ACCURACY OF FRACTAL AND MULTIFRACTAL MEASURES FOR SIGNAL ANALYSIS

by

HONGJING CHEN

A Thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCENCE
in
COMPUTER ENGINEERING

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba, Canada

Thesis Advisor: W. Kinsner, Ph. D., P. Eng.

(xii+115+A37+B29=) 193 pp.

0-612-23247-6

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES
*****
COPYRIGHT PERMISSION PAGE

ACCURACY OF FRACTAL AND MULTIFRACTAL MEASURES FOR SIGNAL ANALYSIS

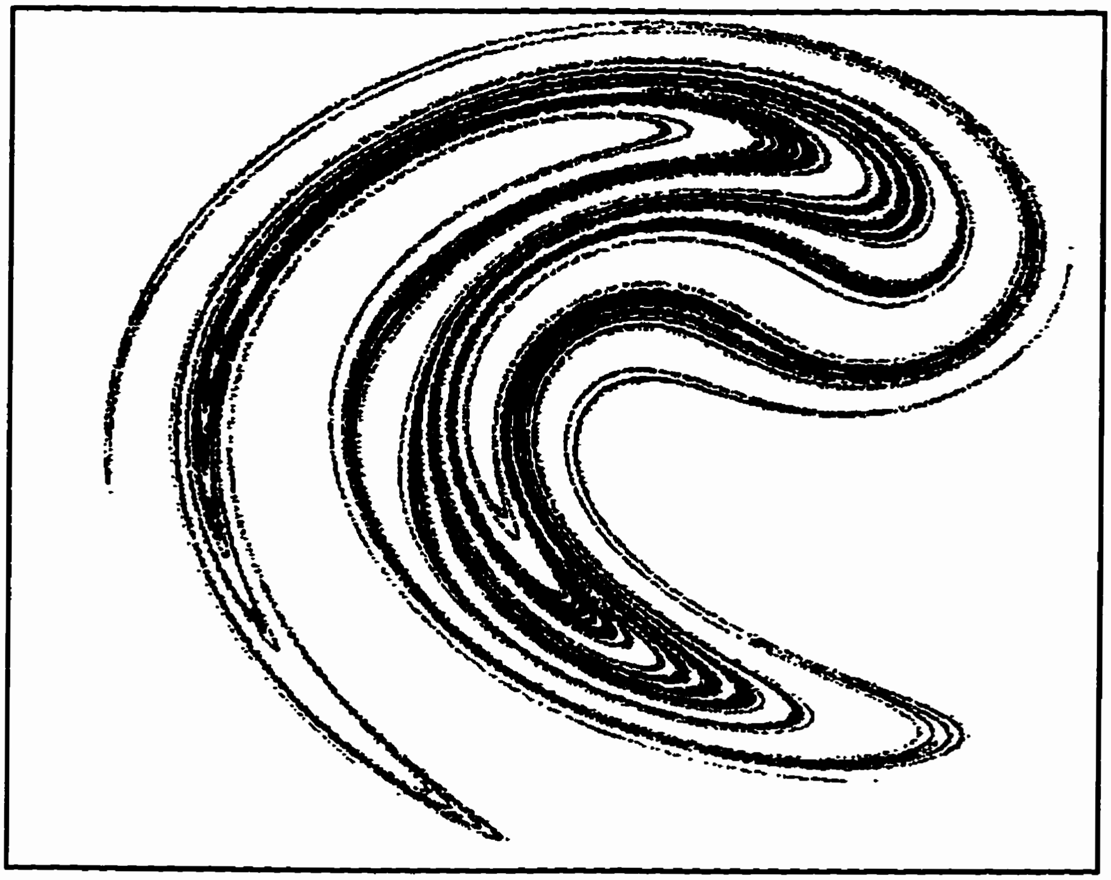
BY

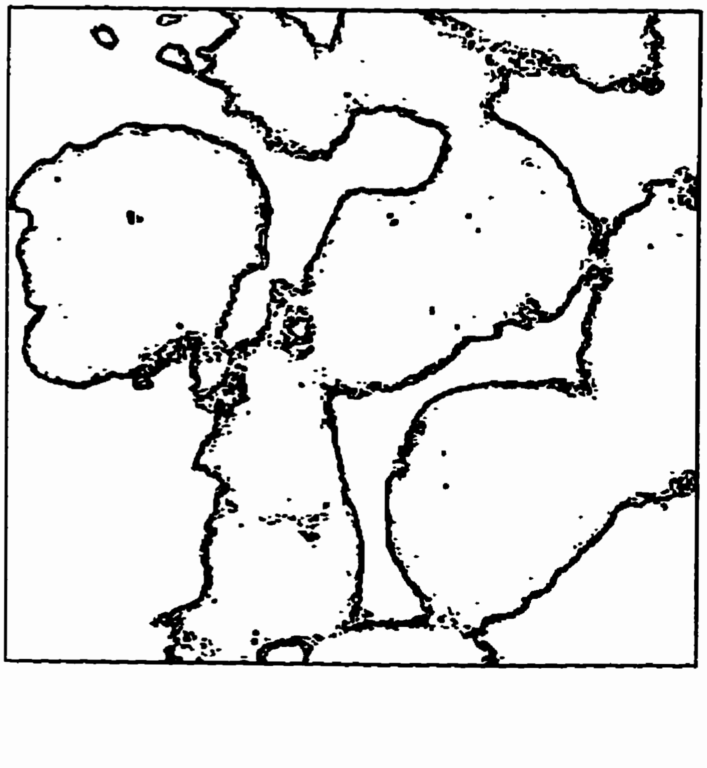
HONGJING CHEN

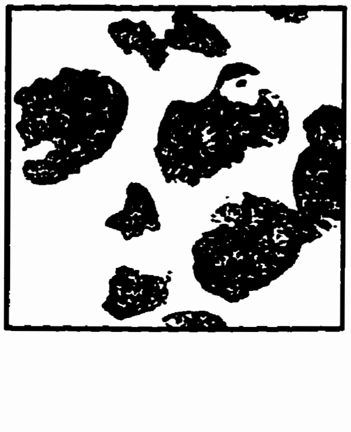
A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree

of

MASTER OF SCIENCE







ABSTRACT

This thesis is concerned with accurate approximations of multifractal measures of strange attractors and analysis of the spatial signals (images) from the point of view of strange attractors through multifractal measures.

The approximations of multifractal measures of strange attractors is studied through the Rényi dimension, singularity spectrum, and the Mandelbrot dimension. It is based on the probability of each volume element (vel) intersected by the points on the strange attractor. Since the complete strange attractor consists of an infinite number of points, we cannot obtain the theoretical value of the probability; instead, we consider a finite number of points in the vels. Therefore, this study reduces to a finite number of points and finite size of vels. We have shown that, for a given vel size, the Rényi dimension is sensitive to the number of points used in the attractor, and that for a given number of points in the strange attractor, it is also sensitive to the vel size. We also find that for a given vel size, there is a minimum bound on the number of points required. The smaller the vel size, the larger the minimum bound. Furthermore, the Rényi dimension converges when the number of points increases above the minimum bounds. The convergence can be a guideline to determine the number of points required to compute the dimension.

The results of the study of the strange attractors can be applied to spatial signals. In this thesis, spatial signals such as images are modelled as strange attractors, and multifractals are used to characterize complicated distributions of the grey levels in images. Similar textures generally will have the homogeneous property which can be characterized by the singularity and Mandelbrot dimension of the fractal sets. Consequently, the Mandelbrot spectrum reveals the structures of the spatial signals. With multifractal representation of images, better image segmentation and edge detection are achieved. Since accurate approximation of multifractal measures is obtained, detailed textures can also be detected and separated.

ACKNOWLEDGEMENTS

I would like to thank God, blessing me in all the difficulties I have encountered during the course of this thesis. I would like to thank Dr. Kinsner for proposing this interesting topic and giving me his time and patience in the course of this thesis. I would also like to thank Dr. Kinsner for his improvements and editing this thesis documents.

I would like to thank Richard Dansereau for his help. He was always there for me. I would also like to thank Randy Allan for providing the benign and malignant cancerous breast cells images.

I would like to acknowledge everybody in our Delta Group, past and present, including Ken Ferens, Armein Langi, Lawrence Arendt, Warren Grieder, Eric Jang, Tom Tessier, Jason Toonstra, Rasekh Rifaat, Don Shaw, Fan Mo, Pradeepa Yahampath, Jizong Li.

This thesis is for my father and my elder brother for their unconditional support and inspiration. Above all this thesis is for my wife, Qionghua and my son, Diyuan.

TABLE OF CONTENTS

Abstract iii
Acknowledgements iv
Table of Contents v
List of Figures viii
List of Tables xi

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 BACKGROUND 4

2.1 Fractal Sets 4
Mathematical Fractals 4
Fractal Objects in Nature 7
Common Geometric Properties of Fractal Sets 10
2.2 Fractal Dimensions 11
Self-Similarity Dimension 13
Hausdorff Dimension 13
Information Dimension 14
Correlation Dimension 15
2.3 Is Single Dimension Sufficient? 15
2.4 Multifractals 18
Rényi Entropy 18
Rényi Dimension 18
Formalism of the Geometric Description of Multifractals 20
Characteristic Features of the Functions D_q, $\alpha(q)$, and $f(\alpha(q))$ 23
2.5 Overview of Chapter 2 25

CHAPTER 3 STRANGE ATTRACTORS AND THEIR APPROXIMATIONS OF MULTIFRACTAL MEASURES 26

3.1 Chaotic Dynamical Systems 26
3.2 Examples of Strange Attractors 29
The Poincaré Section of the Rössler Attractor 29
The Hénon Attractor 32
The Ikeda Attractor 32
3.3 Characterization of Strange Attractors 34
Approximation of Multifractal Measures 34
Approximations of the Rényi Dimension 35
Approximations of the Singularity Spectrum 37
Approximations of the Mandelbrot Dimension 39
Compromise Between the Vel Size and the Number of the Points 41
3.4 Overview of Chapter 3 42

CHAPTER 4 MULTIFRACTAL MEASURES IN SPATIAL SIGNAL ANALYSIS 43

4.1 Texture Analysis of Images 43
4.2 Measuring a Pixel Through Its Random Binary Dithering Process 44
Electronic Techniques for Pictorial Image Reproduction. 44
Binary Marking/Displaying Technologies 44
Continuous Tone Imagery and Halftone Imagery 45
Measure a Pixel of an Image. 45
4.3 Determining the Size of a Pixel in Fractal Analysis. 47
4.4 Multifractal Model for Images 51
Multifractal measure for Images. 51
Approximations of the Rényi Dimension for Images. 52
Approximations of Singularity Spectrum for images. 53
Approximations of Mandelbrot Dimension for Images 55
Multifractal Measures on Single Fractal Images 56
Multifractal Measures on Multifractal Images 60
4.5 Overview of Chapter 4 64

CHAPTER 5 EXPERIMENTAL RESULTS AND DISCUSSION 65

5.1 Multifractal Characterization of Strange Attractors 65
Sensitivity to the Number of Points in the Numerical Representation. 66
Sensitivity to the Vel Size in the Computational Schemes. 69
Compromise of the Number of Points and the Vel Size. 71
5.2 Application of Multifractal Measures in Image Analysis. 78
Multifractal Feature Map 79
Relationship Between Edges and Textures. 81
Edge Detection 82
Edge Detection of Aerial Images Using Multifractal Feature Maps 82
Classical Edge Detectors for the Aerial Image. 85
Boundary Extraction For Cancer Cells. 86
Boundary Extraction for a Benign Cancer Cells Image 87
Boundary Extraction for a Malignant Cancer Cells Image. 90
Texture Extraction Using Multifractals 96
5.3 Decomposition of the Mandelbrot Spectrum 97
The Mandelbrot Spectra of Sub-Images. 100
The Decomposition of the Mandelbrot Spectrum. 102
Set-valued Multifractal Feature Map 102
Decomposition of the Mandelbrot Spectrum through Singularity 103
An Application of the Decomposition of the Mandelbrot Spectrum 105
5.4 Overview of Chapter 5 106

CHAPTER 6 CONCLUSIONS AND RECOMMENDATIONS 107

6.1 Conclusions 107
6.2 Contributions 108
6.3 Recommendations. 109

REFERENCES 110

APPENDICES

A: STRUCTURE OF PROGRAM AND SOURCE CODE IN C.......... A-1 - A-37

B: STRUCTURE OF PROGRAM AND SOURCE CODE IN C++........ B-1 - B-29

LIST OF FIGURES

Fig. 2.1 Generation of the Sierpinski blanket. (a) The initiator (step zero). (b) The generator (step one). (c) Step two. (d) Step three. 7

Fig. 2.2 The rugged border of the northeastern part of Canada. 8

Fig. 2.3 (a) The Hénon attractor. (b) A zoomed portion of the Hénon attractor. 9

Fig. 2.4 (a) A picture of Lena consisting of 256x256 pixels, 8 bits each, and (b) its zoomed part. 10

Fig. 2.5 Log-log diagram for the total length of the northeast coast of Canada L versus the inverse of the ruler size r. 12

Fig. 2.6 (a) The merged fractal set. (b) The log-log plots for the merged fractal. 17

Fig. 2.7 An illustration of a recursive process of generating a nonuniform fractal. 21

Fig. 2.8 An simple example showing $f(\alpha)$. 22

Fig. 2.9 Visualization of $D(q)$ and $\alpha(q)$. 24

Fig. 2.10 The visualization of the Mandelbrot spectrum. 25

Fig. 3.1 An orbit in a three-dimensional (N=3) phase space. 27

Fig. 3.2 The sensitivity to initial conditions in chaotic dynamical systems. (After [Kins94b]) 28

Fig. 3.3 The Rössler attractor. 30

Fig. 3.4 An illustration of the Poincaré section. 31

Fig. 3.5 The Poincaré section of the Rössler attractor at $z = 22.0$. 31

Fig. 3.6 The Hénon attractor. 32

Fig. 3.7 The Ikeda attractor. 33

Fig. 3.8 An illustration of covering a strange attractor by nonintersecting vels. 34

Fig. 4.1 (a) An image of Lena (128x128 pixels with 8 bpp). (b) A cutout image (34x34 pixels) from (a). (c) A zoomed image (544x544 pixels with 8 bpp) from the cutout image. 46

Fig. 4.2 The dithering image (544x544 pixels with 1 bpp) from the cutout image in Fig. 4.1b. 47

Fig. 4.3 (a) The dimension of a line is 1. (b) The dimension of a plane is 2. 48

Fig. 4.4 An image of a plane (a) and an image of a line (b). 49

Fig. 4.5 Grey level images. (a) The images of the Koch curve. (b) The image of the Minkowski curve. (c) The image of the Sierpinski carpet. (d) The image of the Sierpinski gasket. 57

Fig. 4.6 The the Rényi dimension, singularity spectrum, and the Mandelbrot dimension of a line, the Koch curve, the Minkowski curve, Sierpinski gasket, Sierpinski carpet, and a plane. 58

Fig. 4.7 The degraded Mandelbrot spectrum for a line, the Koch curve, the Minkowski curve, the Sierpinski gasket, the Sierpinski carpet, and a plane. 58

Fig. 4.8 Multifractal images. (a) An image of Lena which is 256 by 256 pixels with 8 bpp. (b) An image of Baboon which is 256 by 256 pixels with 8 bpp. 60

Fig. 4.9 The Rényi dimension of the image of Lena and Baboon. 61

Fig. 4.10 The singularity spectrum of the image of Lena and Baboon. 61

Fig. 4.11 The Mandelbrot dimension f(q) of the image of Lena and Baboon. 62

Fig. 4.12 The Mandelbrot spectrum of the image of Lena and Baboon. 63

Fig. 4.13 The histograms. (a) for image of Lena. (b) for image of Baboon..................63

Fig. 5.1 The Rényi dimension D_q is changed when the number of the points N from the Hénon attractor varies from 50 to 50,000. 67

Fig. 5.2 The singularity spectrum is changed when the number of the points N from the Hénon attractor varies from 50 to 50,000. 67

Fig. 5.3 The Mandelbrot dimension f_q is changed when the number of the points N from the Hénon attractor varies from 50 to 50,000. 68

Fig. 5.4 The Mandelbrot spectrum fα is changed when the number of the points N from the Hénon attractor varies from 50 to 50,000. 68

Fig. 5.5 The spectrum of the Rényi dimension varies as the vel size changes. (a) ε= 1/64. (b) ε= 1/128. (c) ε= 1/256. (d) ε= 1/512. (e) ε= 1/1024.........................69

Fig. 5.6 The spectrum of the singularity varies as the vel size changes. (a) ε= 1/64. (b) ε= 1/128. (c) ε= 1/256. (d) ε= 1/512. (e) ε= 1/1024. 70

Fig. 5.7 The spectrum of the Mandelbrot dimension varies as the vel size changes. (a) ε= 1/64. (b) ε= 1/128. (c) ε= 1/256. (d) ε= 1/512. (e) ε= 1/1024.................70

Fig. 5.8 The Mandelbrot spectrum varies as the vel size changes. (a) ε= 1/64. (b) ε= 1/128. (c) ε= 1/256. (d) ε= 1/512. (e) ε= 1/1024...71

Fig. 5.9 The convergence of the approximation of the Rényi dimension for the Hénon attractor for N = 50, 500, 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ..., 90000, 95000, 100000.................................73

Fig. 5.10 The convergence of the approximation of the Mandelbrot spectrum for the Hénon attractor for N = 50, 500, 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ... , 90000, 95000, 100000...................74

Fig. 5.11 The convergence of the approximation of Rényi dimension of the Poincaré section of the Rössler attractor for N=10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ... , 90000, 95000, 100000...............................74

Fig. 5.12 The convergence of the approximation of the Mandelbrot spectrum of the Poincaré section of the Rössler attractor for N=10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ... , 90000, 95000, 100000...................75

Fig. 5.13 The convergence of the approximation of the Rényi dimension of the Ikeda attractor, N = 100, 5000, 50000, 100000, 15000, 200000, 25000, ..., 900000, 95000, 1000000...75

Fig. 5.14 The convergence of the approximation of the Mandelbrot spectrum of the Ikeda attractor, N = 100, 5000, 50000, 100000, 15000, 200000, 250000, ..., 900000, 95000, 1000000...76

Fig. 5.15 The convergence of the approximation of the Hausdoff dimension of the Hénon attractor for N = 50, 500, 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ... , 90000, 95000, 100000...............................77

Fig. 5.16 The convergence of the approximation of the Hausdorff dimension of the Poincaré section of the Rössler attractor for N = 50, 500, 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ..., 90000, 95000, 100000. 77

Fig. 5.17 The convergence of the approximation of the Hausdoff dimension of the Ikeda attractor for N = 100, 10000, 50000, 100000, 15000, 200000, ..., 900000, 95000, 1000000...78

Fig. 5.18 The mapping from an image to the multifractal feature matrix.......................80

Fig. 5.19 An illustration of edges and a texture. 81
Fig. 5.20 An illustration of corners and corners texture......................................81
Fig. 5.21 An aerial image (512x512x8). 82
Fig. 5.22 The multifractal feature vector at row 249. (a) The rate of change of the singularity spectrum. (b) The density function DF_q..84
Fig. 5.23 Edge detection result for an aerial image using the multifractal method. 85
Fig. 5.24 Edge detection using classical operator. (a) Thresholding Sobel operator at T=77. (b) Thresholding Prewitt operator at T=77. (c) Thresholding Robert operator at T=19. (d) Thresholding Laplacian operator at T=140. 86
Fig. 5.25 An image of benign cancer cells. 87
Fig. 5.26 The multifractal feature vector at row 76. (a) The rate of change of the singularity spectrum. (b) The density function DF_q. 88
Fig. 5.27 The detected boundary from the benign cells image. 89
Fig. 5.28 Edge detection using classical operators. (a) Thresholding Sobel operator at T=77. (b) Thresholding Prewitt operator at T=77. (c) Thresholding Robert operator at T=19. (d) Thresholding Laplacian operator at T=140. 90
Fig. 5.29 An image of malignant cancer cells. 91
Fig. 5.30 The multifractal feature vector at row 245. (a) The rate of change of the singularity spectrum. (b) The density function DF_q..92
Fig. 5.31 The detected boundary from the malignant cancer cell. 93
Fig. 5.32 Sobel operator is used in the malignant cancer cell image. (a) T=27. (b) T=54. 94
Fig. 5.33 Thresholding Prewitt operator is used in the malignant cancer cell image. (a) T=7. (b) T=15. 94
Fig. 5.34 Thresholding Robert operator is used in the malignant cancer cell image. (a) T=25. (b) T=45. 95
Fig. 5.35 Thresholding of Laplacian is used in the malignant cancer cells image (T=153). 95
Fig. 5.36 The extracted textures from the malignant cancer cells..................................96
Fig. 5.37 The selected texture image..97
Fig. 5.38 An image of 256 by 256 pixels with 8 bits representing each pixel. 99
Fig. 5.39 The Mandelbrot spectrum of the image shown in Fig. 5.38. 99
Fig. 5.40 The four images cut from the original image. (a) Upper left part. (b) Upper right part. (c) Lower left part. (d) Lower right part. 100
Fig. 5.41 The Mandelbrot spectra of the four images (Fig. 5.40a to 5.40d) and the original image..101
Fig. 5.42 The histograms of the four sub-images in Fig. 5.40. (a) Upper left part. (b) Upper right part. (c) Lower left part. (d) Lower right part. 102
Fig. 5.43 The mapping from an image to a set-valued multifractal feature set 103
Fig. 5.44 A decomposition of the original Mandelbrot spectrum. 104
Fig. 5.45 A non-intersecting decomposition of the original Mandelbrot spectrum. 105
Fig. 5.46 The textures extracted using the decomposition elements of the original image. 106

LIST OF TABLES

Table 2.1 The generation of the Koch curve 6
Table 4.1 The Theoretical and Experimental Dimensions 59
Table 5.1 Classical Edge Detection Masks 85

CHAPTER 1

INTRODUCTION

The behaviour of many phenomena found in areas such as physics, chemistry geology, and electronics may range from simple (uniform or periodic) to very complex (quasiperiodic or chaotic). Chaotic dynamical systems are examples of complex phenomena. This thesis is related to the characterization of strange attractors from chaotic systems.

The characterization of the strange attractors can be done through either single fractal dimension or multifractal dimension. Since single fractal dimension masks out many important features in the objects, this thesis will address more accurate multifractal characterization.

Mandelbrot [Mand83] and other researchers (*e.g.*, [GrPr83] [HaJK86], [Kins96]) have developed the concept of multifractals to describe such complex objects adequately. The reason for multifractal measures is that a strange attractor cannot be fully characterized by a single fractal measure since any simple fractal measure reveals only a single property of the multifractal object; instead, multifractal measures must be used to reveal the underlying structure of the multifractals.

The objectives of this thesis are to achieve accurate estimation of multifractal measures for strange attractors and to apply this approach to spatial signals such as images.

Is it easy to estimate such measures accurately? Since a strange attractor consists of an infinite number of points, but any practical computation involves a finite number of points only, there is an inherent residual error in the numerical values obtained. One purpose of this thesis is to study this residual error, and to establish the conditions for its accurate estimation.

In the past, Smith [Smit88] and Eckmann and Ruelle [EcRu92] presented lower bounds on the number of points required to estimate a single fractal dimension, the correlation dimension, D_2. This thesis extends the study of lower bounds from single-fractal dimension to multifractals through the generalized Rényi dimension [Kins96] covering all the fractal dimensions from $D_{+\infty}$ to $D_{-\infty}$.

The techniques in accurate approximation of the multifractal measures for strange attractors can be applied to nonstationary signals such as images, speech and transients, this thesis focuses on the study of images. Image analysis is an important research field which has a number of applications in areas such as satellite imaging, medical imaging, and robotic vision. Image segmentation and texture description has long been an area of active computer vision research. The ability of humans to distinguish different textures is apparent, but automated description and estimation is very difficult. The existing approaches to texture description are based on statistical and structural properties of the image. Haralick [Hara79] and Davis [Davi80] gave good surveys of these methods.

Mandelbrot introduced [Mand83] fractal geometry in texture analysis. Pentland [Pent84] [Pent93] developed a three dimensional model which has been used in texture segmentation and classification, estimation of 3D shape information and to distinguish between perpetually smooth and perpetually textured surfaces in the scene. Peleg *et al.* [PNHA84] also used fractal dimension as a scale-insensitive ruggedness measure by which silhouettes of tree lines and mountain lines are described.

In extending the concept of a single fractal dimension to surfaces, we have discovered that the fractal dimension does not provide sufficient capability in texture analysis. Although there is a link between roughness and fractal dimension, the roughness is not sufficient to describe a textured surface, because other characteristics have to be involved, such as arrangements and spatial distribution of grey levels. Multifractal theory can avoid the drawback of the single fractal dimension.

In this thesis, spatial signals such as images are to be analyzed from the point of view of strange attractors. Images actually consist of many different grey levels which distribute nonuniformly. A model of an image as a strange attractor will be set up. With this model, we will treat the images as strange attractors, the grey levels of pixels in an image are translated to the number of points in the strange attractor. Multifractals are used to characterize the complicated distribution of the grey levels.

In this thesis, we shall show the potential of multifractals in image analysis. Unlike the classical fractal approaches which use fractal dimensions to describe some processes on images (*e.g.*, [KeCh89] [SaCh92] [ChSK93]), this new fractal approach is used to describe the image structure such as textures. Different textures can be separated because similar textures generally will have an homogeneous property which can be characterized by the singularity and Mandelbrot dimension of the fractal sets. Consequently, we can achieve not only image segmentation and edge detection, but also separation of different textures.

This thesis is organized into six chapters. Chapter 2 provides a general introduction to fractals and multifractals, and discusses some of their properties. Chapter 3 presents the theory of approximation of multifractal measures. Chapter 4 develops a model to match the grey level images to strange attractors. With this model, we develop the multifractal measures. In this chapter, based on the singularity spectrum, the decomposition of the Mandelbrot spectrum is performed. In Chapter 5, we present experimental results. In the multifractal characterization of strange attractors, since the sensitivity of the number of points in the numerical representation and sensitivity of the volume size in the computational schemes, a compromise of the number of the points and the volume size is obtained. With the accurate estimation of the multifractal measures, we apply them in our image model. The experimental results show that this method is better than the classical methods in the edge detection and feature extraction. In this chapter, we also give preliminary results on the application of the decomposition of the Mandelbrot spectrum. Conclusions, contributions and recommendations are presented in Chapter 6.

CHAPTER 2

BACKGROUND

The concept of a fractal, recognized already at the end of the last century, and popularized by Mandelbrot in the 1960s, is now widely used in different fields. Knowledge of fractals is expanding because of their usefulness in understanding either pure growth or the growth/death phenomena. Determining the dimension *D* of the fractals occupies a central place in the theory of fractals.

The purpose of this chapter is to give an introduction to the basic concepts, properties of fractals and multifractals. Section 2.1 will discuss the fractal sets, both mathematical and natural. The basic methods for determining the fractal dimension are presented in Section 2.2. The multifractal dimensions and their geometric formalism are given in Section 2.4.

2.1 Fractal Sets

The formal definition of a fractal is "a set for which the Hausdorff-Besicovitch dimension strictly exceeds the topological dimension" [Mand83]. This definition is very mathematical; and it does not convey much intuitive meaning of fractals and how it is related to various applications. In addition, the word *strictly* has been removed from today's definition to accommodate the fractal sets with integer dimensions. A good way to understand fractals is to start from a few examples: both mathematical fractals and natural fractals.

2.1.1 Mathematical Fractals

One of the common features of a mathematical fractal object is its strict self similarity. If one enlarges a portion of such an object, the new, enlarged object is identical to the initial one. This is the scale-invariance property of the fractal objects. We use Koch's curve as an example of mathematical self-similar objects. A mathematical fractal satisfies the self-similarity property exactly; *i.e.*, no physical limitations are considered. Since such mathematical objects are built with infinite number of iterations, they are rigorously self similar at all scales.

One can generate fractals through iterative procedures from an initiator. The initiator of the Koch curve is a straight line (shown in Step 0 of Table 2.1). The generator is an algorithm describing the transformation from the initiator to the largest scale object. In the Koch curve case, the initiator is transformed into four line segments with equal length which is one third of the length of the initiator as shown in Step 1 of Table 2.1. In the second step, we begin from the results of the first step; the generator is applied to each of the four line segments in Step 1, then we obtain the results of the second step. At this stage, there are 16 line segments. The generator is then applied to each of these line segments. This recursive procedure is repeated to infinity.

It is important to observe that the form of the initiator is not very important. For example, the initiator can be a line, a circle, or any other geometric objects. After recursively applying the generator as described above, the final results will be the same. But the generator is very important as it determines the outcome after the recursive procedure. Although the generator is so simple, the final result can be infinitely complicated!

