$$V(\epsilon) = \sum_{ij}(U_\epsilon(i,j) - B_\epsilon(i,j))$$

Step 2) Calculate the area $A(\epsilon)$,

$$A(\epsilon) = \frac{V(\epsilon)}{2\epsilon}$$

Step 3) Take the log of both sides of $A(\epsilon) = K\epsilon^{2-D}$ yielding

$$\log A(\epsilon) = (2 - D)\log(\epsilon) + K$$

Plot $\log A(\epsilon)$ vs. $\log(\epsilon)$.

Step 4) Choose ϵ_{max} and ϵ_{min}.

Step 5) Apply least-squares linear regression to fit a straight line to plot of $\log A(\epsilon)$ vs. $\log(\epsilon)$.

Step 6) D = 2 - slope.

For example, a natural image and its plot of measured surface area, $A(\epsilon)$ versus ϵ in log-log scale are shown in Figure 3.2 and Figure 3.3 respectively for $\epsilon = 1, \cdots, 7$. The fit is good for $\epsilon = 1, \cdots, 5$, which implies that the natural image is a fractal surface for $\epsilon = 1, \cdots, 5$. Therefore, ϵ_{max} is 5 and ϵ_{min} is 1. The value of the estimated slope is -0.62729 using a least-squares linear regression. Therefore, the estimated fractal dimension is 2.62729.

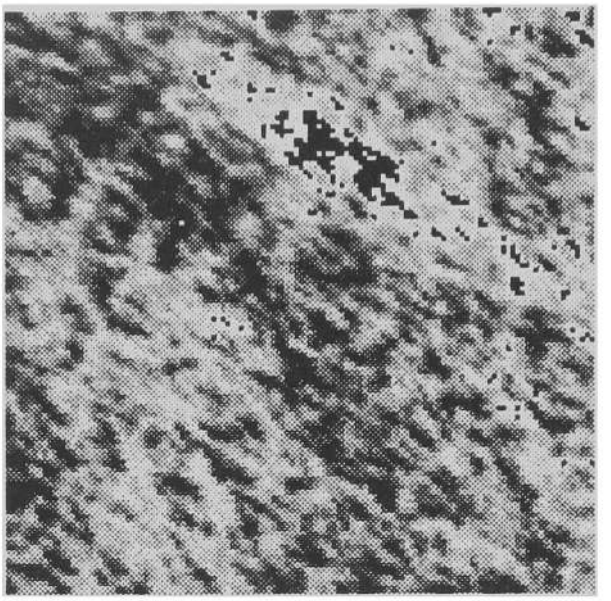


Figure 3.2: A natural image

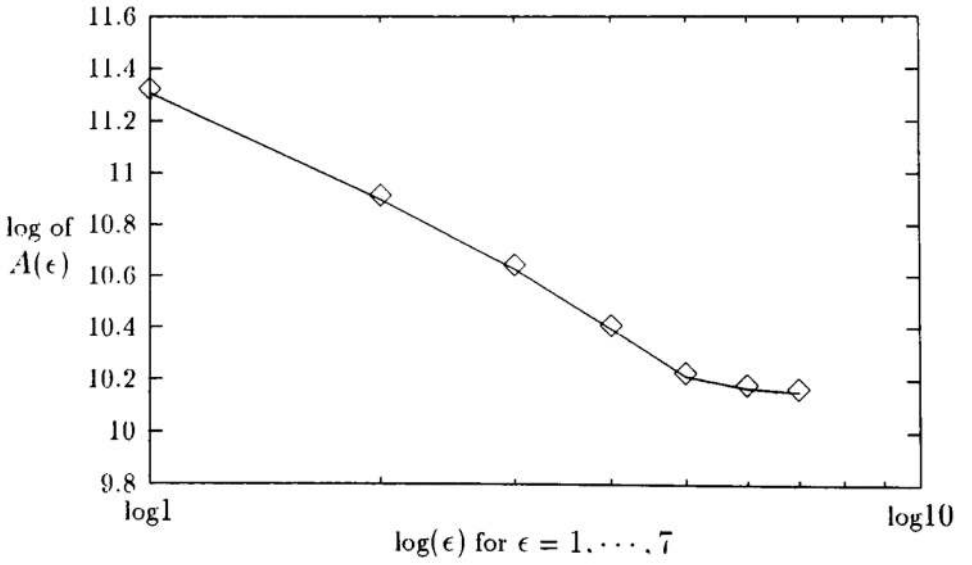


Figure 3.3: The calculation of the fractal dimension of a natural image. The values of the estimated slope and estimated fractal dimension are -0.62729 and 2.62729, respectively.

3.2.2 Fractional Brownian Function

One of the most useful models for random fractals found in nature such as mountain terrain, clouds, and trees is the fractional Brownian function (FBF), $I(x;H)$ introduced by Mandelbrot and Van Ness [51]. It has been shown that the FBF produces surfaces that closely resemble natural surfaces. By modeling natural surfaces using the FBF, we can extract important information about the texture surfaces. For example, one can relate the fractal dimension and the power spectrum density of the FBF. As will be discussed later, we take advantage of this relation in our proposed coding scheme.

The fraction Brownian Function $I(x)$ is an expansion of the Brownian function. Variations $I(\Delta x) = I(x2) - I(x1)$ are zero-mean Gaussian distributed with variance proportional to the displacement difference magnitude $\Delta x = |x2 - x1|$ raised to the power $2H$ [59].

$$< I(\Delta x)^2 >= K\Delta x^{2H} \tag{3.8}$$

where the brackets $<$ and $>$ denote statistical expectation. K is a constant, $0 < H < 1$, and H is called the Hurst coefficient. The case $H = 1/2$ corresponds to standard Brownian function in which $< I(\Delta x)^2 >= K\Delta x$. The mean modulus of a Gaussian variable is proportional to its standard deviation so the relationship can be written

$$< |I(\Delta x)| >= K\Delta x^{H} \tag{3.9}$$

where K is a constant.

Voss [59] proved the fractal dimension of the FBF in one-dimensional case

$$D = 2 - H = ED + 1 - H \tag{3.10}$$

where D and ED are the fractal dimension and the Euclidean space dimension respectively. In two-dimensional case,

$$D = 3 - H = ED + 1 - H \tag{3.11}$$

A more detailed derivation of the fractal dimension of the FBF in a two dimensional case is given in a paper [10].

Let us now derive a relationship between the fractal dimension and the slope of the power spectrum density (PSD) of the FBF. If one defines $I(k,T)$ as the Fourier transform a specific sample of $I(x)$ for $0 < x < T$,

$$I(k,T) = \frac{1}{T}\int_0^T I(x)\epsilon^{-j2\pi kx}dx, \tag{3.12}$$

then the PSD of $I(k,T)$ is given by

$$S_I(k) = KT|I(k,T)|^2 \text{ as } T \to \infty \tag{3.13}$$

where K is a constant.

The autocorrelation function of $I(x)$ is given by

$$G_{I(\Delta x)} = < (I(x) - < I(x) >)(I(x+\Delta x) - < I(x+\Delta x) >) > \tag{3.14}$$

$$= < I(x)I(x+\Delta x) > - < I >^2 \tag{3.15}$$

The autocorrelation function $G_{I\Delta x)}$ is directly related to the mean square increments of the FBF using Eq. (3.8)

$$< I(\Delta x)^2 > = < |I(x2 - x1) - I(x1)|^2 > \tag{3.16}$$

$$= -2 < I(x)I(x+\Delta x) > +2 < I(x)^2 > \tag{3.17}$$

$$= -2G_{I(\Delta x)} + 2 < I^2 > \tag{3.18}$$

$$= -2G_{I(\Delta x)} + K_1 \tag{3.19}$$

$$= K_2 \Delta x^{2H} \tag{3.20}$$

where K_1 and K_2 are constants. Therefore, the autocorrelation function is given by

$$G_{I(\Delta x)} = K_1 + K_2 \Delta x^{2H} \tag{3.21}$$

Recall, the autocorrelation function and the PSD are related by the Wiener Khintchine relation [37].

$$G_{I(x)} = \int_{-\infty}^{\infty} S_I(k) \epsilon^{2\pi k x} dk \tag{3.22}$$

For certain simple power laws for $S_I(k)$, $G_{I(\Delta x)}$ can be calculated exactly. Thus, the PSD of the FBF is given by

$$S_I(k) = K_1 \delta(k) + K_2 \frac{1}{k^{2H+1}} \tag{3.23}$$

$$= K_1 \delta(k) + K_2 \frac{1}{k^{\beta_1}} \tag{3.24}$$

and, $\beta_1 = 2H + 1$ for $0 < H < 1$.

The procedure is easily extended to the two-dimensional function, $I(\vec{x})$. $I(\vec{x})$ must satisfy the property given by

$$< I(\Delta \vec{x})^2 >= K |\vec{x_2} - \vec{x_1}|^{2H} \tag{3.25}$$

Eq. (3.20) can be extended to the $\vec{x}$ plane

$$< I(\Delta \vec{x})^2 > = -2G_{I(\Delta \vec{x})} + K_1 \tag{3.26}$$

$$= K_2 \Delta \vec{x}^{2H} \tag{3.27}$$

as can Eq. (3.22) using Eq 3.27

$$G_{I(\vec{x})} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} S(\vec{k}) \epsilon^{(j\pi \vec{k} \bullet \vec{\delta})} 2\pi \vec{k} d\vec{k} \tag{3.28}$$

$$= \int_{-\infty}^{\infty} S(\vec{k}) \epsilon^{(j\pi \vec{k} \bullet \vec{\delta})} 2\pi \vec{k} d\vec{k} \tag{3.29}$$

$$= K_1 + K_2 \Delta \vec{x}^{2H} \tag{3.30}$$

Here $\vec{x}$ is a point in the two-dimensional (xy) plane and $\Delta\vec{x}$ is a displacement in the two-dimensional plane.

For this $\Delta\vec{x}$ dependence to correspond to the Δx dependence of $< I(\Delta x)^2 >$, $S(\vec{k})$ is given taking the Fourier transform of Eq. 3.30 and using $\beta_1 = 2H + 1$

$$S(\vec{k}) = K_1\delta(k) + K_2\frac{1}{k^{2H+2}} \tag{3.31}$$

$$= K_1\delta(k) + K_2\frac{1}{k^{\beta_1+1}} \tag{3.32}$$

$$= K_1\delta(k) + K_2\frac{1}{k^{\beta_2}} \tag{3.33}$$

$$\tag{3.34}$$

Using the above equations, the relation between the slope β_2 of the two-dimensional PSD and H is given by

$$\beta_2 = 2H + 2 \tag{3.35}$$

Using $D = 3 - H$ in two-dimensional space, the relation the fractal dimension D and slope β_2 is given by

$$D = 3 - H \tag{3.36}$$

$$= 3 - \frac{\beta_2 - 2}{2} \tag{3.37}$$

$$= 4 - \frac{\beta_2}{2} \tag{3.38}$$

When D is close to either 2 or 3, β_2 is close to 4 or 2 respectively. β_2 is the negative slope of the PSD. Therefore, the higher the value of D, the higher the spatial frequency content, the rougher the waveform.

3.3 Conclusions

In this chapter we provided a brief overall review of texture models in applications including the classification, the segmentation, and the image coding. We then discussed the fractal Browian function in detail. The relation of the fractal dimension and slope of the power spectrum density of the fractional Brownian function was obtained. It was shown that higher values of D provide the smaller negative slope of the power spectrum density and the rougher image surface. We use this important relation for segmenting an image into texturally homogeneous regions with respect to the degree to roughness.

4

A New Approach for Segmentation-Based Image Coding

4.1 Introduction

In Chapters 2 and 3 we presented previous results in the areas of image coding and texture analysis with particular attention to structurally-based coding and texture analysis using fractals. In the remainder of this report we present a method for combining fractal-based texture analysis and properties of the human visual system to segment an image for application in image coding.

In this chapter we define the structure of the new segmentation-based image coder. A description of both the transmitter and receiver will be given. Fundamental to this coder is the technique for segmenting an image using properties of the HVS and the fractal dimension. In Chapter 5, we present in detail the proposed segmentation technique and characterize its performance through computer simulation. In Chapter 6, we present the details of the mixed encoder and analyze the performance of the coder by determining the sensitivity of the system performance to changes in system parameters.

4.2 The Codec

A general description of the system for segmentation-based image coding is shown in Figure 4.1. It consists of three components; the transmitter, the channel, and

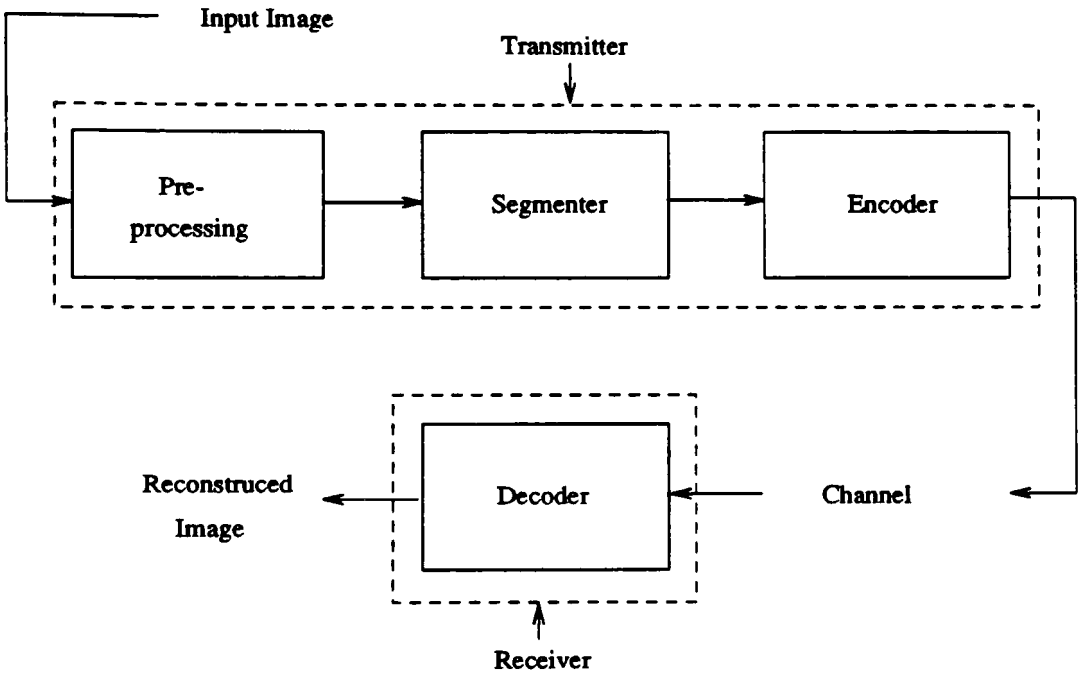


Figure 4.1: The overall block diagram of the segmentation-based coder.

the receiver. The transmitter is responsible for generating the coded information which is then sent over some communication channel. The output of the encoder is a string of bits that represent the input image. In this work, it is assumed that the communication channel is noiseless or any errors have been corrected. In this ideal case, the output of the transmitter is equal to the input of the receiver. At

the receiver, the image decoder reconstructs the image from the string of For human viewing, the reconstructed image is typically displayed on a CRT monitor.

4.3 The Transmitter

Figure 4.2 shows a more detailed block diagram of the transmitter. It includes three main stages: the preprocessor, the segmenter, and the coder. The main purpose of the preprocessor is to alter the image so that the segmenter produces a high quality segmented image. The segmenter divides the preprocessed image into disjoint regions of common textural content. The segmenter is followed by a coder which generates the codes for the boundaries and the three texture classes.

4.3.1 Preprocessing

In any segmentation-based compression algorithm, the information describing the content of each segment must be encoded. Thus, the number of image segments and the number of bits representing the textures of the segments are directly proportional to the bit rate of the coded image. Because of this, a minimum number of segments and an efficient representation of the textures are critical. The preprocessing is designed to alter the image in such a way that fewer segments and textures are produced by the segmenter, but without degrading the visual quality of the segmented image.

Preprocessing will be required when the image data has been contaminated by noise or the dynamic range has been compressed. In the former case, the preprocessor is a linear or nonlinear filter designed to remove the noise from the signal. In the latter case, the preprocessor is a clamping operation to enforce the dynamic range reduction. It should be noted, however, that it is possible that no preprocessing will

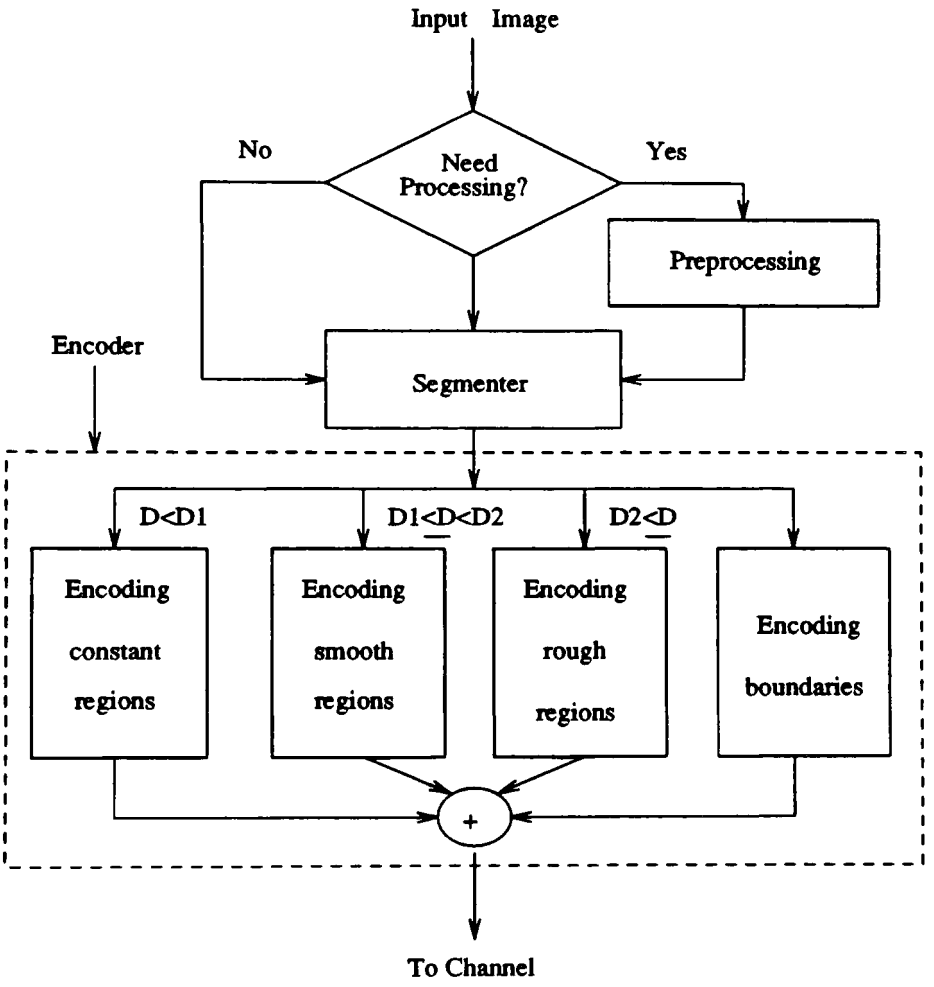


Figure 4.2: The block diagram of the transmitter characteristics.

be required.

Many different noise cleaning filters exist in the literature [37, 87]. The particular choice of filter depends on the type of noise one needs to remove. If the noise is additive or white Gaussian noise, then it is useful to use a linear filter. If the noise is salt-and-pepper or shot noise, then it is useful to use an out-of-range filter or a

median filter.

The clamping process proposed in [71] reduces the dynamic range of the image by setting all pixels with gray level above a threshold to that threshold and setting all pixels below a second threshold to the second threshold. This can be expressed:

$$p = \begin{cases} th_1 & p < th_1 \\ p & th_1 \leq p \leq th_2 \\ th_2 & p > th_2 \end{cases} \tag{4.1}$$

where p is the gray level of a pixel in the image, and th_1 and th_2 are the two clamping thresholds. Clamping is motivated by the contrast sensitivity of the eye, which is known to decrease as the intensity of the visual stimulus moves away from the middle range of intensity values [14]. The reasoning is that, since the eye has reduced sensitivity to differences in very high gray levels and differences in very low gray levels, variety in gray levels at these extremes of the gray level range is unnecessary. However, the clamping operation greatly depends on the original dynamic range of the image data and the experimental environment [65]. The clamping operation can noticeably degrade the subjective quality of the image in a bad environment. Therefore, one must be careful in using the clamping operation.

4.3.2 Image Segmentation

After preprocessing, the next step in the compression algorithm is the segmentation of the image. In this work, centroid-linkage region growing [32] is used. An important attribute of region growing segmentation is the production of disjoint segments with closed boundaries. This is important because segmentation-based compression requires a description of the boundary and texture of each image segment. Such a

description would be impossible if the segments overlapped or did not have closed boundaries.

An image is segmented into texturally homogeneous regions with respect to the degree of roughness as perceived by the HVS. The segmentation is accomplished by thresholding the fractal dimension so that membership is in one of three textural classes. The three classes are perceived constant intensity (class I), smooth texture (class II), and rough texture (class III). Regions belonging to the perceived constant intensity have a fractal dimension less than D_1. The second class contains regions with the fractal dimension between D_1 and D_2. The third class contains regions with the fractal dimension greater than D_2.