As we can see, the span of the Koch curve is finite. Its area is zero, but its length is infinite. In fact, In the n-th step, the total length of the curve is $L(n)$

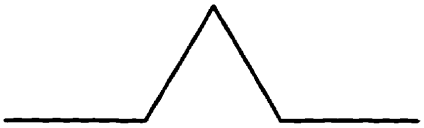
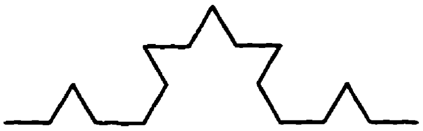
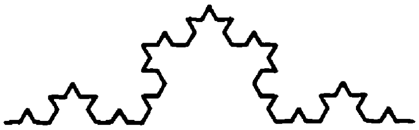
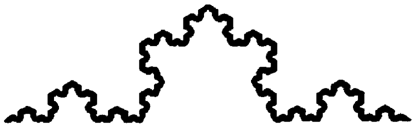
$$L(n) = \left(\frac{4}{3}\right)^n \tag{2.1}$$

as $n \to +\infty$, $L(n) \to +\infty$.

The Sierpinski blanket is another mathematical fractal object. The initiator is a square box of unit length shown in Fig. 2.1a, and the generator is an algorithm that cut the square box into nine equal sub-square boxes, and eliminate the middle square box. The result is shown in Fig. 2.1b. Applying the generator to each of the eight square boxes in the result of the first step, we achieve the result of the second step which is shown in Fig. 2.1c. Recursively applying the generator to infinity, the Sierpinski blanket can be obtained.

Just like the Koch's curve, the Sierpinski blanket is scale invariant, and has self-similarity property. The area of the Sierpinski blanket is zero, but its perimeter goes to infinity. So, we get the incompatible results in both one dimensional and two dimensional spaces. This gap can be reconciled through fractal dimension which is a fraction rather than an integer.

Table 2.1: The generation of the Koch curve

Step	Graph	Segments Length	No. of the Segments	Total length
0		1	1	1
1		$\frac{1}{3}$	4	$\frac{4}{3}$
2		$\frac{1}{9}$	16	$\frac{16}{9}$
3		$\frac{1}{27}$	64	$\frac{64}{27}$
⋮	⋮	⋮	⋮	⋮
n		$\frac{1}{3^n}$	4^n	$\left(\frac{4}{3}\right)^n$

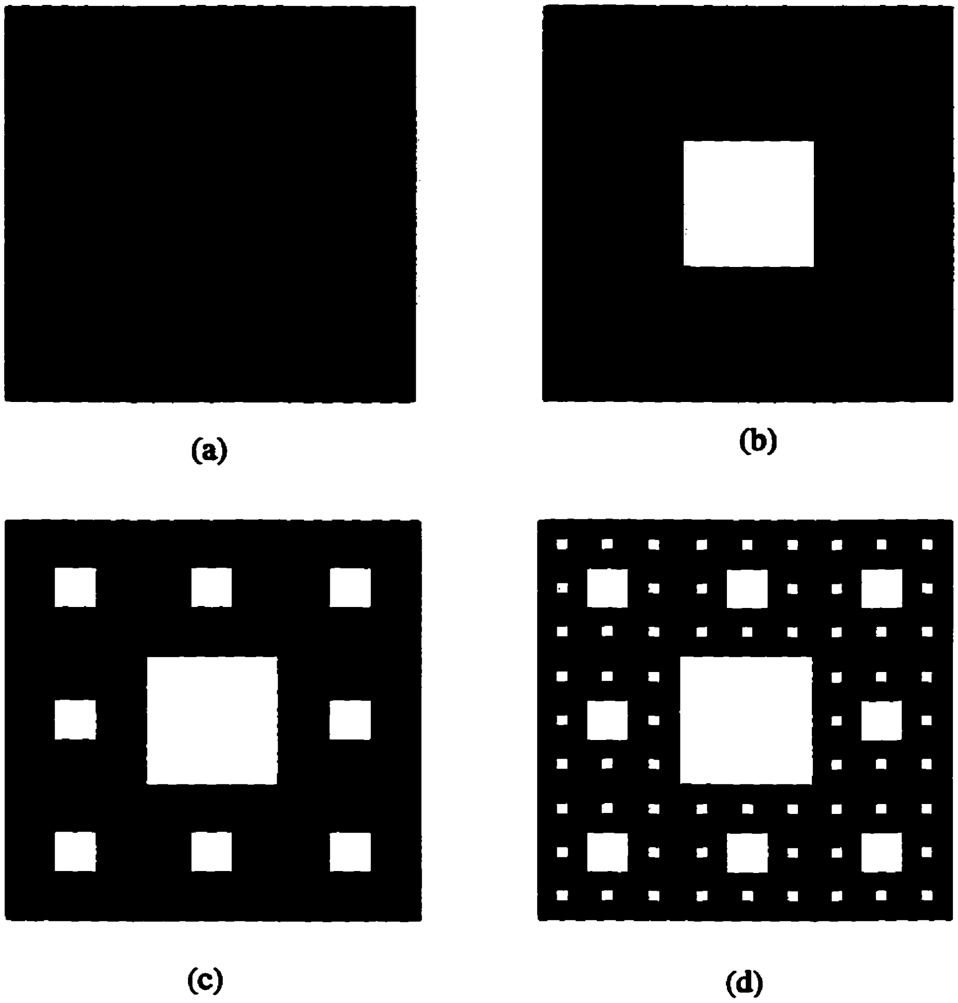


Fig. 2.1 Generation of the Sierpinski blanket. (a) The initiator (step zero). (b) The generator (step one). (c) Step two. (d) Step three.

2.1.2 Fractal Objects in Nature

The strict self-similarity property described in the previous section only exists in mathematical fractal objects; many objects in nature are not strictly self similar, but they are statistically self similar. For example, the northeast coastline of Canada shown in Fig. 2.2 is very irregular. It contains a multitude of bays, inlets, and peninsulas. Comparing two different-scale maps of the northeast of Canada, we will find a single bay or peninsula in the lower scale map might consist of many subbays and subpeninsulas in the larger scale map. In fact, regardless of the scale that one uses to measure a bay, there exists a higher resolution or larger scale under which more bays and peninsulas will be shown. This is a kind of self similarity. Although they are not strictly self similar, they are statistically self similar.

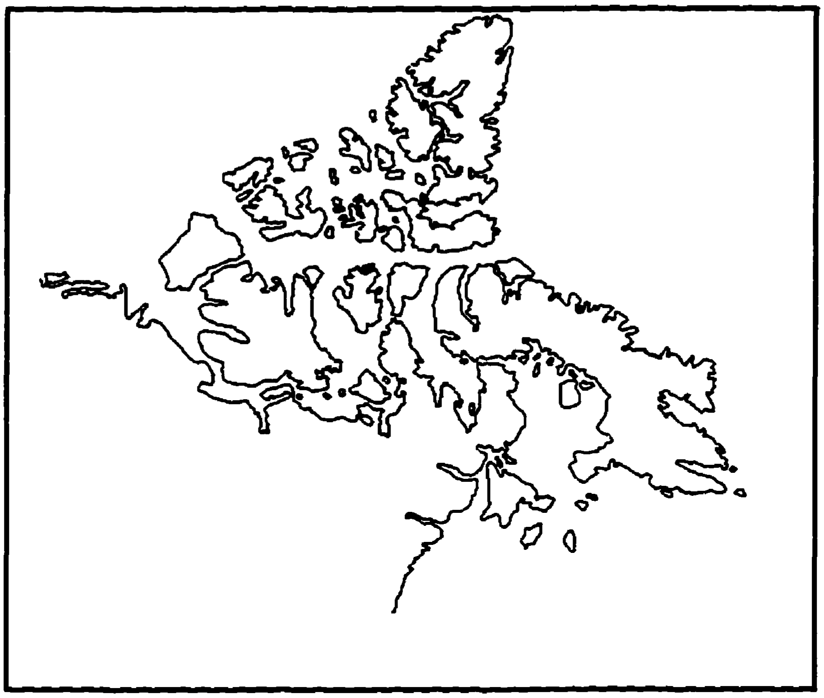


Fig. 2.2 The rugged border of the northeastern part of Canada.

A strange attractor in a chaotic dynamical system is another example. The Hénon attractor is used as a case which is shown in Fig. 2.3a. In Fig. 2.3a, in order to see the self similarity, we focus on the dotted box, the range of x is from -1 to -0.5, the range of y is from 0.3 to 0.37. You notice that there are two dotted "lines" in the box. This portion of the attractor is zoomed, the lower dotted "line" in Fig. 2.3a consists of two other dotted "lines" in Fig. 2.3b, and the upper dotted "line" in Fig. 2.3a consists of four other dotted "lines" in Fig. 2.3b. Furthermore, if we expand one of the dotted lines in Fig. 2.3b, a similar phenomenon will occur. So, the Hénon attractor also has the self-similarity property. One of the differences between the Koch's curve and a strange attractor is that the "lines" in a strange attractor are not real lines, they actually consist of many points that appear as lines due to the limitation of the resolution or scale. Furthermore, the densities of the points in different places of the lines may be very different.

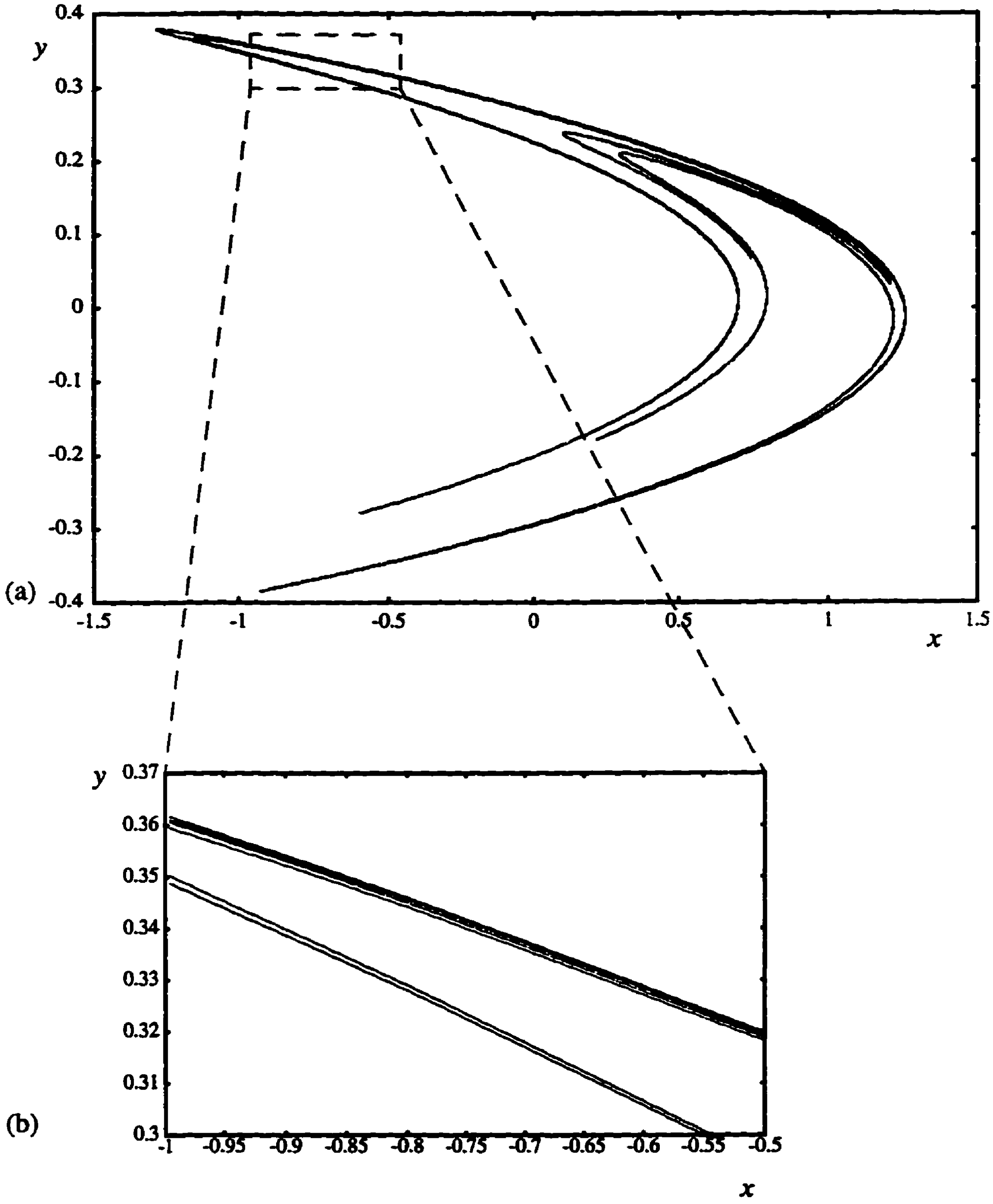
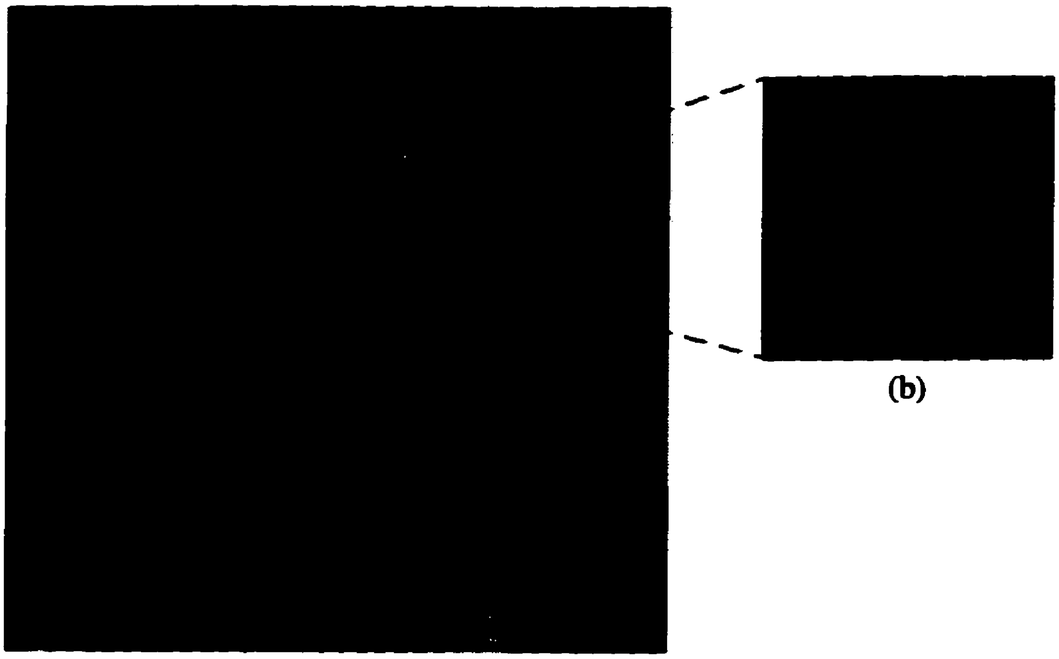


Fig. 2.3 (a) The Hénon attractor. (b) A zoomed portion of the Hénon attractor.

Spatial signals, such as images shown in Fig. 2.4, can also be modelled as fractal sets. A dithered image (Fig. 2.4b) actually consists of a finite number of picture elements (pixels). Each grey level can be considered as a density of points in the strange attractor. The arrangements of some groups of the pixels might be similar to other groups. So the images can also be considered as statistically self-similar objects.



(b)

(a)

Fig. 2.4 (a) A picture of Lena consisting of 256x256 pixels, 8 bits each, and (b) its zoomed part.

2.1.3 Common Geometric Properties of Fractal Sets

From the above examples of fractal sets, involving both theoretical (mathematical) fractal objects and natural fractal objects, we can see that fractal sets exhibit the following irregular property:

1) Fractal sets have scale invariance property.

2) Fractal sets are self similar or statistically self similar.

3) Fractal sets cannot be described by traditional geometry.

4) Most of fractal sets cannot be measured properly in integer dimensional space.

Most of the fractal sets in nature and applied sciences are not strict fractals. When the scale level reduces to a molecule or atomic level, fractal sets do not exist, so the ideal fractal only exists in the theoretical study.

2.2 Fractal Dimensions

We know that the length of the Koch curve is infinite, so we cannot measure it in one dimensional space. We also know that Koch's curve does not take any area when measured the Koch's curve in two dimensional space, so we also cannot effectively measure the Koch's curve. The same thing happens when measuring the Sierpinski blanket.

For the statistically self-similar objects, the measuring problem in integer dimensional spaces will occur again. For example, if we try to measure the length of the coastline of northeast of Canada which is shown in Fig. 2.2. The obvious way that one can measure the coastline is by piecewise linear approximation. Assume that it is possible to get an accurate approximation of the length of the curve in Fig. 2.2, consider what will happen if we acquire a map at larger scale. As we have already seen, more details such as sub-bay and sub-peninsula will be visible. These details have to be taken into account, so the problem becomes: in doing so, the approximate measure of the length of the coastline will increase without bound as the scale of the map becomes larger and larger. Figure 2.5 shows the empirical data representing the length of the coastline of northeast of Canada at the different scales which substantiates the measuring problem. In Fig. 2.5, the total length of the coastline L is measured in small ruler size r. The total length resulting from this measurement is plotted on a log-log scale against the inverse of the ruler size. As the length of the coastline is measured using smaller and smaller ruler size, the total length increases without bound.

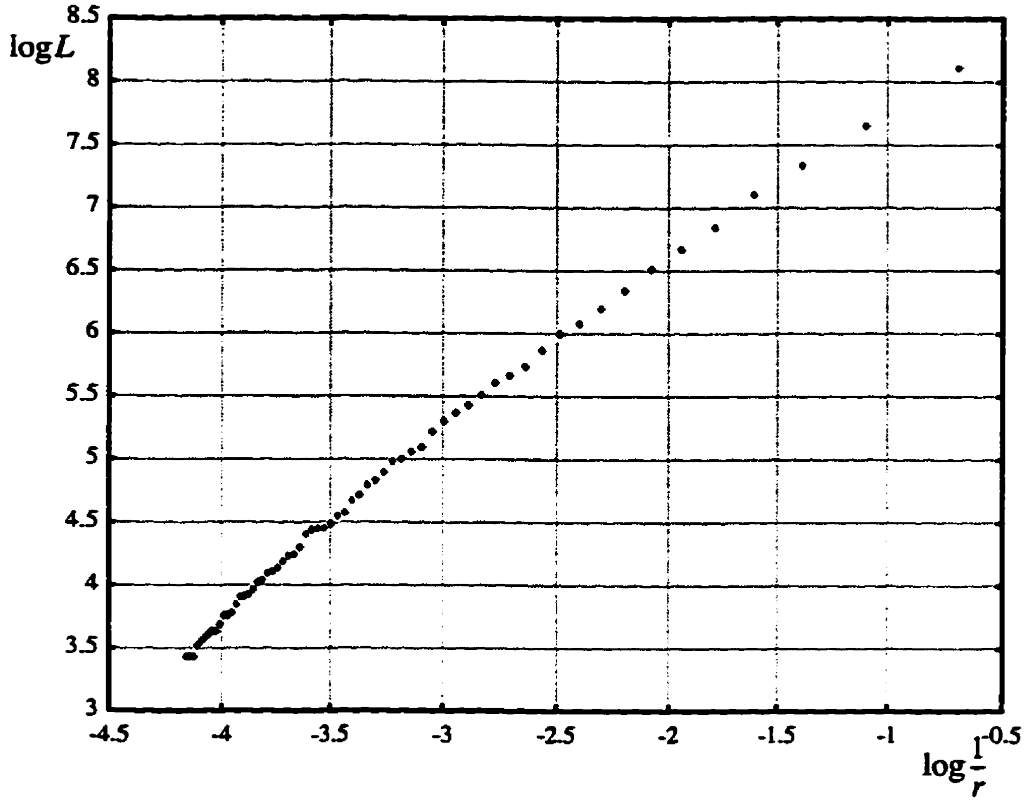


Fig. 2.5 Log-log diagram for the total length of the northeast coast of Canada: L versus the inverse of the ruler size r.

These observations indicate that integer dimensions are not sufficient to describe complicated fractal sets, *i.e.*, they cannot describe the degree to which the fractal sets fill the space. Furthermore, the degree of roughness of two fractal sets cannot be compared in integer dimensional spaces.

Is there a dimension that can be used to measure the Koch curve, the Sierpinski blanket and the irregular coastline? The answer is positive. In the following section, we will see that the dimension is fractional.

The irregularity of the fractal sets makes them different from traditional smooth sets, but how could we measure the degree of the irregularity? The fractal dimension provides an objective tool. The importance of the fractal dimension is that it can be defined not only theoretically, but also through experimental data. A comprehensive coverage of fractal dimensions has been provided by Kinsner [Kins94a]. The next several subsections will summarize several of the commonly used fractal dimensions.

2.2.1 Self-Similarity Dimension

Consider a bounded set A in the Euclidean n dimensional space. The set A is said to be self similar when A is the union of $N(r)$ distinct (non-overlapping) copies of itself, each of which has been scaled down by a ratio r in all coordinates. The similarity dimension of A is given by the following power-law relation,

$$N(r) \sim r^{D_s} \tag{2.2}$$

$$D_s = \frac{\log N(r)}{\log(1/r)} \tag{2.3}$$

Koch's curve shown in Table 2.1 is not a line because its length is infinite, though it is continuous and spans a finite distance. At r=1/3 and $N(r)$=4, the self-similarity dimension of Koch's curve is

$$D_s = \frac{\log 4^n}{\log 3^n} \approx 1.2619 \tag{2.4}$$

Similarly, the self-similarity dimension of the Sierpinski blanket shown in Fig. 2.1 is

$$D_s = \frac{\log 8}{\log 3} \approx 1.8927 \tag{2.5}$$

2.2.2 Hausdorff Dimension

Self-similarity dimension is suitable only for strictly self-similar fractal sets. However, the majority of fractals are not exact, or self similar, and they may even be random. So the Hausdorff dimension plays an important role. The Hausdorff dimension can be approximated by experimental data.

The way of measuring the Hausdorff dimension of an object is by defining a covering set of *volume elements* (*vels*) with vel size r and then counting the number of vels, $N(r)$, required to cover the object completely. The covering vels can be circles or square boxes. The Hausdorff dimension is defined as follows

$$D_H = \lim_{r \to 0} \frac{\log N(r)}{\log(1/r)} \tag{2.6}$$

If the vels are not overlapping and square, the dimension is called the box-counting dimension.

The Hausdorff dimension is one kind of morphological dimension [Kins96]. The morphological dimension uses the number of the geometrical covering vels as a basic measure. The disadvantage of this measurement is that the non-uniform property of the fractal set is not taken into account. At certain level *r*, the morphological dimension ignores the difference of some covering sets which have different densities. Consequently, the morphological dimension cannot accurately describe complex objects such as strange attractors in chaotic dynamical systems and non-stationary signals in various practical applications. The information dimension is used to overcome this disadvantage.

2.2.3 Information Dimension

The Shannon entropy is widely used to reflect the complexity of objects. It is the amount of the information needed to specify the state of a system to an accuracy of *r*. Let us consider covering a fractal object with *N*(*r*) vels. The size of the vels is *r*, where *r* is either radius or diameter or some other estimate of the size. The frequency in the fractal object may be distributed non-uniformly. The Shannon entropy is defined as

$$H_r = -\sum_{j=1}^{N(r)} p_j \log p_j \tag{2.7}$$

The probability of p_j is the relative frequency n_j with which the fractal enters (intersects) the *j*-th vel of the covering to the total number *N* of intersects of the fractal with all the vels

$$p_j = \lim_{N \to \infty} \frac{n_j}{N} \tag{2.8}$$

where

$$N = \sum_{j=1}^{N(r)} n_j \tag{2.9}$$

Based on the Shannon entropy, the information dimension is defined as

$$D_I = \lim_{r\to 0} \frac{H_r}{\log(1/r)} = \lim_{r\to 0} \frac{-\sum_{j=1}^{N(r)} p_j \log p_j}{\log(1/r)} \tag{2.10}$$

The correlation dimension is another fractal dimension which also has major advantages over the morphological dimension.

2.2.4 Correlation Dimension

As described in Section 2.2.3, assume the following power-law relationship holds between the sum of *squared* probabilities over all the vels with size r

$$\sum_{j=1}^{N(r)} p_j^2 \sim r^{D_C} \tag{2.11}$$

The correlation dimension is defined as

$$D_C = \lim_{r\to 0} \frac{\log\left(\sum_{j=1}^{N(r)} p_j^2\right)}{\log(1/r)} \tag{2.12}$$

The ensemble has the meaning of a correlation between pairs of neighbouring points on the fractal set [Kins96].

2.3 Is a Single Dimension Sufficient?

Suppose a fractal set F_M is made up by merging two fractal sets F_1 and F_2 (nonintersecting) each with a different fractal dimension. Assume that the fractal sets F_1 and F_2 have the box-counting dimensions $D^{(1)}$, $D^{(2)}$ ($D^{(1)} > D^{(2)}$), respectively.

$N_1(r)$, $N_2(r)$ are the numbers of vels to cover the fractal sets F_1 and F_2, respectively, then the following power-law relationships hold:

$$N_1(r) = r^{-D^{(1)}} \qquad (2.13)$$

$$N_2(r) = r^{-D^{(2)}} \qquad (2.14)$$

then

$$D^{(1)} = \lim_{r \to 0} \frac{\log N_1(r)}{\log(1/r)} \qquad (2.15)$$

$$D^{(2)} = \lim_{r \to 0} \frac{\log N_2(r)}{\log(1/r)} \qquad (2.16)$$

When the two fractal sets merge without intersecting, we use vels to cover the new fractal set. The number of the vels will be $N_M(r) = N_1(r) + N_2(r)$. So the box-counting dimension of the merged fractal sets is

$$D^{(M)} = \lim_{r \to 0} \frac{\log N_M(r)}{\log(1/r)} \qquad (2.17)$$

or

$$D^{(M)} = \lim_{r \to 0} \frac{\log(N_1(r) + N_2(r))}{\log(1/r)} \qquad (2.18)$$

Consequently, we get

$$D^{(M)} = \lim_{r \to 0} \frac{\log\left(r^{-D^{(1)}} + r^{-D^{(2)}}\right)}{\log(1/r)} \qquad (2.19)$$

The above equation tells us that $D^{(M)} = D^{(1)}$. Although the merged fractal set F_M and F_1

are very different, a single dimension cannot distinguish them.

We use an example to explain this. The morphological dimension of Koch's curve is 1.2619 and the dimension of Minkowski's curve is 1.5. Koch's curve and Minkowski's curve are merged as is shown in Fig. 2.6.

We use the box-counting dimension to measure the dimension of the merged set, due to experimental error, the dimension of the merged set is 1.49, which is slightly different from 1.5. This example tells us that the dimension of the two merged fractal set is equal to the dimension of the larger one. From this example, we also see a single dimension really cannot distinguish two fractal sets.

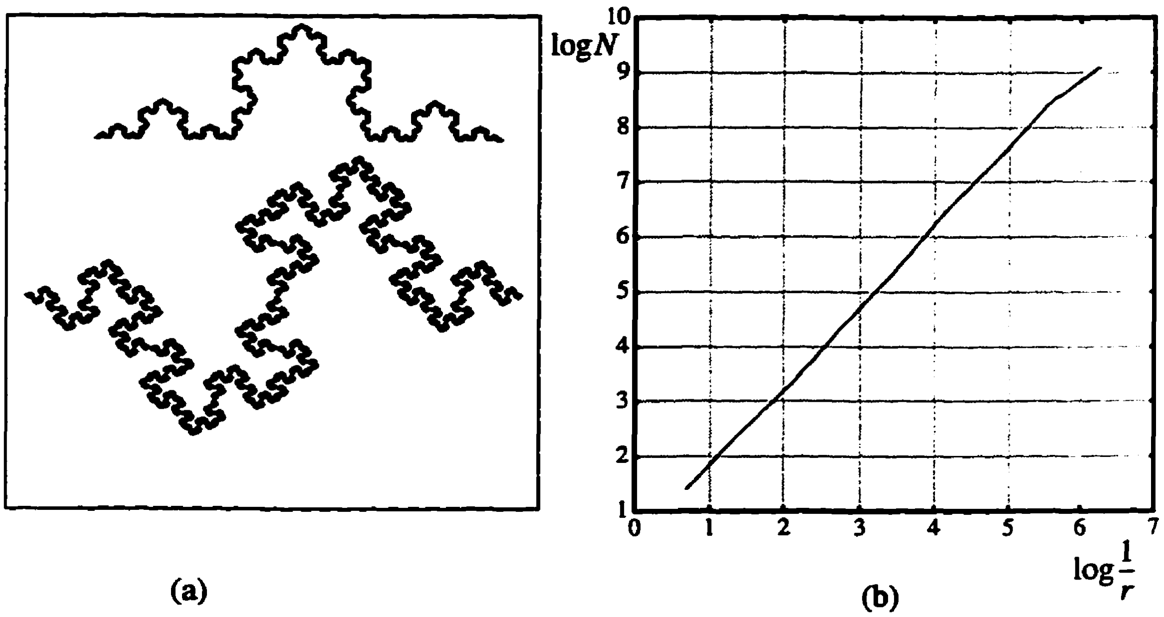


Fig. 2.6 (a) The merged fractal set. (b) The log-log plots for the merged fractal.