The output of the segmenter is a gray level image consisting of many segments. The images are partitioned so that each segment contains the same degree of roughness as perceived by the HVS. The objective now is to apply an efficient coding technique to the boundaries and each texture class to achieve high compression with small visual degradation.

4.3.3 The Mixed Coder

The last stage in the transmitter is the encoding of the segments and their boundaries. During the segmentation, the segments are classified as one of the three texture classes. The objective of the coding is to obtain an efficient representation of the segmented image data for transmission or storage. The image coder should use more bits to encode the information for which the HVS is more sensitive and use fewer bits to encode the information which the HVS is less sensitive. To accomplish this we propose a mixed coder. It consists of four separate stages; the boundary encoding and three textural class encodings, see Figure 4.2.

4.4 The Receiver

As can be shown in Figure 4.3, the two types of coded information come into the mixed decoder.

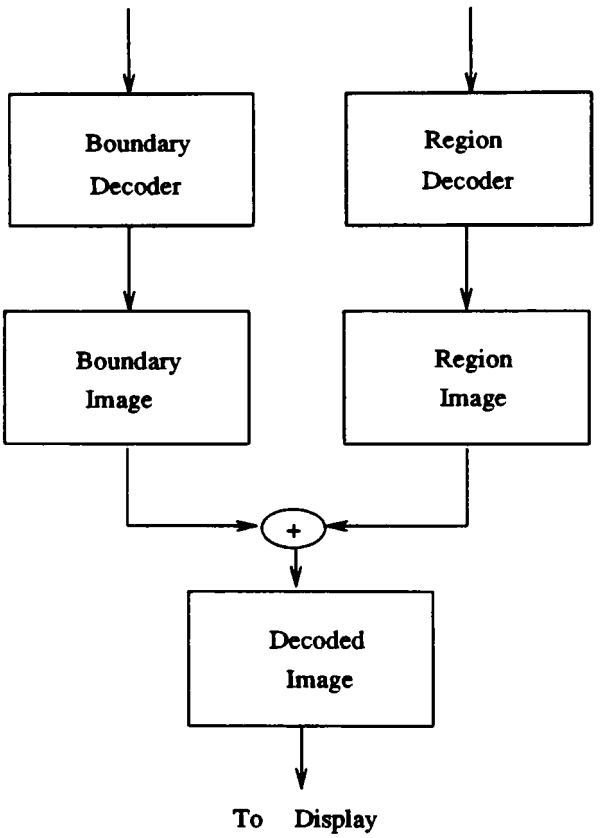


Figure 4.3: The block diagram of the receiver characteristics.

The boundary decoder generates the boundaries of the decoded image from the coded data received the communication channel. A general description of boundary coders

will be given in the boundary coding.

The missing part of the decoded image after the boundaries are decoded is texture information within regions. Since regions belonging to class I are perceived constant regions, their mean values are painted within the appropriate regions. Regions belonging to class II and III are reconstructed by reproducing polynomial functions. The information of the boundaries and the textures are combined to form the reconstructed image.

4.5 The Basic Principle of the Mixed Coder

The goal of the mixed coder is to encode the boundaries and their regions in a segmented image. Accurate representation of the boundary is necessary to describe the location of the region boundary because of the HVS sensitivity of the edges. As a result, we chose an errorless coding scheme to represent the boundaries. A binary image representing the boundaries is created. Then, the binary data is encoded using an adaptive arithmetic codec [62, 72, 76].

After boundary encoding, we then need to encode the three texture regions. For regions belonging to the perceived constant intensity class, only the mean intensity values need be transmitted. It should be noted that in this case lossy compression has already taken place since we are approximating each regions texture with a constant value. We do not wish to introduce any further compression, so that the lossless arithmetic code is again employed to achieve further compression. Since an intensity mean requires 8 bits, the mean intensity value is converted into a vector of an $8 \times N$ binary array, where N is the number of segments belonging to perceived constant regions. The mean vector is then encoded using the arithmetic code.

The texture information in regions belonging to class II and III are not directly encoded. To achieve compression the texture information is first modeled. Polynomial models are used for each class. Compression is achieved by adjusting the amount of error tolerated between the original image data and the modeled image data. The lower the amount of error the better the approximation, but the higher the bit rate. Because of the sensitivity of the HVS to middle range spatial frequencies, a lower error is chosen for class II than class III.

There are two critical components in this image coding system. The first is the segmentation process and the second is the mixed coder. The segmentation algorithm is a region growing-based technique which incorporates estimates of the fractal dimension and the expected value of the block to describe the texture content and the JND to better adapt its decisions to the HVS. The mixed coder defines appropriate compression schemes for the textural classes and the segmentation boundary information. The constant intensity regions (class I) are modeled by the mean intensity value of each region while the smooth and rough texture classes (II and III) are modeled using polynomial models. After modeling, the data from class I and the boundary information is encoded using an errorless arithmetic code. The data from class II and III are encoded using the arithmetic code.

In Chapter 5 we describe in detail the texture segmentation technique. In addition, the results of an evaluation of the performance of the segmenter to changes in various parameters is presented. Parameters considered are the blocksize used to estimate the fractal dimension and threshold of the fractal dimension.

In Chapter 6 we present the mixed codec and evaluated the performance for the entire image coding system to parameter variations. The parameters are the fractal dimension thresholds D_1 and D_2 and the adjusting amount of error tolerated between

the original image data and the modeled image data for the smooth and the rough texture classes.

5

Image Segmentation Using Properties of the HVS and Fractals

In this chapter we describe a new technique for segmenting discrete gray level images. In image segmentation, the pixels in an image are divided into mutually exclusive spatial regions based on some criteria, each having certain properties. Segmentation is an important step in scene analysis, image understanding systems, and image coding. In a scene analysis application, for example, an image is first segmented into regions. Regions are used to identify objects in the image scene. Such identification requires accurate segmentation so that a one-to-one correspondence between the image segments and the objects can be made. Many image segmentation techniques have been proposed in the past [32, 42, 80]. These segmentation techniques can be categorized into the following six categories: (1) amplitude thresholding, (2) component labeling, (3) boundary-based approaches, (4) region-based approaches and clustering, (5) template matching, (6) and texture-based segmentation. Although a great deal of work has been done on segmentation techniques, there are only a few texture-based segmentation techniques [11, 53, 69, 82, 88].

The purpose of image segmentation in image coding is fundamentally different from that in scene analysis discussed above. For compression applications, it is not

necessary to have a one-to-one correspondence between physical objects and image segments. It is only important to design the segmentation algorithm so that image segments are allocated in a way that achieves high compression with small visual quality loss. In the proposed new technique, this is accomplished by incorporating properties of the HVS at various stages in the segmentation algorithm. By using knowledge of properties of the HVS to guide the image segmentation, the segments can be chosen to produce a visually pleasing segmented image.

In segmentation-based image compression algorithms [6, 7, 43, 71, 74], the information that is encoded describes the boundaries and interiors of the segments in the segmented image. Thus, the number of image segments and the method of coding the interiors within segments will determine, for the most part, the bit rate of the compressed image. For this reason, it is critical that an image with a minimum number of segments is produced and that the interior of each segment is encoded efficiently. The goal of the segmentation algorithm we propose is, for a given desired image quality, to produce a segmented image which has not only the minimum number of image segments but also will result in an efficient bit allocation of the boundary and interior of each segment.

The segmentation technique we present segments an image into texturally homogeneous regions with respect to the degree of roughness as perceived by the HVS. The segmentation algorithm uses a variation of centroid-linkage region growing [32]. The region growing is directed by the texture distance measure between image blocks. In Section 5.1, we describe segmentation algorithm. The measure of the roughness of the textural regions is represented by the fractal dimension. In the actual segmentation, the fractal dimension is thresholded so that the textural regions are classified into three textural classes: perceived constant intensity, smooth texture, and rough

texture. A description of the fractal dimension was given in Chapter 3. Further analysis of the sensitivity of the calculation of fractal dimension to blocksize will be described in Section 5.2. In Section 5.3 we describe technique for choosing the fractal dimension thresholds that are used in the segmentation process. The other features used in segmentation are the JND and the expected mean. A method for measuring the JND and the experimental results are given in Section 5.4.

Experiments were performed on three test images given in Figure 5.1. Different types of imagery were chosen to show that the proposed algorithm works well for a wide variety of images. The first is a head and shoulder image, referred to as Miss USA. It is typical of those found in video-telephone or video-conferencing applications. In general, these images do not have highly complicated textural regions and thus, the segmented image contains a small number of segments with relatively large regions. The second is a complex image with many edges and is referred to as Lena. The last is a natural outdoor scene and is referred to as House. In general, segmentation-based compression techniques have not worked well for natural scenes such as in House. That is because it has highly textured areas which produce numerous segments in the segmented image. The approach we propose overcomes these difficulties because the image is segmented into texturally homogeneous regions. Each image consists of 256×256 pixels, with 256 gray levels. The images are viewed on a Sun 4 Workstation with a Sony monitor with 256 possible gray levels. The monitor was calibrated so that there was a linear relationship between gray level numeric value and output luminance using a technique proposed in [30].

(a)

(b)

(c)

Figure 5.1: Original test images. Each image is 256×256 pixels, with 256 gray levels. (a) Miss USA. (b) Lena. (c) House.

5.1 Image Segmentation

The goal of the image segmentation process is to decompose an image into texturally homogeneous regions with respect to the degree of roughness as perceived by the HVS. Textural regions are classified into three classes; perceived constant intensity, smooth texture, and complicated texture. For example, the background in a head and shoulder image or the sky in a natural image are considered as perceived constant intensity, the face or the shoulder is considered as smooth texture, and the trees and the bushes in a natural image is considered as rough texture. To extract texture information for accomplishing textural-based image segmentation, the fractal dimension, mean, and just noticeable difference (JND) are used in the segmentation algorithm. The segmentation algorithm is based on a region growing technique. A unique of feature of the region growing process used in this research is that it is directed by the texture feature distance between image blocks. The region growing is achieved through a merging test condition between texturally homogeneous neighboring blocks. If the condition for merging is satisfied, an observing block can be merged into a neighboring block. Otherwise, a new region is declared.

For our segmentation, we have used a centroid linkage region growing method because it is guaranteed to produce disjoint segments with close boundaries and provides a sequential algorithm for growing region. The centroid linkage region growing method is illustrated in Figure 5.2. The observing block [OB] is examined along with its neighboring blocks [NB1], [NB2], [NB3], and [NB4]. A classification of block [OB] is made after comparing the feature set of block [OB] with the feature sets of its neighboring blocks. The texture features used are the mean, JND, and the class type based on the fractal dimension of the image block.

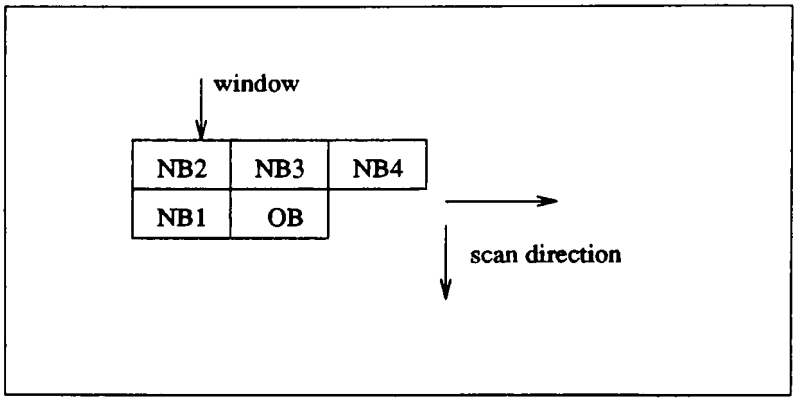


Figure 5.2: Centroid linkage window

The class type is determined by thresholding the fractal dimension. If a block has fractal dimension less than D_1, it is assigned to class I (perceived intensity value). If a block has fractal dimension greater than D_1 and less than D_2, it is assigned to class II (smooth texture). If a block has fractal dimension greater than D_2, it is assigned to class III (rough texture).

Using the class type value, the mean of the image block and the JND, the segmentation algorithm is as follows.

Texture-based segmentation algorithm:

Step 1) Divide the image into $NR \times NC$ blocks (NR and NC are the numbers of row and column blocks, respectively).

Step 2) Calculate the feature set: the class type, the mean, and the JND look-up table for each block.

Step 3) Calculate the distance between the observing block and its 4-connected

neighboring blocks. The distance is given by

$$\mathbf{D(OB,NB)} = \begin{cases} \mathbf{0} & \mathbf{if} \quad \begin{cases} F(OB) \le D_1,\ C_{OB} = C_{NB}, \\ |M(OB) - M(NB)| < \mathrm{JND}(OB, NB) \\ \mathbf{or} \\ D_1 < F(OB) \le D_2,\ C_{OB} = C_{NB} \\ \mathbf{or} \\ F(OB) > D_2,\ C_{OB} = C_{NB} \end{cases} \\ \mathbf{1} & \mathbf{otherwise} \end{cases}$$

where$F(OB)$ is the fractal dimension of the observing block. $C(OB)$ and $C(NB)$ are the class types of an observing block and its neighboring block respectively. $M(OB)$ and $M(NB)$ are the means for the observing block and its neighboring block respectively. JND(OB, NB) is the just noticeable difference between the observing block and its neighboring block.

Step 4) If there is a neighboring block with distance 0, then merge the observing block into it; else declare a new region. If there are more than two good neighboring blocks, merge the observing block into a neighboring block whose mean value is closest to the mean value of the observing block.

Step 5) Repeat step 3 to step 5 until all blocks are segmented.

Step 6) Stop.

The algorithm scans the image from top to bottom and from left to right.

In image segmentation, the block size for estimating the fractal dimension is a key component. The appropriate block size is critical for good estimation of the fractal dimension and control of the computation requirements. In section 5.2, we investigate the variation in the fractal dimension estimate for variable block sizes.

Another issue is the choice of the fractal dimension thresholds D_1 and D_2 in the segmentation process because the values of D_1 and D_2 affect the compression

ratio and the image quality. Higher values of D_1 give more regions belonging to the perceived constant intensity in an image. This results in a high compression ratio because only the perceived gray level and boundary information of these regions need to be transmitted, but generally the image quality is degraded. Lower values of D_2 produce more regions belonging to the rough textural class. In the proposed segmentation-based compression system, a larger amount of error is tolerated between the original image data and the modeled image data for the rough texture class than for the smooth texture class. Regions belonging to the smooth texure class contain middle range spatial frequencies, thus require more attention and emphasis in order to maintain image quality. We determine the range of fractal dimension for each texture class. Based on the range of the fractal dimension for each class, the threshold values for D_1 and D_2 are proposed.

The last issue in segmentation process is selection of the threshold used to determine when regions belonging to the perceived constant intensity merge or split. We investigate several different thresholds based on the HVS properties. The threshold that produces the best image quality is chosen in the proposed segmentation-based image compression system.

5.2 Determination of the Block Size for Estimating the Fractal Dimension

When we compute the fractal dimension in an image, the pixel intensity in an image is considered as a surface above a plane. All points in the three-dimensional space at distance ϵ from the surface were considered, covering the surface with a blanket of thickness 2ϵ. The algorithm for computing the fractal dimension was given in

Subsection 3.2.1. A brief summary of that algorithm is reiterated here. First, calculate the volume $V(\epsilon)$ using Eq. (3.5). Second, calculate the surface area $A(\epsilon)$ obtained from the volume divided by 2ϵ. Third, plot $\log A(\epsilon)$ versus $\log(\epsilon)$. Fourth, use a least squares linear regression to fit a straight line to the plot $\log(\epsilon)$ vs. $\log(\epsilon)$. Finally, the fractal dimension dimension D is equal to 2 minus to the slope of the straight line.

In the proposed compression algorithm, an image is divided into blocks and the fractal dimension of each block is computed. It is important to choose the block size in the image so that good estimates of the fractal dimensions are obtained and good image quality can be maintained. When the block size is small, there may not be enough pixels to describe the texture within the block. For example, if the block is 1×1, there is only one pixel in the block and it is not possible to characterize its texture. When the block size is large, several different textures may be present within the block and the estimated fractal dimension will not accurately represent the characteristics of the multiple textures. Another issue to be considered when choosing the block size is the computation requirements. The computation requirement for the large block is more expensive than the one for the small block. For example, consider the block sizes 8×8 and 16×16. The number of pixels in the block size 16×16 is four times larger than the number in the block size 8×8. Therefore, the computation time to compute the fractal dimension in the larger block is more expensive. Considering the issues discussed above, we conclude that the smallest feasible block size is the best block alternative.

5.2.1 Experimental Results for Determining the Best Block Size

To determine the best block size in terms of the fractal dimension, three 30×30 subimages in each of the test images are taken. Each subimage is assumed to belong

to one of the texture classes. We investigated the variation of the fractal dimension versus the block size for each class. Block size is varied from 2 to 30 and is increased from the left, top corner. Curves of the fractal dimension versus block size are given in Figures 5.3, 5.4, and 5.5. In each plot, the x-axis represents the block size and the y-axis the fractal dimension. The curve with a diamond symbol corresponds to the perceived constant intensity (class I), the curve with a cross symbol to the smooth texture (class II), and the curve with the square symbol to the rough texture (class III).

Examining the curves in Figure 5.3, the fractal dimension corresponding to perceived constant intensity ($\diamond$) has almost a constant value, 2.0, for blocksize $2, \cdots, 30$. That is, there exists only a single texture of perceived constant intensity in each block. The shape of the curve corresponding to smooth texture ($+$) are quite variable for blocks between $(2, \cdots, 7)$ and $(19, \cdots, 30)$ but are nearly constant for the middle block sizes $(8, \cdots, 18)$. The reason for variability in the smaller $2, \cdots, 7$ blocks is the small number of pixels to characterize the texture. In larger $19, \cdots, 30$ blocks more than one texture is present. The middle $8, \cdots, 18$ blocks have only one texture and provide the least variable estimate of the fractal dimension. Note, the 30×30 subimage at the bottom of Figure 5.3 has three different textures; the neck and two sweaters. In general, when the block size is large, there is more likelihood that several textures will be in the block, and the value of the fractal dimension will not remain constant. The curve ($\sqsubset$) corresponding to the rough texture blocks looks similar to that of the smooth texture. At the small block sizes, there are too few pixels to estimate the texture and at the large block sizes, multiple textures are present in the block. The curve is nearly constant for middle $(8, \cdots, 18)$ block sizes. In Lena and House, the shape of the curve corresponding to rough texture is nearly constant for

blocks greater than 7×7 block. Since the 30×30 subimages in the feather in Lena and in the tree in House have one texture, the fractal dimension of the regions within the block remain nearly constant. In summary, the larger block sizes may not give good estimates of the fractal dimension because they contain several textures and the smaller block may not contain enough pixels to characterize the texture.

Through extensive experimentation, we have found that block sizes of 8×8 up to 14×14 have almost a constant value. Thus these blocks meaningfully represent the textural characteristics of a region. An estimate of the means and the standard deviations of the fractal dimensions for the blocks in each class for the three test images as a result of these simulations are given in Tables 5.1, 5.2, and 5.3.

We chose an 8×8 block size for the block-by-block segmentation algorithm since the smaller block size reduces the computation and storage requirement and as will be seen later is consistent with giving the best image quality. Furthermore, by comparing the set of curves in each plot, curves on square, cross, and diamond symbols are the top, middle, and bottom respectively for the mid sized blocks from 8×8 to 14×14, rougher texture produces higher fractal dimension.

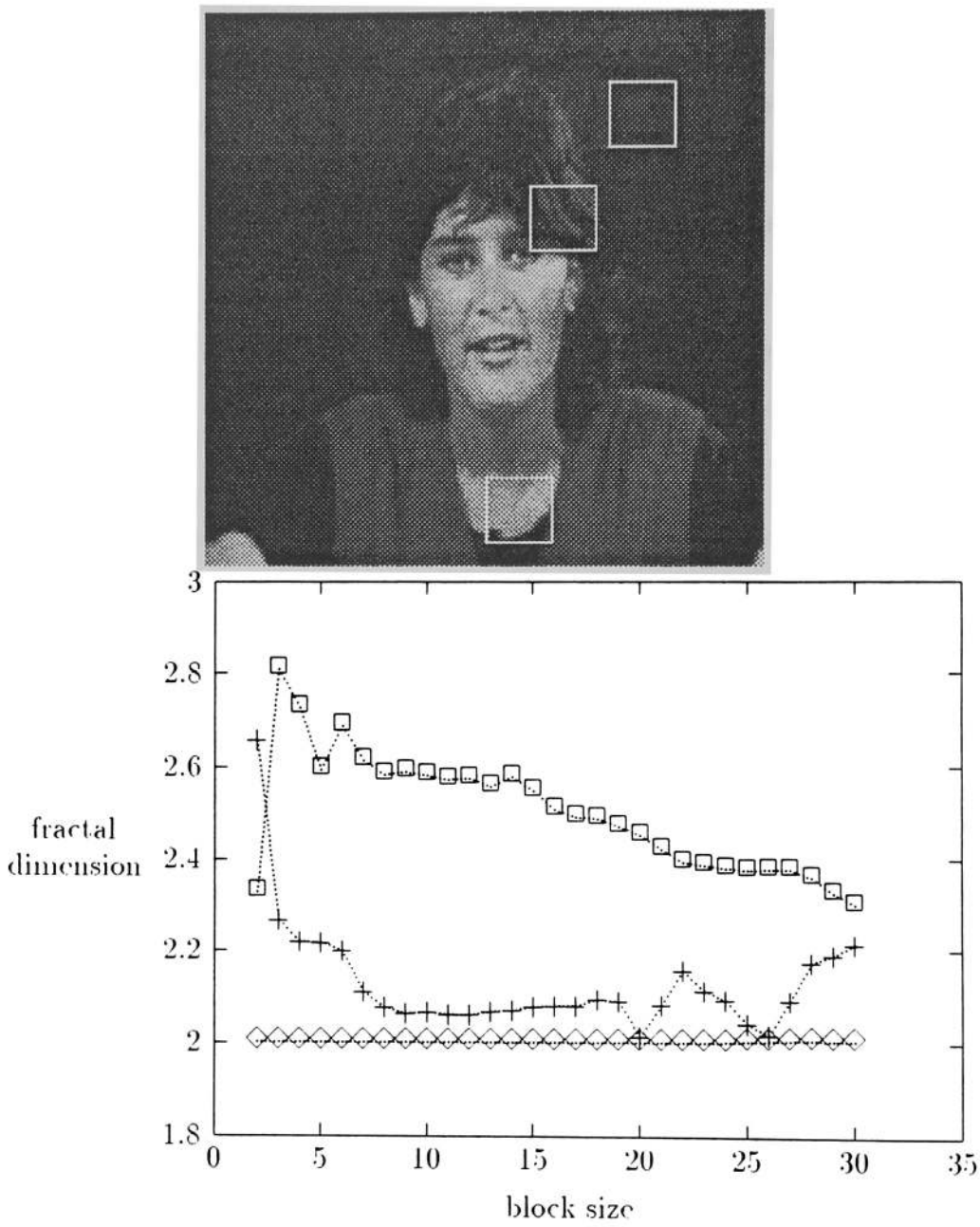
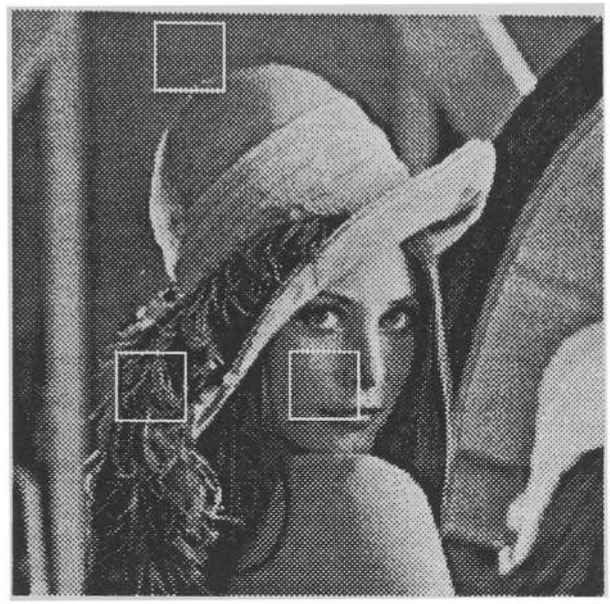


Figure 5.3: Plot of fractal dimension versus block size in Miss USA. Miss USA with three subimages is given on the top. Three subimages on the top, middle, and bottom belong to perceived constant intensity, rough texture, and smooth texture respectively. A plot of fractal dimension versus block size is given on the bottom. The curves with a diamond symbol, a cross symbol, and a square symbol correspond to perceived constant intensity, rough texture, and smooth texture respectively.



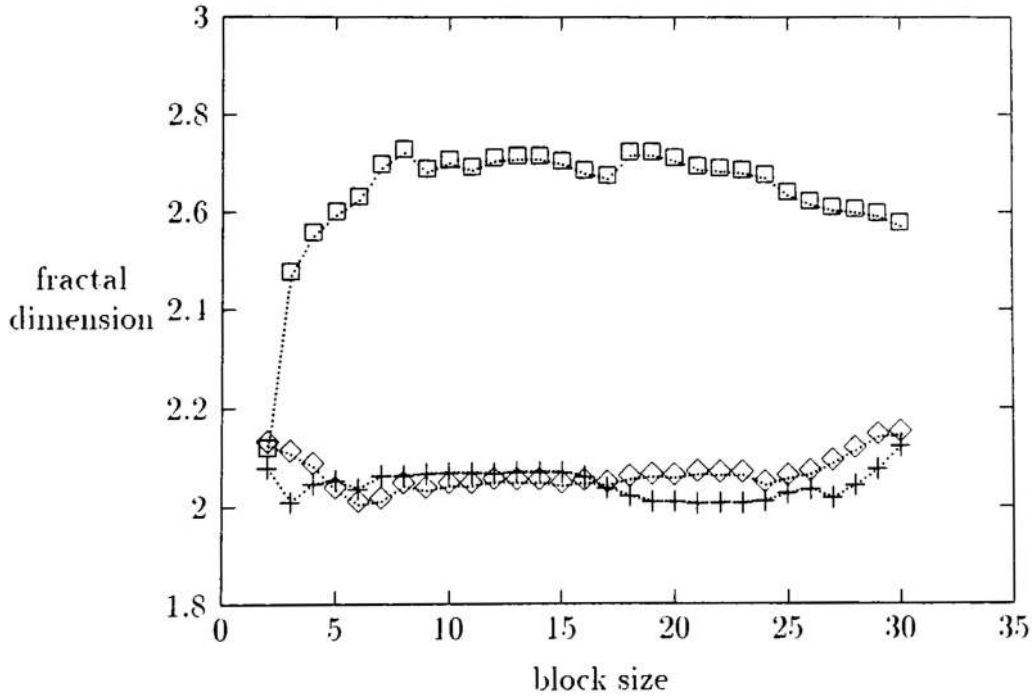
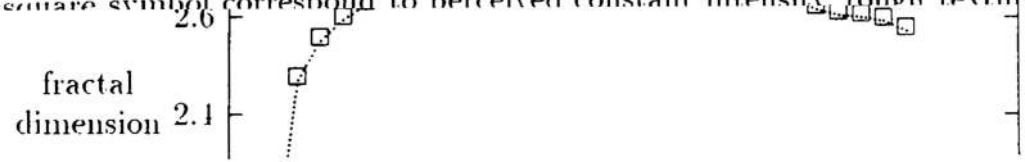


Figure 5.4: Plot of fractal dimension versus block size in Lena. Lena with three subimages is given on the top. Three subimages on the top, the bottom and left corner, and the bottom and right corner belong to perceived constant intensity, rough texture, and smooth texture respectively. A plot of fractal dimension versus block size is given on the bottom. The curves with a diamond symbol, a cross symbol, and a square symbol correspond to perceived constant intensity, rough texture, and



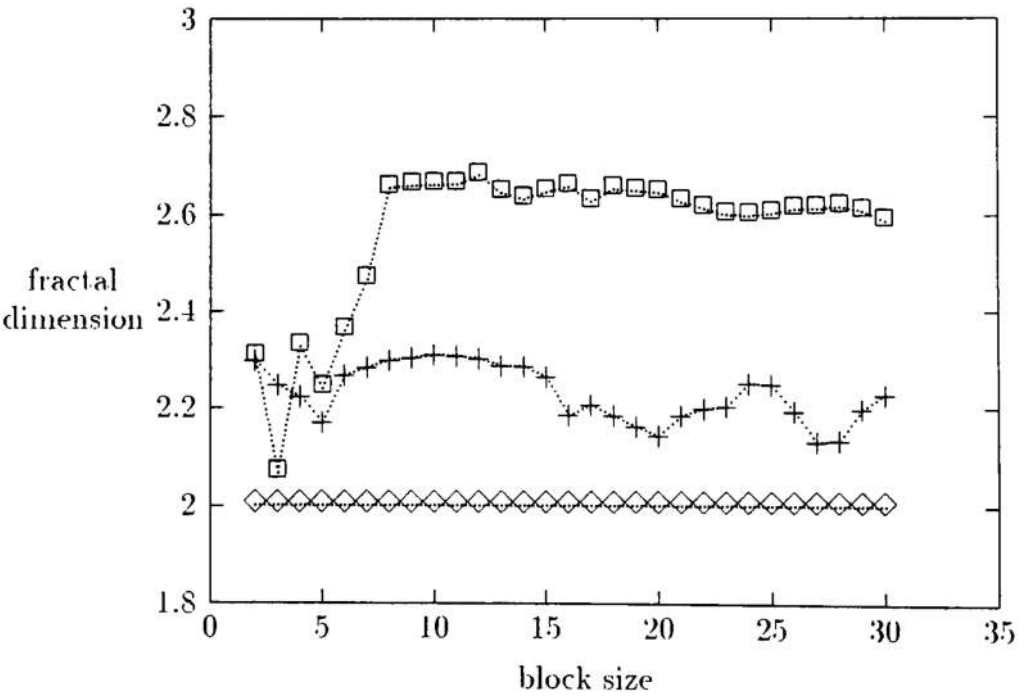


Figure 5.5: Plot of fractal dimension versus block size. House with three subimages is on the top. Subimages on the top and left corner, the top and right corner, and the bottom and right corner in the image belong to rough texture, perceived constant intensity, and smooth texture respectively. A plot of fractal dimension versus block size is given on the bottom. The curves with a diamond symbol, a cross symbol, and a square symbol correspond to perceived constant intensity, rough texture, and

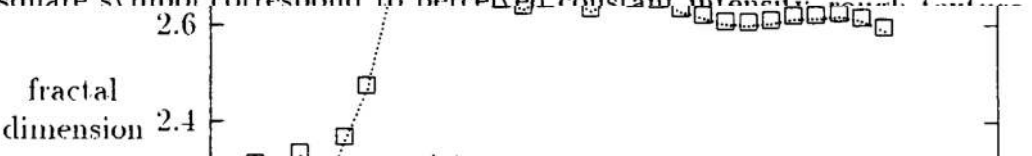


Table 5.1: Statistics of fractal dimension in Miss USA

class	block size	mean	standard deviation
class I	2×2 to 28×28	2.000705	0.000002
class II	8×8 to 19×19	2.072152	0.000110
class III	8×8 to 14×14	2.576650	0.000086

Table 5.2: Statistics of fractal dimension in Lena

class	block size	mean	standard deviation
class I	8×8 to 14×14	2.040403	0.000026
class II	7×7 to 15×15	2.066312	0.000007
class III	7×7 to 15×15	2.698575	0.000140

Table 5.3: Statistics of fractal dimension in House

class	block size	mean	standard deviation
class I	2×2 to 28×28	2.000000	0.000000
class II	7×7 to 14×14	2.296010	0.000091
class III	8×8 to 20×20	2.650985	0.000172

5.3 Thresholds for the Fractal Dimension

A key feature of the fractal dimension in the segmentation algorithm is the thresholds D_1 and D_2 used in order that the textural regions can be separated into textural classes and better image quality is obtained with a larger compression ratio. The values of the thresholds D_1 and D_2 affect the compression ratio and the image quality. Higher values of D_1 give more regions belonging to the perceived constant intensity class in an image. It results in high compression ratio because only the perceived gray level and boundary information of these regions need to be transmitted. But generally the image quality is degraded. Lower values of D_2 produce more regions belonging to rough texture class. In the proposed segmentation-based compression system, a higher amount of error tolerated between the original image data and the modeled image data is chosen for class III than class II because of the sensitivity of the HVS to middle range of spatial frequencies. Higher values of D_1 and lower values of D_2 result in high compression system, however, the image quality is degraded. Therefore, it is important to have suitable thresholds D_1 and D_2 for compression and image quality.

The relationship between the fractal dimension D and the negative slope β_2 of the PSD of the FBF in two-dimensional case is given as derived in Subsection 3.2.2

$$D = 4 - \frac{\beta_2}{2}$$

As D is close to either 2 or 3, β_2 is close to 4 or 2, respectively. This means that a lower value of D corresponds to a larger negative slope of the PSD and a lower spatial frequency content. Using this relationship, we would like to find the thresholds D_1 and D_2 based on properties of the HVS. One simple and good HVS model uses

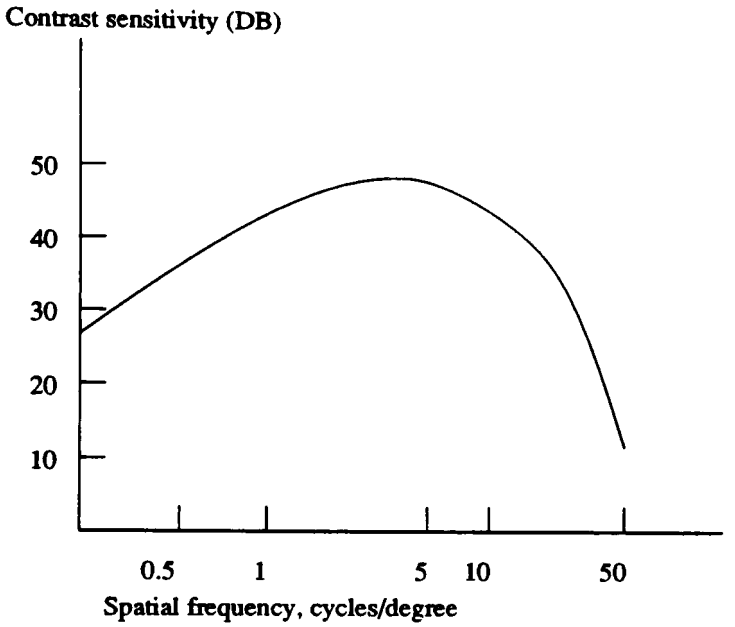


Figure 5.6: A typical MTF curve

the MTF. The curve of a typical MTF is given in Figure 5.6. Using the MTF has the advantage that it tells us which frequencies are more important for maintaining quality. This can be used in the image coding. Once the MTF is specified, it would be desirable to obtain an exact relation between the PSD of the textural regions in an image and the MTF of the HVS. The relation could then be used to specify the thresholds for the fractal dimension. However, it has proven to be very difficult to derive an analytic relationship between the two. Therefore, another alternative is to define the thresholds for the fractal dimension empirically.

The shape of the MTF of the HVS is similar to a band-pass filter and suggests

that the HVS is more sensitive to midfrequencies and less sensitive to low and high frequencies in Figure 5.6. Thus we have chosen to segment an image into three textural classes (perceived constant intensity, smooth texture, and rough texture) based on spatial frequency content. Since the fractal dimension of a shape corresponds to roughness, regions which belong to the perceived constant intensity have a fractal dimension less than D_1. The second class contains regions with the fractal dimension between D_1 and D_2 which corresponds to the midfrequencies, those regions which are more sensitive to the human being. This implies that midfrequencies play a more important role in perceived image quality than other frequencies and that these regions require more attention and emphasis in order to maintain image quality. The third class contains regions with fractal dimension greater than D_2. These regions are perceived as the most rough regions, but ones to which a human has less sensitivity; thus they can be more highly compressed.

To define the thresholds for the fractal dimension empirically, many subimages which may belong to each class in the images are chosen. Fractal dimensions of the subimages in each class are investigated and the range of fractal dimensions is determined.

5.3.1 The Experimental Results for Determining Thresholds D_1 and D_2

Five 8×8 subimages belonging to each class in each test image with 256×256 pixels, and 256 gray levels are chosen. In this experiment, an 8×8 block size is used for each subimage because the block size is the smallest one to characterize meaningfully the texture of a region as discussed in Subsection 5.2.1. For Miss USA, 5 subimages in the background and sweater are chosen to represent class I, 5 subimages in the neck, cheeks, and shoulder for class II, and 5 subimages in the hair, eyes, mouth, and

nose for class III. For Lena, 5 subimages in the background and middle of the mirror are chosen for class I, 5 subimages in the hat, cheek, and shoulder for class II, and 5 subimages in the feather and eyes for class III. For House, 5 subimages in the sky and concrete wall are chosen for class I, 5 subimages in the lawn and car for class II, and 5 subimages in the trees in the left and right of the image for class III.

Images with five 8×8 subimages for each class are given in Figures 5.7 to 5.15. The mean and the standard deviation of the five subimages fractal dimensions are given in Tables 5.4 to 5.12. The mean and the standard deviation of the fractal dimension of the fifteen subimages corresponding to class I are $\mu = 2.007853$ and $\sigma = 0.000053$, respectively. The mean and the standard deviation of the fractal dimensions of the fifteen subimages corresponding to class II are $\mu = 2.194596$ and $\sigma = 0.005807$, respectively. The mean and the standard deviation of fractal dimensions of the fifteen subimages corresponding to class III are $\mu = 2.660297$ and $\sigma = 0.023209$, respectively. A plot of the fractal dimensions of the fifteen subimages for each class is given in Figure 5.16. In the plot, the x-axis represents the fractal dimension and the y-axis the number of blocks at that fractal dimension. The curve with a diamond symbol corresponds to the perceived constant intensity (class I), the curve with a cross symbol to the smooth texture (class II), and the curve with the square symbol to the rough texture (class III). The fractal dimensions belonging to class I are distributed around the fractal dimension, 2.0. The curves of the fractal dimensions belonging to class II and class III are approximately bell-shaped around their means respectively. There are gaps between the curves for each class. Through these results, the value of D_1 should not be greater than the minimum fractal dimension belonging to class II and the value of D_2 should lie between the fractal dimension means of class II and class III. In Chapter 6, we determine the sensitivity of the codec to variations in D_1 and

D_2 as a starting point we propose to let $D_1 = \frac{D_{Imax}+D_{IImin}}{2}$ and $D_2 = \frac{D_{IImax}+D_{IIImin}}{2}$, where D_{Imax} is the maximum fractal dimension belonging to class I, D_{IImax} is the maximum fractal dimension belonging to class II, and D_{IIImin} is the minimum fractal dimension belonging to class III.

Figure 5.7: Estimation of the fractal dimension for the perceived constant intensity blocks in Miss USA. Five 8×8 blocks are used.

Table 5.1: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension
(109,17)	2.008697
(211,44)	2.017080
(35,63)	2.000000
(100,221)	2.001205
(228,236)	2.001842

(a)

μ	σ
2.008697	0.000011

(b)

Table 5.1: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension

Figure 5.8: Estimation of the fractal dimension for the smooth texture blocks in Miss USA. Five 8×8 blocks are used.

Table 5.5: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension
(105,126)	2.137630
(150,126)	2.214584
(180,180)	2.167482
(138,206)	2.101584
(138,222)	2.324481

(a)

μ	σ
2.189152	0.005951

(b)

Table 5.5: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension

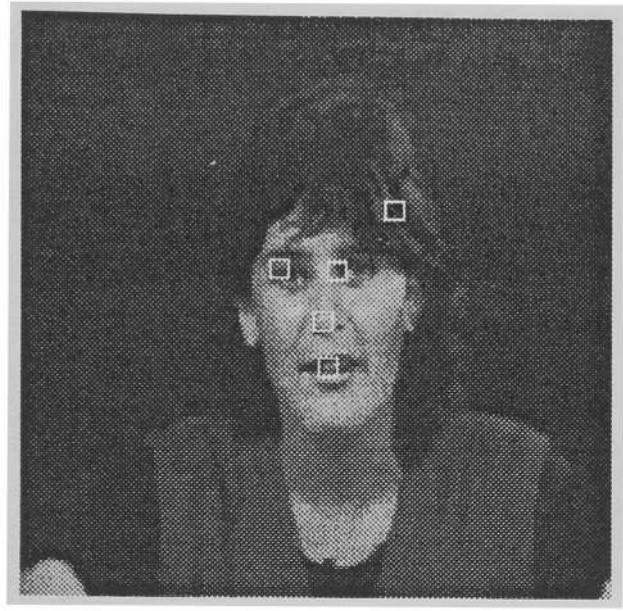


Figure 5.9: Estimation of the fractal dimension for the rough texture blocks in Miss USA. Five 8×8 blocks are used.

Table 5.6: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension
(157,81)	2.631716
(107,107)	2.861825
(132,108)	2.468696
(126,130)	2.403171
(128,150)	2.617450

(a)

μ	σ
2.596571	0.025157

(b)

Table 5.6: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension

Figure 5.10: Estimation of the fractal dimension for the perceived constant intensity blocks in Lena. Five 8×8 blocks are used.

Table 5.7: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension
(148,10)	2.009648
(37,97)	2.001016
(239,97)	2.021052
(2,107)	2.013692
(242,164)	2.011178

(a)

μ	σ
2.011323	0.000042

(b)

Table 5.7: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension

Figure 5.11: Estimation of the fractal dimension for the smooth texture blocks in Lena. Five 8×8 blocks are used.