Inductively, any finite number of fractal sets are merged to form a new fractal set; the dimension will also equal the maximum value of the dimensions of the fractal sets. In this sense, the single fractal dimension is not sufficient to characterize the composite fractal sets. A multifractal set can be considered as a complex fractal set obtained from the merging of an infinite number of single fractal sets. In order to characterize the multifractal set fully, an infinite number of dimensions needs to be introduced.

2.4 Multifractals

2.4.1 Rényi Entropy

The Rényi entropy is a generalization of the Shannon entropy. We know that the Shannon entropy is just a single value. It is suitable for describing the amount of the information in a random process. However, Rényi realized that Shannon entropy was inadequate for describing the distribution of different densities in the fractal sets.

Similar to the setting used in determining the information dimension and the correlation dimension, consider a covering of $N(r)$ vels, each with vel size r. Assume that the j-th vel is intersected by a fractal set with frequency n_j, the probability of j-th vel is

$$p_j = \lim_{N \to \infty} \frac{n_j}{N} \tag{2.20}$$

where N is the total number of points in all the vels

$$N = \sum_{j=1}^{N(r)} n_j \tag{2.21}$$

Rényi has generalized an entropy of order q expressed as

$$H_q = \frac{1}{q-1} \log \sum_{j=1}^{N(r)} p_j^q \qquad -\infty < q < \infty \tag{2.22}$$

Since q could be any real number, the Rényi entropy has an infinite number of possible values. When q goes to positive infinity, the entropy reflects the contribution of the largest probability p_j. When q goes to negative infinity, the entropy reflects the contribution of the smallest probability p_j. Each value of q reflects the contribution from some group of p_j. It can be shown [Kins94] that when q is 1, the Rényi entropy is the first-order entropy, *i.e.,* Shannon entropy which is related to the information dimension. When q equals to 2, the Rényi entropy is the second-order entropy which is related to the correlation dimension.

2.4.2 Rényi Dimension

The Rényi entropy as a measure, changes as the vel size changes. Assuming the Rényi entropy follows the following power-law relationship

$$H_q \sim r^{D_q} \tag{2.23}$$

the Rényi dimension is defined as follows

$$D_q = \lim_{r\to 0} \frac{H_q}{\log r} = \lim_{r\to 0} \frac{1}{q-1} \frac{\log\left(\sum_{i=1}^{N(r)} p_j^q\right)}{\log r} \tag{2.24}$$

In this thesis, the Rényi dimension is also denoted by $D(q)$. When q=0, the Rényi dimension reduces to the similarity dimension D_s and the Hausdorff dimension D_H

$$D_0 = \lim_{r\to 0} \frac{1}{0-1} \frac{\log \sum_{i=1}^{N(r)} p_j^0}{\log r}$$

$$= \lim_{r\to 0} -\frac{\log N(r)}{\log r}$$

$$= \lim_{r\to 0} \frac{\log N(r)}{\log \frac{1}{r}} \tag{2.25}$$

When q=1, with the help of the equation [Kins94b]

$$\log \sum_{i=1}^{N(r)} p_j^q \sim (q-1) \sum_{i=1}^{N(r)} p_j \log p_j \tag{2.26}$$

the Rényi dimension reduces to the information dimension D_I

$$D_1 = \lim_{r \to 0} \frac{q-1}{q-1} \frac{\log \sum_{i=1}^{N(r)} p_j \log p_j}{\log r}$$

$$= \lim_{r \to 0} \frac{\log \sum_{i=1}^{N(r)} p_j \log p_j}{-\log r}$$

$$= \lim_{r \to 0} \frac{H_I}{\log (1/r)} \qquad (2.27)$$

When q=2, the Rényi dimension reduces to the correlation dimension D_C

$$D_2 = \lim_{r \to 0} \frac{1}{2-1} \frac{\log \sum_{i=1}^{N(r)} p_j^2}{\log r}$$

$$= \lim_{r \to 0} \frac{\log \sum_{i=1}^{N(r)} p_j^2}{\log r}$$

$$= D_C \qquad (2.28)$$

Therefore, the Rényi dimension includes the Hausdorff dimension, the information dimension and the correlation dimension as special cases.

2.4.3 Formalism of the Geometric Description of Multifractals

We now present the general scheme of the geometric description of multifractals [Kins94b]. Consider generating a nonuniform fractal (*i.e.*, rescaled regions of different sizes r_j) with inhomogeneous measures (*i.e.*, regions with different probabilities p_j) at each rescaled region, recursively. An example of such a process at its commencement is shown below:

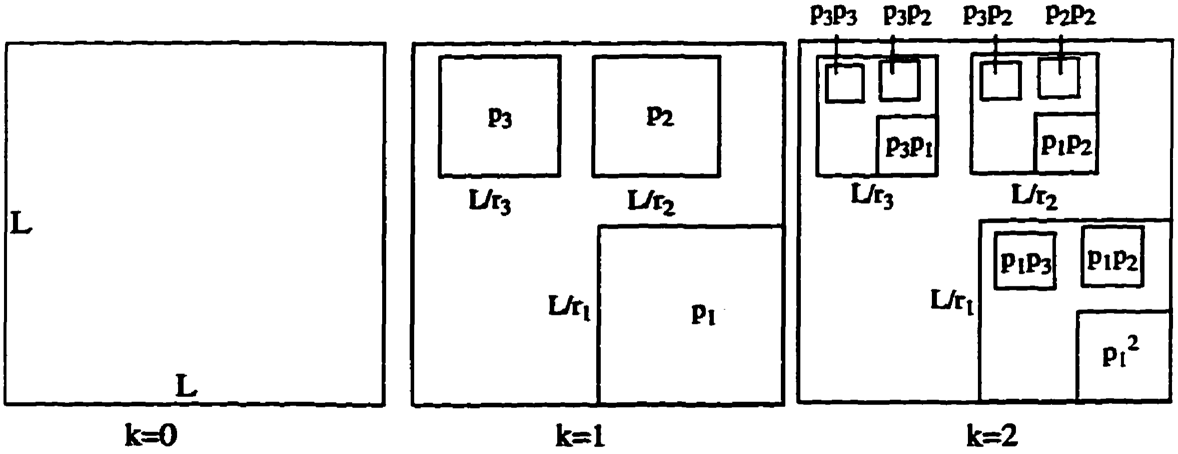


Fig. 2.7 An illustration of a recursive process of generating a nonuniform fractal.

We know that for a uniform fractal with homogenous measures, the distribution of the probabilities p for a given vel with size r satisfies the following power-law relation

$$p = r^{D_s} \tag{2.29}$$

However, for a nonuniform fractal with inhomogeneous measures, the relationship between the probability p_j in j-th vel and the vel size r_j satisfies

$$p_j(r_j) = r_j^{\alpha_j} \qquad -\infty < \alpha_j < +\infty \tag{2.30}$$

where α_j (j=1, 2, ..., $N(r)$) is a *scaling parameter* which is a noninteger. The scaling parameter depends on the selected region of the measure. It reflects the strength of the local singularity of the measure, and describes how the variation of the measure versus the variation of vel size r departs from linearity. For decreasing r, the measure decreases much faster for large than for small α. Large values of α characterize the rarefied subsets, where small values of α represent the dense subsets. For a particular nonuniform fractal with inhomogeneous measures, a limited range $\alpha_{min} < \alpha < \alpha_{max}$ is obtained. The value of $\alpha_{max} - \alpha_{min}$ roughly measures the degree of nonuniformity. The scaling parameter is also called the Hölder exponent or singularity dimension.

Besides the singularity dimension α, we are also interested in how many vels have the same singularity α. In the example shown in Fig. 2.7, let us specify the probabilities: $p_1=1/2, p_2=1/6, p_3=1/3$. Therefore, at the second step there will be two vels with probability of 1/6, two vels with probability of 1/12, another two vels with probability of 1/18. There will also be one vel with probability of 1/9, one vel with probability of 1/36, another one vel with probability of 1/36. The illustration is shown in Fig. 2.8. In this example, we just introduce three probabilities p_1, p_2, p_3 at the beginning of the recursive process. If the number of the probabilities increases, and some of the probabilities may be the same, after enough recursion, the result can be very complicated.

In general, the number of the vels with a specific α satisfies the following power-law relation

$$N_\alpha(r) \sim r^{-f(\alpha)} \tag{2.31}$$

where $f(\alpha)$ is the Mandelbrot dimension of the α subset.

The exponent α is analogous to energy, while $f(\alpha)$ is analogous to the entropy as a function of the energy, and is reminiscent of plots in thermodynamical systems [StMe88].

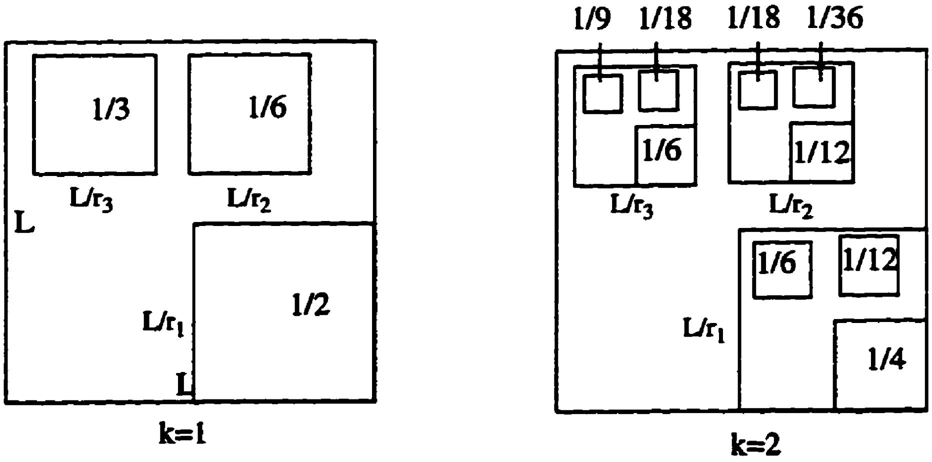


Fig. 2.8 An simple example showing $f(\alpha)$.

It can be shown [AtSV88] that singularity dimension α and the Mandelbrot dimension $f(\alpha)$ are related to the Rényi dimension D_q. The singularity dimension can be obtained by taking the derivative with respect to q

$$\alpha(q) = \frac{d}{dq}[(q-1)D_q] \tag{2.32}$$

and the Mandelbrot dimension can be obtained by

$$f[\alpha(q)] = q\alpha(q) - (q-1)D_q \tag{2.33}$$

2.4.4 Characteristic Features of the Functions D_q, $\alpha(q)$, and $f(\alpha(q))$

It has been shown [HePr83] that the Rényi dimension is monotonically decreasing

$$D_{q'} \le D_q \qquad\qquad q' \le q \tag{2.34}$$

The visualization of the Rényi dimension is given in Fig. 2.9. If the fractal is strictly self similar, then $D_{q'} = D_q$. Thus, the range of the variation of D_q can indicate the complexity of the fractal.

It also can be shown [HePr83] that the singularity dimension $\alpha(q)$ is also monotonically decreasing, and that the singularity dimension α is equal to the Rényi dimension D_q at $q = \pm\infty$. If we rewrite Eq. 2.32 in the following form

$$\alpha(q) = (q-1)\frac{dD_q}{dq} + D_q \tag{2.35}$$

It is easy to see that when q=1, the singularity dimension is equal to the Rényi dimension D_q. So the two curves intersect at the q=1. The visualization of the singularity dimension is also shown in Fig. 2.9.

The function of $f(\alpha)$ shows the following universal properties [AtSV88]

$$\frac{df}{d\alpha} = q \tag{2.36}$$

$$\frac{d^2 f}{d\alpha^2} < 0 \tag{2.37}$$

It can be seen that the Mandelbrot dimension $f(\alpha)$ assumes positive values in the bounded interval $[\alpha_{min}, \alpha_{max}]$. It reach its maximum value at $q = 0$ where $\frac{df}{d\alpha} = 0$; at this point, the value of the Mandelbrot dimension is equal to D_0 which is the Hausdorff dimension. As one can see, at the both ends of α_{min} and α_{max} of the domain of the function $f(\alpha)$, the derivative $\frac{df}{d\alpha} = \pm\infty$, where the parameter q assumes the values $q = \pm\infty$. The value of the $f(\alpha)$ at α_{max} corresponds to the value $q = -\infty$, where this derivative is negative, and the value of $f(\alpha)$ at α_{min} corresponds to the value $q = +\infty$, where this derivative is positive. So, the Mandelbrot spectrum has a characteristic plot with a single hump. The illustrative diagram of $f(\alpha)$ is shown in Fig. 2.10.

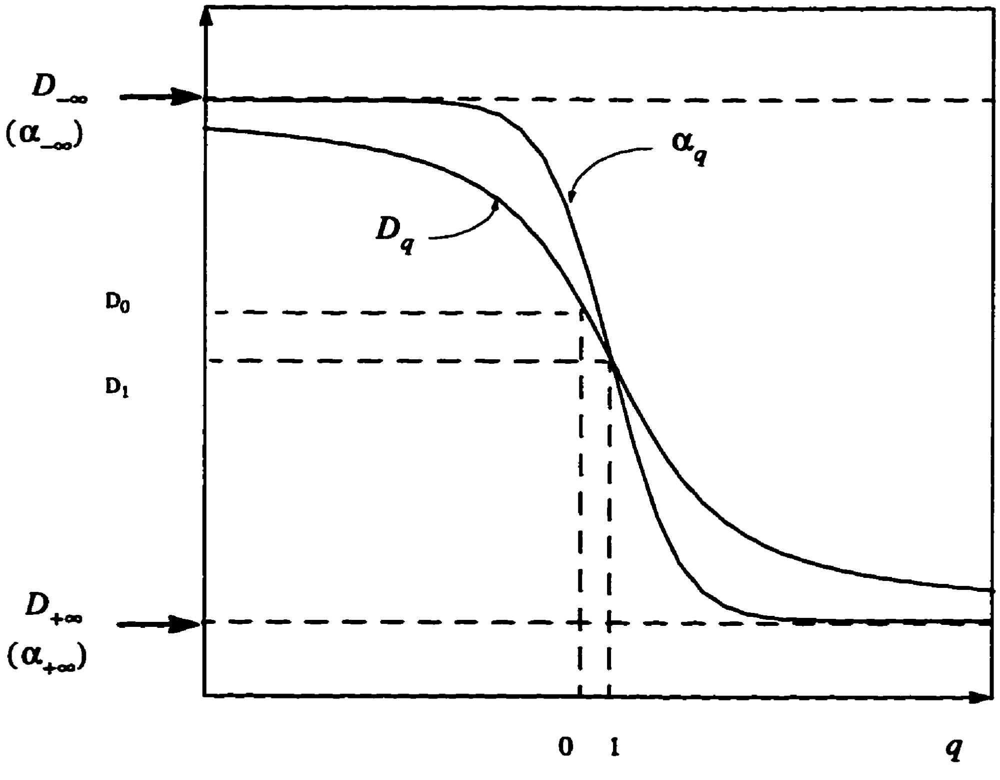


Fig. 2.9 Visualization of $D(q)$ and $\alpha(q)$.

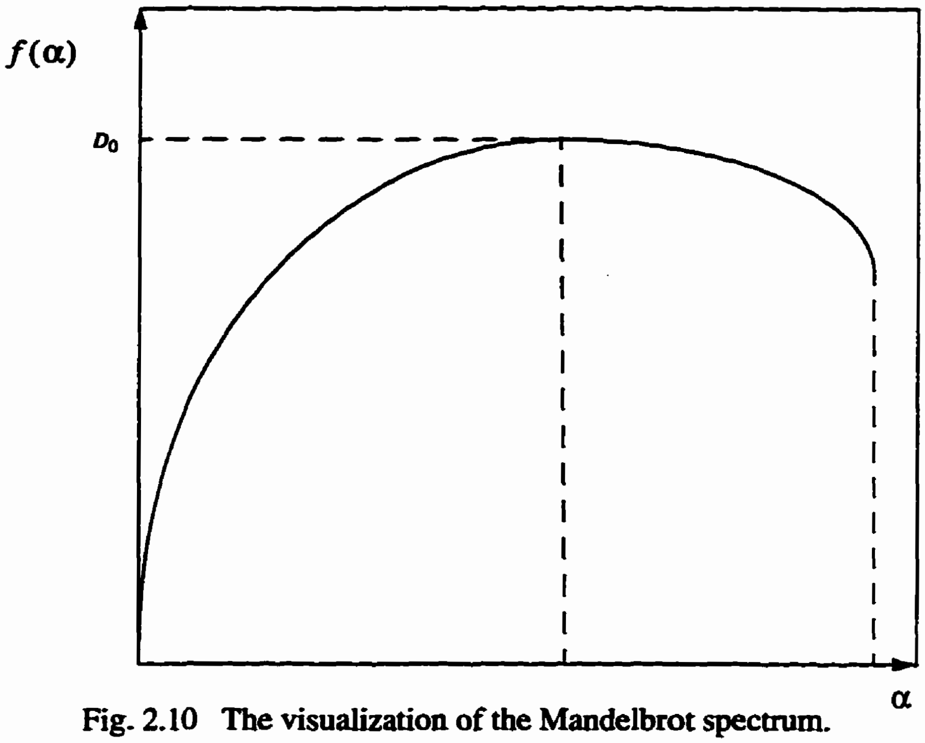


Fig. 2.10 The visualization of the Mandelbrot spectrum.

2.5 Overview of Chapter 2

In this chapter, both mathematical fractal objects and natural fractal objects are described. Several commonly-used-fractal dimensions such as self-similarity dimension, the Hausdorff dimension, the information dimension, and the correlation dimension are summarized. Since the limitation of the single fractal dimension (see Section 2.4), multi-fractal objects and multifractal measures are presented. Multifractals are the main theme in this thesis, they play a central rule in later chapters.

CHAPTER 3

STRANGE ATTRACTORS AND THEIR APPROXIMATIONS OF MULTIFRACTAL MEASURES

3.1 Chaotic Dynamical Systems

A dynamical system can be defined as a deterministic mathematical prescription for evolving the state of a system forward in time. The time here either may be a continuous variable, or else it may be a discrete integer-valued variable. An example of a dynamical system in which time (denoted t) is a continuous variable is a system of N first-order, autonomous, ordinary differential equations

$$\begin{cases} \dfrac{dx^{(1)}}{dt} = F_1\left(x^{(1)}, x^{(2)}, \ldots, x^{(N)}\right) \\ \dfrac{dx^{(2)}}{dt} = F_2\left(x^{(1)}, x^{(2)}, \ldots, x^{(N)}\right) \\ \quad\quad\quad\cdot \\ \quad\quad\quad\cdot \\ \quad\quad\quad\cdot \\ \dfrac{dx^{(N)}}{dt} = F_N\left(x^{(1)}, x^{(2)}, \ldots, x^{(N)}\right) \end{cases} \tag{3.1}$$

which we also can write in vector form as

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{F}[\mathbf{x}(t)] \tag{3.2}$$

where $\mathbf{x}(t)$ is a N-dimensional vector. This is a dynamical system because, for any initial state of the system $\mathbf{x}(0)$, we can in principle solve the equations to obtain the future system state $\mathbf{x}(t)$ $(t > 0)$. Figure 3.1 shows the path followed by the system state as it evolves with time in a case where N=3. The space $(x^{(1)}, x^{(2)}, x^{(3)})$ in the figure is referred to as *phase space*, and the path in phase space followed by the system as it evolves with time is referred to as an *orbit* or *trajectory*.

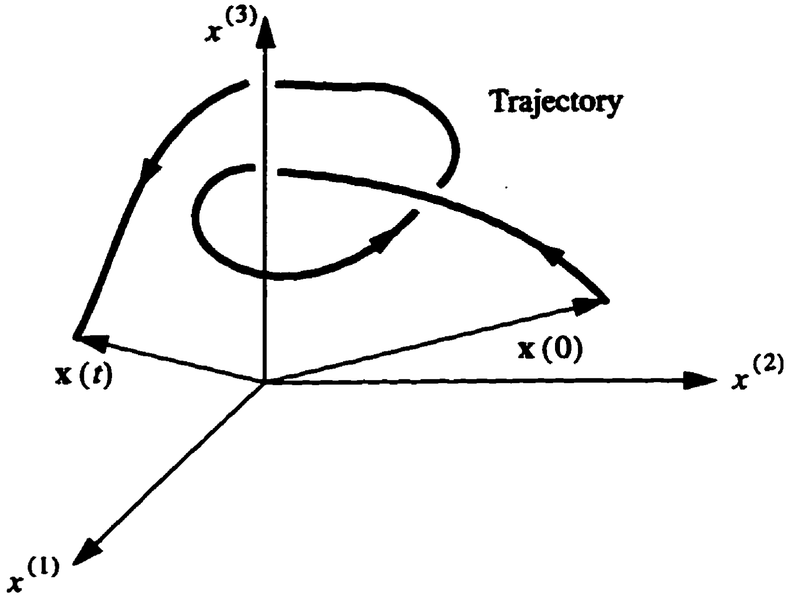


Fig. 3.1 An orbit in a three-dimensional (N=3) phase space.

In the case of discrete, integer-valued time (with n denoting the time variable, n = 0, 1, 2, ...), an example of a dynamical system is a map, which we write in vector form as

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

where $\mathbf{x}_n$ has N components, $\mathbf{x}_n = (x_n^{(1)}, x_n^{(2)}, ..., x_n^{(N)})$ and $\mathbf{M}$ is a map. Given an initial state $\mathbf{x}_0$, we obtain the state at time n = 1 by $\mathbf{x}_1 = \mathbf{M}(\mathbf{x}_0)$. Having determined $\mathbf{x}_1$, we can then determined the state at n = 2 by $\mathbf{x}_2 = \mathbf{M}(\mathbf{x}_1)$, and so on. Thus, given an initial condition $\mathbf{x}_0$, we generate an orbit (or trajectory) of the discrete time system: $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$, ... As we shall see, a continuous time system of dimensionality N can often be reduced to a discrete time map of dimensionality N-1 via the Poincaré section technique.

The dynamical systems fall into three classes: stable, unstable and chaotic. Many systems exhibit dynamical behaviour which can be fully analyzed mathematically. After some transient period such systems settle either into a periodic motion (a limit cycle) or into a steady state (a limit point). On the other hand, an unstable system has no limit cycle or limit point, and the trajectory has no bound.

However, there are many dynamical systems which are neither stable nor unstable but display random behaviour. Their trajectories are limited to a bounded region. We call these dynamical systems *chaotic dynamical systems*. They have not been appreciated until recently due to the complexity of the numerical solutions that were not attainable without fast digital computers. The chaotic dynamical systems are very sensitive to initial conditions. It is practically impossible (although possible in principle) to predict the future of the systems because any error, no matter how small, in either specifying the initial conditions or made during the numerical computation leads to erroneous predictions of the outcome at some future time, as shown in Fig. 3.2. the sensitivity to the initial conditions is often called the Lorenz's *butterfly effect* (a flapping of a butterfly's wings in the Amazon tropics could lead to a hurricane on other continents).

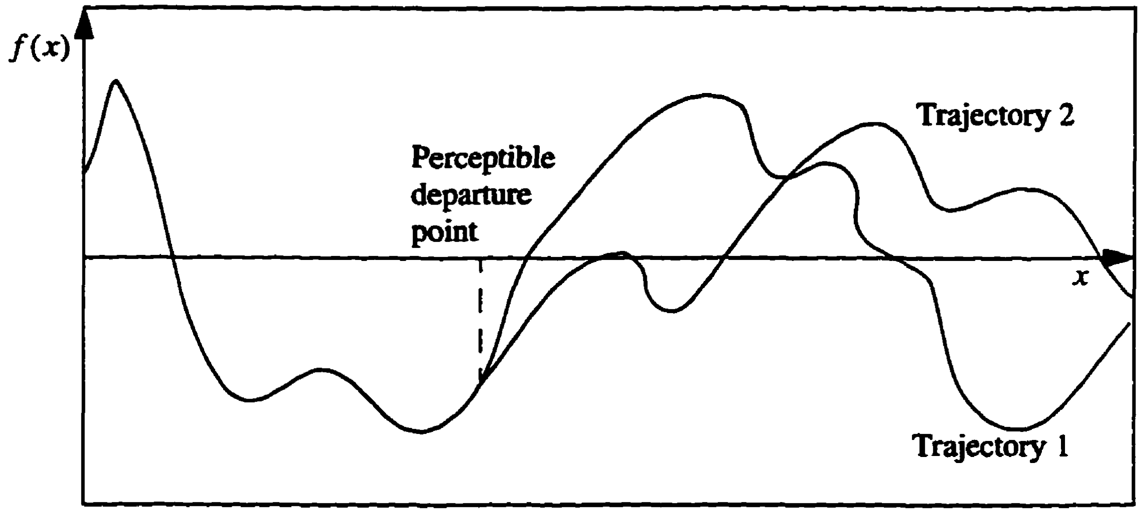


Fig. 3.2 The sensitivity to initial conditions in chaotic dynamical systems. (After [Kins94b])

In chaotic dynamical systems, the periodic points are *topologically dense*. This means that given a point in a system, one can find another point of the chaotic dynamical system in any infinitely small neighbourhood of the first point. This is like the indecomposibility of the rational set and irrational set. Given any two points, one can also find an orbit that comes arbitrarily close to both points, so the chaotic dynamical system is *topo-*

logically transitive [Deva92].

Although the trajectories of the chaotic dynamical systems are neither periodic nor even asymptotically-periodic, the systems are still deterministic (*i.e.*, they have equations governing their behaviour), and thus their behaviour has a sense of order and pattern. It is a very important concept in dynamics that chaotic dynamical systems typically can be characterized by the presence of attracting sets or strange attractors in the phase space. The strange attractor is not a simple geometrical object like a circle, torus, or even a manifold. It cannot be well characterized in any integer dimensional spaces. Actually the strange attractor is a very complicated object like the Cantor-type dust with a fractal dimension.

3.2 Examples of Strange Attractors

3.2.1 The Poincaré Section of the Rössler Attractor

In 1976, Otto E. Rössler found a chaotic dynamical system which is probably the most elementary geometric construction of chaos in continuous systems. We interpret the system as a collection of laws of motion for a point at the coordinates (x, y, z) in three-dimensional space. For any given initial coordinates (x_0, y_0, z_0), the system defines a unique trajectory which is parametrized by time t and satisfy the equations at all times. Denoting the coordinates of the trajectory by $(x(t), y(t), z(t))$ for time $t \geq 0$, the system is

$$\begin{aligned} \frac{d}{dt}x(t) &= -(y(t)+z(t)) & x(0) &= x_0 \\ \frac{d}{dt}y(t) &= x(t)+ay(t) & y(0) &= y_0 \\ \frac{d}{dt}z(t) &= b+x(t)z(t)-cz(t) & z(0) &= z_0 \end{aligned} \tag{3.4}$$

In the above equation, we fix the parameters a, b and c at $a = 0.2$, $b = 0.2$, $c = 5.7$ and set the initial condition of $(x_0, y_0, z_0) = (1, 0, 0)$. The trajectory is generated using the

Runge-Kutta method, with a fixed time step of 0.01. After discarding 10,000 steps to allow the trajectory to fall to the attractor, the Rössler attractor looks like a set of spirals in x-y plane distorted into the two directions as shown in Fig. 3.3.

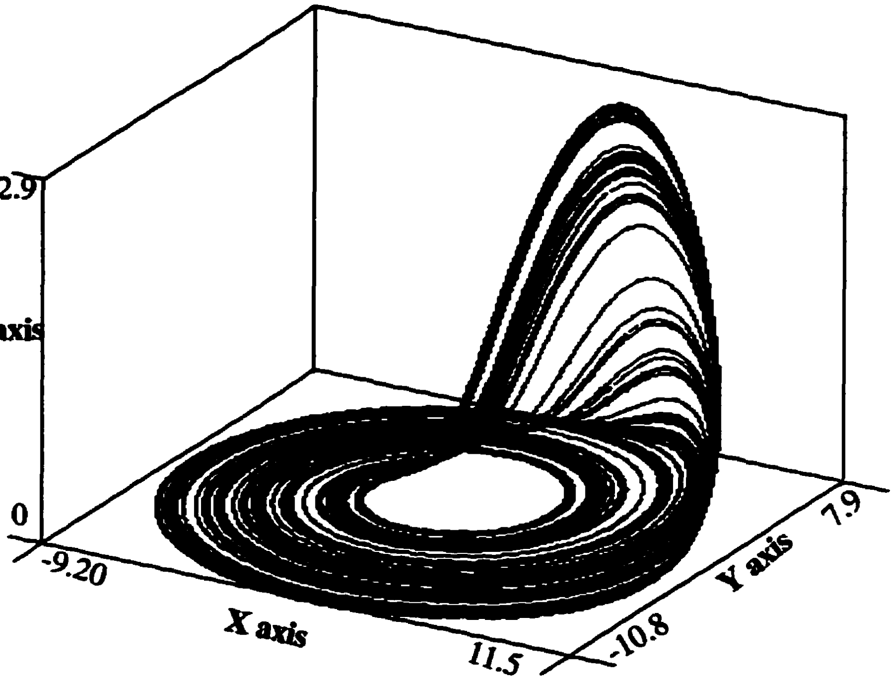


Fig. 3.3 The Rössler attractor.

We often use a part of a strange attractor to characterize it. A convenient subset is the Poincaré section of the strange attractor. In Fig. 3.4, a surface is intersected by a strange attractor. A point *A* from the trajectory of the strange attractor is also on the surface. Use *A* as an initial point, we follow the corresponding trajectory, as time goes, the section will be entered by the trajectory again at Point *B*. A Poincaré section consists of all the points from the trajectory of the system which pierce the surface, but are not tangential to it. Figure. 3.5 is the Poincaré section of the Rössler Attractor at $z = 22.0$.

There is a fundamental difference between the two images of the Rössler attractor: the trajectory shown in Fig. 3.3 is continuous, while the Poincaré section of the Rössler attractor shown in Fig. 3.5 is discontinuous everywhere. This means that the trajectory never intersects itself.

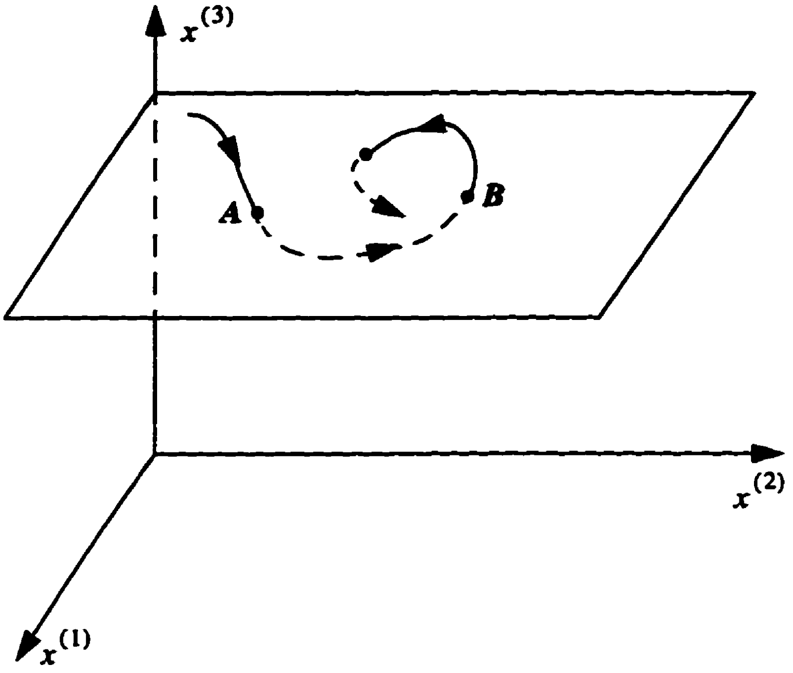


Fig. 3.4 An illustration of the Poincaré section.

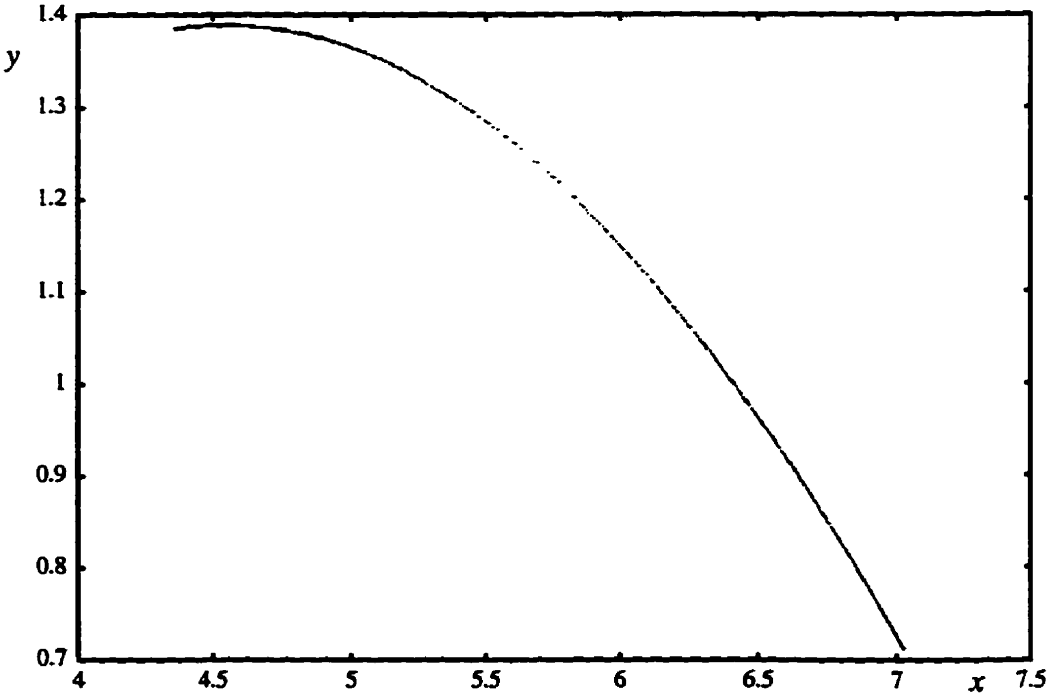


Fig. 3.5 The Poincaré section of the Rössler attractor at $z = 22.0$.

3.2.2 The Hénon Attractor

As an example of a chaotic dynamical system of discrete, integer-valued time, consider the strange attractor obtained from the two-dimensional Hénon map,

$$\begin{aligned} x_{n+1} &= a - x_n^2 + by_n \\ y_{n+1} &= x_n \end{aligned} \qquad (3.5)$$

We fix parameters a and b at $a = 1.4$, and $b = 0.3$. Initiating at (0, 0), and discarding 10,000 steps, the Hénon attractor is plotted in Fig. 3.6.

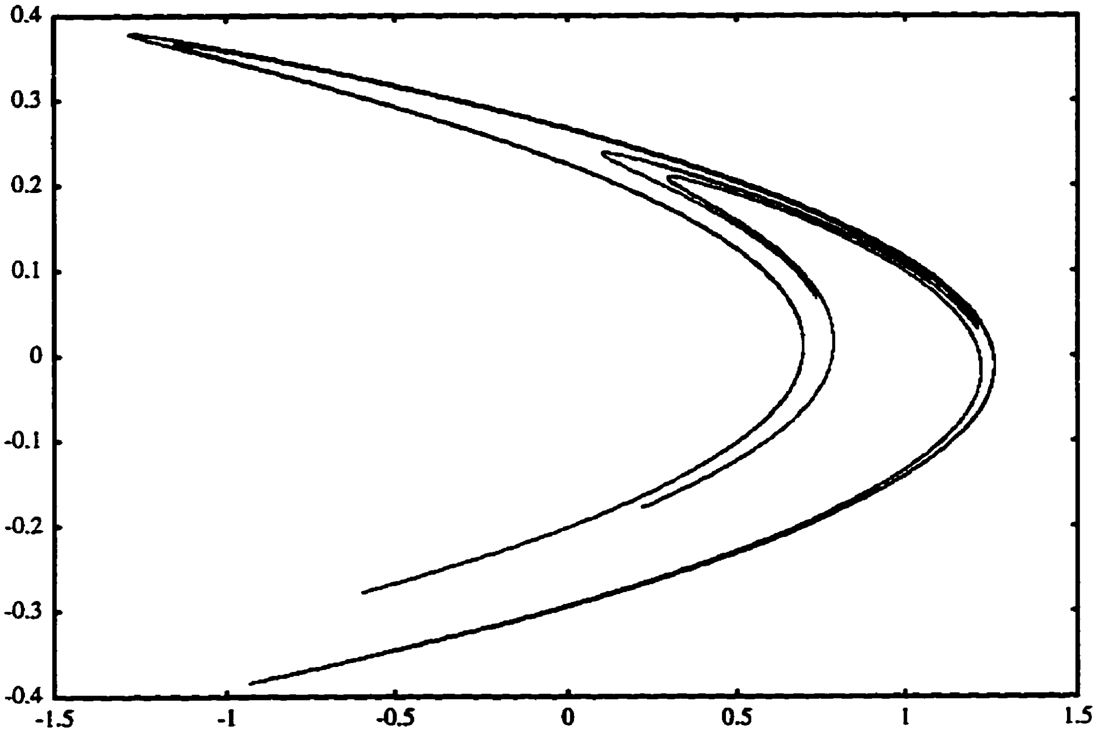


Fig. 3.6 The Hénon attractor.

3.2.3 The Ikeda Attractor

As another example of a chaotic dynamical system of discrete, integer-valued time, consider the strange attractor obtained from a plane-wave map [HaJo85]

$$g_{n+1} = a + R \exp\left[i\left(K - \frac{p}{1 + |g_n|^2}\right)\right] g_n \tag{3.6}$$

where g is a dimensionless intracavity complex field amplitude, a is a dimensionless input amplitude, $(1\text{-}R)$ is the input and output mirror intensity transmission function, K is the laser empty-cavity detuning, p is related to linear absorption per pass and $i = \sqrt{-1}$. The above map can be rewritten as follows if we assume $g_n = x_n + y_n i$

$$x_{n+1} = r + Rx_n \cos\left(K - \frac{a}{1 + x_n^2 + y_n^2}\right) + y_n \sin\left(K - \frac{a}{1 + x_n^2 + y_n^2}\right) \tag{3.7}$$

$$y_{n+1} = Rx_n \sin\left(K - \frac{a}{1 + x_n^2 + y_n^2}\right) + Ry_n \cos\left(K - \frac{a}{1 + x_n^2 + y_n^2}\right) \tag{3.8}$$

If we take K=0.4, p=5.5, a=0.85, R=0.9, starting at (0.5, 0.2), and discarding 10,000 steps, the Ikeda attractor emerges as that shown in Fig. 3.7.

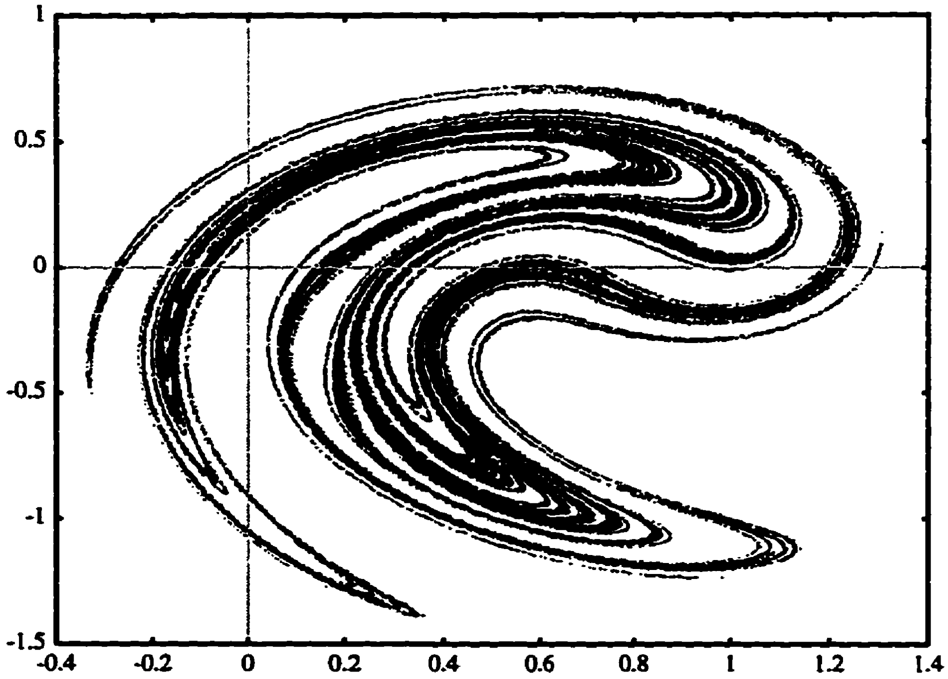


Fig. 3.7 The Ikeda attractor.

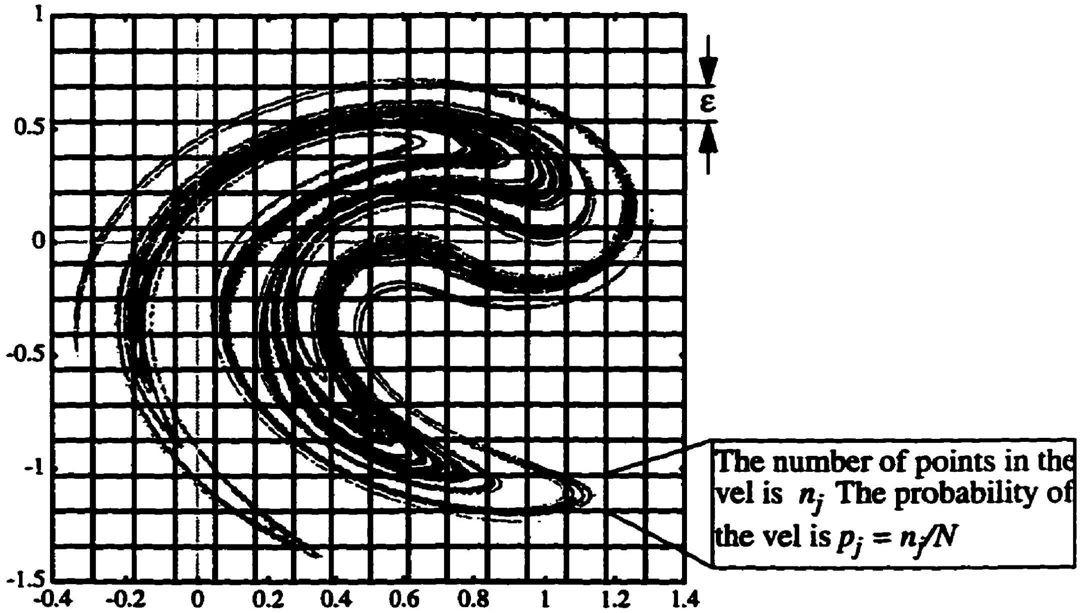
3.3 Characterization of Strange Attractors

3.3.1 Approximation of Multifractal Measures

Let us assume that a strange attractor is covered by N_ε volume elements (vels), each of size ε. The characterization of strange attractors is based on the probability of each covering vel V_j intersecting the points on the strange attractor. The probability p_j ($j = 1, 2, ..., N_\varepsilon$) is used to characterize the density distribution of points in each vel. Since the complete strange attractor consists of an infinite number of points, we cannot obtain the theoretical value of the probability; instead, the probability in each vel is approximated by a finite number of points intersected by the vel. Therefore, this study reduces to a finite number of points and finite size of vels.

Consider a covering as shown in Fig. 3.8, assume that the j-th vel is intersected by n_j points of the attractor and the total number of points in the strange attractor is N. We can then approximate the probability of the j-th vel in terms of its relative frequency

$$p_j = \lim_{N \to \infty} \frac{n_j}{N} \qquad\qquad N = \sum_{j=1}^{N_\varepsilon} n_j \qquad (3.9)$$



The number of points in the vel is n_j The probability of the vel is $p_j = n_j/N$

Fig. 3.8 An illustration of covering a strange attractor by nonintersecting vels.

3.3.1.1 Approximations of the Rényi Dimension

With the above assumption, the Rényi dimension D_q is approximated by

$$\tilde{D}_q = \frac{1}{q-1} \frac{\log \sum_{j=1}^{N_\varepsilon} n_j^q - q \log N}{\log \varepsilon} \tag{3.10}$$

If the vel size ε diminishes to infinitesimal, and the number of the points N increases to infinity, the estimated value is the theoretical value

$$\lim_{\substack{N \to \infty \\ \varepsilon \to 0}} \tilde{D}_q = D_q \tag{3.11}$$

This provides another way to study the property of the Rényi dimension. The total number of points and the distribution of these points in each vel determine the Rényi dimension D_q; consequently, $\alpha(q)$ and $f(\alpha(q))$ can also be approximated from the known property Eq. (2.32) and Eq. (2.33). $\alpha(q)$ and $f(\alpha(q))$ are also denoted as α_q and f_q in short, respectively.

As shown in the previous chapter, $\tilde{D}_q$ is monotonically decreasing and $\tilde{D}_q$ is bounded. In fact, we can get the extreme values of $\tilde{D}_q$. If we take the maximum number $n_{\max}$ and the minimum $n_{\min}$ of $\{n_j\}$ j=1, 2, ..., N_ε, as $q \to +\infty$, from Eq. (3.10) we get the minimum value of $\tilde{D}_q$

$$\tilde{D}_{\min} = \lim_{q \to +\infty} \frac{1}{q-1} \frac{\log \sum_{j=1}^{N_\varepsilon} n_j^{\,q} - q\log N}{\log\varepsilon}$$

$$= \lim_{q \to +\infty} \frac{q}{q-1} \frac{\log n_{\max} - \log N}{\log\varepsilon}$$

$$= \frac{\log n_{\max} - \log N}{\log\varepsilon} \tag{3.12}$$

and as $q \to -\infty$, we get the maximum value of $\tilde{D}_q$

$$\tilde{D}_{\max} = \lim_{q \to -\infty} \frac{1}{q-1} \frac{\log \sum_{j=1}^{N_\varepsilon} n_j^{\,q} - q\log N}{\log\varepsilon}$$

$$= \lim_{q \to -\infty} \frac{q}{q-1} \frac{\log n_{\min} - \log N}{\log\varepsilon}$$

$$= \frac{\log n_{\min} - \log N}{\log\varepsilon} \tag{3.13}$$

For a fixed ε and N, the following inequality holds

$$\tilde{D}_{\min} \le \tilde{D}_q \le \tilde{D}_{\max} \tag{3.14}$$

From Eq. (3.12) and Eq. (3.13), we know that $\tilde{D}_{\min}$ and $\tilde{D}_{\max}$ are independent of q. So the approximation of the Rényi dimension $\tilde{D}_q$ has upper bound $\tilde{D}_{\max}$ and lower bound $\tilde{D}_{\min}$. If $\tilde{D}_{\min}$ and $\tilde{D}_{\max}$ are equal, $\tilde{D}_q$ must be a constant.

When $N \to +\infty$ and $\varepsilon \to 0$ simultaneously, the above equations and inequality still hold. This means the bounds for $\tilde{D}_q$ are also the approximation of the bounds of the Rényi dimension D_q. If $\tilde{D}_q$ is constant, the fractal set is a uniform fractal or a nearly uniform fractal.

3.3.1.2 Approximations of the Singularity Spectrum

From Eq. (2.32) and Eq. (3.10), we can get the approximation of α_q as follows

$$\alpha_q = D_q + (q-1)\frac{dD_q}{dq} \tag{3.15}$$

$$\frac{dD_q}{dq} = \frac{(q-1)\sum_{j=1}^{N\varepsilon} n_j^q \log n_j + (\log N)\left(\sum_{j=1}^{N\varepsilon} n_j^q\right) - \sum_{j=1}^{N\varepsilon} n_j^q \log \sum_{j=1}^{N\varepsilon} n_j^q}{(\log\varepsilon)\ (q-1)^2 \sum_{j=1}^{N\varepsilon} n_j^q} \tag{3.16}$$

By substituting Eq. (3.16) into Eq. (3.15), and then simplifying Eq. (3.15), we obtain the approximation of the singularity dimension

$$\tilde{\alpha}_q = \frac{\sum_{j=1}^{N\varepsilon} n_j^q (\log n_j - \log N)}{(\log\varepsilon) \sum_{j=1}^{N\varepsilon} n_j^q} \tag{3.17}$$

Similarly to Eq. (3.11), we have

$$\lim_{\substack{N \to \infty \\ \varepsilon \to 0}} \tilde{\alpha}_q = \alpha_q \tag{3.18}$$

We calculate the first derivative of $\tilde{\alpha}_q$ as

$$\frac{d}{dq}\tilde{\alpha}_q = \frac{\left(\sum_{j=1}^{N\varepsilon} n_j^q\right)\sum_{j=1}^{N\varepsilon} n_j^q (\log n_j)^2 - \left(\sum_{j=1}^{N\varepsilon} n_j^q \log n_j\right)^2}{\left(\sum_{j=1}^{N\varepsilon} n_j^q\right)^2 \log\varepsilon} \tag{3.19}$$

Using Hölder inequality, we have

$$\left(\sum_{j=1}^{N\varepsilon} n_j^q\right)\sum_{j=1}^{N\varepsilon} n_j^q (\log n_j)^2 \leq \left(\sum_{j=1}^{N\varepsilon} n_j^q \log n_j\right)^2 \tag{3.20}$$

So the function $\tilde{\alpha}_q$ decreases monotonically, and from Eq. (3.19), α_q also decreases monotonically. When q goes to infinity, we can get the extreme value of $\tilde{\alpha}_q$ from Eq. (3.17). Let $\kappa_{n_{\max}}$ denote the number of vels which have $n_{\max}$ points in them, then from Eq. (3.17)

$$\lim_{q\to+\infty}\tilde{\alpha}_q = \lim_{q\to+\infty}\frac{\kappa_{n_{\max}} n_{\max}^q (\log n_{\max} - \log N)}{\kappa_{n_{\max}} n_{\max}^q \log\varepsilon}$$

$$= \frac{(\log n_{\max} - \log N)}{\log\varepsilon}$$

$$= \tilde{D}_{\min} \tag{3.21}$$

Similarly, if $\kappa_{n_{\min}}$ denotes the number of vels with $n_{\min}$ points in them, then we have

$$\lim_{q \to -\infty} \tilde{\alpha}_q = \lim_{q \to -\infty} \frac{\kappa_{n_{\min}} n_{\min}^q (\log n_{\min} - \log N)}{\kappa_{n_{\min}} n_{\min}^q \log \varepsilon}$$

$$= \frac{\log n_{\min} - \log N}{\log \varepsilon}$$

$$= \tilde{D}_{\max} \tag{3.22}$$

So for $\tilde{\alpha}_{\max} = \tilde{D}_{\max}$ and $\tilde{\alpha}_{\min} = \tilde{D}_{\min}$, we have $\tilde{\alpha}_{\min} \leq \tilde{\alpha}_q \leq \tilde{\alpha}_{\max}$. The difference between $\tilde{\alpha}_{\max}$ and $\tilde{\alpha}_{\min}$ is

$$\tilde{\alpha}_{\max} - \tilde{\alpha}_{\min} = \frac{\log n_{\min} - \log n_{\max}}{\log \varepsilon} \tag{3.23}$$

This difference roughly reflects the degree of nonuniformity (or inhomogeneity) of the fractal.

3.3.1.3 Approximations of the Mandelbrot Dimension

We shall now study the intensity of every spectral component, *i.e.* how often a spectral component will occur. From Eq. (2.33), the approximation of f_q can be evaluated by the approximate values of *D(q)* and *α(q)*

$$\tilde{f}_q = (q-1)\tilde{D}(q) - q\tilde{\alpha}(q) \tag{3.24}$$

or

$$\tilde{f}_q = \frac{\sum_{j=1}^{N\varepsilon} n_j^{\,q} \log n_j - \sum_{j=1}^{N\varepsilon} n_j^{\,q} \log \sum_{j=1}^{N\varepsilon} n_j^{\,q}}{\sum_{j=1}^{N\varepsilon} n_j^{\,q} \log\varepsilon} \tag{3.25}$$

Consequently, $\tilde{f}_q \geq 0$, and the following limits also hold

$$\lim_{q \to +\infty} \tilde{f}_q = \lim_{q \to +\infty} \frac{\sum_{j=1}^{N\varepsilon} n_j^{\,q} \left(\log n_j - \log \sum_{j=1}^{N\varepsilon} n_j^{\,q} \right)}{\sum_{j=1}^{N\varepsilon} n_j^{\,q} \log\varepsilon}$$

$$= \lim_{q \to +\infty} \frac{\kappa_{n_{\max}} n_{\max}^q \left(\log n_{\max}^q - \log \kappa_{n_{\max}} n_{\max}^q \right)}{\kappa_{n_{\max}} n_{\max}^q \log\varepsilon}$$

$$= -\frac{\log \kappa_{n_{\max}}}{\log\varepsilon} \tag{3.26}$$

Similarly

$$\lim_{q \to -\infty} \tilde{f}_q = \lim_{q \to -\infty} \frac{\sum_{j=1}^{N\varepsilon} n_j^{\,q} \left(\log n_j - \log \sum_{j=1}^{N\varepsilon} n_j^{\,q} \right)}{\sum_{j=1}^{N\varepsilon} n_j^{\,q} \log\varepsilon}$$

$$= \lim_{q \to +\infty} \frac{\kappa_{n_{\min}} n_{min}^q \left(\log n_{min}^q - \log \kappa_{n_{min}} n_{min}^q \right)}{\kappa_{n_{min}} n_{min}^q \log\varepsilon}$$

$$= -\frac{\log \kappa_{n_{\min}}}{\log\varepsilon} \tag{3.27}$$

Furthermore

$$\lim_{\substack{N \to \infty \\ \varepsilon \to 0}} \tilde{f}_q = f_q \tag{3.28}$$

From Eq. (3.26) and Eq. (3.27), we can also see that if we use the same number of points in the attractor, while making ε smaller and smaller, both $\kappa_{n_{\max}}$ and $\kappa_{n_{\min}}$ reach the value of 1, thus making $\tilde{f}_q$ constant. In this case, from Eq. (3.23), the range of $\tilde{\alpha}_q$ will be zero, so $\tilde{\alpha}_q$ is also constant. Obviously, this result is incorrect. Thus we conclude that a compromise between N and ε must be used. This will lead to an optimal approximation of the theoretical value of D_q.

3.3.2 Compromise Between the Vel Size and the Number of the Points

Let us study Eq. (3.11) again. If $a = 1/N$, then it becomes

$$\lim_{\substack{a \to 0 \\ \varepsilon \to 0}} \tilde{D}_q = D_q \tag{3.29}$$

Theoretically, for any $\tilde{\varepsilon}$, there exists δ, such that

$$\| (a, \varepsilon) - (0, 0) \| \leq \delta \tag{3.30}$$

Thus, the following inequality holds

$$\| \tilde{D}_q - D_q \| \leq \tilde{\varepsilon} \tag{3.31}$$

Although we would like to make $\| (a, \varepsilon) - (0, 0) \|$ very small in order to get a better approximation of D_q, in practice, δ cannot become arbitrarily small, because ε is nonzero. Consequently, from

$$\varepsilon^2 + \left(\frac{1}{N}\right)^2 < \delta^2 \tag{3.32}$$

we get the following relationship between N and ε

$$N > \frac{1}{\sqrt{(\delta+\varepsilon)\ (\delta-\varepsilon)}} \approx \frac{1}{\sqrt{(2\varepsilon)\ (\delta-\varepsilon)}} \tag{3.33}$$

For example, suppose the smallest usable ε is 0.001, and we want δ as small as possible to get a good approximation. If we take $\delta = 0.0011$, then N should be larger than 224. This will not give a good approximation. If we take $\delta = 0.001001$, then N should be larger than 7071 points, so the smaller the δ, the larger the N, leading to a better approximation of D_q.

3.4 Overview of Chapter 3

A brief introduction to chaotic dynamical systems and strange attractors is presented in this chapter. Because of the complexity of the strange attractors, multifractal measures are used to characterize them. The study of the approximations of the multifractal measures reduces to a finite number of points N from the strange attractors and a finite size of vels ε used in their coverings. In this study, some useful results on the extreme values of the multifractal measures are obtained, they will be used in Chapter 4 for spatial signal analysis. In this chapter we also find that a compromise between N and ε must be used in the multifractal characterization of the strange attractors.

CHAPTER 4

MULTIFRACTAL MEASURES IN SPATIAL SIGNAL ANALYSIS

4.1 Texture Analysis of Images

Although texture is one of the key parameters in image processing, it is very difficult to estimate. The determination of visual texture has long been an area of active computer vision research. While the ability of humans to distinguish different textures is apparent, the automated description and recognition of these same patterns has proven to be very complicated. The classical approaches to texture description are based on statistical or syntactical analysis. The use of the statistical features is motivated by the conjecture of Julez [JuBa75] that second order probability distributions suffice for human discrimination of two texture patterns. This conjecture has been shown not to hold strictly, particularly when the textures have some structure [CaJu78] [PaRi77].