Table 5.8: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension
(94,40)	2.273487
(129,63)	2.284637
(227,116)	2.231022
(120,156)	2.071251
(158,218)	2.049282

(a)

μ	σ
2.181936	0.010237

(b)

Table 5.8: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension

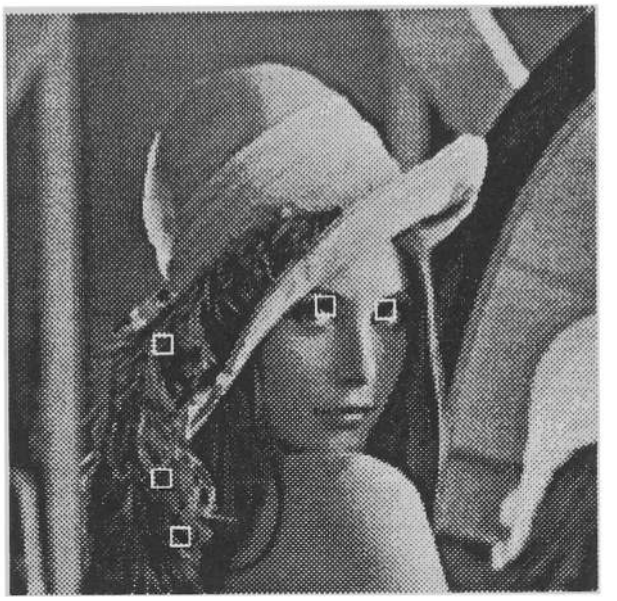


Figure 5.12: Estimation of the fractal dimension for the rough texture blocks in Lena. Five 8×8 blocks are used.

Table 5.9: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension
(134,125)	2.847927
(160,127)	2.591966
(63,141)	2.728663
(62,200)	2.521942
(70,226)	2.890308

(a)

μ	σ
2.716161	0.020198

(b)

Table 5.9: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension

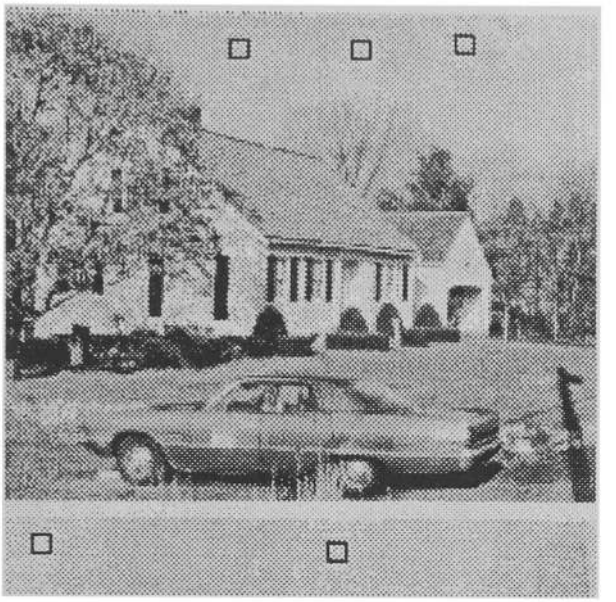


Figure 5.13: Estimation of the fractal dimension for the perceived constant intensity blocks in House. Five 8×8 blocks are used.

Table 5.10: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension
(194,9)	2.000000
(96,12)	2.000000
(149,12)	2.000000
(11,229)	2.014151
(139,232)	2.018201

(a)

μ	σ
2.006470	0.000064

(b)

Table 5.10: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension

Figure 5.14: Estimation of the fractal dimension for the smooth texture blocks in House. Five 8×8 blocks are used.

Table 5.11: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension
(226,140)	2.214396
(181,154)	2.255769
(107,181)	2.180522
(168,189)	2.188989
(13,216)	2.223822

(a)

μ	σ
2.212699	0.000716

(b)

Table 5.11: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

Figure 5.15: Estimation of the fractal dimension for the rough texture blocks in House. Five 8×8 blocks are used.

Table 5.12: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension
(30,23)	2.715114
(144,53)	2.729585
(9,68)	2.840850
(217,93)	2.457473
(227,108)	2.597764

(a)

μ	σ
2.668157	0.017029

(b)

Table 5.12: Fractal dimensions for each blocks. The column and row index are ordered from the top left. The fractal dimension of each subimage is given in table (a). The mean μ and the standard deviation σ are given in table (b).

(column,row)	fractal dimension

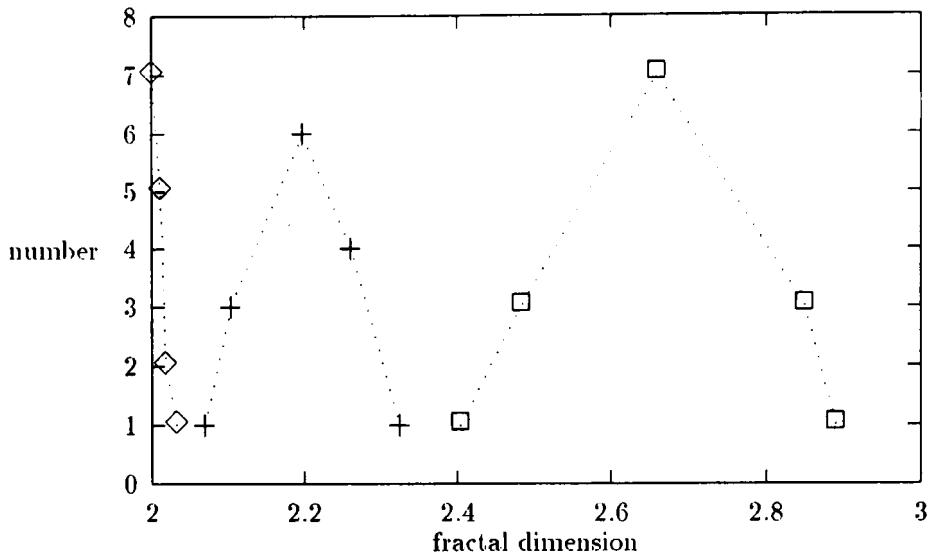


Figure 5.16: A plot of the fractal dimensions of the fifteen subimages for each class. The x-axis represents the fractal dimension and the y-axis the number of blocks at that fractal dimension. The curve with a diamond symbol corresponds to the perceive constant intensity, the curve with a cross symbol to the smooth texture, and the curve with the square symbol to the rough texture.

5.4 Selection of the Threshold for Regions Belonging to Perceived Constant Intensity

In our proposed texture-based segmentation algorithm, textural regions are classified into three classes: perceived constant intensity, smooth texture, and rough texture. To determine the merging-split condition between the regions belonging to the perceived constant intensity, we use the visual threshold in the segmentation algorithm. We investigate three different thresholds, two are based on HVS properties. The HVS-based thresholds we propose are adapted to local intensity characteristics of the image. As the segmentation algorithm progresses spatially through the image, the segmentation threshold is varied, depending on the intensity of the image in a local area. The thresholds have been designed for use on an image with 256 gray levels. The third threshold is the simplest threshold possible, a single constant that is used for the entire image. We refer to the constant threshold as th_{con}.

We incorporate the HVS properties in our segmentation algorithm using the JND as the visual threshold. The split-field method [30] to measure the JND discussed in Subsection 2.3.1 is used since it measures the JND quickly and reliably. The image display device is divided down the middle into two equal-size fields. The left field is a constant reference intensity and the right field begins at the top with the constant reference intensity and increases linearly up to 40 steps above the constant reference intensity. To perform the tests, the viewer simply clicks the mouse at the point where the difference between the left and right fields is no longer discernible. This point is the JND between the reference intensity on the left and the test intensity on the right. A plot of the average of five viewers is shown in Figure 5.17.

As a third threshold, the JND curve is approximated by three straight line com-

ponents. The motivation behind the straight line component approximation is to determine whether a simple approximation to Figure 5.17 can be used. The threshold is largest in the highest and lowest intensity areas of the image and smallest and almost constant in the middle intensity areas. The JND curve is approximated by using a least squares linear regression to fit straight lines to the JND curve. The third threshold th_{ap} is given by

$$th_{ap} = \begin{cases} a_1 p + b_1 & for \ \ 0 \leq p < p_1 \\ a_2 p + b_2 & for \ \ p_1 \leq p < p_2 \\ a_3 p + b_3 & for \ \ p_2 \leq p \leq 255 \end{cases}$$

where a_i is a slope, b_i is y-intercept, i is a line index, and p is a gray level, see Figure 5.18.

5.4.1 The Experimental Results

The thresholds described above were used to segment the test images shown in Figures 5.1. These test images are 256 × 256 pixels, with 256 gray levels. The three segmentation thresholds were compared to each other in order to determine which threshold resulted in the most visually pleasing segmented image.

First th_{con} was examined. th_{con} is used to decide when to merge two blocks belonging to class 1. We wished to determine approximately which th_{con} resulted in the subjectively best visual quality segmented image. Comparing the segmented images with a variety of th_{con}, the best visual quality segmented images were obtained with $th_{con} = 5.5$. The segmented images of test images are shown in Figure 5.19 using $th_{con} = 5.5$. The numbers of segments belonging to the perceived constant intensity are given in Table 5.13.

The second threshold was found using the JND. The test image size is 256×256 with 256 gray levels. Five test subjects were asked to take a measurement of JND. Each test subject sat a distance of approximately six times the image height away from the screen. The test subject was given approximately three minutes before the start of the experiments, to allow for adaptation to the laboratory's illumination. The test subject was asked to take five seconds in each click to allow for adaptation to the screen. The mean of five subjects JND measurements is shown in Figure 5.17. In the figure we see that the experiment agrees with the HVS contrast sensitivity properties [14]. The JND is largest in the lowest and highest intensity areas of the image. The JND is smallest and nearly constant in the middle intensity areas of the image. The segmented images of test images are shown in Figure 5.20 using this threshold. The numbers of segments of the perceived constant intensity are given in Table 5.14.

The third threshold is obtained by approximating the JND curve with three straight line components using a least-square linear regression. The approximated curve is described by six parameters: three slopes a_1, a_2, and a_3, and three y-intercepts b_1, b_2, and b_3. The parameters calculated are $a_1 = -0.57282$, $a_2 = 0.00931$, $a_3 = 0.73864$, $b_1 = 42.96975$, $b_2 = 4.40066$, and $b_3 = 6.05303$. The original JND curve and the approximated JND curve are overlaid for comparison in Figure 5.18. The bold line corresponds to the approximated JND curve. The values of the JND and the approximated JND are almost the same for the middle and high intensity areas of the image and the values are a little different only for the lower intensity areas of the image. However, since the human has the least sensitivity in the lower intensity areas, the difference does not affect the image quality. The segmented images of test images are shown in Figure 5.21 using the approximated JND. The number of

segments belonging to the perceived constant intensity are given in Table 5.15

Comparing the segmented images in Figures 5.19 to 5.21, th_{con} has been shown to be inferior to th_{JND} and th_{ap}. From these sets of images it can be seen that th_{JND} produces as good or slightly better quality segmented images than th_{ap}. This is as expected because th_{JND} better adapts the local properties of the images. The number of segments belonging to the perceived constant intensity is approximately the same for th_{JND} and th_{ap} and the number of the segments for th_{JND} and th_{ap} is less than the one for th_{con}. In a segmentation-based image compression system, the number of the segments is critical to the bit rate because the boundaries interiors of the segments are encoded. Therefore, the best threshold is the one that produces not only a better image quality but also the minimum number of image segments. th_{JND} or th_{ap} can be chosen as the threshold. We use th_{JND} as the threshold in the proposed segmentation algorithm because th_{JND} produces slightly better quality segmented image than th_{ap}.

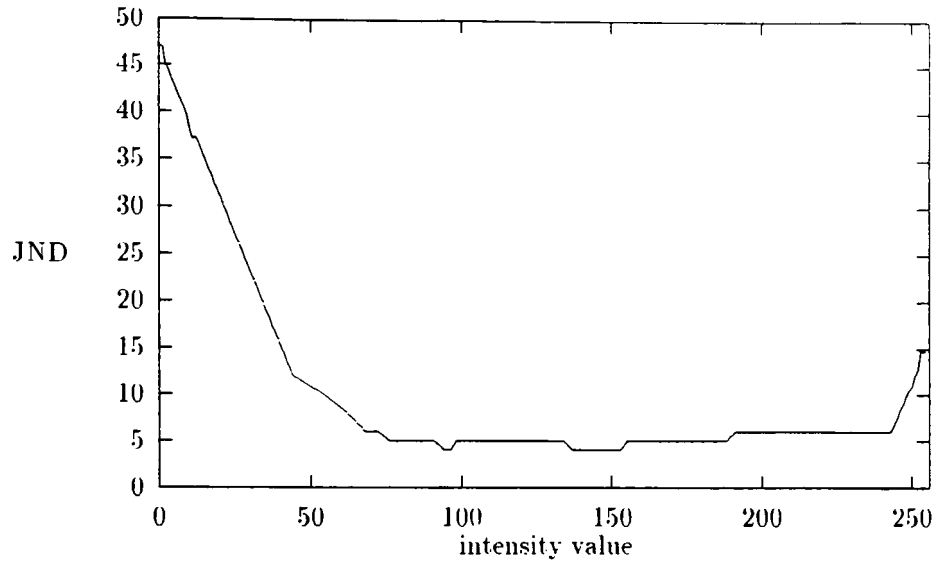


Figure 5.17: The mean of five subjects' JND measurements

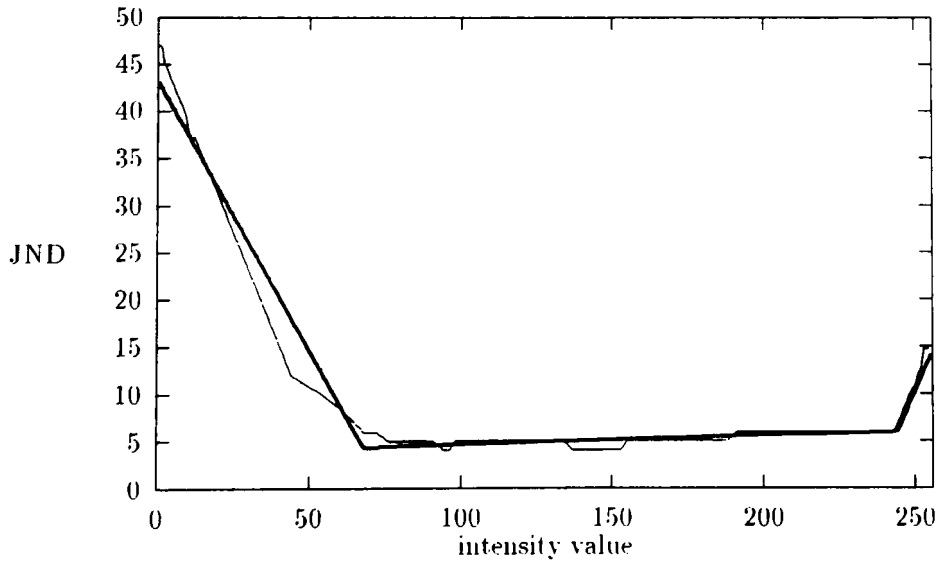


Figure 5.18: The original JND curve and the approximated JND curve. The bold line corresponds to the approximated JND curve

(a)

(b)

(c)

Figure 5.19: The segmented images using $th_{con} = 5.5$. (a) Miss USA. (b) Lena. (c) House.

Table 5.13: The number of segments for test images using $th_{con} = 5.5$.

Image	Miss USA	Lena	House
Num. of class I	448	821	851

Table 5.13: The number of segments for test images using $th_{con} = 5.5$.

Image	Miss USA	Lena	House
Num. of class I	448	821	851

(a)

(b)

(c)

Figure 5.20: The segmented images using th_{JND}. (a) Miss USA. (b) Lena. (c) House.

Table 5.14: The number of segments for test images using th_{JND}.

Image	Miss USA	Lena	House
Num. of class I	391	774	680

Image	Miss USA	Lena	House
Num. of class I	391	774	680

(a)

(b)

(c)

Figure 5.21: The segmented images using th_{ap}. (a) Miss USA. (b) Lena. (c) House.

Table 5.15: The number of segments for test images using th_{ap}.

Image	Miss USA	Lena	House
Num. of class I	395	781	693

Image	Miss USA	Lena	House
Num. of class I	395	781	693

5.5 Conclusion

In this chapter we discussed how to determine the block size for estimating the fractal dimension because the block size is critical in good estimates of the fractal dimension. Then we examined how to threshold the fractal dimension. The threshold values of the thresholds D_1 and D_2 affect the compression ration and the image quality. Finally, the selection of the threshold used to determine when regions belonging to the perceived constant intensity merge was discussed. It was shown that the JND or the approximated JND rather than the constant threshold produced better quality segmented images and less number of the segments.

6

A Texture Segmentation-Based Image Coder

6.1 Introduction

In this chapter 4, a complete description of the new codec model was given. The transmitter was divided in three major components; the transmitter, the segmenter, and the mixed encoder. The preprocessor and the segmenter have been discussed in detail in Chapters 4 and 5, respectively. In this chapter, we present the details of the mixed encoders. The purpose of using the mixed encoder rather than a single encoder is that a higher compression image can be obtained by applying an efficient coding technique to the boundary and texture classes. Furthermore, more bits are used to encode the information for which the HVS is more sensitive and fewer bits are used to encode the information for which the HVS is less sensitive.

After presenting the details of the mixed encoding, the performance of the proposed image coding technique will be addressed. The first issue concerns the segmentation method. A modification will be proposed that allows for increased spatial detail without decreasing the block size. The modification uses the concept of overlaped blocks. The choice of overlap or nonoverlap blocks affects the number of segments belonging to each class. Specifically, the number of segments belonging to the per-

ceived constant intensity is critical to the compression ratio since their perceived gray levels need to be transmitted. We investigate the number of segments for each class and the number of pixels for each class in the non-overlap and overlap segmentation.

The second issue is the performance of the proposed coding system with respect to variation in D_1 and D_2. The values of D_1 and D_2 affect the compression and the image quality. Higher values of D_1 give more regions belonging to the perceived constant intensity in an image and lower values of D_2 produce more regions belonging to the rough texture class. We examine bit rates and the change of the number of segments for each class with D_1 and D_2 variable.

The texture information in regions belonging to class II and class III are encoded using polynomial functions. The amount of error that can be tolerated between the original image data and the modeled image data is. A lower error is chosen for class II than class III because of the sensitivity of the HVS. The last issue is to evaluate the performance with variation in the amount of error.

6.2 The Mixed Encoder

The last stage in the transmitter encodes the boundaries and the interiors of the regions in the segmented image. In the proposed image segmentation, the regions are classified as belonging to one of three classes. Classes I, II, and III are perceived constant value, smooth texture, and rough texture, respectively. In the proposed coding approach, separate coding schemes are used for each class. We would like the image coder to use more bits to encode the regions for which the HVS is more sensitive, and use fewer bits to encode the regions for which the HVS is less sensitive. To accomplish this, the encoder is designed as separate stages: the encoding of the

boundaries, and the encoding of each textural class. In the next section, the encoding of the boundaries will be discussed.

6.2.1 The Boundary Coding

Since the HVS is sensitive to the edges, the accurate representation of the boundary is necessary to describe the location of the region boundary. Therefore, we chose an errorless coding scheme to represent the boundaries. A binary map is created to represent the boundaries. We examine three different errorless binary codes: runlength code [22], crack code [77], and arithmetic code [73, 76].

Runlength coding is a technique which works well on sparse binary signals, for example an image made up mostly zero, with a few ones. The image rows are catenated together to form a vector, and all runs of consecutive 0's are found. The lengths of these runs, separated by a symbol (referred to as a comma) to mark the end of a run (i.e. the presence of a 1), completely describe the original image. The runlengths and commas are then coded using a source coding technique such as the one described in [22]. This technique involves using n symbols in an n-ary arithmetic system to represent the runlengths, and an n+1'th symbol to represent a comma. Details are described in Appendix 9.1.

Crack coding traces the boundaries between regions. The lines that separate pixels belonging to different regions are coded. This is a four-way connected line diagram, in which links are from the four element set up, down, left, right. The coded segmentation image consists of two independent sources; a list of coordinates from which to start tracing the edges and the edge description. The crack coding is described in detail in Appendix 9.2.

Arithmetic coding, introduced in the LIFO-form (last-in first-out) in Rissanen [76] and subsequently modified to the important FIFO-form (first-in first-out) in Pasco [58], encodes the strings, S consisting of the symbols $i = 0, 1, \cdots, m$. The basic encoding operation in an arithmetic code requires an update of of the probability $P(S)$ of the so-far processed string S, which can be done by $P(S)P(i/S)$, where $P(i/S)$ is a conditional probability of the symbol i given S. The arithmetic coding is described in detail in Appendix 9.3.

In our approach, the boundary information was represented using blocks, not pixels. Therefore, the number of bits to represent the boundaries is almost reduced by the block size. The compression ratio of the boundaries does not govern the overall compression ratio.

6.2.2 The Perceived Constant Regions

For regions which belong to perceived constant intensity class (class I), only the mean intensity values need be transmitted to describe the textures of the regions. In this case, lossy compression has already taken place since we are approximating each region texture with a constant value. We do not wish to introduce any further compression.

Since an intensity mean requires 8 bits, the mean intensity value is converted into a vector of an $8 \times N$ binary array, where N is the number of segments belonging to perceived constant regions. The mean vector is then encoded using an arithmetic code.

6.2.3 The Smooth and Rough Textural Regions

The texture information in regions belonging to class II and class III are not directly encoded. To get higher compression, these regions are modeled first using polynomial functions. The coefficients of the polynomial functions are encoded because the variance of the coefficients is less than that of the original data. Arithmetic code is used to encode the coefficients. A lower amount of error tolerated between the original image data and the modeled image data is chosen for class II than class III because of the sensitivity of the HVS. In general, modeling these regions by functions of higher order polynomials is computationally excessive. In this section, the use of one or two dimensional polynomial functions is investigated and the performance of the encoding system to variation of tolerated amounts for class II and class III are investigated also.