A new branch of mathematics, fractal geometry, as developed and explored by Mandelbrot and others, has a major impact on modelling and analysis in the natural and physical sciences. They have demonstrated that fractal geometry is able to describe shapes and phenomena similar to natural ones. Pentland [Pent93] has presented evidence that most natural surfaces are spatially isotropic fractals and the intensity images of these surfaces are also fractals. He obtained interesting results by using a tool in the characterization of natural surfaces. In particular, he found a link between the fractal dimensions of surfaces and their roughness.

The roughness, however, is not sufficient to describe a textured surface, because other characteristics have to be considered, such as arrangements, spatial distribution of grey levels. Furthermore, this approach assumes that the 2D grey level image can be seen as a 3D surface, or, equivalently, that the grey levels can be assimilated in a spatial coordinate on the z-axis. This assumption has no theoretical basis, because the scaling property of the of the grey levels are generally different from those of the space coordinates. Instead we should look at the grey levels as a measure, laid upon a generally compact set, totally inhomogeneous to space coordinates. This leads to the multifractal analysis.

In this thesis, an image will be considered as a strange attractor, a natural question is how to define the measure for the basic unit of an image (a pixel) in terms of points which are used in the multifractal analysis for strange attractors.

4.2 Measuring a Pixel Through a Random Binary Dithering Process

In order to consider an image as a strange attractor and apply the multifractal measures to it, a measure of a pixel should be defined. The assignment of a measure for a pixel in this section is motivated by the electronic techniques for pictorial image reproduction. So at the beginning of this section, we will give some background on this.

4.2.1 Electronic Techniques for Pictorial Image Reproduction

In this section, a brief review of the printing technologies and the display of continuous tone image are presented.

4.2.1.1 Binary Marking/Displaying Technologies

The mass reproduction technologies for pictorial imagery would include relief printing (letter press), intaglio (gravure), and lithography. Although the history of these technologies dates back to the 8th century, one will find that the most efficient mode for their operation was binary, *i.e*, they generate two-tone microstructure composed of regions with or without ink. Today, nearly all printed pictorials are composed of such a microstructure which is "binary"; *i.e.*, either ink or paper is located at a given spot. Even full colour pictorials in magazines, etc., are essentially composed of four inks operating in this binary mode.

A large number of printing and display techniques have been developed in the 20th century, exploiting the advances being made in photoconductors and electronics. Included among these techniques are cathode ray tube displays, xerography, stylus printing, plasma displays, panels, and ink-jet printing. These techniques do not conveniently fall into the three classical printing categories developed above, but they do represent commercially viable methods of printing or displaying imagery. It is sufficient to examine the dominant modes of their operation and note that they are frequently driven as binary output processes.

4.2.1.2 Continuous Tone Imagery and Halftone Imagery

A continuous tone image is one which contains an apparent continuum of gray levels. Some scenes, when viewed by humans, may require more than 256 discrete gray levels to give the appearance of a continuum of gray levels from one shade to another. Continuous tone images are exemplified by television images, photographic images, and real world scenes viewed by a Vidicon or CCD camera. Continuous tone imagery, therefore, is composed of "natural" images and the approximations to them.

As an approximation to continuous tone images, the printing industry developed, in the mid-19th century, a technique for approximating the continuum of the gray scales available in "natural imagery". There were, as there are now, three dominant printing technologies: lithography, letterpress, and gravure; and all of these produced a page via the presence or absence of opaque ink on a page. In order to represent "natural" scenes to yield a varying percent reflectance across the page. The end result is that when such images are viewed at normal viewing distances the dot and line structure is not noticeable, but the varying average gray level produces an approximation to a natural scene.

Since the relationship between grey level and its display in the printing techniques, we would obtain a method to measure the grey levels through their binary display in printing technology.

4.2.2 Measure a Pixel of an Image

In halftoning, a halftone pattern is very important. The halftone pattern may be approximated on the bilevel device in various ways. One of the simplest techniques is to use character overstrikes. Generally, overstriking is the preferred technique when the output device is a line printer. Another approach for halftoning on a bilevel device consists of forming various patterns of block dots within a white square area. For example, a two-by-two pixel region of a bilevel device may be used to produce five gray levels. A three-by-three area can be used to produce ten grey level [GoWo92]. In these techniques, the halftone patterns are deterministic. Unlike these techniques with fixed patterns, a random pattern halftoning will be discussed as follows.

Consider transforming a grey level image into a random dithering image. The black pixels in the dithering image are also called *points* in this thesis. Each grey level pixel is translated into a dithering block with either a uniform or nonuniform distribution of points. Such a block is a $T \times T$ pixel binary image. Different pixels correspond to different dithering blocks. The generation of a dithering block utilizes a random approach. In this study, for an image with 256 different grey levels, the dithering block size is T=16. With any value a of the grey level between 0 and 255, a dithering block is obtained with $(255 - a)$ black pixels or points randomly produced.

As an example, an image of Lena (128x128 pixels with 8 bpp) is shown in Fig. 4.1a. For convenience, we work on a cutout image (34x34 pixels with 8 bpp) which is shown in Fig. 4.1b. Applying the random dithering process to each pixel of the cutout image as described above, the resulted binary image is shown in Fig. 4.2

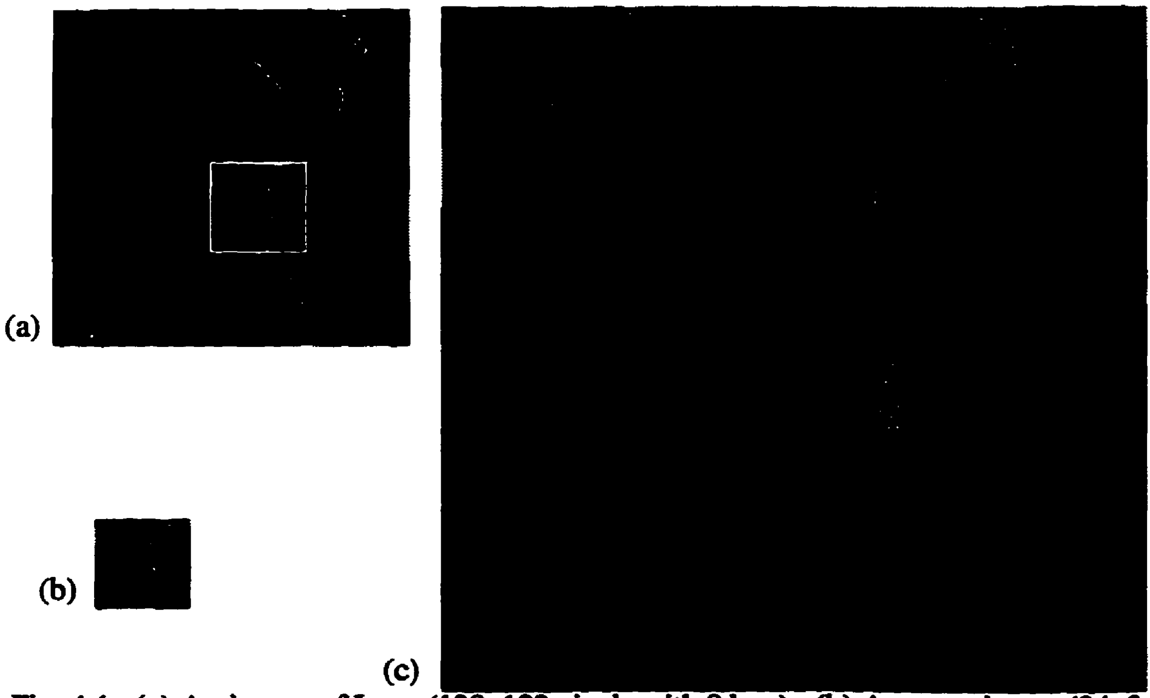


Fig. 4.1 (a) An image of Lena (128x128 pixels with 8 bpp). (b) A cutout image (34x34 pixels) from (a). (c) A zoomed image (544x544 pixels with 8 bpp) from the cutout image.

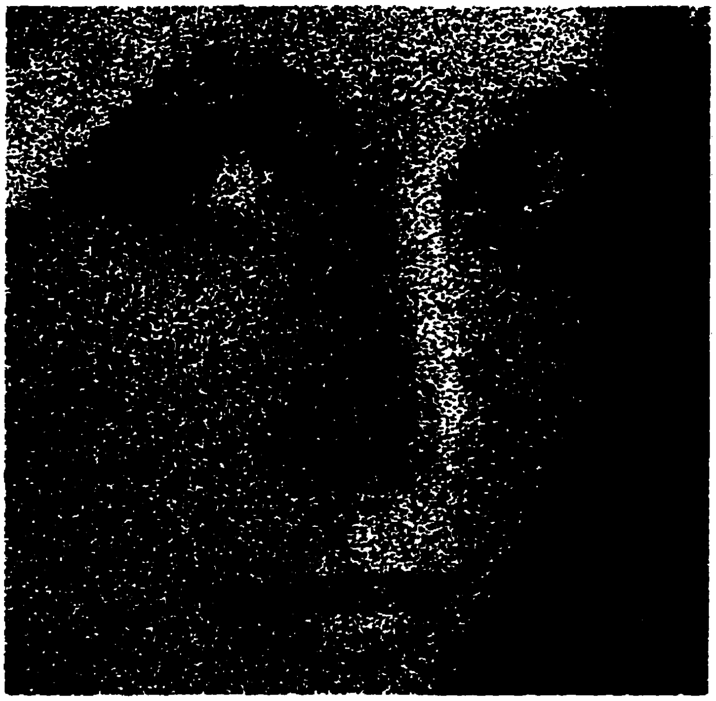


Fig. 4.2 The dithering image (544x544 pixels with 1 bpp) from the cutout image in Fig. 4.1b.

From the binary image shown in Fig. 4.2, it can be seen that the distribution of the binary pixels in different blocks is nonuniform. We notice that some areas may have rarefied subsets of points while some other areas may have dense subsets of points. This binary image is just like the strange attractors shown in Figs. 3.5, 3.6, and 3.7, and will be treated as a strange attractor!

A pixel is the smallest unit in an image, in order for us to use multifractal measures to measure the grey level image, a measure is assigned to each pixel: *the measure of the pixel is the number of the points in the dithering block described above*. So, a bridge between the grey level and number of the dithering points is established. With the measure assignment to the basic unit, any portion of the image can be measured.

4.3 Determining the Size of a Pixel in Fractal Analysis

Chapter 2 summarized several fractal dimensions, including the self-similarity dimension, the Hausdorff dimension, information dimension and the Rényi dimension.

For each definition of fractal dimension, some kind of measure is used. For the self-similarity dimension, the measure is *N(r)* distinct (non-overlapping) copies of itself (see 2.2.1). For the Hausdorff dimension, the measure is the number of covering vels, *N(r)*, required to cover the fractal object completely (see 2.2.2). For the information dimension, the measure is the Shannon entropy H_r (see 2.2.3). The Rényi entropy H_q is the measure in the Rényi dimension. All the values of the measures change as the vel size changes. One can see that the fractal dimensions are actually reflecting the rate of change of the measures as the vel sizes change. We have already discussed the measure used for each pixel of an image.

The next step is how we can determine the size of a pixel when images are analyzed as single fractals or multifractals. All the dithering blocks can be considered as vels which are used to obtain a nonintersecting cover of the dithering image. Since each pixel is translated into a unique dithering block, we will not distinguish between the size of the dithering block (called a vel later) and the size of a grey level pixel in fractal context.

We know that the dimension of a plane is 2, and the dimension of a line is 1. If we apply the definition of the fractal dimension to these regular geometric objects (Fig. 4.3), the values of the dimension should also be 2 and 1, respectively.

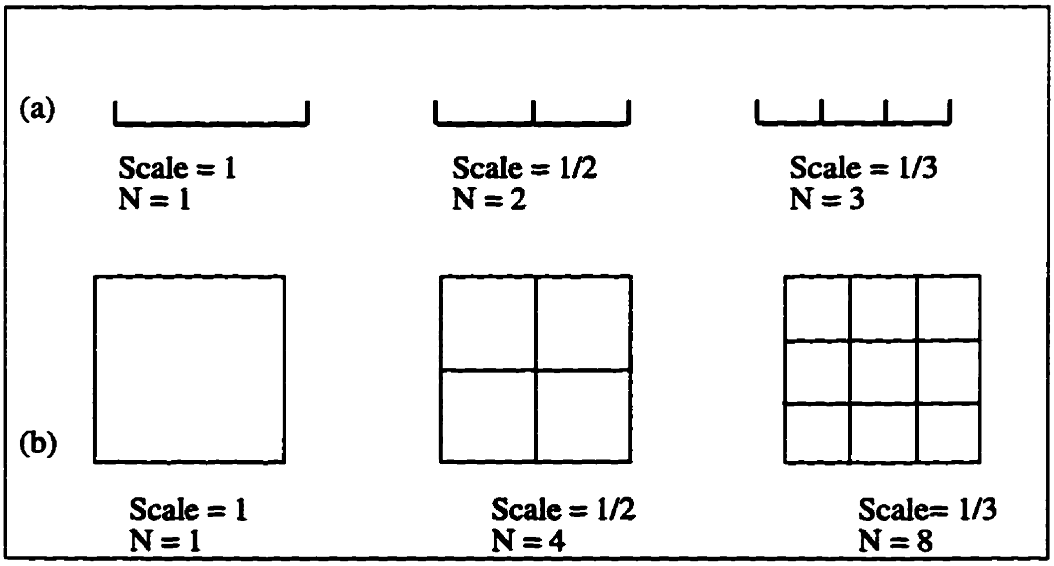


Fig. 4.3 (a) The dimension of a line is 1. (b) The dimension of a plane is 2.

We apply the multifractal formalism to the plane. Since the plane is a uniform object, the property in any location of the plane is the same. So the Rényi dimension should always be 2. It will not change no matter what value of q is.

If one displays a plane using an image, when we apply the multifractal measure to it, the values of the Rényi dimension are supposed to be 2. Fig. 4.4a is an image of a plane. Each pixel in the image has the same grey level and is translated to its corresponding dithering block, so the measure of each pixel is known. Now the natural question is what the size of one pixel is when we try to calculate the fractal dimension.

Suppose the grey level of a uniform $s \times s$-pixel image is G. Since each pixel is represented by 8 bits. The measure of each pixel is denoted by n_j, using the number of points in the dithering block, we have $n_j = 255 - G(j=1, 2, \ldots s \times s)$. The total number of points N in the dithering image is $s \times s \times (255 - G)$. Suppose the size of each pixel or dithering block is r, then the minimum number of the nonoverlapping vels $N(r)$ used to cover the image is $s \times s$, then we are ready to apply the multifractal model which has been discussed in Chapter 3.

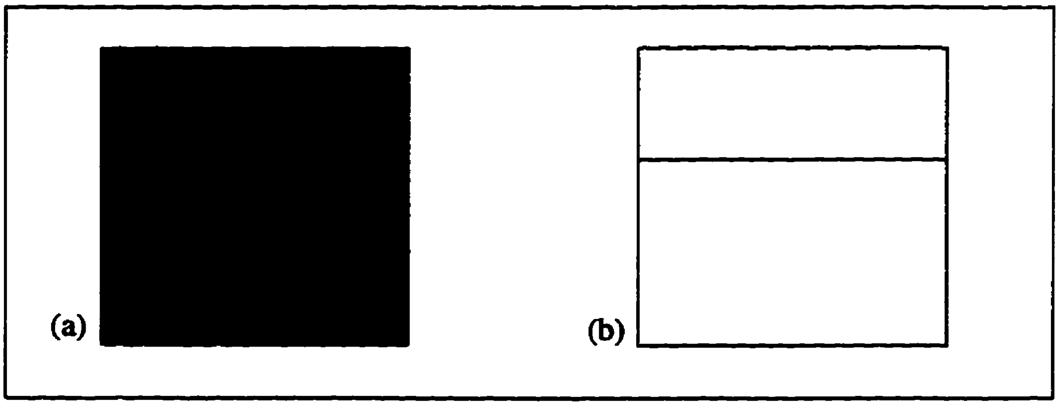


Fig. 4.4 An image of a plane (a) and an image of a line (b).

According to the definition of the Rényi dimension

$$D_q = \lim_{r \to 0} \frac{H_q}{\log 1/r} = \lim_{r \to 0} \frac{1}{q-1} \frac{\log\left(\sum_{j=1}^{N(r)} p_j^2\right)}{\log 1/r} \qquad (4.1)$$

The estimation of D_q for the image can be easily obtained from

$$\tilde{D}_q = \frac{1}{q-1}\frac{\log\sum_{j=1}^{s^2}\left(\frac{n_j}{N}\right)^q}{\log r} \tag{4.2}$$

From Eq. (4.2), we can represent r with the dimension as follows

$$r = \exp\left(\frac{\log\sum_{j=1}^{s^2}\left(\frac{n_j}{N}\right)^q}{\tilde{D}_q(q-1)}\right) \tag{4.3}$$

In the above equation, we fit $\tilde{D}_q$=2, $N = s \times s \times (255-G)$, $n_j = 255-G$, then we obtain the equation as follows

$$r = \exp\left(\frac{\log\sum_{j=1}^{s^2}\left(\frac{255-G}{s^2(255-G)}\right)^q}{2(q-1)}\right) = \frac{1}{s} \tag{4.4}$$

Let us consider a $s \times s$-pixel image of a line, the line in the image is s-pixel long and each grey level on the line is G. So the measure of each pixel is $n_j = 255-G$. Similar to the discussion for the plane, the size of each pixel is

$$r = \exp\left(\frac{\log\sum_{j=1}^{s}\left(\frac{n_j}{N}\right)^q}{\tilde{D}_q(q-1)}\right) \tag{4.5}$$

note that the number of the vels $N(r)$ used to cover the line is s, and the total number of the points N in the dithering image is

$$N = s(255-G) \tag{4.6}$$

In Eq. (4.5), we fit $\tilde{D}_q$=1, $n_j = 255 - G$, the size of each pixel is

$$r = \exp\left(\frac{\log \sum_{j=1}^{s} \left(\frac{255-G}{s(255-G)}\right)^q}{(q-1)}\right) = \frac{1}{s} \tag{4.7}$$

Both Eq. (4.2) and Eq. (4.7) mean that if we analyze an image in fractal model, the size of a pixel in the fractal analysis is the inverse of the image size.

4.4 Multifractal Model for Images

4.4.1 Multifractal measure for Images

We know from Chapter 3 that in the estimation of multifractal measures of strange attractors, the points from the strange attractors and size of the vels used to cover the strange attractors play a central role. With the discussion in Section 4.2 and 4.3, we apply the approximations of multifractal measures of the strange attractors to image analysis.

Suppose we deal with an image which is $s \times s$ pixels, with each pixel represented by 8 bits. There are s^2 pixels in the image with grey level value of G_j (j = 1, 2, 3, ... s^2). As we discussed in Section 4.2, when an image is considered as a multifractal or a strange attractor, a pixel of the image is considered as the smallest vel that can be used, so there will be s^2 nonintersecting vels covering the image. The number of the points n_j in the dithering block is used as a measure of the pixel, and it relates to the grey level by

$$n_j = 255 - G_j \qquad\qquad j = 1, 2, 3, \dots s^2 \tag{4.8}$$

Suppose the total number of the dithering points in the image is N, and N should satisfy the equation as follows

$$N = \sum_{j=1}^{s^2} (255 - G_j) = 255s^2 - \sum_{j=1}^{s^2} G_j \tag{4.9}$$

On the other hand, suppose the size of each pixel is r, since the size of the image is $s \times s$, according to the discussion in Section 4.2, the size of each pixel r follows the relationship

$$r = \frac{1}{s} \tag{4.10}$$

With the above, we apply the equations in Chapter 3.3, the multifractal measures can be estimated as follows.

4.4.1.1 Approximations of the Rényi Dimension for Images

Based on Eq. (3.10), we obtain the estimation of the Rényi dimension for images

$$\tilde{D}_q = \frac{1}{q-1} \frac{\log \sum_{j=1}^{s^2} (255 - G_j)^q - q\log\left(255s^2 - \sum_{j=1}^{s^2} G_j\right)}{\log\frac{1}{s}} \tag{4.11}$$

The extreme values can also be easily obtained similarly to Eq. (3.12) and Eq. (3.13). If we take the maximum number $G_{\max}$ and the minimum $G_{\min}$ of $\{G_j\}$ $j=1, 2, ..., s^2$, as $q \rightarrow +\infty$, we get the minimum value of $\tilde{D}_q$ for images as follows:

$$\tilde{D}_{\min} = \lim_{q \to +\infty} \frac{1}{q-1} \frac{\log \sum_{j=1}^{s^2} (255 - G_j)^q - q\log\left(255s^2 - \sum_{j=1}^{s^2} G_j\right)}{\log\frac{1}{s}}$$

$$= \lim_{q \to +\infty} \frac{q}{q-1} \frac{\log(255 - G_{min}) - \log\left(255s^2 - \sum_{j=1}^{s^2} G_j\right)}{\log\frac{1}{s}} \tag{4.12}$$

$$= \frac{\log(255 - G_{min}) - \log\left(255s^2 - \sum_{j=1}^{s^2} G_j\right)}{\log\frac{1}{s}}$$

As $q \to -\infty$, we get the maximum value of $\tilde{D}_q$ for the image as follows

$$\tilde{D}_{\max} = \lim_{q \to -\infty} \frac{1}{q-1} \frac{\log \sum_{j=1}^{s^2} (255 - G_j)^q - q\log\left(255s^2 - \sum_{j=1}^{s^2} G_j\right)}{\log\frac{1}{s}}$$

$$= \lim_{q \to -\infty} \frac{q}{q-1} \frac{\log(255 - G_{max}) - \log\left(255s^2 - \sum_{j=1}^{s^2} G_j\right)}{\log\frac{1}{s}}$$

$$= \frac{\log(255 - G_{max}) - \log\left(255s^2 - \sum_{j=1}^{s^2} G_j\right)}{\log\frac{1}{s}} \tag{4.13}$$

4.4.1.2 Approximations of Singularity Spectrum for images

According to Eq. (3.17), we have the estimation of the singularity spectrum for images

$$\tilde{\alpha}_q = \frac{\sum_{j=1}^{s^2} (255-G_j)^q \left(\log (255-G_j) - \log\left(255s^2 - \sum_{j=1}^{s^2} G_j \right) \right)}{\left(\log\frac{1}{s} \right) \sum_{j=1}^{s^2} (255-G_j)^q} \qquad (4.14)$$

Let $\kappa_{G_{\min}}$ denote the number of pixels which has grey level value $G_{\min}$ in the images, then from Eq. (4.14), we have the following equation

$$\lim_{q \to +\infty} \tilde{\alpha}_q = \lim_{q \to +\infty} \frac{\kappa_{G_{\min}} (255-G_{min})^q \left(\log (255-G_{min}) - \log\left(255s^2 - \sum_{j=1}^{s^2} G_j \right) \right)}{\kappa_{G_{min}} (255-G_{min})^q \log\frac{1}{s}}$$

$$= \frac{\log (255-G_{min}) - \log\left(255s^2 - \sum_{j=1}^{s^2} G_j \right)}{\log\frac{1}{s}} \qquad (4.15)$$

$$= \tilde{D}_{\min}$$

Similarly, if $\kappa_{G_{max}}$ denotes the number of vels with G_{max} points in them, then we have

$$\lim_{q \to -\infty} \tilde{\alpha}_q = \lim_{q \to -\infty} \frac{\kappa_{G_{max}} (255 - G_{max})^q \left(\log (255 - G_{max}) - \log \left(255 s^2 - \sum_{j=1}^{s^2} G_j \right) \right)}{\kappa_{\kappa_{G_{max}}} (255 - G_{min})^q \log \frac{1}{s}}$$

$$= \frac{\log (255 - G_{max}) - \log \left(255 s^2 - \sum_{j=1}^{s^2} G_j \right)}{\log \frac{1}{s}}$$

$$= \tilde{D}_{\max} \qquad (4.16)$$

So we also get $\tilde{\alpha}_{\max} = \tilde{D}_{\max}$ and $\tilde{\alpha}_{\min} = \tilde{D}_{\min}$. We can get the difference between $\tilde{\alpha}_{\max}$ and $\tilde{\alpha}_{\min}$

$$\tilde{\alpha}_{\max} - \tilde{\alpha}_{\min} = \frac{\log (255 - G_{max}) - \log (255 - G_{min})}{\log \frac{1}{s}} \qquad (4.17)$$

This difference reflects the degree of nonuniformity (or inhomogeneity) of the image. The greater the difference of the gray levels, the larger the range of the singularity spectrum.

4.4.1.3 Approximations of Mandelbrot Dimension for Images

From Eq. (3.25), the approximation of f_q can be easily evaluated as follows

$$\tilde{f}_q = \frac{\sum_{j=1}^{s^2} (255 - G_j)^q \log (255 - G_j) - \sum_{j=1}^{s^2} (255 - G_j)^q \log \sum_{j=1}^{s^2} (255 - G_j)^q}{\sum_{j=1}^{N_t} (255 - G_j)^q \log \frac{1}{s}} \qquad (4.18)$$

Similar to Eq. (3.26) and Eq. (3.27), from the above equation, we can get the following equations

$$\lim_{q \to +\infty} \tilde{f}_q = \lim_{q \to +\infty} \frac{\sum_{j=1}^{s^2} (255 - G_j)^q \left(\log (255 - G_j) - \log \sum_{j=1}^{s^2} (255 - G_j)^q \right)}{\sum_{j=1}^{s^2} (255 - G_j)^q \log \frac{1}{s}}$$

$$= \lim_{q \to +\infty} \frac{\kappa_{G_{min}} (255 - G_{min})^q \left(\log (255 - G_{min})^q - \log \kappa_{G_{min}} (255 - G_{min})^q \right)}{\kappa_{G_{min}} (255 - G_{min})^q \log \frac{1}{s}}$$

$$= \frac{\log \kappa_{G_{min}}}{\log \frac{1}{s}} \tag{4.19}$$

$$\lim_{q \to -\infty} \tilde{f}_q = \lim_{q \to -\infty} \frac{\sum_{j=1}^{s^2} (255 - G_j)^q \left(\log (255 - G_j) - \log \sum_{j=1}^{s^2} (255 - G_j)^q \right)}{\sum_{j=1}^{s^2} (255 - G_j)^q \log \frac{1}{s}}$$

$$= \lim_{q \to -\infty} \frac{\kappa_{G_{max}} (255 - G_{max})^q \left(\log (255 - G_{max})^q - \log \kappa_{G_{max}} (255 - G_{max})^q \right)}{\kappa_{G_{max}} (255 - G_{max})^q \log \frac{1}{s}}$$

$$= \frac{\log \kappa_{G_{max}}}{\log \frac{1}{s}} \tag{4.20}$$

4.4.2 Multifractal Measures on Single Fractal Images

In order to test our model, we apply the model to single fractal objects whose theoretical values are known. The results under this model should equal or be close to the theoretical results. From experiments, we find that using this model, the estimated fractal dimensions of some of the fractals are very accurate, for example, the estimated dimension of the Minkowski curve is always 1.50000 no matter which q (between -30 and 30) we use. But for some of the other fractals, we have the estimation dimension slightly different from theoretical results, for example, the estimation dimension D_q for Koch's curve is 1.244456 for all the q between -30 and 30. The theoretical result should be 1.2618595.

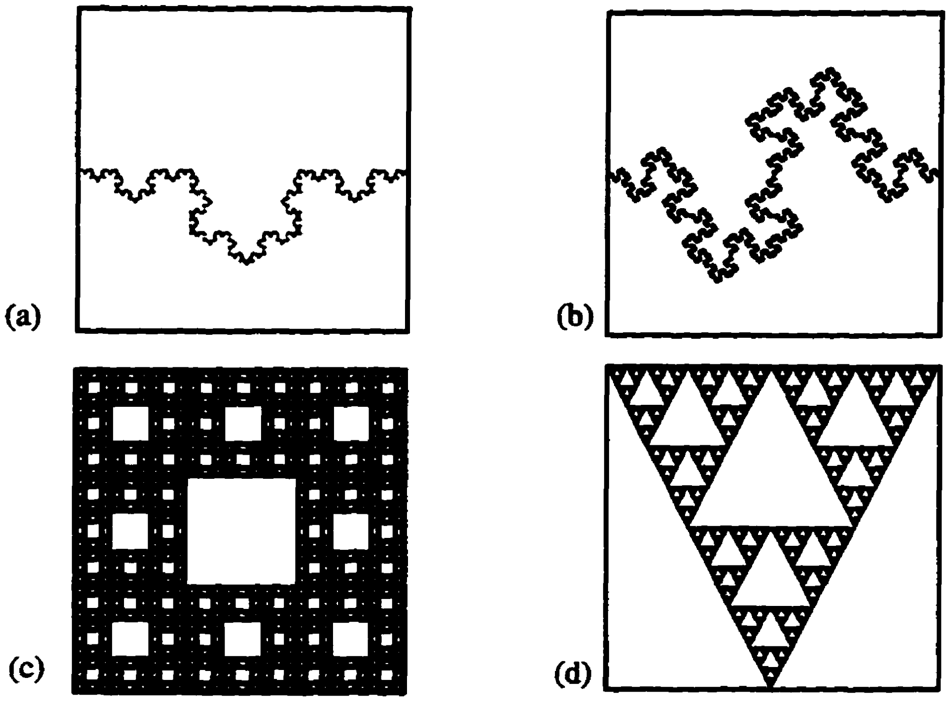


Fig. 4.5 Grey level images. (a) The images of the Koch curve. (b) The image of the Minkowski curve. (c) The image of the Sierpinski carpet. (d) The image of the Sierpinski gasket.