The functions that approximate a signal within each region using 2-D polynomials, have the form:

$$\begin{aligned}
\text{order 0:}\quad & z(i,j) = a_0 \\
\text{order 1:}\quad & z(i,j) = a_0 + a_1 i + a_2 j \\
\text{order 2:}\quad & z(i,j) = a_0 + a_1 i + a_2 j + a_3 ij + a_4 i^2 + a_5 j^2
\end{aligned}$$

In the 1-D case these reduce to:

$$\begin{aligned}
\text{order 0:}\quad & z(i,j) = a_0 \text{ for all j} \\
\text{order 1:}\quad & z(i,j) = a_0 + a_1 i \text{ for all j} \\
\text{order 2:}\quad & z(i,j) = a_0 + a_1 i + a_2 i^2 \text{ for all j}
\end{aligned}$$

Regions belonging to the smooth and rough texture class are modeled using 1-D or

2-D polynomials by minimizing the SSE of the approximation given by

$$\sum_i \sum_j (g(i,j) - z(i,j))^2$$

where g(i,j) is the two-dimensional intensity field of the original image and i,j are defined within each region. The selection of the 1-D or 2-D representation for regions belonging to class II and class III depends on the SSE. Representation of the smooth and the rough textures by the zero-order, first-order, and second-order 2-D polynomial functions produces large SSE because these regions are changed to plateau, tilted plane, and quadratic surfaces, respectively. If the SSE is large, the approximated error is large and the image quality is very noticeably degraded. The other alternative is to use the 1-D representation for these regions.

In the one-dimensional case, the sum of deviations between the original image data and the modeled image data using the 1-D polynomial functions is used as the amount of error. The amount of error is compared with the threshold, and if it is less than the threshold, the next pixel is taken as the temporary end point. This procedure is repeated until the sum of deviations is larger than the threshold. At this point, the previous pixel examined will be taken as the end point of the modeled function. The same pixel is also taken as the start point for the fitting function.

The sum of deviations for class II is set to be smaller than one for class III. That is why regions which belong to the class II require more attention in order to maintain image quality because of the sensitivity of the HVS. For regions which belong to the class III like the class I, compression error can be tolerated because the HVS is least sensitive to these regions.

6.3 Nonoverlap and Overlap Segmentation Method

During image segmentation, the image is segmented into three texture classes. The number of pixels belonging to each class in an image affects the overall compression ratio. In general, the more regions belonging to the class I, the higher the compression ratio. An approach to obtain higher compression ratio and maintain high image quality is considered. It is referred to as the overlap method. In the non-overlap method, an image is divided into mutually exclusive blocks. In the overlap method, the block subimages are allowed to overlap at their perimeters. The pixels at the perimeter are computed in two or more blocks. If the pixels at the perimeter represent perceived constant intensity, the overlap method produces more regions which belong to the perceived constant intensity. Thus, the compression ratio is increased while the image quality is maintained or improved. An example of the overlap method is shown in Figure 6.1.

In the figure, an 8×8 image consists of three strips of textures. The block size is 4×4. Pixels in the first and second columns represent texture which corresponds to the class II. Pixels from the third through the sixth columns represent texture which corresponds to the class I. Pixels in the last two column represents texture which corresponds to the class III. The 50 percent overlap method characterizes the texture in the middle four columns while the non-overlap method does not characterize the texture exactly, as shown in the figure. However, some pixels are computed more than once, and this increases the computation load, as shown in the figure. When an image of 256×256 pixels is divided into 8×8 blocks, the number of blocks with 0, 50, and 75 percent overlaps are 32×32, $4 \times 32 \times 32$, and $16 \times 32 \times 32$, respectively. There must be a trade-off between the overlap and compression ratio. Experimental

results are given in the next section.

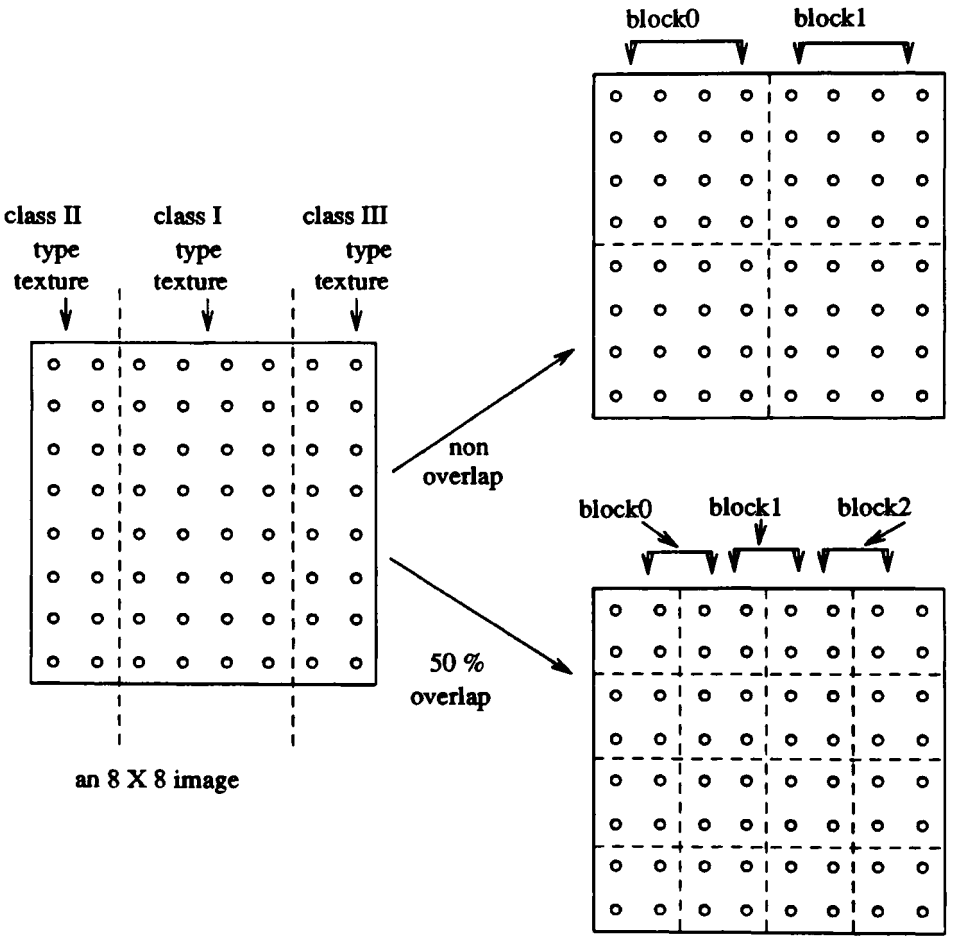


Figure 6.1: Comparison of the nonoverlap and overlap method. An image size and block size are 8×8 and 4×4 respectively.

6.4 The Experimental Results for the Nonoverlap and Overlap Segmentation

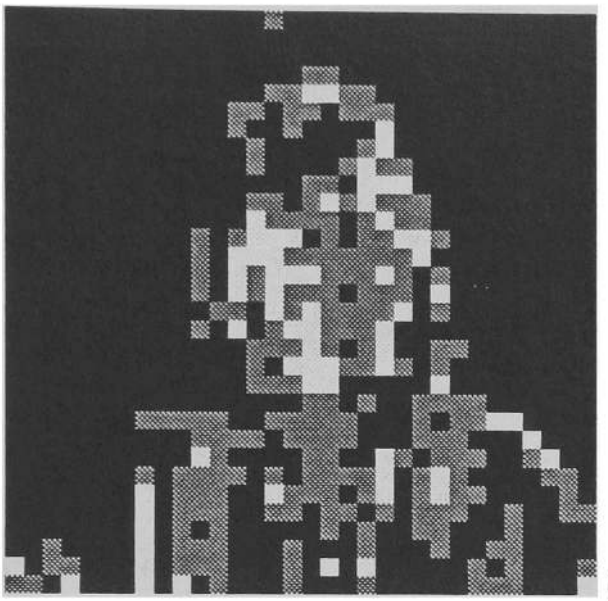
6.4.1 The Pixel Percentage for Each Class

In Section 5.3 of Chapter 5, we discussed how to determine the thresholds D_1 and D_2 of the fractal dimension. It was shown that the curve of the fractal dimensions belonging to the perceived constant intensity are approximately semi bell-shaped around the zero and the curves of the fractal dimensions belonging to class II and class III are approximatedly bell-shaped around their mean respectively. In addition, there are gaps between the curves of the fractal dimensions for each class. By these experiments, the value of D_1 should not be greater than the minimum of the fractal dimensions belonging to class II and the value of D_2 should lie between the means of class II and class III. Therefore, it is reasonable that the value of D_1 is chosen to be half of the sum of the maximum fractal dimension corresponding to the perceived constant intensity and the mininum fractal dimension corresponding to the smooth texture, $D_1 = \frac{2.021+2.049}{2} = 2.035$. The value of D_2 is chosen to be half of the sum of the maximum fractal dimension corresponding to the smooth texture and the minimum fractal dimension corresponding to the rough texture, $\frac{2.324+2.403}{2} = 2.363$. With $D_1 = 2.035$ and $D_2 = 2.363$, the class type images for the test images with 0, 50, and 75 % overlap are shown in Figures 6.2, 6.3, and 6.4. Blocks with fractal dimension less than D_1, between D_1 and D_2, and greater than D_2 are represented with an intensity value of 0, 127, and 255 respectively at all pixels in their blocks for visual purposes.

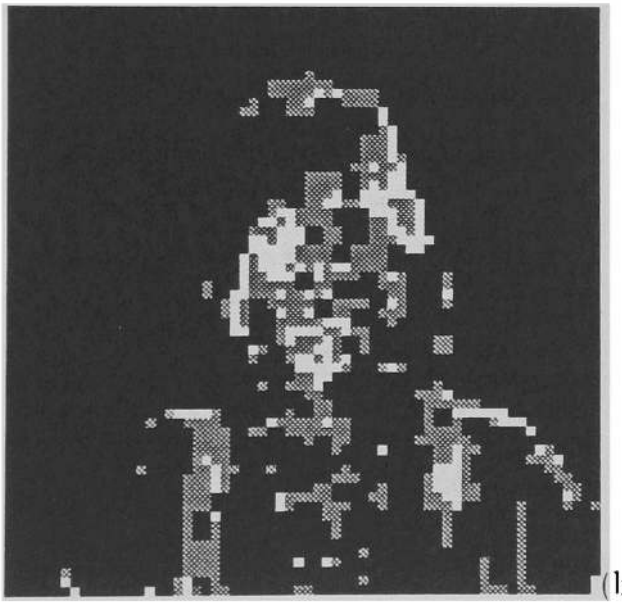
In Miss USA, almost all of the blocks in the large background and some blocks on the sweater correspond to class I, some blocks around the neck correspond to class II,

and blocks around the eyes, noses, and mouth correspond to class III. In Lena, most blocks on the background and many blocks on the black frame of the mirror and the within mirror correspond to class I, blocks on the cap, the chin, and the shoulder class II, and all the blocks on the feather correspond to class III. In House, all the blocks on the sky on the top and some blocks on the wall on the bottom correspond to class I. Some of the blocks on the lawn correspond to class I and II. Almost all of the blocks on the trees on the left and right, and the bushes, correspond to class III. In addition, blocks on the windows of the house correspond to class III since shadows of trees are on the house.

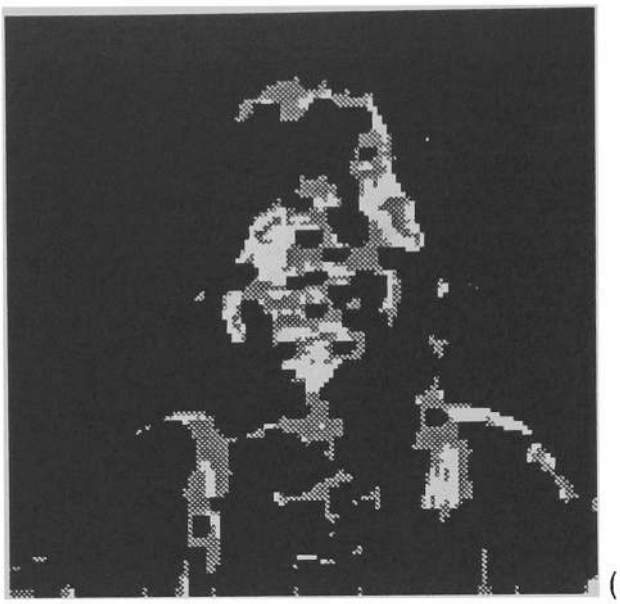
The percentage of pixels within each class in the test images is given in Tables 6.1, 6.2, and 6.3 with the different overlaps. When the overlap is 50 percent, the percentage of pixels belonging to class I is greatly increased. This means that the compression ratio for 50 percent overlap will be much higher than that for a zero percent overlap. When the overlap is 75 percent, the percentage of pixels belonging to class I is almost the same as for 50 percent overlap. Since the number of computation is increased for the 75 overlap method and computation rate is not significantly decreased, we determined that the best overlap is 50 percent.



(a)



(b)



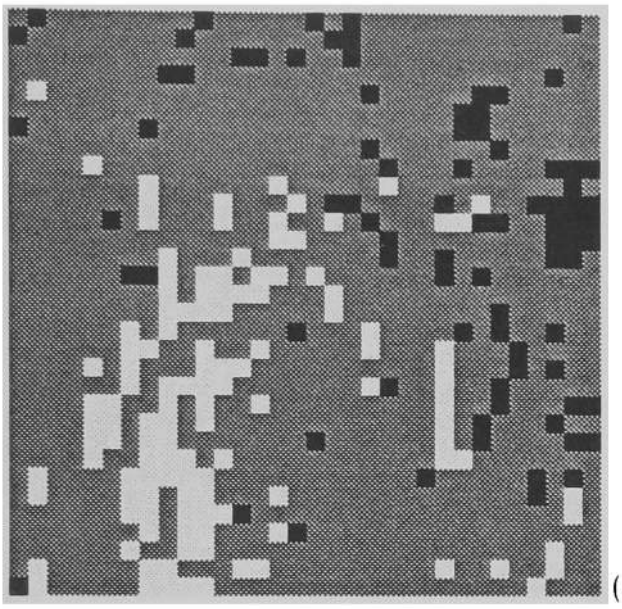
(c)

Figure 6.2: The class type images of Miss USA. (a) 0 percent overlap. (b) 50 percent overlap. (c) 75 percent overlap.

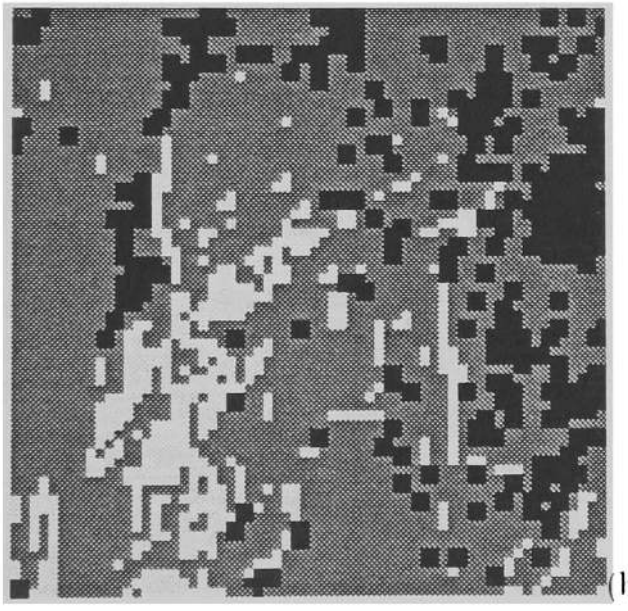
Table 6.1: Pixel percentage of each class in Miss USA

overlap	**class I %**	**class II %**	**class III %**
0 %	73.24	20.11	6.65
50 %	85.49	10.29	4.22
75 %	88.59	7.56	3.85

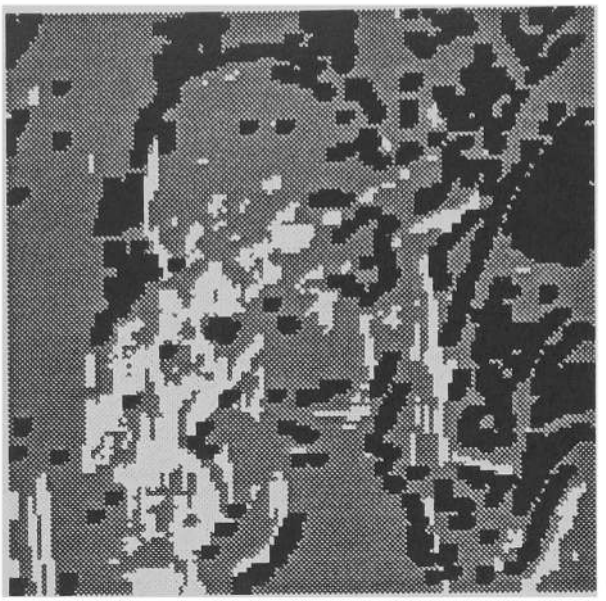
Table 6.1: Pixel percentage of each class in Miss USA



(a)



(b)



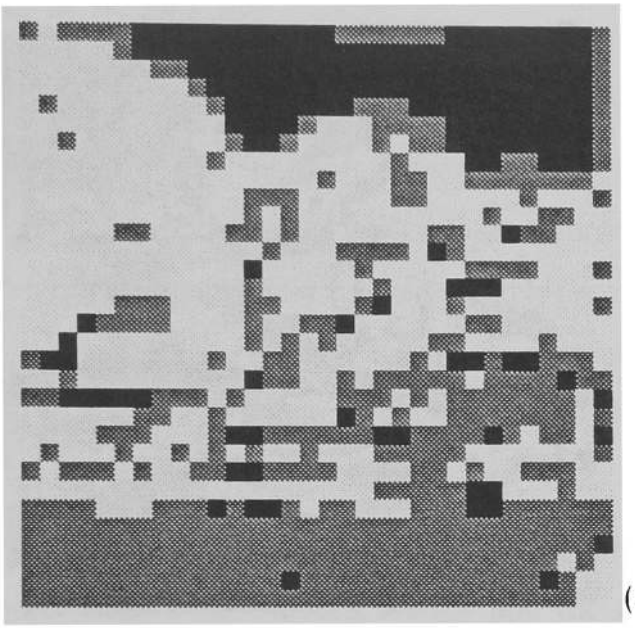
(c)

Figure 6.3: The class type images of Lena. (a) 0 percent overlap. (b) 50 percent overlap. (c) 75 percent overlap.

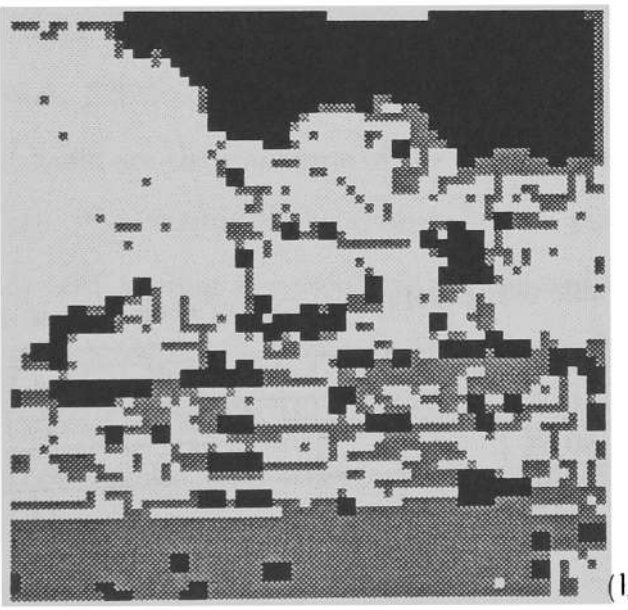
Table 6.2: Pixel percentage of each class in Lena

overlap	class I %	class II %	class III %
0 %	29.49	45.89	24.62
50 %	52.05	26.14	21.81
75 %	59.79	20.08	20.13

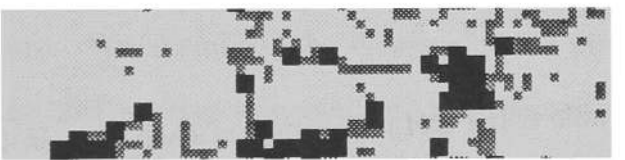
Table 6.2: Pixel percentage of each class in Lena

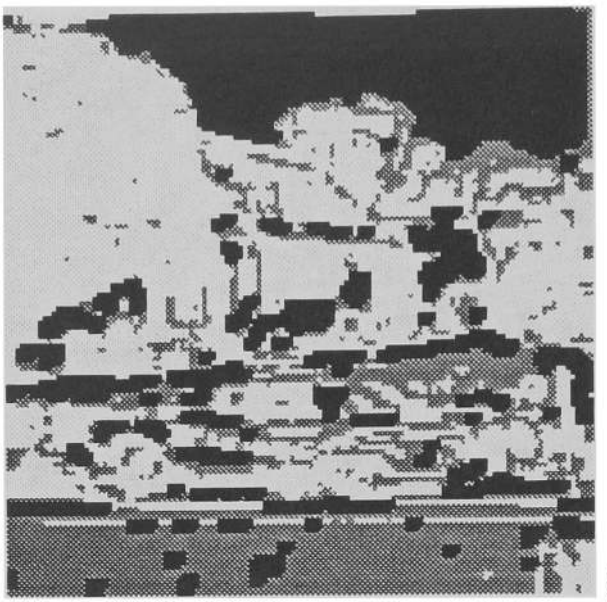


(a)



(b)





(c)

Figure 6.4: The class type images of House. (a) 0 percent overlap. (b) 50 percent overlap. (c) 75 percent overlap.