The theoretical results and the experimental results using our model are listed in Table. 4.1. We can see the error is very small, this also indicates that the multifractal models for images can hold the single fractals fairly well. From Section 4.4.1, we should have $D_q = \alpha_q = f_q$. Our experimental results are also plotted in Fig. 4.6. Since both the singularity spectrum and the Mandelbrot dimension are constant, the Mandelbrot spectrum will be degraded to single points for the different objects, as shown in Fig. 4.7.

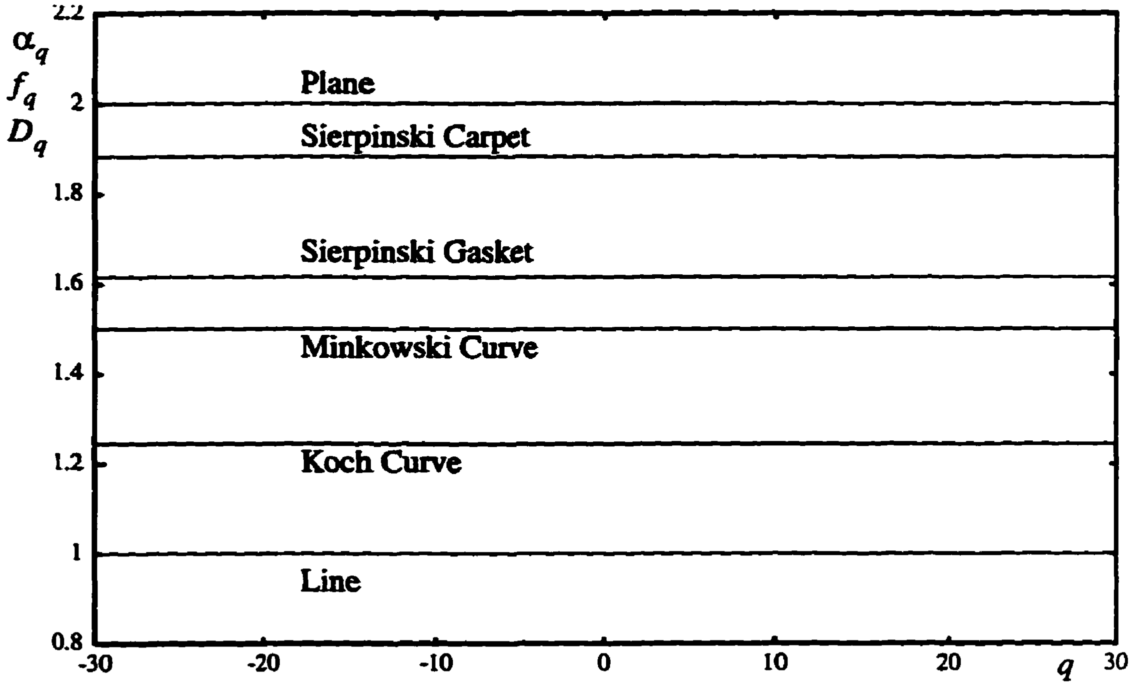


Fig. 4.6 The Rényi dimension, singularity spectrum, and the Mandelbrot dimension of a line, the Koch curve, the Minkowski curve, the Sierpinski gasket, the Sierpinski carpet, and a plane.

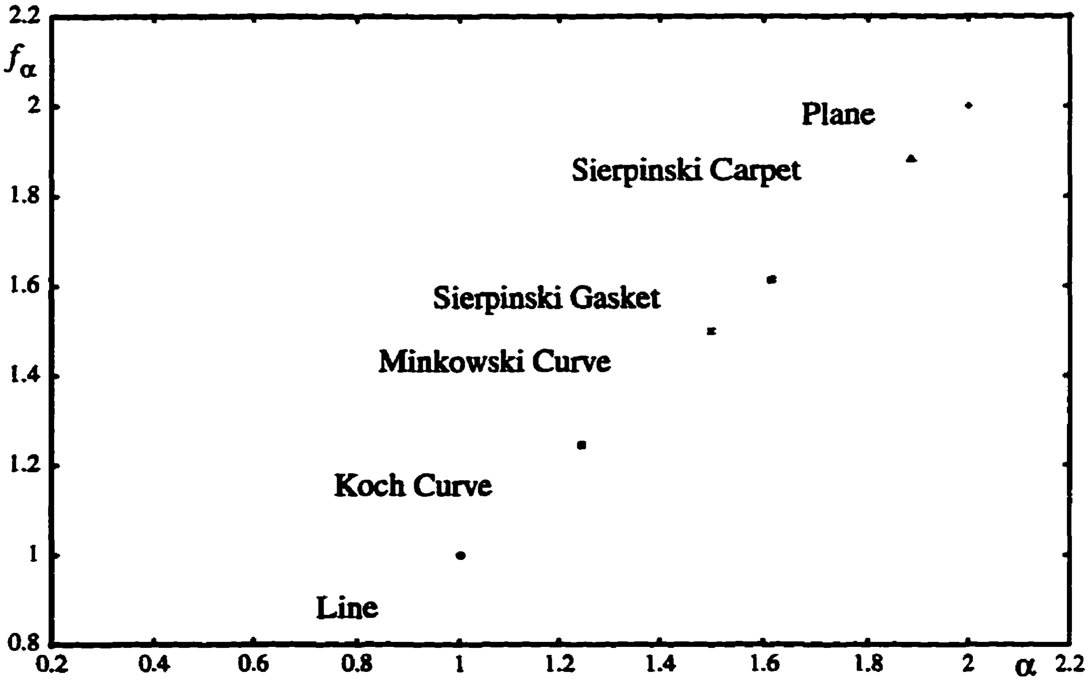


Fig. 4.7 The degraded Mandelbrot spectra for a line, the Koch curve, the Minkowski curve, the Sierpinski gasket, the Sierpinski carpet, and a plane.

Table 4.1: The Theoretical and Experimental Dimensions

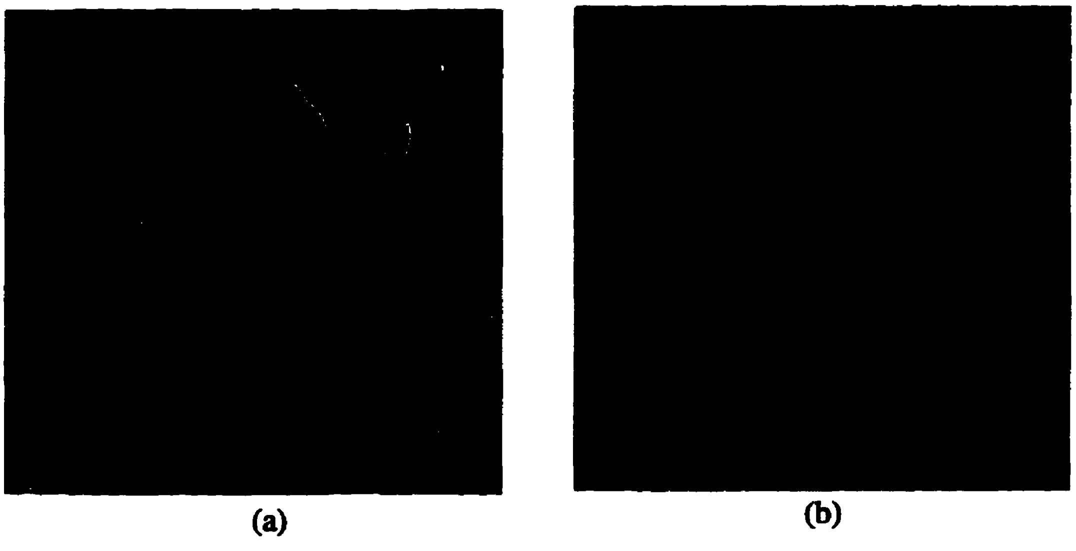
Type of Regular Geometric Objects and Single Fractals	Theoretical Dimension (D_0)	Multifractal Measure $D_q = \alpha_q = f_q$ $-30 \le q \le 30$	Error
Square	$\frac{\log n^2}{\log n} \approx 2.0$	$D_q = 2.000000$	+0.0000000
Line	$\frac{\log n}{\log n} \approx 1.0$	$D_q = 1.000000$	+0.0000000
Koch Curve	$\frac{\log 4}{\log 3} \approx 1.2618595$	$D_q = 1.245181$	−0.0166785
Minkowski Curve	$\frac{\log 8}{\log 4} \approx 1.5000000$	$D_q = 1.500000$	+0.0000000
Sierpinski Gasket	$\frac{\log 3}{\log 2} \approx 1.5849625$	$D_q = 1.614881$	+0.0299185
Sierpinski Carpet	$\frac{\log 8}{\log 3} \approx 1.8927893$	$D_q = 1.882249$	−0.0105403

From Table 4.1, it can be seen that although the experimental results are very close to the theoretical results, minor errors still exist. We notice that for a square, a line and the Minkowski Curve, the results are almost perfect. But for the Koch curve, the Sierpinski carpet, and the Sierpinski gasket, larger errors are produced. The errors are mainly caused by the image representation of the fractal objects. Since the resolution of an image is limited, if only horizontal or/and vertical lines are used in generating a fractal, the experimental result is more accurate. However, if slant lines are used in the generation of the fractal, more errors will be introduced because the slant lines look zig-zag when higher resolution is used. The numerical rounding error in producing the fractal images is another reason. For example, the generation of the Sierpinski carpet is only related to the horizontal and vertical lines, but the experimental result is not as good as either a plane, a line, or the Minkoski curve. This rounding error is reflected in the image which is shown in Fig.

4.5(c). As the number of iteration increases, some of the smaller white boxes are not really square in many locations.

4.4.3 Multifractal Measures on Multifractal Images

Multifractal images are images whose grey levels are distributed nonuniformly. Two multifractal images are shown in Fig. 4.8, one is the image of Lena which is 256 by 256 pixels, with 8 bpp, the other image is an image of a Baboon which is also 256 by 256 pixels with 8 bpp. Each grey level is measured by the number of points in the corresponding dithering block, the Rényi dimension D_q, the singularity spectrum $\alpha(q)$, and the Mandelbrot dimension $f(\alpha(q))$ are shown in Figs. 4.9, 4.10, and 4.11, respectively. From these figures, we find the Rényi dimension, the singularity spectrum, and the Mandelbrot spectrum are no longer a straight line. It makes sense that the singularities of the image of Lena and the image of Baboon have some differences, respectively.



(a)

(b)

Fig. 4.8 Multifractal images. (a) An image of Lena which is 256 by 256 pixels with 8 bpp. (b) An image of Baboon which is 256 by 256 pixels with 8 bpp.

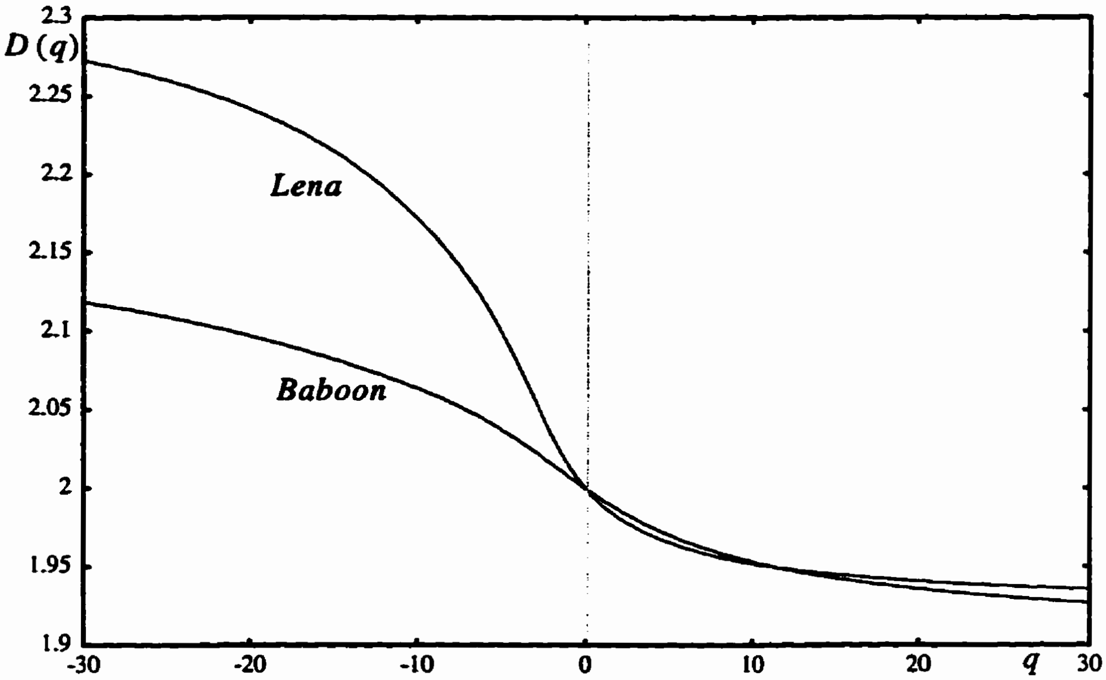


Fig. 4.9 The Rényi dimension D_q of the image of Lena and Baboon.

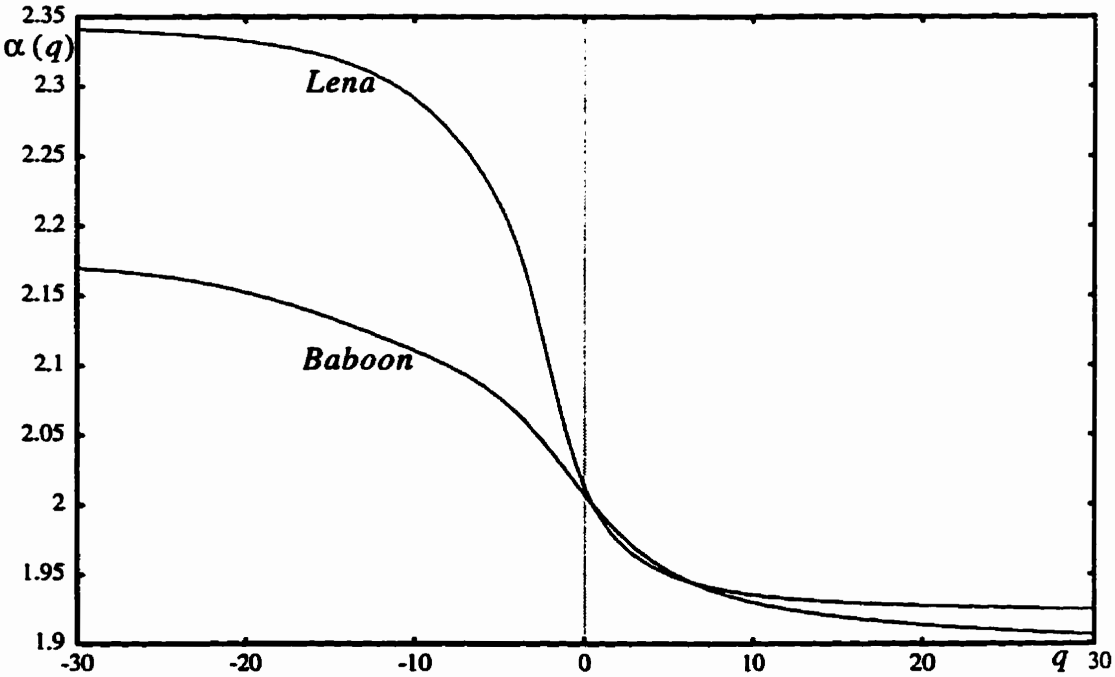


Fig. 4.10 The sigularity spectrum $\alpha(q)$ of the image of Lena and Baboon.

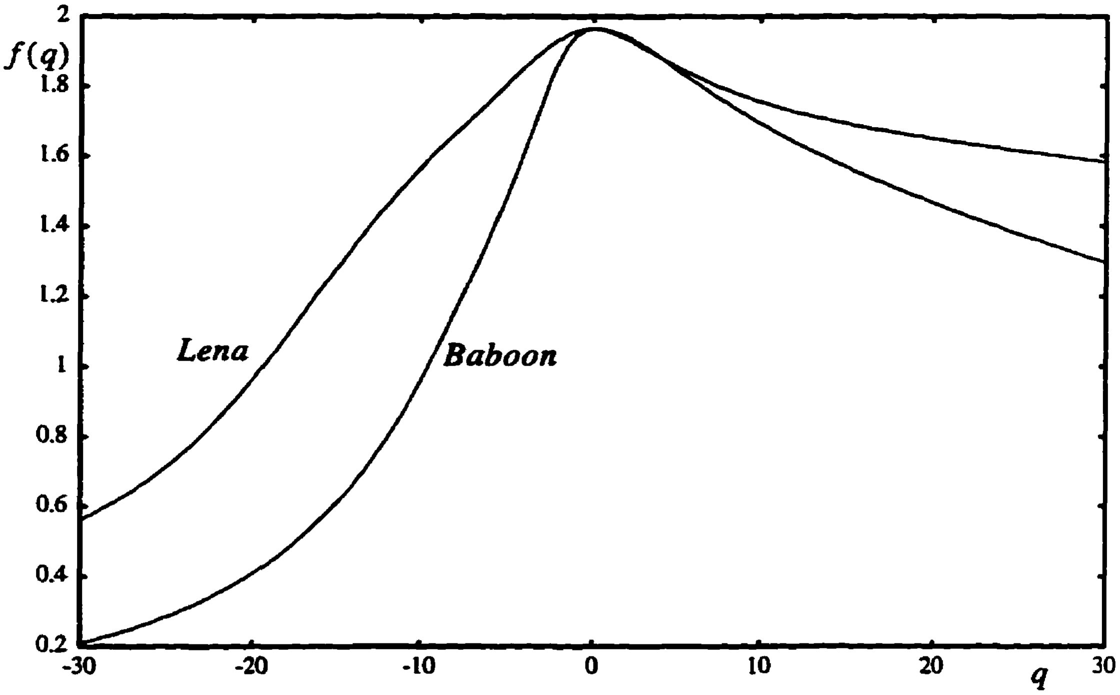


Fig. 4.11 The Mandelbrot dimension $f(q)$ of the image of Lena and Baboon.

We know that in multifractal measures, the most important one is the Mandelbrot spectrum. We notice that the range of the singularity spectrum for the image of Lena is larger than that for the image Baboon (shown in Fig. 4.12). This means the degree of the nonuniformity of Lena is larger than that of Baboon. From Eq. (4.17), we can see that this range is related to the difference between the maximum grey level and minimum grey level. It reflects the range of the grey levels in the image of Lena is large than that of Baboon. The histograms of Lena and Baboon which are shown in Fig. 4.13 exhibit this property.

From the multifractal formalism discussed in Chapter 2, Eq. (4.19) and Eq. (4.20), it can be seen that the value of the Mandelbrot dimension $f(\alpha(q))$ at right end of the singularity spectrum $\alpha(q)$ indicates how dense the population of the higher grey level pixels is located there is. The value of $f(\alpha(q))$ at left end of the singularity spectrum $\alpha(q)$ indicates how dense the population of the lower grey level pixels located there is. At the right end of $\alpha(q)$ for Baboon, we can see the value of the $f(\alpha(q))$ is much smaller than that for Lena. So, the population of the higher grey level pixels is less than that for Lena. This result coincides with the histogram results shown in Fig. 4.13.

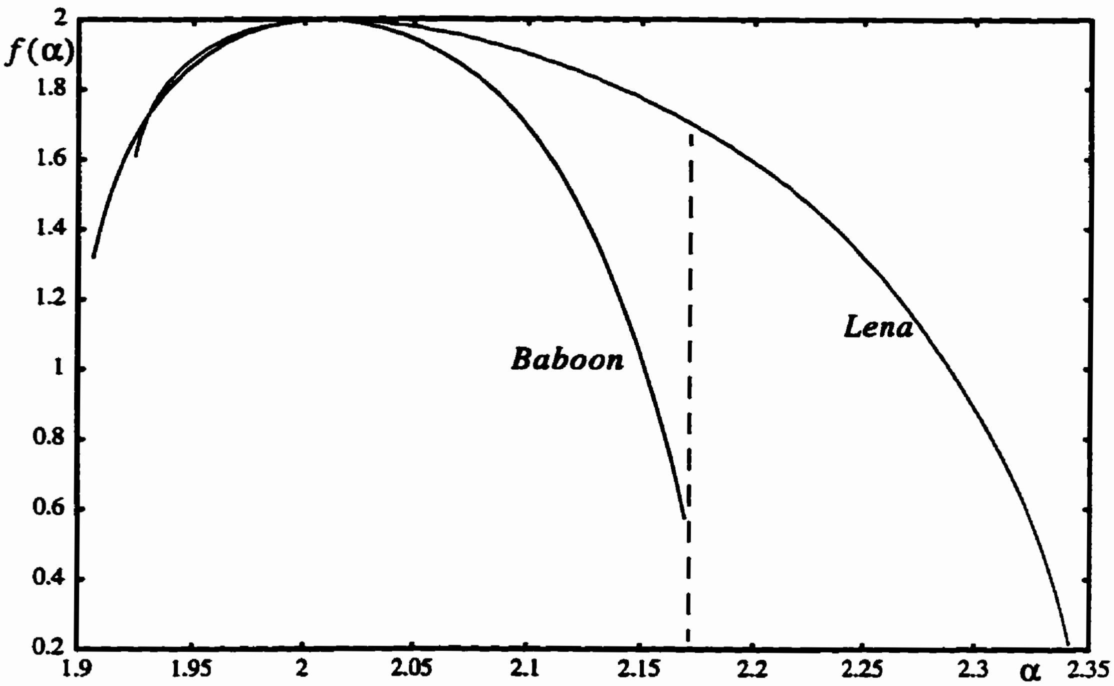


Fig. 4.12 The Mandelbrot spectrum $f(\alpha)$ of the image of Lena and Baboon.

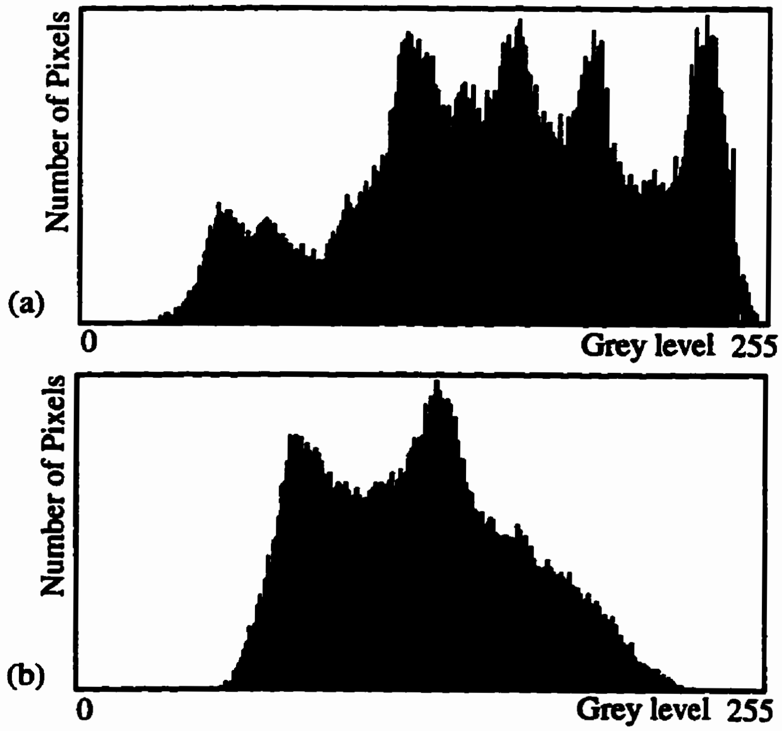


Fig. 4.13 The histograms. (a) for the image of Lena. (b) for the image of Baboon.

4.5 Overview of Chapter 4

This chapter is an extension of the multifractal measures discussed in Chapter 3 to image analysis. In this chapter, the measure of the grey level in the image is defined, and the size of a pixel is determined for fractal analysis. These are the two most important factors in the approximation of the multifractal measures. With this measures, very accurate results on some regular objects and single fractals are obtained. Based on this model, the multifractal feature map will be defined in the Chapter 5.

CHAPTER 5

EXPERIMENTAL RESULTS AND DISCUSSION

All the experiments and results will be shown in this chapter, with pertinent software listed in Appendices. In Section 5.1, we implement approximations of the multifractal measures for strange attractors, using the C programming language on a Sun SparcStation-5 running the Unix operating system (*disp.c* in Appendix A). Generating the Poincaré section of the Rössler attractor with 100,000 points takes 50 min, generating the Hénon attractor with 100,000 points takes less than one minute, and generating the Ikeda attractor with 1,000,000 points takes 5 min 30 sec. It takes 1h 30 min to calculate the multifractal measures for strange attractors when the smallest vel size 1/1024 is used. In Section 5.2, we implement the segmentation of grey-level images using a multifractal feature map (see the C program package *MulFracSeg* in Appendix A). Fifty seconds is needed for segmenting the images with size of 256 x 256 pixels. Four minutes and ten seconds is needed for images of size of 512 x 512 pixels. In Section 5.3, the C++ programming language is used on a Sun Sparcstation-5 running the Unix operating system to implement the decomposition of the Mandelbrot spectrum and its application (see the program package *DecomMand* in Appendix B). The time needed to compute the Mandelbrot spectrum components is 5 min 40 sec. The times provided here may be different on different platforms, or even on the same platform, due to the multitasking and networking functions of the Unix operating system.

5.1 Multifractal Characterization of Strange Attractors

We know from Chapter 3 that strange attractors are very complicated objects as they consist of an infinite number of single points. In practice, computation involves a finite number of points only, leading to an inherent residual error in the numerical values obtained. In the past, Smith [Smit88] and Eckmann and Ruelle [EcRu92] presented lower bounds on the number of points required to estimate a single fractal dimension, the correlation dimension D_2. In this section, we will extend the study of lower bounds from a single-fractal dimension to multifractal measures through the generalized Rényi entropy, the Rényi dimension, covering all the fractal dimensions from $D_{-\infty}$ to $D_{+\infty}$.

The accuracy of estimating multifractal measures of a strange attractor is also related to the size of volume elements (vels) used to cover the strange attractor. The sensitivity of the multifractal measures to the vels will also be discussed here.

5.1.1 Sensitivity to the Number of Points in the Numerical Representation

As we can see in Sec. 3.3.1.1, the Rényi dimension D_q is approximated by

$$\tilde{D}_q = \frac{1}{q-1}\frac{\log\sum_{j=1}^{N_\varepsilon} n_j^q - q\log N}{\log\varepsilon} \tag{5.1}$$

where N_ε is the number of volume elements (vels), each of size ε, which cover the strange attractor, the j-th vel is intersected by n_j points of the attractor and the total number of points in the strange attractor is N, and $-\infty < q < +\infty$. The value of $\tilde{D}_q$ depends on the value of the number of the points from the strange attractor. We use the Hénon attractor as an example (see Fig. 3.6). It is seen from Fig. 5.1, as the number of the points from the strange attractor changes, the estimation of the Rényi dimension D_q varies greatly. So are the singularity spectrum α_q, the Mandelbrot dimension f_q and the Mandelbrot spectrum f_α which are shown in Figs. 5.2, 5.3, 5.4, respectively.

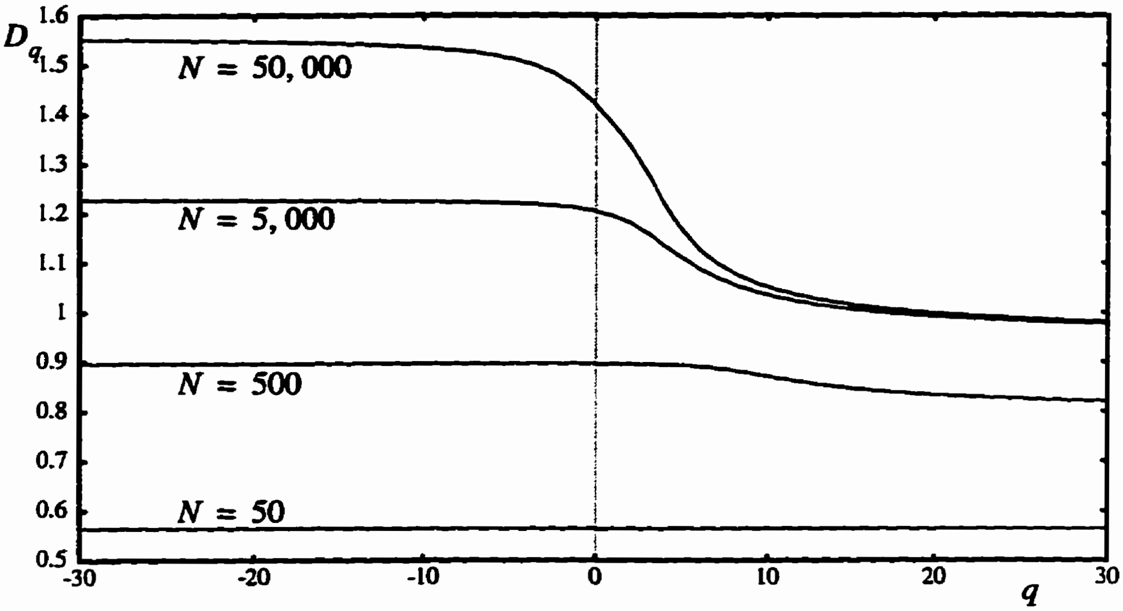


Fig. 5.1 The Rényi dimension D_q is changed when the number of the points N from the Hénon attractor varies from 50 to 50,000.

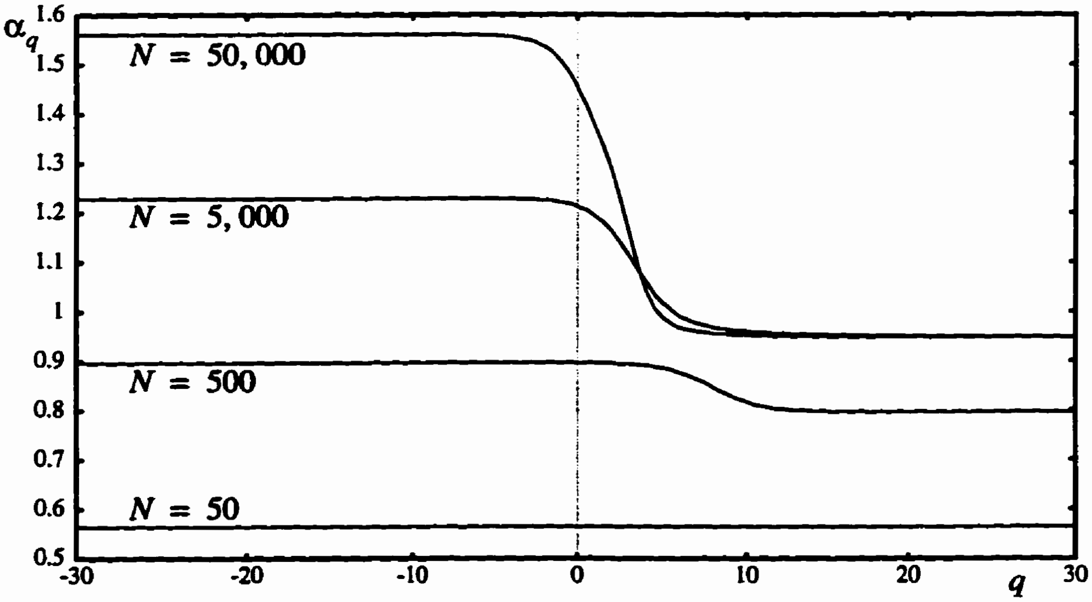


Fig. 5.2 The singularity spectrum is changed when the number of the points N from the Hénon attractor varies from 50 to 50,000.

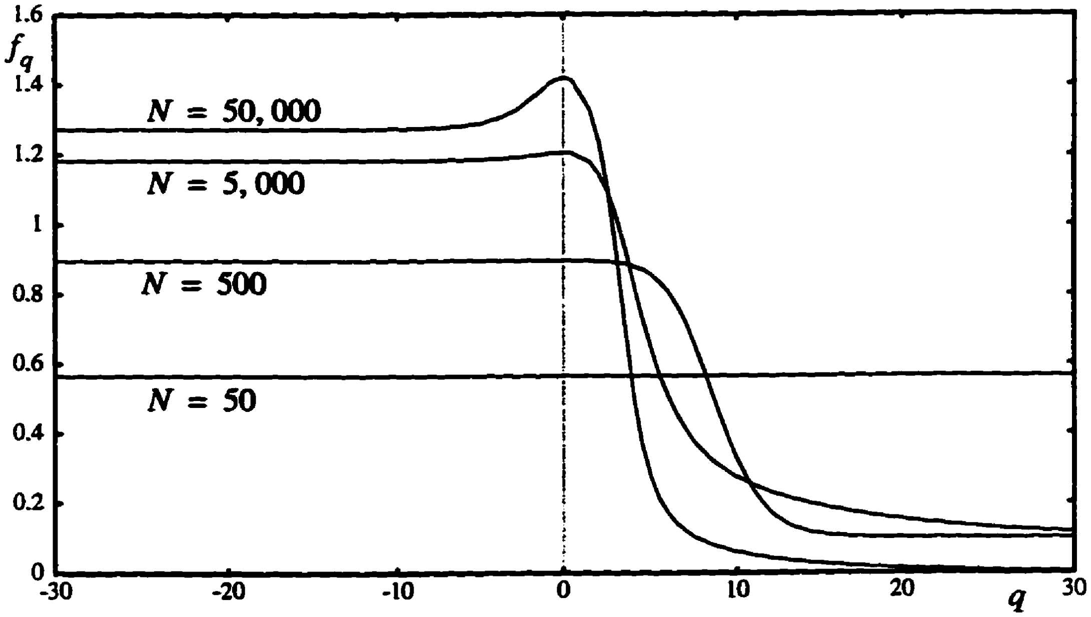


Fig. 5.3 The Mandelbrot dimension f_q is changed when the number of the points N from the Hénon attractor varies from 50 to 50,000.

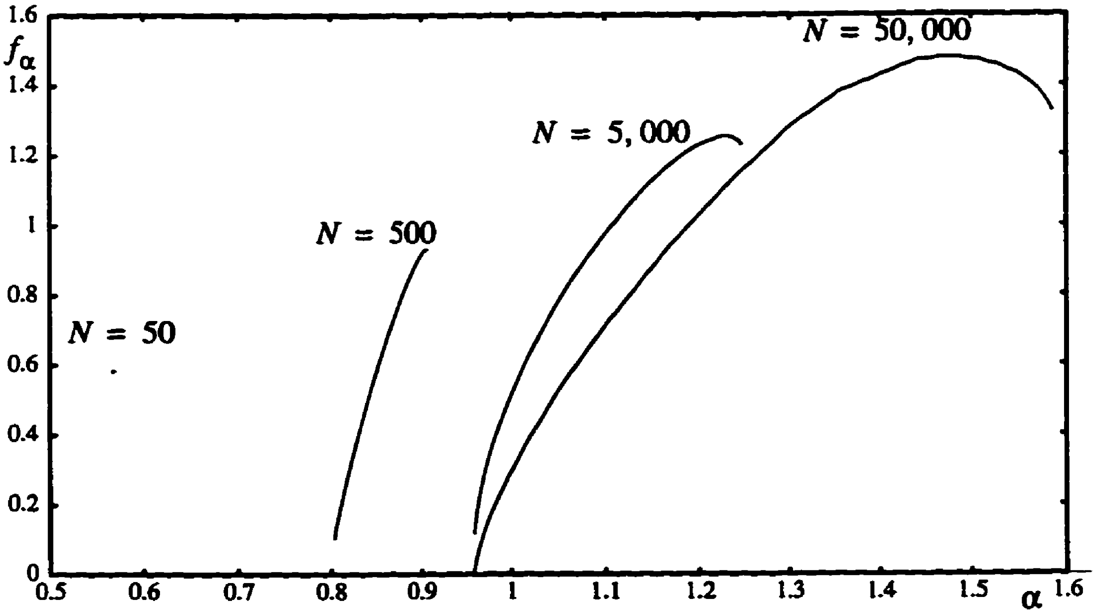


Fig. 5.4 The Mandelbrot spectrum f_α is changed when the number of the points N from the Hénon attractor varies from 50 to 50,000.

5.1.2 Sensitivity to the Vel Size in the Computational Schemes

From Eq. (5.1), the estimation $\tilde{D}_q$ of the theoretical Rényi dimension D_q not only depends on the number of the points used from the strange attractor, but also depends on the size of the volume elements. In this section, we will show the sensitivity of the estimation of the Rényi dimension $\tilde{D}_q$ to the volume size in the computational schemes. The Hénon attractor is used again as an example.

In Fig. 5.5, five different sizes of the volume element are used. ε= 1/64, 1/128, 1/ 256, 1/512, 1/1024. The experimental results show that as the vel size changes, the estimation of the Rényi dimension $\tilde{D}_q$ varies very much. As the volume size becomes smaller, the range of $\tilde{D}_q$ becomes smaller. So is the singularity spectrum α_q (see Fig. 5.6). The Mandelbrot dimension f_q and the Mandelbrot spectrum f_α are also very sensitive to the change of the vel size (see Figs. 5.7, 5.8, respectively).

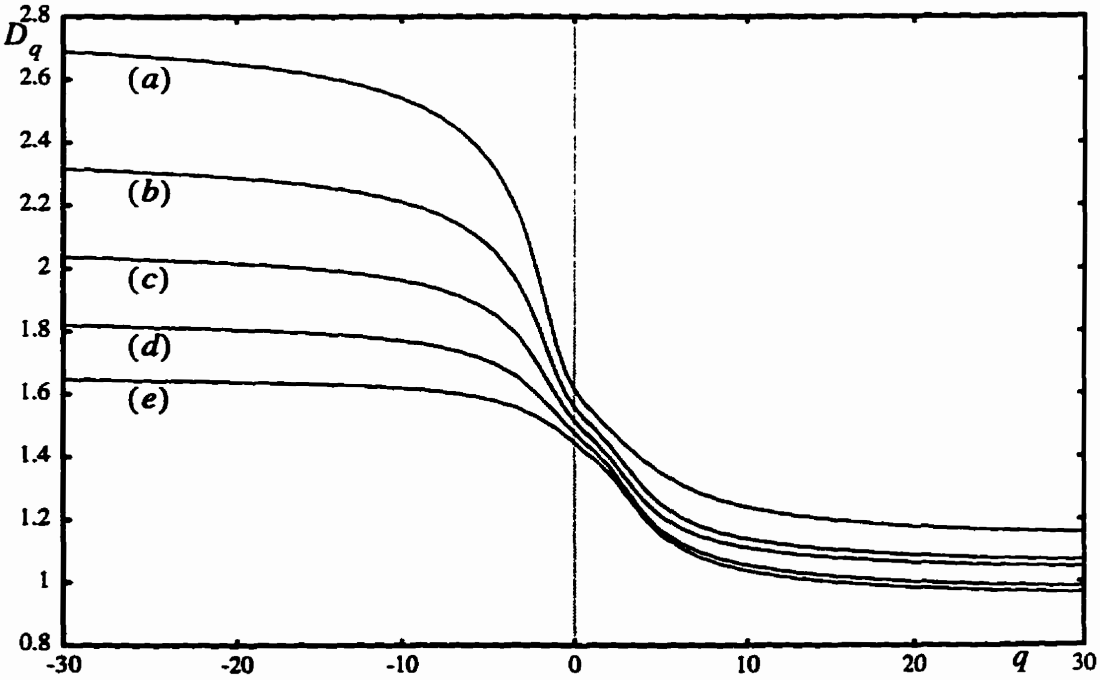


Fig. 5.5 The spectrum of the Rényi dimension varies as the vel size changes. (a) ε= 1/64. (b) ε= 1/128. (c) ε= 1/256. (d) ε= 1/512. (e) ε= 1/1024.

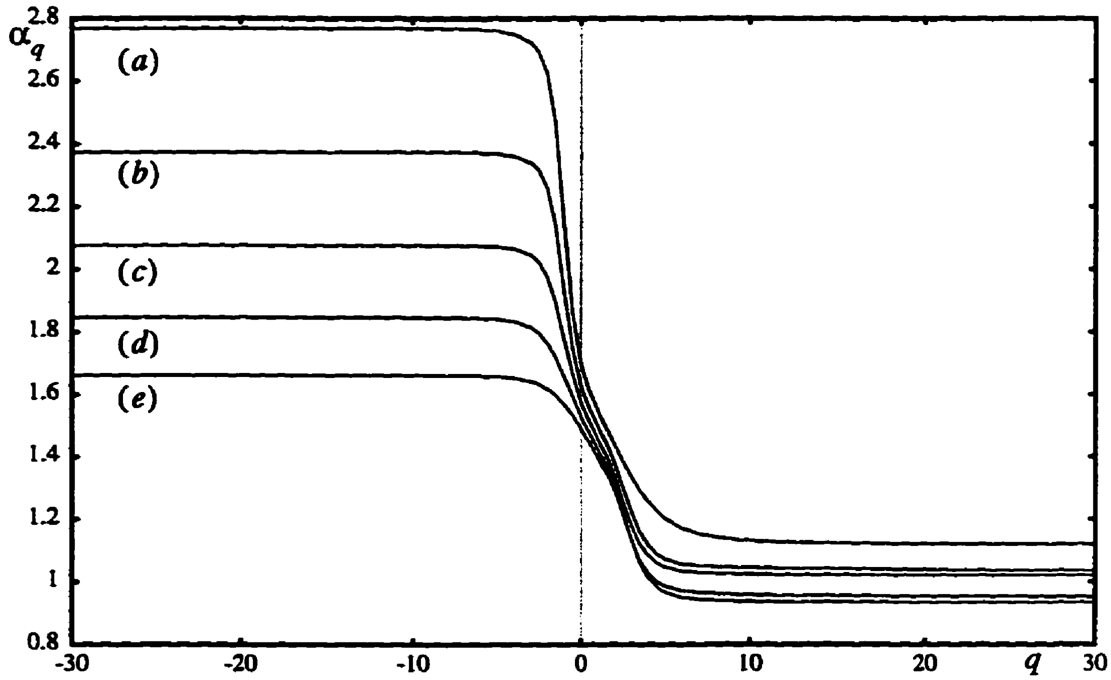


Fig. 5.6 The spectrum of singularity varies as the vel size changes. (a) ε= 1/64. (b) ε= 1/128. (c) ε= 1/256. (d) ε= 1/512. (e) ε= 1/1024.

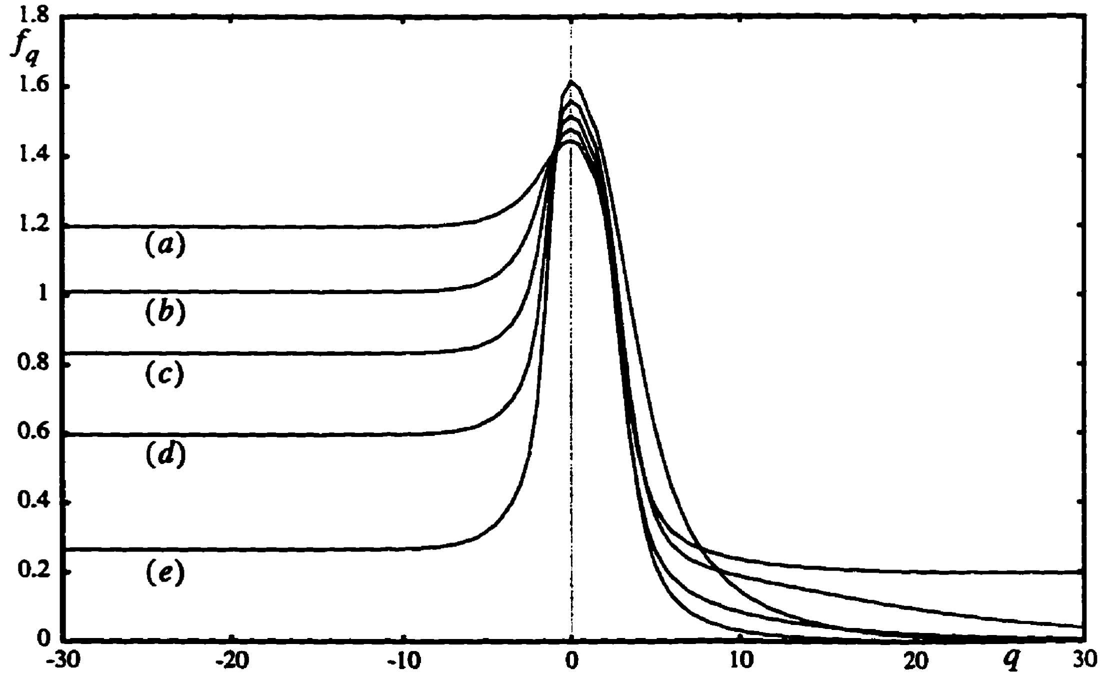


Fig. 5.7 The spectrum of the Mandelbrot dimension varies as the vel size changes. (a) ε= 1/64. (b) ε= 1/128. (c) ε= 1/256. (d) ε= 1/512. (e) ε= 1/1024.

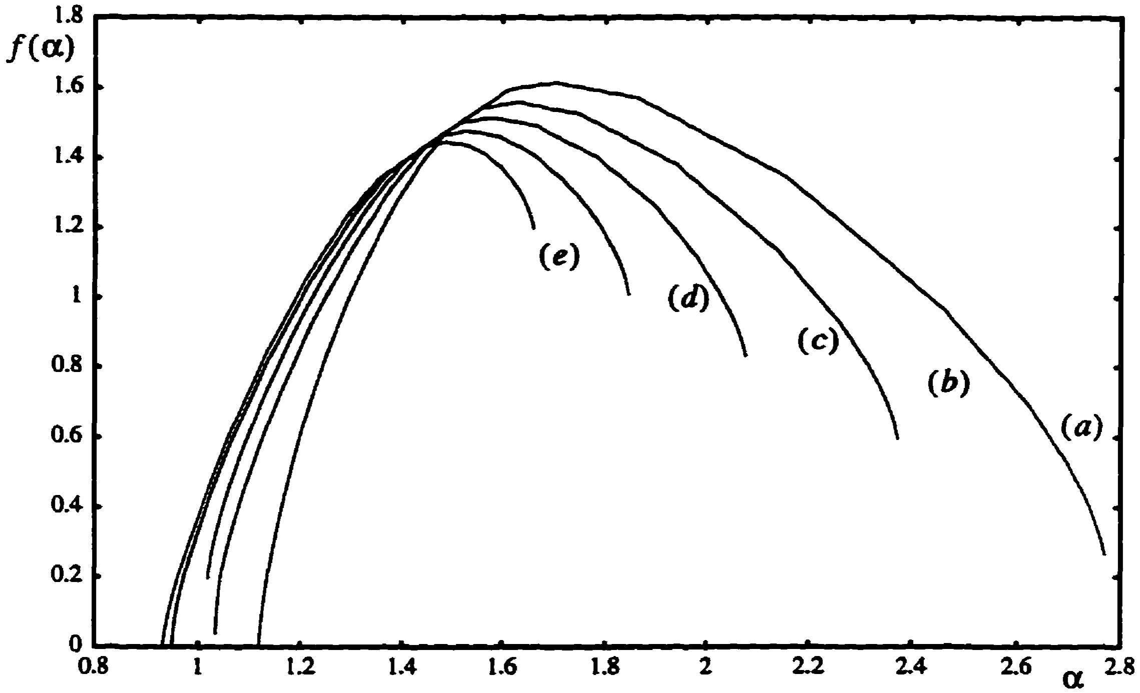


Fig. 5.8 The Mandelbrot spectrum varies as the vel size changes. (a) $\varepsilon= 1/64$. (b) $\varepsilon= 1/128$. (c) $\varepsilon= 1/256$. (d) $\varepsilon= 1/512$. (e) $\varepsilon= 1/1024$.

5.1.3 Compromise of the Number of Points and the Vel Size

Since the estimations of multifractal measures are sensitive to both the number of the points used from the strange attractor and the size of volume elements used to cover the strange attractor, a compromise of the number of points and the vel size should be made. In order to get more accurate information about the distribution density of the points in the strange attractor, the vel size should be as small as possible. We find that the smaller the vel size we use, the larger the number of points from the strange attractor is required in the estimation to obtain more accurate results.

Figures. 5.9 shows the experimental results for the Rényi dimension of the Hénon attractor. The result shows the convergence property when N is increased to 100,000. We use $N = 50$, 500, 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ... , 90000, 95000, 100000. When N=50, we find that the approximation of the Rényi dimension is actually a straight line. A straight line for the Rényi dimension means the attractor is not a multifractal fractal set. This conclusion is obviously wrong. The reason that it is a straight line is because too few points have been used, no vel has more than one point in it. So, 50 points in the Hénon attractor never means a good approximation of the Hénon attractor. However, when N increases, the ranges of D_q are also increased. When N is large enough, the Rényi dimension tends to converge. It can be seen that when q is larger than zero, the convergence property is shown very well for relatively smaller N. But when q is smaller than zero, larger N is required for the Rényi dimension to converge. For example, when N increases from 500 to 5,000, the Rényi dimension at $q = -30$ increases by 1.227272 − 0.89654 = 0.330732, but this difference diminishes monotonically to 0.014259 for N from 90,000 to 100,000.

The experimental results for the Mandelbrot spectrum are shown in Fig. 5.12. As N increases from 50 to 100,000, the Mandelbrot spectrum also tends to converge. We notice that when $N = 50$, the Mandelbrot spectrum degrades to a point. This means that there is no variation for the singularity, because at this time, just as the Rényi dimension, the singularity spectrum is also a straight line.

Figures 5.11 and 5.12 show the experimental results for the Poincaré section for the Rössler attractor. The smallest vel size ε we use is 1/1024. The experimental results show the convergence property when N is increased to 100,000. In Fig. 5.9, we use $N =$ 1000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ... , 90000, 95000, 100000. When N increases, the ranges of D_q are also increased. When N is large enough, the Rényi dimension tends to converge. For example, when N increases from 1,000 to 10,000, the Rényi dimension at $q = -30$ increases by 1.316415 − 0.993553 = 0.322862, but this difference diminishes monotonically to 1.631421 − 1.616944 = 0.014477 for N from 90,000 to 100,000. The experimental results for the Mandelbrot spectrum are shown in Fig. 5.10. As N increases to 100,000, the Mandelbrot spectrum also tends to converge.

The experimental results for the Ikeda attractor (see Fig. 3.7) are shown in Figs. 5.13 and 5.14. This attractor looks more complicated than the other two attractors we have just discussed. Unlike the Poincaré section of the Rössler attractor and the Hénon attractor, the Ikeda attractor is very scattered.

If we use the same numbers of points from the Ikeda attractor as the numbers of the points from the Poincaré section of the Rössler attractor and the Hénon attractor, either the Rényi dimension or the Mandelbrot spectrum will not tend to converge. But when we continue to increase the points from the Ikeda attractor, the convergence property occurs when N is close to 1,000,000. For example, when N increases from 5,000 to 50,000, the Rényi dimension at $q = -30$ increases by $1.560204 - 1.228651 = 0.331553$, but this difference diminishes monotonically to $1.983433 - 1.968865 = 0.014568$ for N from 900,000 to 1,000,000.

This indicates that different attractors may require different number of points to get more accurate estimations of the multifractal measures. This can also be understood from the probability point of view, the scattered attractors need relatively more points from the attractors to make probability in the strange attractor meaningful.

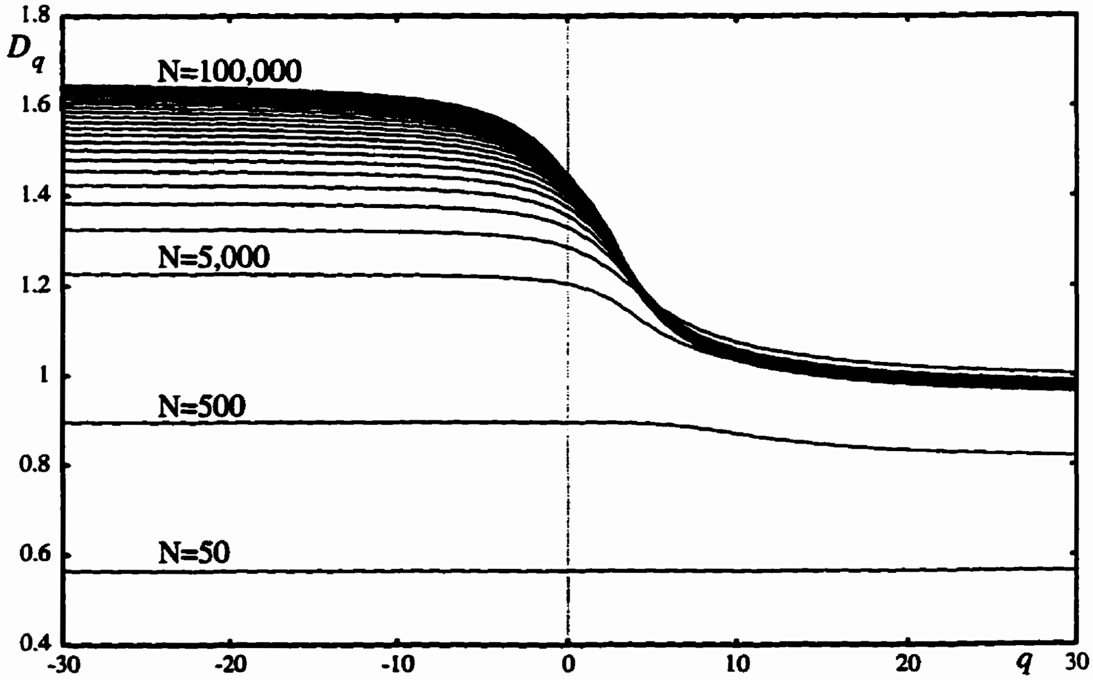


Fig. 5.9 The convergence of approximation of the Rényi dimension for the Hénon attractor for N = 50, 500, 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ..., 90000, 95000, 100000.

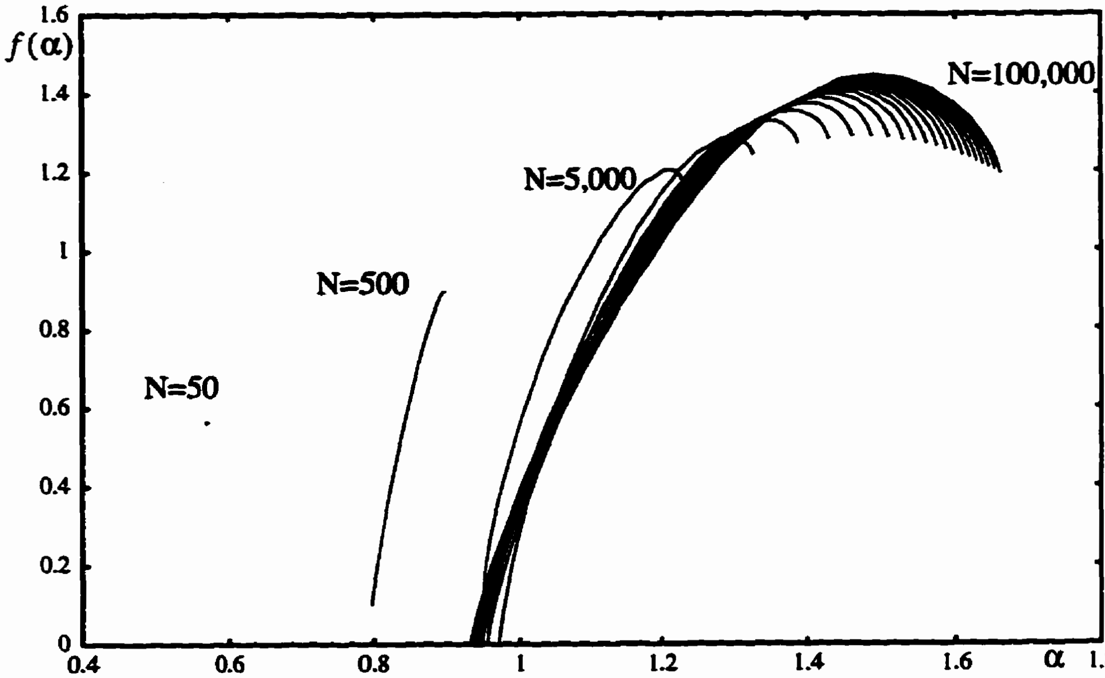


Fig. 5.10 The convergence of the approximation of the Mandelbrot spectrum for the Hénon attractor for N = 50, 500, 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ... , 90000, 95000, 100000.