Table 6.3: Pixel percentage of each class in House

overlap	class I %	class II %	class III %
0 %	16.99	37.89	45.12
50 %	24.26	30.26	45.48
75 %	29.48	26.37	44.15

Table 6.3: Pixel percentage of each class in House

6.4.2 Variability of D_1 and D_2

In this subsection, we examine the percentage of pixels within each class as a function of D_1 and D_2. Higher values of D_1 force more pixels into the perceived constant intensity class and generally there will be both a larger number of regions belonging to the perceived constant intensity class and more pixels per region. Lower values of D_2 produce more regions belonging to the rough texture class. We investigate the changes of the percentage in pixels within each class for the following values of D_1 and D_2. $D_1 = 2.035$ is the half of the sum of the maximum of the curve corresponding to the perceived constant intensity, $D_2 = 2.363$ is the half of the sum of the maximum of the curve corresponding to the smooth texture and the minimum of the curve corresponding to the rough texture, 2.021 is the maximum of the curve corresponding to the perceived constant intensity, 2.049 is the minimum of the curve corresponding to the smooth texture, 2.324 is the maximum of the curve corresponding to the smooth texture, and 2.408 are the minimum of the curve corresponding to the rough texture in Figure 5.16. We examined the changes of the pixels of each class with 50% overlap since the 50% overlap was used for the proposed coding algorithm. The percentage of the pixels within each class with variation in D_1 and D_2 are given in Tables 6.4 to 6.9. These results show that higher values of D_1 produce more pixels belonging to the perceived constant intensity and lower values of D_2 produces more pixels belonging to the rough texture.

Table 6.4: Percentage of the pixels in each class in Miss USA with 50% overlap, D_1 variable, and $D_2 = 2.363$.

D_1	class I %	class II %	class III %
2.021	76.70	18.40	4.88
2.035	85.49	10.27	4.22
2.049	90.62	5.88	3.49

Table 6.5: Percentage of the pixels in each class in Miss USA with 50% overlap, $D_1 = 2.035$, and D_2 variable.

D_2	class I %	class II %	class III %
2.324	85.49	10.27	4.22
2.363	85.49	10.27	4.22
2.408	85.49	10.40	4.10

Table 6.6: Percentage of the pixels in each class in Lena with 50% overlap, D_1 variable, and $D_2 = 2.363$.

D_1	class I %	class II %	class III %
2.021	30.51	44.79	24.68
2.035	52.05	26.14	21.80
2.049	65.89	15.23	18.87

Table 6.7: Percentage of the pixels in each class type in Lena with 50% overlap, $D_1 = 2.035$, and D_2 variable.

D_2	class I %	class II %	class III %
2.324	52.05	26.14	21.80
2.363	52.05	26.14	21.80
2.408	52.05	26.63	21.31

Table 6.8: Percentage of the pixels in each class in House with 50% overlap, D_1 variable, and $D_2 = 2.363$.

D_1	class I %	class II %	class III %
2.201	17.45	35.05	47.48
2.035	22.58	31.42	45.99
2.049	35.18	21.72	43.09

Table 6.9: Percentage of the pixels in each class in House with 50% overlap, $D_1 = 2.035$, and D_2 variable.

D_2	class I %	class II %	class III %
2.324	22.58	31.03	46.38
2.363	22.58	31.42	45.99
2.408	22.58	31.88	45.53

6.4.3 The Boundary Coding

In this subsection we present the results of compressing the boundaries of the regions in the segmented image using the three different lossless coding techniques; runlength code, crack code, and arithmetic code. The segmented images were obtained with $D_1 = 2.035$ and $D_2 = 2.363$ and 50% overlap. Information about the number of segments and the boundary points is summarized in Table 6.10. The number of the bits to represent the boundaries is given in three different coding techniques in Table 6.11. These results show that the arithmetic code achieves lower bit rate than the runlength code and the crack code. Therefore, we will use the arithmetic code in our codec.

Table 6.10: Summary of the numbers of the total segments and the boundary points using $D_1 = 2.035$ and $D_2 = 2.363$.

image	number of total segments	number of boundary points
Miss USA	298	1908
Lena	699	3801
House	289	1984

Table 6.11: Summary of bits to represent the boundary using the three different coding.

image	number of bits for runlength code	number of bits for crack code	number of bits for arithmetic code
Miss USA	3636	1463	370
Lena	7214	2915	737
House	3781	1521	384

6.4.4 Coding of the Constant Regions

In this subsection we present the results of encoding of the constant regions using the three lossless coding techniques; runlength code, crack code, and arithmetic code. To encode the regions belonging to the perceived constant intensity, their mean values are converted into an $8 \times N$ binary. Each mean value is represented by 8 bits and there are N segments belonging to the perceived constant regions. The results are shown in Tables 6.12 and 6.13. The total number of the regions and the number of constant regions are given in Table 6.12 while the number of bits to represent the constant regions for each coding technique is summarized in Table 6.13. These results again show that the arithmetic code is better than the runlength code and the crak code.

Table 6.12: Summary of the numbers of the total segments and the perceived constant regions using $D_1 = 2.035$ and $D_2 = 2.363$.

image	number of total segments	number of constant regions
Miss USA	298	242
Lena	699	576
House	289	199

Table 6.13: Summary of the number of bits to represent the constant region.

image	number of bits for runlength code	number of bits for crack code	number of bits for arithmetic code
Miss USA	3690	1484	375
Lena	8782	3534	879
House	3034	1889	308

6.4.5 Coding of the Smooth and Rough Textural Regions

In this subsection we first examine the SSE's of 1-D and 2-D polynomial functions to determine which dimensional polynomial functions is used to represent textural regions, and then the results of compressing the smooth and rough regions by 1-D dimensional polynomial functions are presented. In the 2-D case, the coefficients a_i are chosen to minimized the SSE of the approximation

$$\sum_i \sum_j (g(i,j) - z(i,j))^2$$

where $g(i,j)$ is the 2-D intensity field of the original image and $z(i,j)$ is the 2-D polynomial function. We examine how well the 1-D and 2-D polynomial functions approximate the original image data. First consider the 2-D case. A 32×32 subimage around the trees in House was chosen to represent a rough textural region and a 32×32 subimage around the chin in Miss USA was chosen to represent a smooth textural region. Recall from Chapter 5, the trees in the House and the chin in Miss USA were classified as rough and smooth textural classes, respectively. The zero-order (plateau), first-order (tilted plane), and second-order (quadratic) representations for the rough textural and the smooth textural subimages are given in Figures 6.5 and 6.6, respectively along with the original subimages, the zero-order model, the first-order model, and the second-order model. The SSE's for each representation are given in Tables 6.14 and 6.15 below the figures. As shown in the tables, the SSE for each order is so large that the representation using lower order 2-D polynomial functions is not likely to be useful in the proposed compression system. An alternative is to use higher order 2-D polynomial functions or 1-D lower order polynomial functions. However, the computation time to minimize a higher order 2-D polynomial functions

is very expensive. Therefore, the 1-D polynomial function are considered.

In the 1-D case, the same original images were used. The zero-order (plateau), first-order (tilted plane), and second-order (quadratic) representations for the rough textural and the smooth textural subimages are given in Figures 6.7 and 6.8, respectively along with the original subimages, the zero-order model, the first-order model, and the second-order model. The SSE's for each representation are given in Tables 6.16 and 6.17 below the figures. As shown in the tables, the SSE in the 1-D case is less than one in the 2-D. It means that the 1-D functions represent the smooth and the rough textures better than the 2-D functions and with fewer computation and lower bit rates.

In the 1-D ase, we adjust the amount of error tolerated between the original image data and the modeled image data. A lower error can be chosen for class II than class III. Information about the number of regions and the number of bits to represent these regions is summarized in Table 6.18 using the 1-D first-order polynomial function. The first order polynomial functions were used because the SSE of the first order polynomial function is not much greater than the SSE of one of the second-order polynomial function The sum of deviation between the original image and the modeled image were used to calculate the amount of error. The thresholds for class II and class III are 25 and 60, respectively. The largest values of the thresholds which produced the good image quality by experiments were chosen because the largest thresholds give the lower bit rates. Further analysis of the codec performance as a function of the choice of thresholds is given in Subsection 6.4.7.

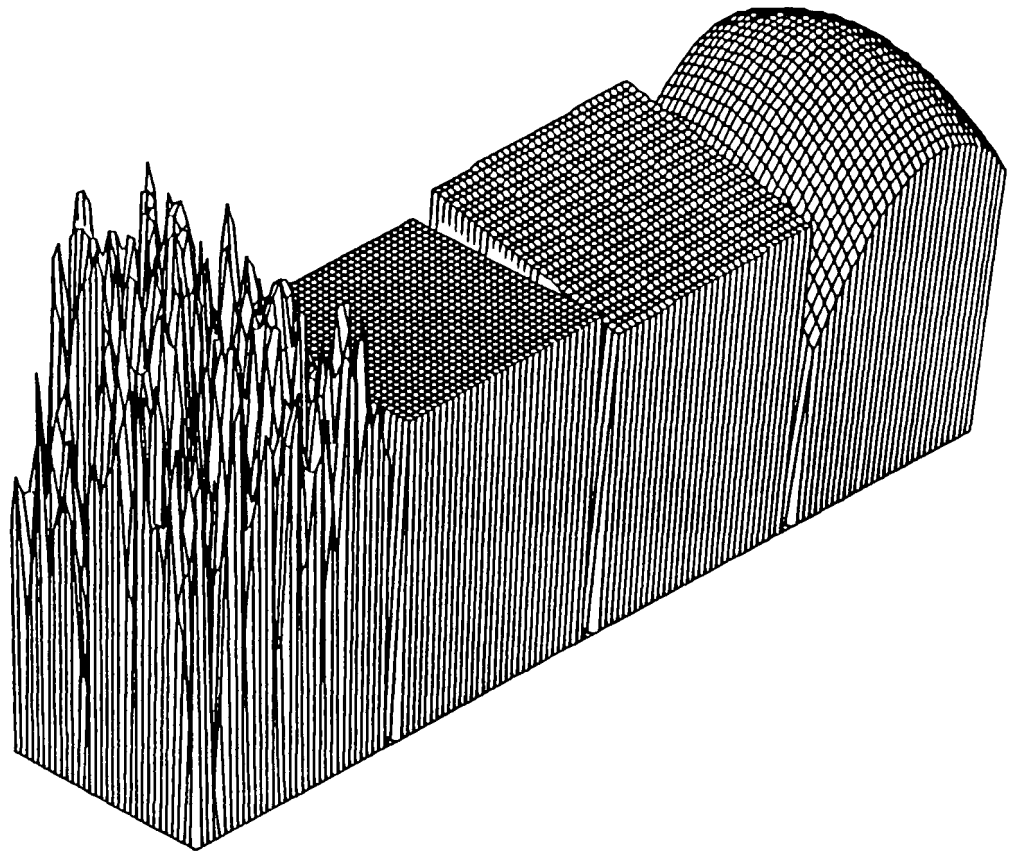


Figure 6.5: Modeling a rough texture region using a 2-D polynomial. From left to to right are the 32×32 tree subimage in House; the original, the zero-order model, the first-order model, and the second-order model.

Table 6.14: The sum of squared error (SSE) values for each model.

order	zero	first	second
SSE	1449139	1326727	1271754

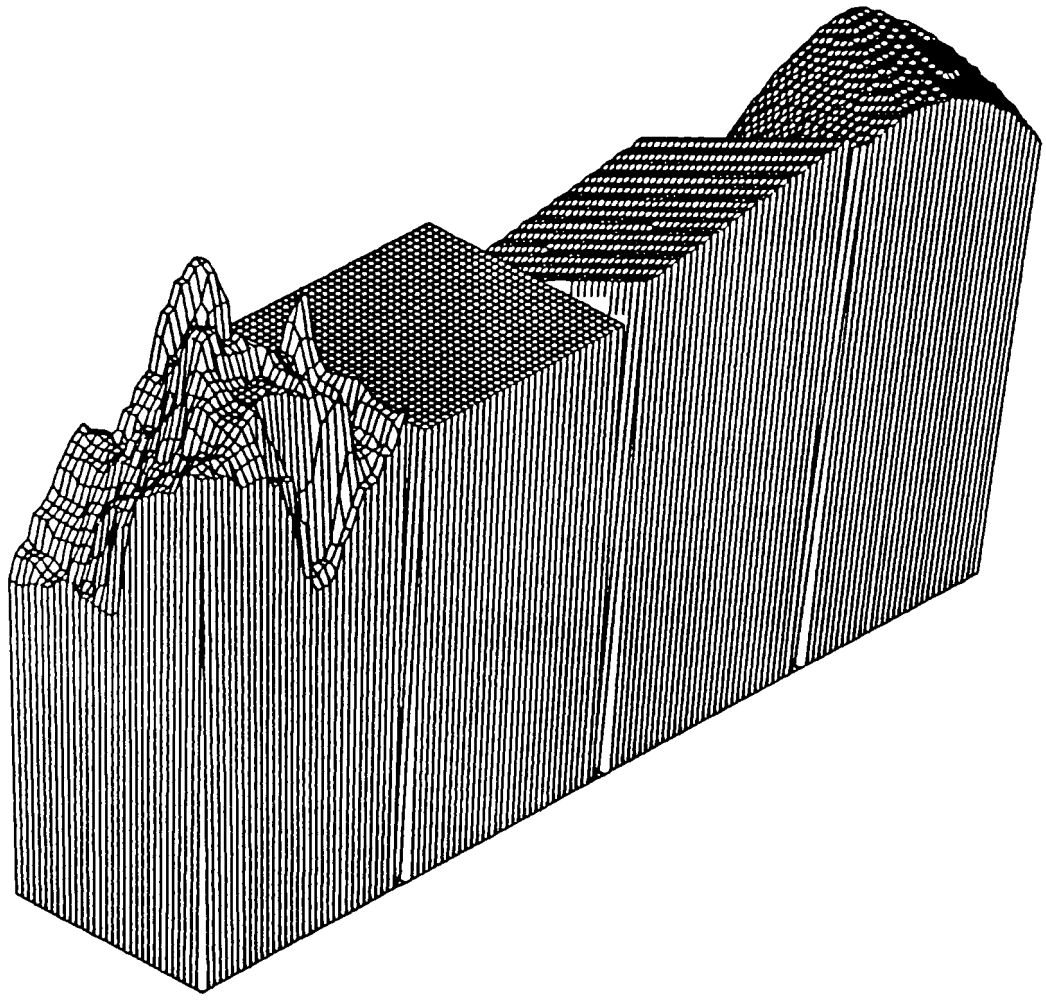


Figure 6.6: Modeling a smooth texture region using a 2-D polynomial. From left to to right are the 32×32 chin subimage in Miss USA; the original, the zero-order model, the first-order model, and the second-order model.

Table 6.15: The sum of squared error (SSE) values for each model.

order	zero	first	second
SSE	371751	250527	180449

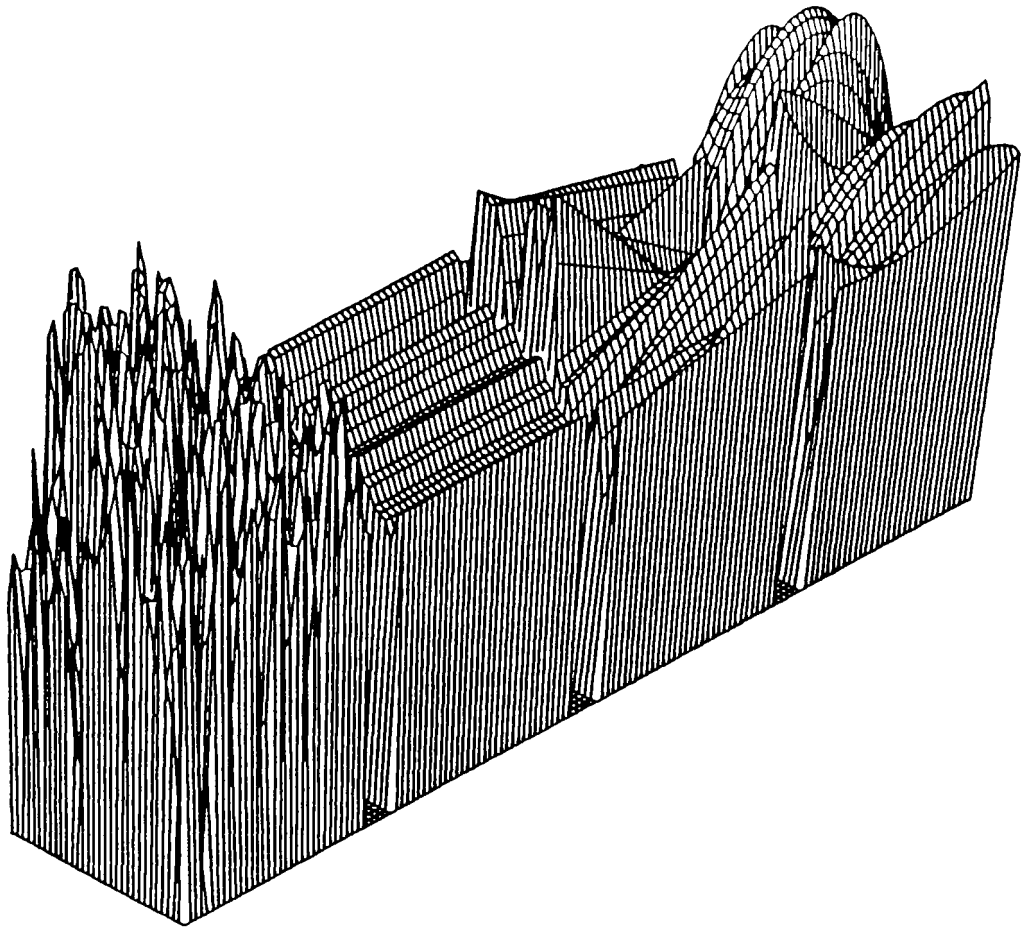


Figure 6.7: Modeling a rough texture region using a 1-D polynomial. From left to to right are the 32×32 tree subimage in House; the original, the zero-order model, the first-order model, and the second-order model.

Table 6.16: The sum of squared error (SSE) values for each model.

order	zero	first	second
SSE	1365865	1146474	953608

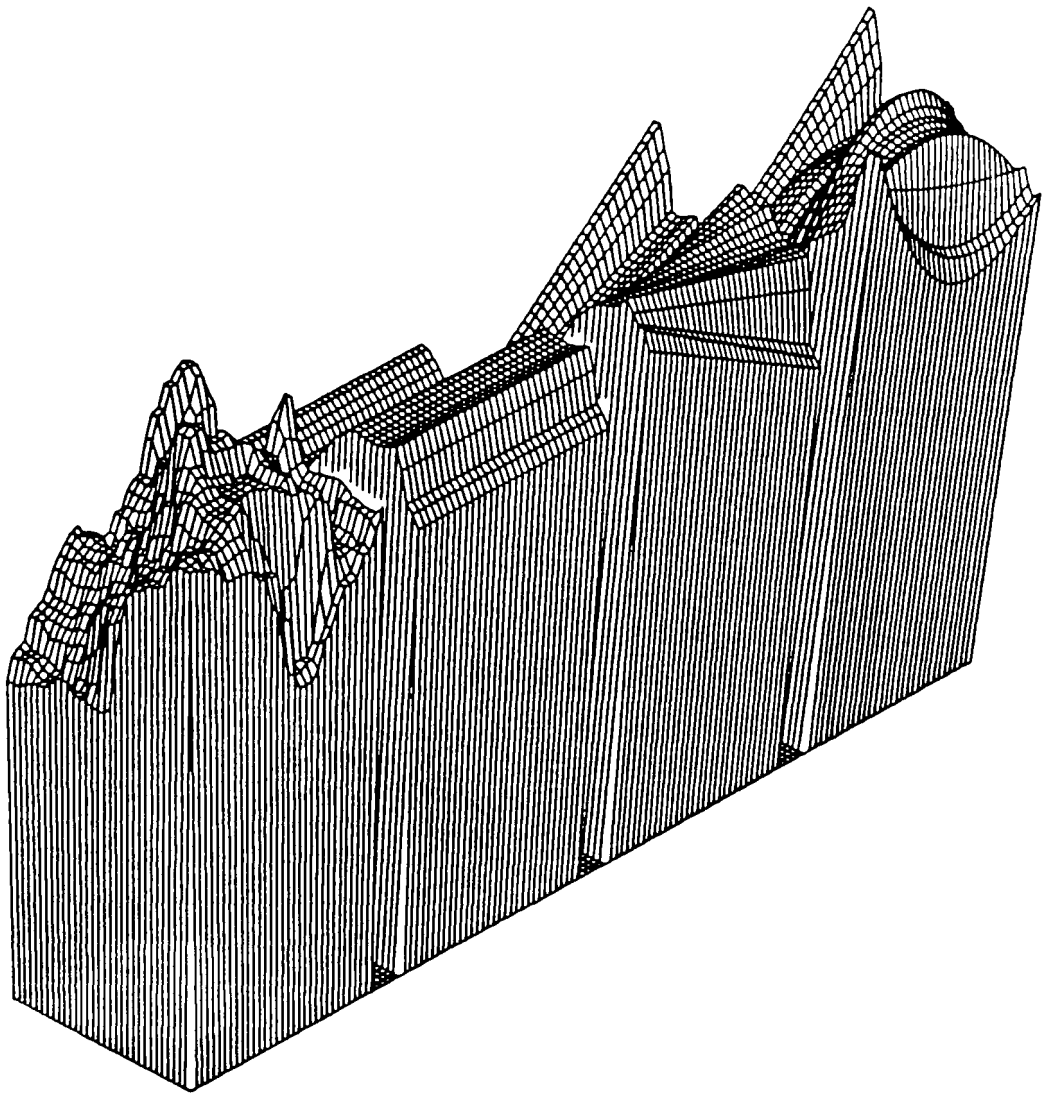


Figure 6.8: Modeling a smooth texture region using a 1-D polynomial. From left to to right are the 32×32 chin subimage in Miss USA; the original, the zero-order model, the first-order model, and the second-order model.

Table 6.17: The sum of squared error (SSE) values for each model.

order	zero	first	second
SSE	217724	113491	79801

Table 6.18: Summary of the numbers of the segments in the smooth and rough textural regions and the number of bits to represent those regions using a 1-D polynomial. The polynomial coefficients were encoded using the arithmetic code.

image	number of segments	number of bits for these regions
Miss USA	56	3127
Lena	123	14320
House	90	28183

6.4.6 Bit Rate Computation

To compute the total number of bits required to transmit an image, the three numbers of bits calculated for the boundary, constant region, smooth/rough texture are added. The bit rate is the sum of the number of bits divided by the total bits of an image. The bit rate, BR is given by

$$BR = \frac{SP + CP + RP}{256 \times 256 = 65536}$$

where SP is the number of bits required for encoding of the boundaries, CP is the number of required for encoding of the constant regions, and RP is the number of required for encoding of the smooth and rough texture regions. For example, we calculate the bit rates of test images with the following paramemters: the thresholds of the smooth and the rough texture classes are 25 and 60 respectively, $D_1 = 0.035$ and $D_2 = 0.363$. The number three numbers of bits calculated for the boundary, constant region, and smooth/rough texture as given in Tables 6.10, 6.12, and 6.18 are added. We see that Miss USA requires 370+375+3127 = 3872 bits for encoding, Lena requires 737+879+14320 = 15936 bits, and House requires 384+308+28183 = 28875 bits for this coding method. The compression ratios, BR for the test images are 3872/65536 = 0.06, 15936/65536 = 0.24, and 28875/65536 = 0.44 bit per pixel, respectively.

The decoded images for each test image are given in Figure 6.9. The image quality of the decoded images are good with these bit rates as shown in Figure 6.9. Through our experiment, our segmentation-based image compression method works well for a wide variety of images including a natural image with highly textural areas referred as House.

(a)

(b)

(c)

Figure 6.9: The decoded images of the test images with $D_1 = 2.035$ and $D_2 = 2.363$. (a) The decoded image of Miss USA. (b) The decoded image of Lena. (c) The decoded image of House.

6.4.7 Performance Evaluation of the CODEC

In this subsection, we evaluate the entire image coding system as parameters are varied. The parameters are the fractal dimensions D_1 and D_2, and the model error thresholds for classes II and III. First, the compression ratios were computed with only D_1 varying and the remaining parameters fixed. The thresholds for the smooth and rough regions are 25 and 60 respectively. Second, the compression ratios were obtained with D_2 varying and the remaining parameters fixed. Third, the compression ratios were computed with the thresholds for the smooth and rough regions varying and D_1, D_2 fixed. Signal-to-ratio (SNR) values are computed in each case. SNR is defined as:

$$SNR = 10log_{10}[\frac{\sigma^2_{image}}{\sigma^2_{error}}]$$

where

$$\sigma^2_{image} = \frac{1}{P^2}\sum_{i=1}^{P}\sum_{j=1}^{P}[g_{ij} - \mu]^2$$

$$\sigma^2_{error} = \frac{1}{P^2}\sum_{i=1}^{P}\sum_{j=1}^{P}[g_{ij} - \hat{g}_{ij} - \mu_\epsilon]^2$$

where g_{ij} the input image, $\hat{g}_{ij}$ the decoded image, μ the mean of g_{ij} and μ_ϵ the mean of the error signal, g_{ij} - $\hat{g}_{ij}$. P is the size of the image ($P \times P$) and i, j are indices to the image array.

The bit rate, the SNR, and the number of segments of each class are summarized in Tables 6.19 to 6.21 with D_1 variable and D_2 fixed. Some plots of the SNR with D_1 variable and D_2 fixed are given in Figures 6.10 and 6.11. In the tables, NT, N1, N2, and N3 represent the total number of segments, the number of segments

belonging to class I, the number of segments belonging to class II, and the number of segments belonging to class III, respectively. These results show that higher values of D_1 produce lower bit rates and lower SNR's. That is why lower values of D_1 increase the number of segments belonging to the perceived constant intensity as shown in the tables.

The bit rate, the SNR, and the number of segments of each class are summarized in Tables 6.22 to 6.24 with D_1 fixed and D_2 variable. A plot of the SNR with D_1 fixed and D_2 variable for House is given in Figure 6.12. These results show that lower values of D_2 produce lower bit rates and lower SNR's. That is why lower values of D_2 increase the number of segments belonging to class III as shown in the tables. In the proposed coding system, regions belonging to class III are less emphasized than regions belonging to class II because of the sensitivity of the HVS.

To evaluate the performance with variations in the amount of error for class II and class III, a summary of information about the bit rate, the SNR, and the number of segments for each class with variation in amount of error is given Tables 6.25 to 6.30. $D_1 = 2.035$ and $D_2 = 2.363$ were used. A lower amount of error for class class II and III produces higher bit rates and higher SNR's while the number of segments for each class remains the same. To increase the image quality of the regions belonging to class II, a lower error for class II is chosen. If we wish to increase the image quality of the regions belonging to class III, a lower error for class III is also chosen. Since regions belonging to class II are sensitive to the HVS, amount of error for class II should, in general, be chosen to lower than class III.

Typical reconstructed images at rates of 1 bit/pixel and 0.2 bit/pixel for all test images are given in Figures 6.13 to 6.15. The reconstructed image at rates of 1 bit/pixel have a very good image quality. This means that the visual loss at these

rates is ignored. The reconstructed images at rates of 0.2 bit/pixel have a good image quality except Lena which contains block boundary artifacts. Since the proposed coding system is based on blocks not pixels, when a higher value of D_1 is chosen, more blocks are classified as the perceived constant intensity and artifacts are more noticeable.

Table 6.19: Summary of coding information in Miss USA. D_1 is variable and D_2 is fixed to 2.363.

D_1	rate, bit/pixel	SNR	NT	N1	N2	N3
2.001	0.15	18.39	73	31	12	30
2.005	0.15	18.39	73	31	12	30
2.01	0.13	18.03	122	81	10	31
2.015	0.11	17.31	171	117	19	35
2.020	0.09	16.64	211	153	25	33
2.025	0.08	16.27	230	177	28	25
2.030	0.07	16.01	252	202	28	22
2.035	0.06	15.90	298	242	31	25
2.040	0.05	15.56	313	262	28	23
2.050	0.04	15.20	358	306	30	22
2.060	0.03	14.73	376	332	27	17
2.080	0.02	14.29	398	377	10	11
2.10	0.02	14.11	401	395	0	6
2.15	0.02	14.11	401	395	0	6

Table 6.20: Summary of coding information in Lena. D_1 is variable and D_2 is fixed to 2.363.

D_1	rate, bit/pixel	SNR	NT	N1	N2	N3
2.001	0.60	17.68	60	2	12	46
2.005	0.59	17.54	105	42	16	47
2.01	0.54	16.77	220	152	21	47
2.015	0.48	15.66	327	216	22	59
2.020	0.45	15.05	424	329	34	61
2.025	0.39	14.51	528	421	51	56
2.030	0.34	14.13	622	503	65	51
2.035	0.30	13.62	699	576	76	47
2.040	0.27	13.21	774	657	74	43
2.050	0.22	12.60	904	793	77	31
2.060	0.18	12.15	968	859	79	30
2.080	0.16	12.01	988	883	75	30
2.10	0.12	11.90	991	889	72	30
2.15	0.10	11.81	991	889	72	30

Table 6.21: Summary of coding information in House. D_1 is variable and D_2 is fixed to 2.363.

D_1	rate, bit/pixel	SNR	NT	N1	N2	N3
2.001	0.55	12.89	60	7	39	14
2.005	0.54	12.86	64	14	36	14
2.01	0.53	12.79	92	35	41	16
2.015	0.52	12.67	136	67	54	15
2.020	0.51	12.53	183	108	57	18
2.025	0.50	12.35	212	136	60	16
2.030	0.49	12.21	255	172	67	16
2.035	0.48	12.12	289	199	73	17
2.040	0.46	11.74	357	266	74	17
2.06	0.42	11.47	436	341	74	18
2.08	0.29	10.19	741	675	45	21
2.10	0.23	9.46	876	841	3	32
2.15	0.23	9.46	876	841	3	32

Table 6.22: Summary of coding information in Miss USA. D_1 is fixed to 2.035 and D_2 is variable.

D_2	rate, bit/pixel	SNR	NT	N1	N2	N3
2.21	0.06	15.90	298	242	31	25
2.26	0.06	15.90	298	242	31	25
2.31	0.06	15.90	298	242	31	25
2.36	0.06	15.90	298	242	31	25
2.41	0.06	15.94	297	241	30	26
2.46	0.06	15.96	296	241	29	26
2.51	0.06	15.99	294	241	28	25
2.56	0.06	16.05	292	240	26	26
2.61	0.06	16.06	286	240	24	22
2.66	0.07	16.10	281	240	21	20
2.71	0.07	16.12	271	240	19	12
2.76	0.07	16.14	269	240	19	10

Table 6.23: Summary of coding information in Lena. D_1 is fixed to 2.035 and D_2 is variable.

D_2	rate, bit/pixel	SNR	NT	N1	N2	N3
2.21	0.30	13.62	699	576	76	47
2.26	0.30	13.62	699	576	76	47
2.31	0.30	13.62	699	576	76	47
2.36	0.30	13.62	699	576	76	47
2.41	0.31	13.63	700	576	74	50
2.46	0.32	13.64	695	576	71	48
2.51	0.33	13.72	682	576	61	45
2.56	0.35	13.79	668	576	51	41
2.61	0.36	13.86	658	575	43	40
2.66	0.38	13.95	645	575	32	38
2.71	0.40	14.06	645	576	29	40
2.76	0.42	14.16	638	578	26	34

Table 6.24: Summary of coding information in House. D_1 is fixed to 2.035 and D_2 is variable.

D_2	rate, bit/pixel	SNR	NT	N1	N2	N3
2.21	0.47	12.11	289	199	73	13
2.26	0.47	12.11	289	199	77	13
2.31	0.47	12.11	287	199	75	13
2.36	0.48	12.12	289	199	73	17
2.41	0.49	12.16	292	199	71	22
2.46	0.49	12.16	292	199	71	22
2.51	0.50	12.21	288	198	69	21
2.56	0.51	12.30	280	198	63	19
2.61	0.54	12.44	281	198	53	30
2.66	0.58	12.67	279	198	41	49
2.71	0.62	12.90	283	198	40	45
2.76	0.67	13.20	289	198	36	55

Table 6.25: Summary of coding information in Miss USA. $D_1 = 2.035$, $D_2 = 2.363$, $TH_{er2} = 60$ and TH_{er1} is variable, where TH_{er1} and TH_{er2} are the thresholds of regions beloinging to the smooth and the rough textures respectively.

TH_{er1}	rate, bit/pixel	SNR
5	0.11	16.19
10	0.08	16.12
15	0.07	16.05
20	0.06	15.98
25	0.06	15.90
30	0.06	15.84
35	0.05	15.77
40	0.05	15.71
45	0.04	15.69

Table 6.26: Summary of coding information in Lena. $D_1 = 2.035$, $D_2 = 2.363$, $TH_{er2} = 60$ and TH_{er1} is variable.

TH_{er1}	rate, bit/pixel	SNR
5	0.51	13.78
10	0.40	13.75
15	0.36	13.71
20	0.32	13.66
25	0.30	13.62
30	0.28	13.57
35	0.27	13.52
40	0.26	13.46
45	0.25	13.42

Table 6.27: Summary of coding information in House. $D_1 = 2.035$, $D_2 = 2.363$, $TH_{er2} = 60$ and TH_{er1} is variable.

TH_{er1}	rate, bit/pixel	SNR
5	0.80	12.28
10	0.63	12.24
15	0.55	12.20
20	0.51	12.16
25	0.48	12.12
30	0.46	12.09
35	0.44	12.06
40	0.43	12.03
45	0.41	12.00

Table 6.28: Summary of coding information in Miss USA. $D_1 = 2.035$, $D_2 = 2.363$, $TH_{er1} = 25$ and TH_{er2} is variable.

TH_{er2}	rate, bit/pixel	SNR
25	0.07	16.22
30	0.06	16.18
40	0.06	16.09
50	0.06	16.01
60	0.06	15.90
70	0.05	15.85
80	0.05	15.76
90	0.05	15.73
100	0.50	15.66

Table 6.29: Summary of coding information in Lena. $D_1 = 2.035$, $D_2 = 2.363$, $TH_{er1} = 25$ and TH_{er2} is variable.

TH_{er2}	rate, bit/pixel	SNR
25	0.35	14.50
30	0.34	14.38
40	0.32	14.12
50	0.31	13.86
60	0.30	13.62
70	0.29	13.37
80	0.28	13.17
90	0.28	12.99
100	0.27	12.85

Table 6.30: Summary of coding information in House. $D_1 = 2.035$, $D_2 = 2.363$, $TH_{er1} = 25$ and TH_{er2} is variable.

TH_{er2}	rate, bit/pixel	SNR
25	0.67	15.91
30	0.63	15.20
40	0.56	13.92
50	0.52	12.95
60	0.48	12.12
70	0.45	11.48
80	0.42	11.03
90	0.41	10.65
100	0.40	10.31

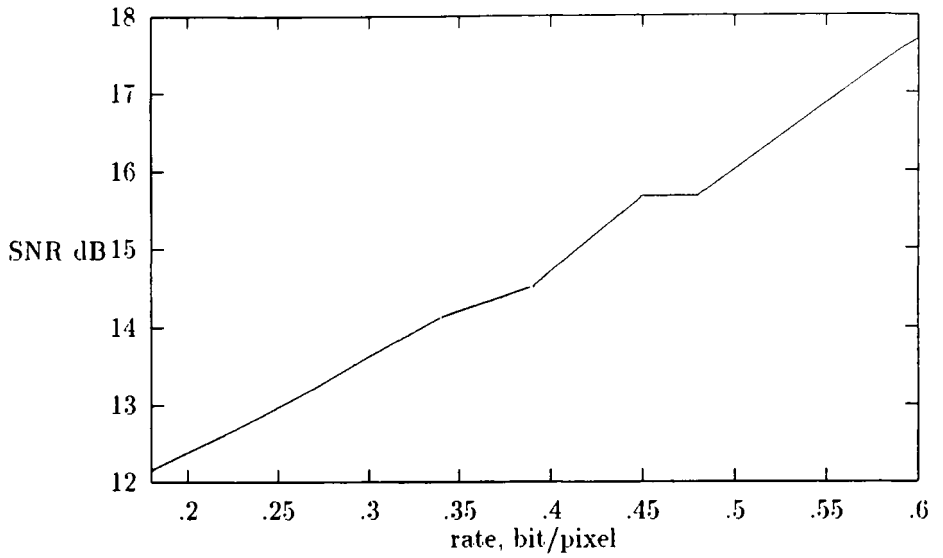


Figure 6.10: Plot of SNR versus rate, bit/pixel for Lena. D_1 is variable and D_2 is fixed to 2.363.

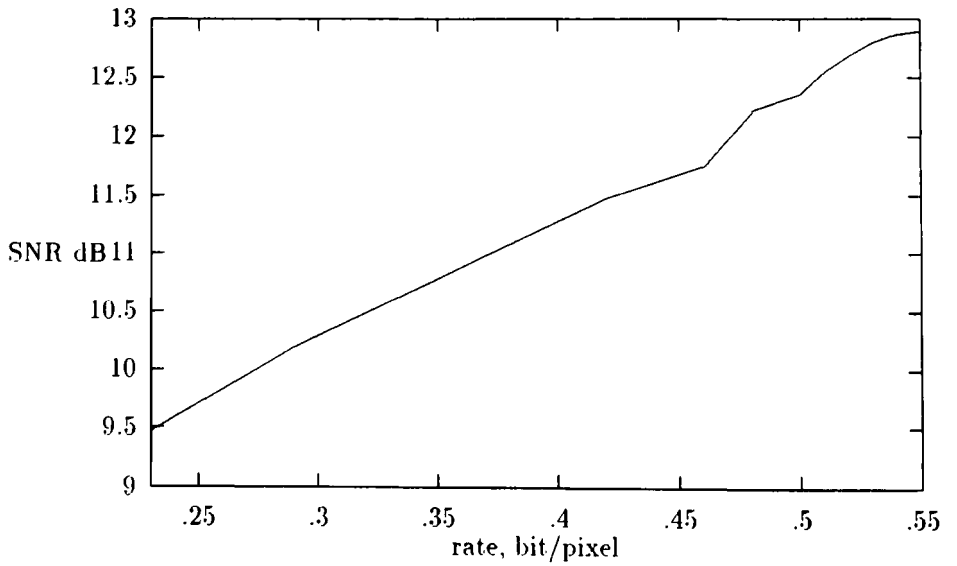


Figure 6.11: Plot of SNR versus rate, bit/pixel for House. D_1 is variable and D_2 is fixed to 2.363.

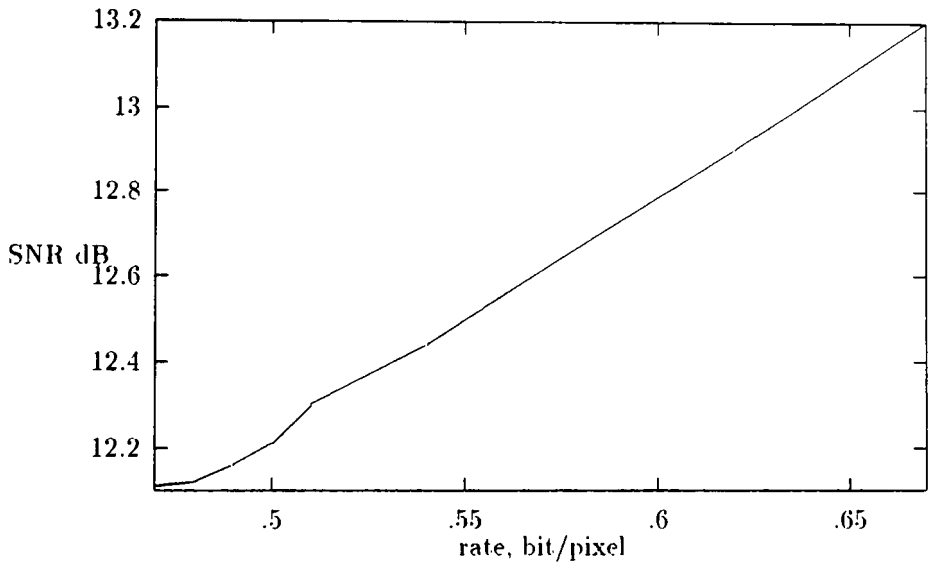
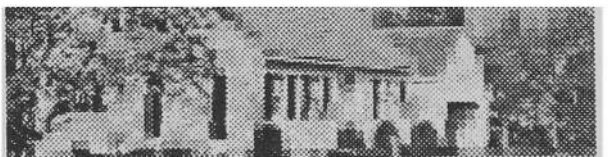


Figure 6.12: Plot of SNR versus rate, bit/pixel for House. D_1 is fixed 2.035 and D_2 is variable.

Figure 6.13: The reconstructed images for Miss USA. The images on the top and the bottom are coded at 1.0 bit/pixel with $D_1 = 2.001$ and $D_2 = 2.76$ and 0.2 bit/pixel with $D_1 = 2.005$ and $D_2 = 2.36$ respectively.

Figure 6.14: The reconstructed images for Lena. The images on the top and the bottom are coded at 1.0 bit/pixel with $D_1 = 2.005$ and $D_2 = 2.71$ and 0.2 bit/pixel with $D_1 = 2.050$ and $D_2 = 2.361$ respectively.

Figure 6.15: The reconstructed images for House. The images on the top and the bottom are coded at 1.0 bit/pixel with $D_1 = 2.005$ and $D_2 = 2.71$ and 0.2 bit/pixel with $D_1 = 2.06$ and $D_2 = 2.41$ respectively.



6.5 Conclusion

These results indicate that, using the texture-based segmentation-based image compression system, compression ratios in the neighborhood of 0.08 to 0.3 bpp are attainable with good image quality. These ratios are almost the same as those achieved by a segmentation-based compression method using flat segments [71]. However, the proposed compression technique produces better image quality and is quite useful for a wider variety of images. This is because our segmentation compression method was developed using texture features as well as gray levels. Specifically, our technique works well for images with highly textured areas, while previous compression techniques were not useful for those images.

In addition to working well for a wide variety of images, there are also other advantages to using the method. One advantage is that the block-by-block method for segmentation-based compression is a more parallel approach than the pixel-by-pixel one. This allows for a fast implementation for the coding algorithm. The algorithm based on the pixel-by-pixel method is not conducive to being done in a parallel fashion.

Another advantage of our segmentation-based compression is that it allows more readily for compression ratio/image quality trade-offs. By varying parameters, the compression ratios can be easily controlled. For example, higher values of D_1 give more regions belonging to the perceived constant intensity in an image. Lower values of D_2 produce more regions belonging to rough textural class.

7

Conclusions

In this report we proposed a new segmentation-based image compression technique using fractals and properties of the HVS, which achieves compression in the neighborhood of 0.08 to 0.3 bpp. The segmentation is good and conforms to the human perception of roughness. The proposed method works well for a variety of images.

The proposed compression technique is different in several key ways from other segmentation-based image compression schemes. First, an image is segmented into regions with respect to perceptual roughness. Regions are classified as belonging to one of three classes; perceived intensity value, smooth texture, and rough texture. Thus the segmentation method takes advantage of properties of the HVS to achieve the image compression system with higher compression and small visual loss. An arithmetic code was used to encode the boundaries and the means of the regions belonging to perceived constant intensity value. A polynomial coding technique was applied to regions belonging to the smooth and rough texture. Second, an overlap method in the segmentation algorithm was proposed to improve a nonoverlap method. The overlap method produces the number of pixels belonging to the perceived constant intensity. This results in higher compression. Third, our compression system is developed based on texture characteristics. Image segments are represented by the degree of roughness

using fractals. Other segmentation-based techniques have typically represented the image segments as the mean gray values within their segments. Their applications are very limited since they are not useful for images with textural areas.

There are many aspects of the work presented here that offer avenues for further research. First, it is possible to improve our compression technique. A better technique for coding each class could be found. In relation to this, a higher order polynomial or adaptive coding technique can be used. In addition, regions belonging to rough class can be coded using an iterated function system developed by Barnsley [2]. Second, other segmentation techniques such as variable block size or quad-tree segmentation can be used. Further work needs to be done to verify which segmentation works best. Third, A postprocessing filter can be used to reduce the artifects. Fourth, our technique can be extended to image sequences. This is why a fractal dimension or segmentation is used as the frame difference signal. Finally, our segmentation-based compression technique for video transmission over a packet-switch network is applied.

8

Bibliography

[1] M. Barnsley. *Fractals Everywhere*. Academic Press, Inc., 1988.

[2] M. F. Barnsley and S. Demko. Iterated function systems and the global construction of fractals. In *The Proceedings of the Royal Society of London*, volume A399, pages 243–275, 1985.

[3] M. F. Barnsley, V. Ervin, D. Hardin, and J. Lancaster. Solution of an inverse problem for fractals and other sets. *Proc. Nat. Acad. Sci.*, 83:1975–1977, 1986.

[4] M. F. Barnsley and A. D. Sloan. A better way to compress images. *Byte*, pages 215–223, Jan. 1988.

[5] M. F. Barnsley and A. D. Sloan. *Byte*, pages 215–223. Managing Megabytes, Jan. 1988.

[6] M. J. Biggar and A. G. Constantinides. Segmented video coding. In *the 1988 IEEE Int. Conf. on Acoustics, Speech, and Signal Proceedings*, pages 1108–1111. April 1988.

[7] M. J. Biggar, O. J. Morris, and A. G. Constantinides. Segmented-image coding: performance comparison with the discrete cosine transform. *Proc. IEE, Part F*, 135(2):121–132, April 1988.

[8] P. Brodatz. *Texture: A photograph Album for Artists and Designers*. Dover, New York, 1956.

[9] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. Commun.*, COM-31:532–540, 1983.

[10] C. Chen, J. S. Daponte, and M. D. Fox. Fractal feature analysis and classification in medical imaging. *IEEE Trans. Medical Imaging*, 8(2):133–142, June 1989.

[11] P. C. Chen and T. Pavlidis. Segmentation by texture using correlation. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-5(1):64–69, Jan. 1983.

[12] M. R. Civanlar, S. A. Rajala, and W. M. LEE. Second generation hybrid image-coding technique. In *Proc. of SPIE*, volume 707, pages 132–137, Sept. 1986.

[13] R. W. Conners and C. A. Harlow. A theorectical comparison of texture algorithm. *IEEE Trans. Pattern and Mach. Intell.*, PAMI-2(3):204–222, May 1980.

[14] T. N. Cornsweet. *Visual Perception*. New York: Academic Press, 1971.

[15] L. D. Davisson. Data compression using straight line interpolation. *IEEE Trans. Inform. Theory*, IT-14:390–394, May 1968.

[16] K. Deguchi and I. Morishita. Texture characterization and texture-based image partitioning using two-dimensional linear estimation technique. *IEEE Trans. Comput.*, 27, 1978.

[17] T. J. Dennis and N. G. Dessipris. Fractal modeling in image texture analysis. *IEE Proceedings*, 136, Pt. F(5):227–235, Oct. 1989.

[18] R. A. Earnshaw, editor. *Fundamental Algorithms for Computer Graphics*, chapter Random fractal forgeries, pages 805–836. New York: Springer-Verlag, 1985.

[19] A. Habibi (Guest Editor). Special issue on image bandwidth compression. *IEEE Trans. Commun.*, COM-25, Nov. 1977.

[20] C. C. Cutler (Guest Editor). Special issue on redundancy reduction. *Proceedings of the IEEE*, 55, March 1967.

[21] L. Ehrman. Analysis of some redundancy removal bandwidth compression techniques. *Proc. IEEE*, 55:278–287, March 1967.

[22] P. Elias. Predictive coding-part i and part ii. *IRE Trans. Inform. Theroy*, Vol. IT-1:16–33, March 1955.

[23] A. Fournier, D. Fussell, and L. Carpenter. Computer rendering of stochastic models. *Commun. Ass. Comput. Mach.*, 25(6), 1982.

[24] S. W. Golomb. Run length encodings. *IRE Trans. on Information Theory*, IT-12:399–401, July 1966.

[25] D. J. Granrath. The role of human visual models in image processing. *Proc. IEEE*, 69:552–561, 1981.

[26] R. M. Gray. *Vector Quantization*, pages 4–29. IEEE Acoustics, Speech, and Signal Processing Magazine, April 1984.

[27] A. Habibi. Survey of adaptive image coding techniques. *IEEE Trans. Commun.*, COM-25:1275–1284, Nov. 1977.

[28] A. Habibi. An adaptive strategy for hybrid image coding. *IEEE Trans. Commun.*, COM-29:1736–1740, Dec. 1981.

[29] A. Habibi and A. N. Netravali (Guest Editors). Special issue on picture communication systems. *IEEE Trans. Commun.*, COM-29, Dec. 1981.

[30] C. A. Hamilton. A quick, portable method for brightness linearization of computer displays. Master's thesis, North Carolina State University, Dec. 1989.

[31] P. M. Haralick, K. Shanmugam, and J. Dinstein. Texture feature for image classification. *IEEE Trans. Syst., Man, Cybern.*, 3, 1973.

[32] R. M. Haralick and L. G. Shapiro. Image segmentation techniques. *Comput. Graphics Image Processing*, 29:100–132, 1985.

[33] J. Y. Y. Huang and P. M. Schulthesis. Block quantization of correlated gaussian random variables. *IEEE Trans. Commun. Systems*, CS-11:289–296, Sep. 1963.

[34] T. S. Huang. Coding of two-tone images. *IEEE Trans. Commun.*, COM-25:1406–1424, Nov. 1977.

[35] A. E. Jacquin. A novel fractal block-coding technique for digital images. In *ICASSP 90*, volume 4, pages 2225–2228, April 1990.

[36] A. K. Jain. Image data compression: a review. *Proc. IEEE*, 69:349–389, March 1981.

[37] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

[38] J. Jang and S. A. Rajala. Segmentation based image coding using fractals and the human visual system. In *ICASSP 90*, volume 4, pages 1957–1960, April 1990.

[39] J. Jang and Sarah A. Rajala. Segmentation based image coding using fractals and the human visual system. In *Proc. ICASSP'90*, pages 1957–1960, April 1990.

[40] N. S. Jayant and P. Noll. *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Prentice-Hall, 1984.

[41] M. Kocher and M. Kunt. Image data compression by contour-texture modelling. In *SPIE Int. Conf. on the Applications of Digital Image Processing*, pages 132–139, 1983.

[42] K. S. Ku and J. K. Mui. A survey on image segmentation. *Pattern Recognition*, 13:3-16, 1981.

[43] M. Kunt, A. Ikonmopoulos, and M. Kocher. Second-generation image coding technique. *Proc. IEEE*, 73:549-574, April 1985.

[44] K. J. Laws. *Texture Image Segmentation*. Univ. Southern Calif., Image Processing Inst., Jan. 1980. USCIPI Rep. 940.

[45] J. O. Limb. Picture coding: The use of a viewer model in source encoding. *Bell System Technical Journal*, 52:1271-1302, Oct. 1973.

[46] B. S. Lipkin and A. Rosenfeld. *Picture Processing & Psychopictorics*. Academic Press: New York/London, 1970. Eds.

[47] T. Lundahl, W. J. Ohley, S. M. Kay, and R. Siffert. Fractional brownian motion: A maximum likelihood estimator and its application to image texuture. *IEEE Trans. Med. Imaging*, MI-5(3):152-161, Sep 1986.

[48] T. Lundahl, W. J. Ohley, S. M. kay, and R. Siffert. Fractional brownian motion: A maximum likelihood estimator and its application to image texture. *IEEE Trans. Medical Imaging*, MI-5(5):152-161, Sept. 1986.

[49] B. B. Mandelbrot. *The Fractal Geometry of Nature*. San Francisco, CA: Freeman, 1982.

[50] B. B. Mandelbrot. *The Fractal Geometry of Nature*. San Francisco: W. H. Freeman and Co., 1983.

[51] B. B. Mandelbrot and B. J. Van Vess. Fractional brownian motion, fractional noises and applications. *SIAM*, 10(4):422-438, 1968.

[52] D. Marr. *Vision*. W. H. Freeman and Company, New York, 1982.

[53] X. Michel, R. Leonardi, and A Gersho. Unsupervised segmentation of texture images. In *Proc. SPIE Visual Communications and Image Processing'88*, pages 582-590, 198e.

[54] H. G. Musmann, P. Pirsch, and H. J. Grallert. Advances in picture coding. *Proc. IEEE*, 73:523-548, April 1985.

[55] A. N. Netravali. Interpolative picture coding using a subjective criterion. *IEEE Trans. Commun.*, COM-25:503-508, May 1977.

[56] A. N. Netravali and J. O. Limb. Picture coding: a review. *Proc. IEEE*, 68:366-406, March 1980.

[57] N. B. Nill. A visual model weighted cosine transform for image compression and quality assessment. *IEEE Trans. Commun.*, 33(6):551–557, 1985.

[58] R. C. Pasco. *Source coding algorithms for fast data compression.* PhD thesis, Dep. of EC'E, Stanford University, May 1976.

[59] Heinz-Otto Peitgen and Dietmar Saupe Ed. *The Science of Fractal Images.* Springer-Verlag, 1988.

[60] S. Peleg, J. Naor, R. Hartley, and D. Avnir. Multiple resolution texture analysis and classifiction. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-6(4):518–523, July 1984.

[61] S. Peleg, J. Naor, R. Hartley, and D. Avnir. Multiple resolution texture analysis and classification. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-6(4):518–523, July 1984.

[62] W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, and R. B. Arps. An overview of the basic principles of the q-coder adaptive binary arithmetic coder. *IBM J. Res. Devel.*, 32(6):717–726, 1988.

[63] A. P. Pentland. Fractal-based description of natural scenes. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-6(6):664–674, Nov 1984.

[64] A. P. Pentland. Shading into texture. *Artificial Intell.*, 29:147–170, 1986.

[65] H. A. Peterson, S. A. Rajara, and E. J. Delp. Image segmentation using human visual system properties with applications in image compression. Technical Report TR-EE 90-4, School of Electrical Engineering, Purdue Univ., Jan. 1990.

[66] W. K. Pratt. *Digital Image Processing.* John Wiley & Sons, Inc., 1978.

[67] W. K. Pratt. *Image Transmisstion Techniques.* Academic Press, New York, 1979.

[68] W. K. Pratt, O. D. Faugeras, and A. Gagalowicz. Visual discrimination of stochastic algorithm for image segmentation and region classification. *IEEE Trans. Syst., Man. Cybern.*, 8, 1978.

[69] H. M. Raafat and A. K. C. Wong. A texture information-directed region growing algorithm for image segmenation and region classification. *Computer Vision, Graphics, and Image Processing*, 43:1–21, 1988.

[70] H. M. Raafat and A. K. C. Wong. A texture information-directed region growing algorithm for image segmentation and region classification. *Comput. Graphics Image Processing*, 43:1–21, 1988.

[71] S. A. Rajala, M. R. Civanlar, and W. M. Lee. A second generation image coding technique using human visual system based segmentation. In *Proc. ICASSP'87*, pages 1362–1365, April 1987.

[72] S. A. Rajala, M. R. Civanlar, and W. M. Lee. A second generation image coding technique using human visual system based segmentation. In *1987 IEEE Int. Conf. on Acoustics, Speech, and Signal Proceedings*, pages 1362–1365, April 1987.

[73] S. A. Rajala, M. R. Civanlar, and W. M. Lee. Video data compression using three-dimensional segmentation based on hvs properties. In *Proc. ICASSP'88*, pages 1092–1095, April 1988.

[74] S. A. Rajala and W. M. Lee. Segmentation-based image coding in a packet-switched network environment. In *Proc. SPIE. Visual Commun. and Image Processing'88*, volume 1001, pages 1006–1010, 1988.

[75] H. K. Reghbati. An overview of data compression techniques. *Computer*, 14(4):71–76, April 1981.

[76] J. Rissanen. Generalized kraft-inequality and arithmetic coding. *IBM J. Res. Devel.*, 20:198–203, 1976.

[77] A. Rosenfeld and A. C. Kak. *Digital picture processing. 2nd. Edn.* Academic Press, 1982.

[78] Ariel Rosenfeld. *Finding Structure in Co-Occurrence Matrics for Texture Analysis*, pages 423–445. Academic Press, 1980. Ed.

[79] W. F. Schreiber. Picture coding. *Proceedings of the IEEE*, 55:320–330, March 1967.

[80] W. F. Schreiber, T. S. Huang, and O. J. Tretiak. *Picture Bandwidth Compression.* Gordon and Breach: New York, 1972. ed. O. J. Tretiak.

[81] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[82] H. C. Shen and A. K. C. Wong. Generalized texture representation and metric. *Computer Vision, Graphics, and Image Processing*, 23:187–206, 1983.

[83] T. G. Stockham. Image processing in the context of a visual model. *Proc. IEEE*, 60:80–87, 1976.

[84] M. Unser. Sum and difference histograms for texture classification. *IEEE Trans. Pattern and Mach. Intell.*, PAMI-8:118–125, 1986.

[85] R. F. Voss. Random fractals:characterization and measurement. *Physica Scripta*, T-13:27–32, 1986.

[86] D. C. C. Wang and A. H. Wagnucci. Gradient inverse filtering weighted smoothing scheme and the evaluation of its performance. *Computer Vision, Graphics, and Image Processing*, 15:167–181, Feb. 1981.

[87] P. A. Wintz. Transform picture coding. *Proc. IEEE*, 60:809–820, July 1972.

[88] A. K. C. Wong and M. A. Vogel. Resolution dependent information measures for image analysis. *IEEE Trans. Syst. Man Cybern.*, SMC-7(1):49–61, Jan. 1977.

[89] Glenn Zorpette. *IEEE Spectrum*, pages 29–31. IEEE, Oct. 1988.

[90] W. Zschunke. Dpcm picture coding with adaptive prediction. *IEEE Trans. Commun.*, COM-25:1295–1302, Nov. 1977.

9

Appendices

9.1 Runlength Coding

In this appendix we describe a technique proposed by Elias for coding a sparse binary image. Suppose a binary image is mostly 0's with only a few 1's. The rows of the $N \times N$ image are concatednated together to form a vector of N^2 gray level values, and all runs of consecutive 0's in this vector are found. The lengths of these runs, separated by a symbol (referred to as a "comma") to mark the end of a run (i.e. the presence of a 1), completely describe the original binary image. Elias has proposed coding these runlengths (viewed as decimal numbers) using an n-ary arithmetic system, and using an n+1'th symbol to represent a comma. For example, for n=3 the runlengths are represented in a ternary system. The comma requires an additional symbol, for a total of four systems. These four symbols are represented using a two bit code. One possible choice to represent the four symbols is: 00=comma, 01=0, 10=1, 11=2.

Consider the following 40 bit binary sequence:

0000110000000000010001000000000100000000

The runlengths for this sequence are: 4, 0, 11, 3, 9,8, and the ternary representations for these runlengths are: 11, 0, 102, 10, 100, 22. Finally, using the representation

described above. the Elias code representation for the original binary sequence is:

1010 00 01 00 100111 1001 00 100101 00 1111

We have represented the original 40 bit sequence using 36 bits.

9.2 Crack Code

Crack code determine a boundary by specifying a starting point and a sequence of moves around the boundary. Figure 9.1 illustrates a way in which this can be done by moving a sequence of cracks between the points S and the adjacent points of the complement $\bar{S}$.

If we follow the cracks around a bounday, at each move we are going either left, righ, up, or down; if we denote direction $90i_o$ by i, these moves can be represented by a sequence of 2-bit numbers; 0, 1, 2, 3. For example, the sequence in Figure 9.1c is represented by 00303332112121. This representation is called as a crack code. A boundary is specified by giving the coordinates of a starting crack together with a crack code.

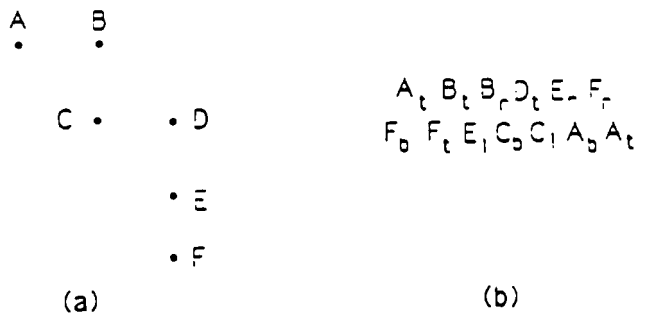


Figure 9.1: A crack code. (a) Set S; each point is labeled with a different latter. (b) Clockwise sequence of cracks around the border, beginning with the crack A_t at the top of A. The subscripts of t, r, b, l denote top, right, bottom, and left, respectively.

described above, the Elias code representation for the original binary sequence is:

1010 00 01 00 100111 1001 00 100101 00 1111

We have represented the original 40 bit sequence using 36 bits.

9.2 Crack Coding

Crack code determine a boundary by specifying a starting point and a sequence of moves around the boundary. Figure 9.1 illustrates a way in which this can be done by moving a sequence of cracks between the points S and the adjacent points of the complement $\bar{S}$.

If we follow the cracks around a boundary, at each move we are going either left, righ, up, or down; if we denote direction $90i_o$ by i, these moves can be represented by a sequence of 2-bit numbers; 0, 1, 2, 3. For example, the sequence in Figure 9.1b is represented by 00303332112121. This representation is called as a crack code. A boundary is specified by giving the coordinates of a starting crack together with a crack code.

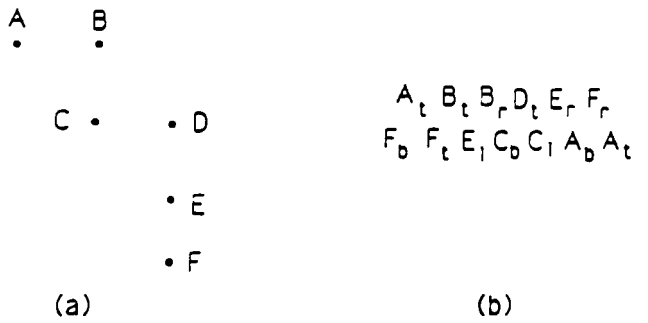


Figure 9.1: A crack code. (a) Set S; each point is labeled with a different latter. (b) Clockwise sequence of cracks around the border, beginning with the crack A_t at the top of A. The subscripts of t, r, b, l denote top, right, bottom, and left, respectively.

9.3 Arithmetic Coding

Let the alphabet consist of the symbols $i = 0, 1, \cdots, m$. An arithmetic code encodes the string to be compressed, symbol for symbol from left to right. When encoding the symbol i, immediately following the so-far processed string s, the code requires as input the data that represent the conditional probability $P(i/s)$ of the symbol's occurrence at its context. Such parameters are provided by the so-called modeling unit. The modeling unit should update for each symbol i not only the count but also the cumulative counts which are affected by the count. An arithmetic code constructs the code string as a cumulative probability of the strings that precede the considered one in the lexical order of the strings.