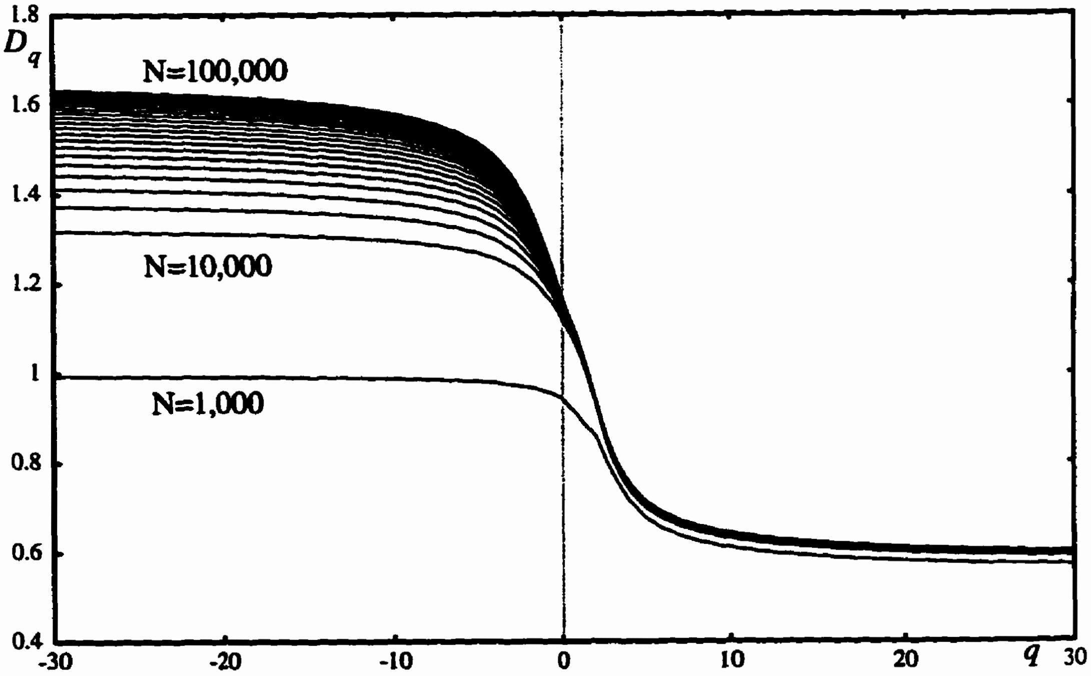


Fig. 5.11 The convergence of the approximation of the Rényi dimension of the Poincaré section of the Rössler attractor for N=1000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ... , 90000, 95000, 100000.

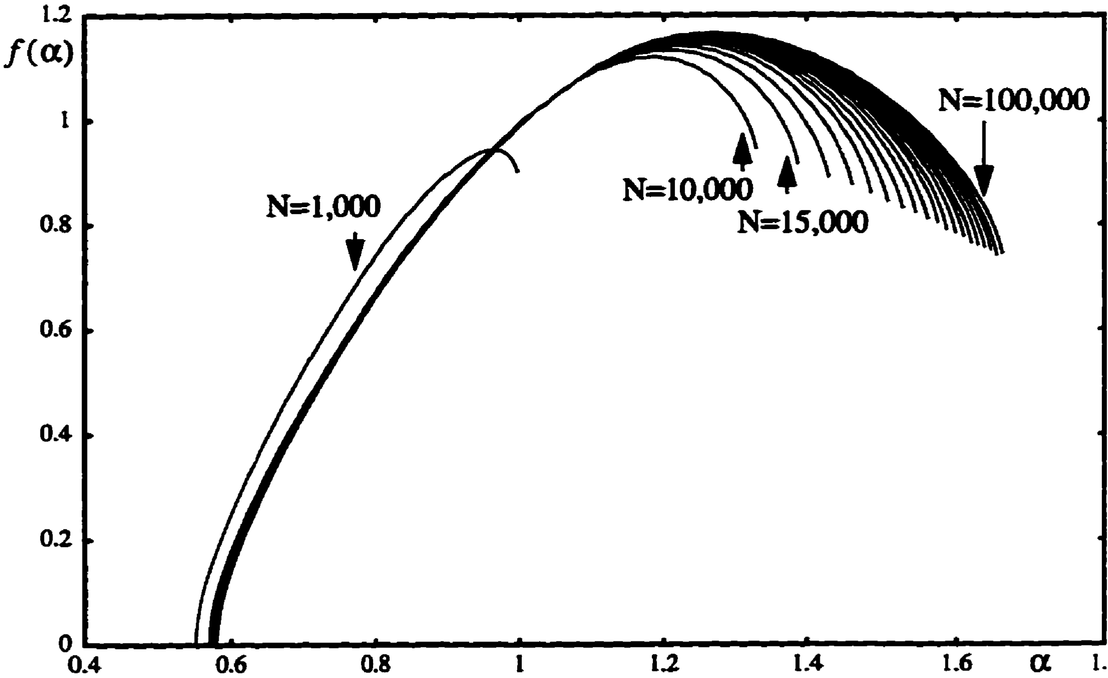


Fig. 5.12 The convergence of the approximation of the Mandelbrot spectrum of the Poincaré section of the Rössler attractor for N=1000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ... , 90000, 95000, 100000.

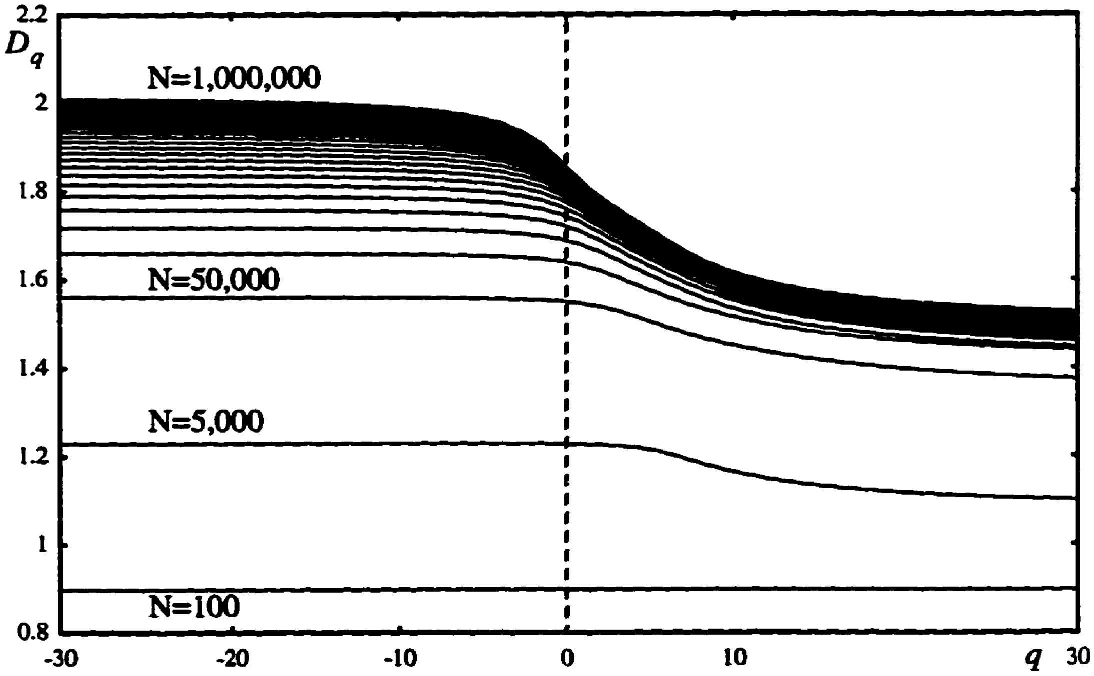


Fig. 5.13 The convergence of the approximation of the Rényi dimension of the Ikeda attractor, N = 100, 5000, 50000, 100000, 15000, 200000, 25000, ..., 900000, 95000, 1000000.

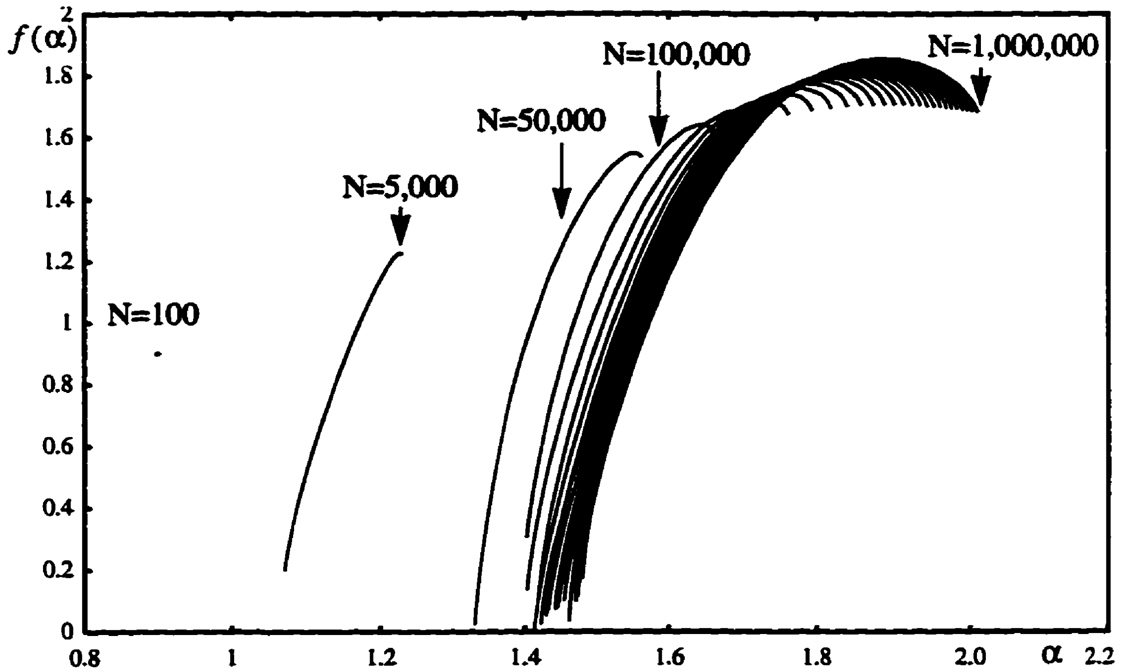


Fig. 5.14 The convergence of the approximation of the Mandelbrot spectrum of the Ikeda attractor, N = 100, 5000, 50000, 100000, 15000, 200000, 250000, ..., 900000, 95000, 1000000.

After the discussion of the multifractal measures of the strange attractors, we come back to the Hausdorff dimension D_0 of the three strange attractors. The Hausdorff dimension D_0 of the Hénon attractor, the Poincaré section of the Rössler attractor and the Ikeda attractor, as a special case of the Rényi dimension, tends to converge to a line as the numbers of the points from the corresponding strange attractors increases. The evolutions of the convergence of the Hausdorff dimension D_0 are shown in Figs 5.15, 5.16 and 5.17, respectively.

Figure 5.15 shows that at least 70,000 points from the strange attractor are needed to get an accurate estimation of the Hausdorff dimension D_0. For the Poincaré section of the Rössler attractor, 30,000 points will give us very good result. We also notice that when $N < 10{,}000$, D_0 increases very fast, after that it increases very slowly. So, if we use fewer than 10,000 points to calculate D_0, the result will not be accurate. Figure 5.15 also shows that fewer points are needed than that of the Hénon attractor for the Rényi dimension to converge. Figure 5.17 shows that the Hausdorff dimension of the Ikeda attractor requires more than 600,000 points to converge.

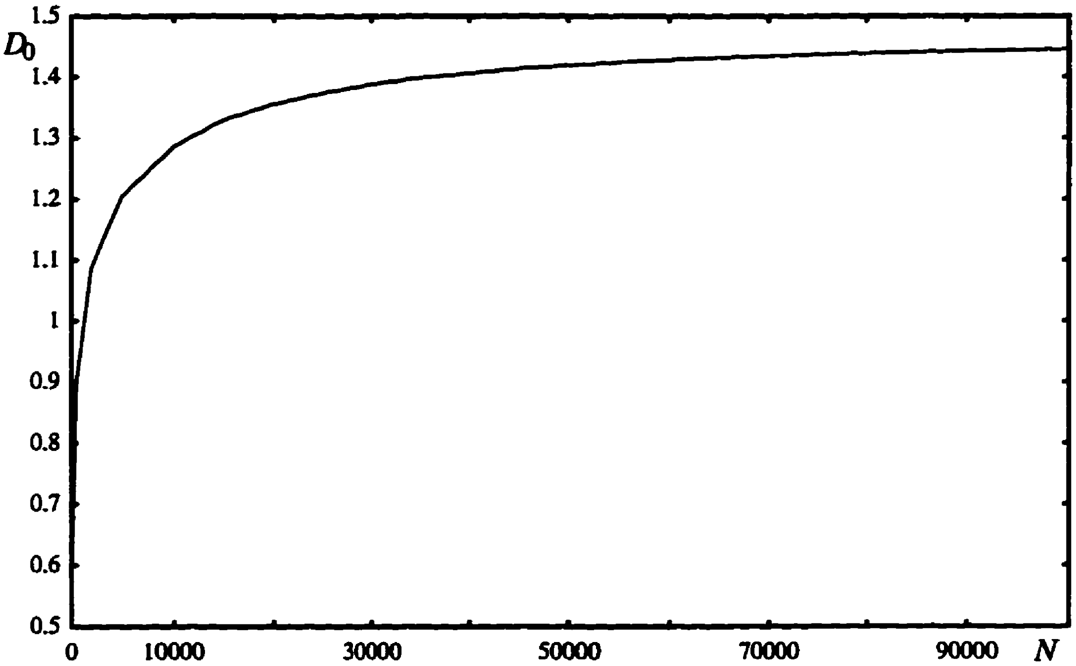


Fig. 5.15 The convergence of the approximation of the Hausdoff dimension D_0 of the Hénon attractor for N = 50, 500, 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ... , 90000, 95000, 100000.

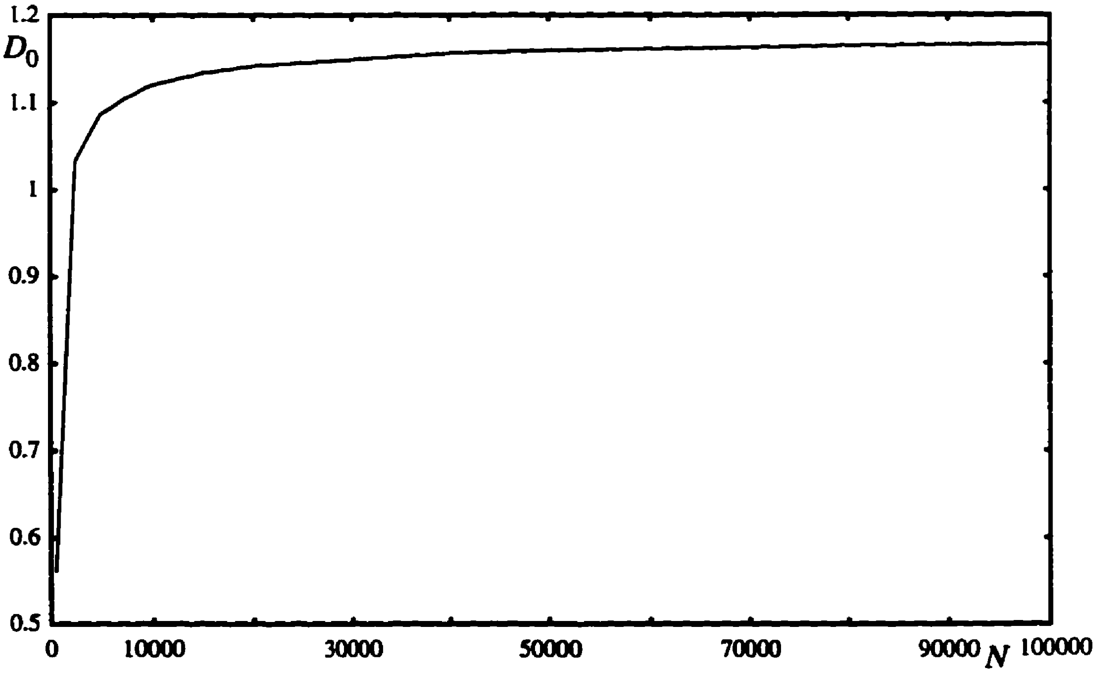


Fig. 5.16 The convergence of the approximation of the Hausdorff dimension D_0 of the Poincaré section of the Rössler attractor for N = 50, 500, 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, ..., 90000, 95000,

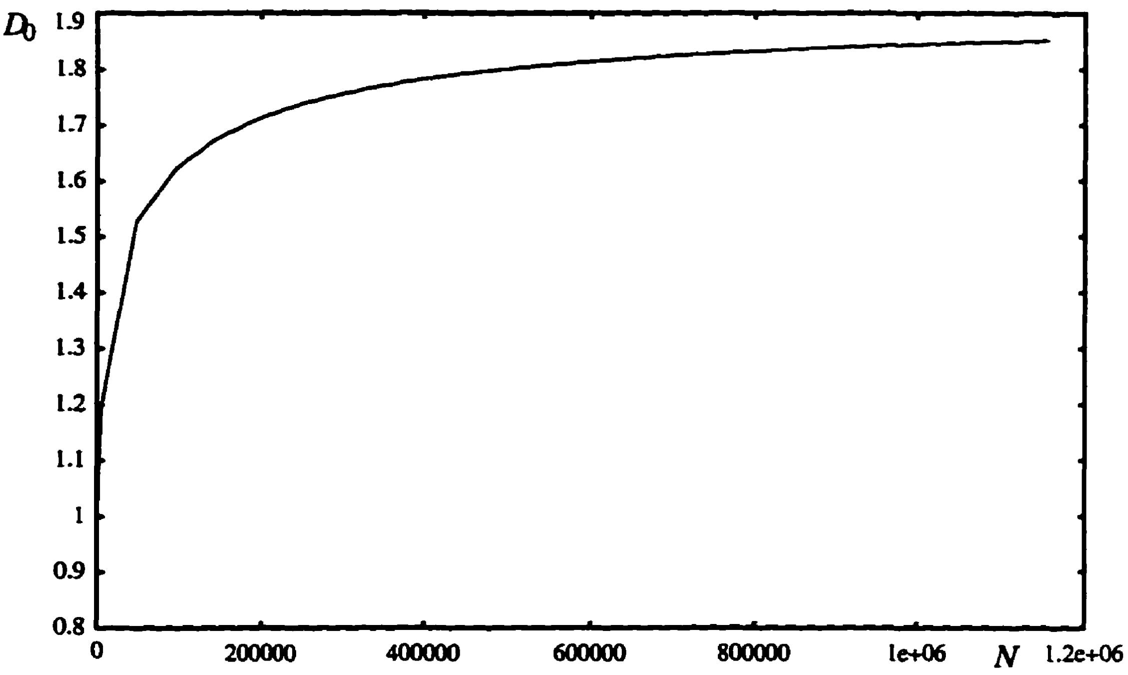


Fig. 5.17 The convergence of the approximation of the Hausdoff dimension D_0 of the Ikeda attractor for N = 100, 10000, 50000, 100000, 15000, 200000, ..., 900000, 95000, 1000000.

From Figs. 5.15, 5.16 and 5.17, it can be seen that a sufficiently large number of the points from the strange attractors should be used in the approximation of the Hausdoff dimension D_0. So, our conclusion goes in the same direction as those of Procaccia [Proc88] discussing work of Tsonis and Elsner [TsEl88], and Nicolis and Nicolis discussing work of Grassberger [Gras86] about a weather attractor in the study of climate and weather over timescales ranging from decades to thousands of years.

5.2 Application of Multifractal Measures in Image Analysis

Image analysis is an important research field which has a number of applications such as satellite imaging, medical imaging, and robotics. We restrict ourselves here to the problem of image segmentation. In segmentation, we want to extract from the image a compact description in terms of edges and/or regions. Essentially, image segmentation consists in finding all the characteristic entities of an image. These entities are either their contours (edge detection) or the region (region extraction). Different from classical methods for edge detection and region extraction, a multifractal feature map will be used in this section.

5.2.1 Multifractal Feature Map

In order to obtain multifractal information from an image, a mask of size *b* x *b* pixels is used to extract the feature from the image. The mask moves over the image vertically and horizontally. With a multifractal feature map, a feature value is returned when the mask moves to each position. So, a feature matrix can be obtained through this procedure. Based on the multifractal feature matrix, the edges can be detected and the textures can be extracted.

Figure 5.18 explains the method that we use to map an image of 3 by 5 pixels to the fractal feature matrix. The shaded area is a 3 by 5 image, each block representing one pixel, and a 3 by 3 mask moves over the image. The mask starts from the first square $A_1A_2A_3A_4$, the multifractal feature map returns a value *a* to the feature matrix at (0,0), then the mask goes to $B_1B_2B_3B_4$, value *b* is returned to the feature matrix at (0,1), finally the mask moves to $C_1C_2C_3C_4$, value *c* is returned to the feature matrix at (0,2) by the multifractal feature map.

When the image size is larger than 3 by 5 pixels, the mask will move horizontally then vertically to obtain the feature values through the multifractal feature map. So, with a mask of size *b* by *b* pixels, through the multifractal feature map, an image of *n* by *n* pixels can be mapped to a feature matrix which is $(n\text{-}b\text{+}1)$ by $(n\text{-}b\text{+}1)$.

The rate of change of the singularity and a density function are used as two multifractal feature maps to detect minor changes of the features. Suppose we deal with an image which is $s \times s$ pixels with 8 bpp, and the grey level of the pixels in the image is represented by G_j ($j = 1, 2, 3, \ldots s^2$). According to Eq. (3.19), the rate of change of the singularity can be rewritten as follows:

$$\frac{d}{dq}\tilde{\alpha}_q = \frac{\left(\sum_{j=1}^{N\varepsilon} n_j^{\,q}\right)\sum_{j=1}^{N\varepsilon} n_j^{\,q}(\log n_j)^2 - \left(\sum_{j=1}^{N\varepsilon} n_j^{\,q}\log n_j\right)^2}{\left(\sum_{j=1}^{N\varepsilon} n_j^{\,q}\right)^2 \log\varepsilon} \tag{5.2}$$

where $n_j = 255 - G_j$, $N_\varepsilon = s^2$ and $\varepsilon = \frac{1}{s}$.

A density function DF_q is defined in order to select different textures in different region as follows:

$$DF_q = \frac{\log \sum_{j=1}^{s^2} (255 - G_j)^q}{\log\frac{1}{s}} \tag{5.3}$$

From Eq. (5.2), the pixels with smaller grey level values correspond to a larger value of DF_q which means high density, and the pixels with larger grey level values correspond to a smaller value of DF_q which means lower density.

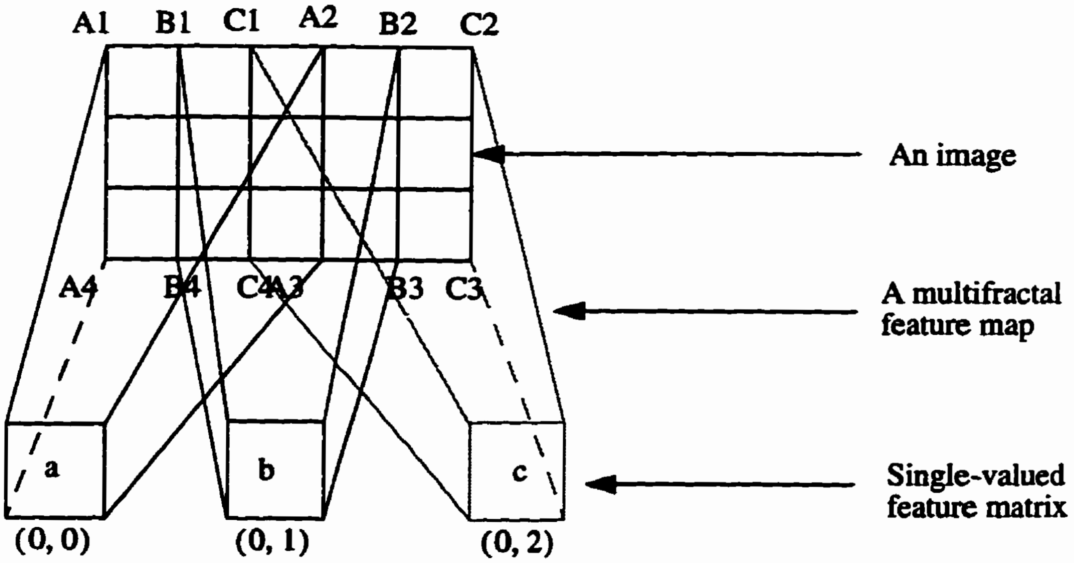


Fig. 5.18 The mapping from an image to multifractal feature matrix.

5.2.2 Relationship Between Edges and Textures

In Fig. 5.19a, three edges are shown, they can be easily detected by any edge detector. In Fig. 5.19b, we have tripled the number of lines in the image. This image can also be interpreted as being composed of nine edges, but it is commonly called a binary texture.

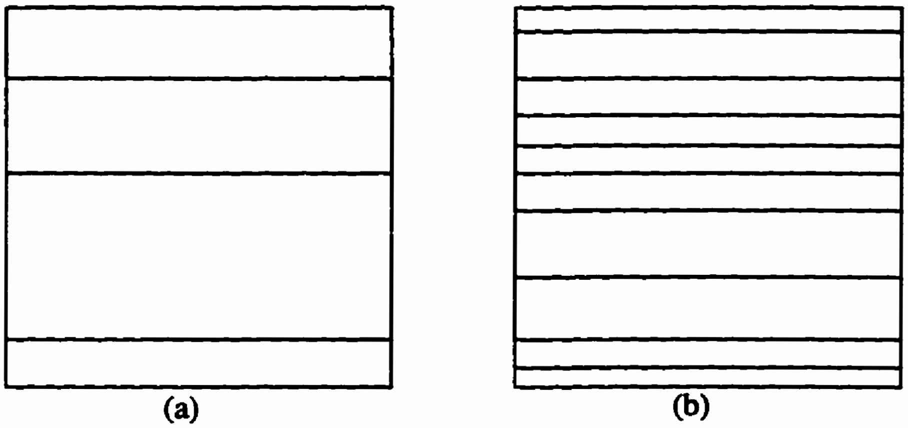


Fig. 5.19 An illustration of edges and a texture.

Figure 5.20 shows another example of four corner edges and texture. The edges correspond to a certain type of the singularity in the image. If too many edges are detected in a portion of an image shown in Fig. 5.20b, then the human visual system will have a tendency to consider it as a textured zone, rather than a concentration of edges.

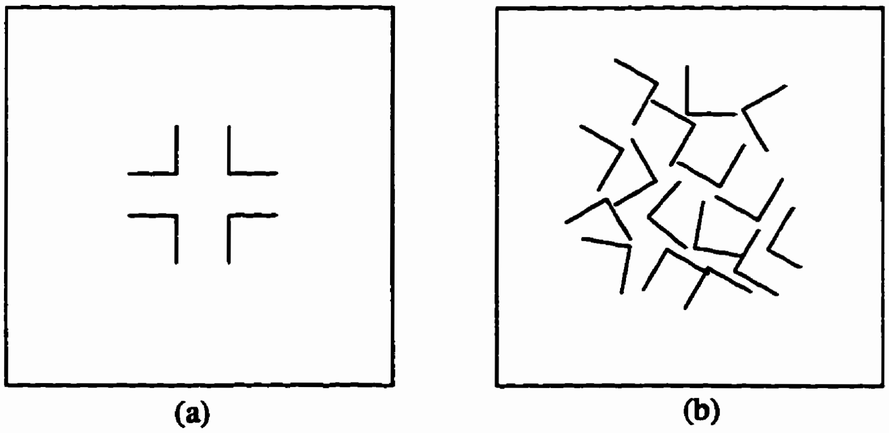


Fig. 5.20 An illustration of corners and corners texture.

5.2.3 Edge Detection

In this section, two kinds of images are used as examples. One is an aerial image, the other is a cancer cells image (both a benign cancer cells image and a malignant cancer image). A malignant cancer cells image is far more complicated than the other images. It is easy to detect the edges for an aerial image and a benign cancer image using classical methods. For these images, we will see that the multifractal method can also produce results as good as the classical methods or even better. For the malignant cancer cell image, classical methods do not perform well, but the multifractal method still produces very good results.

5.2.3.1 Edge Detection of Aerial Images Using Multifractal Feature Maps

In this section, an aerial image shown in Fig. 5.21 is used as an example. The size of the original image is 512 by 512 pixels, with 8 bits representing each pixel. As we discussed earlier, multifractal feature maps are chosen for the image, and the corresponding multifractal feature matrices are computed.

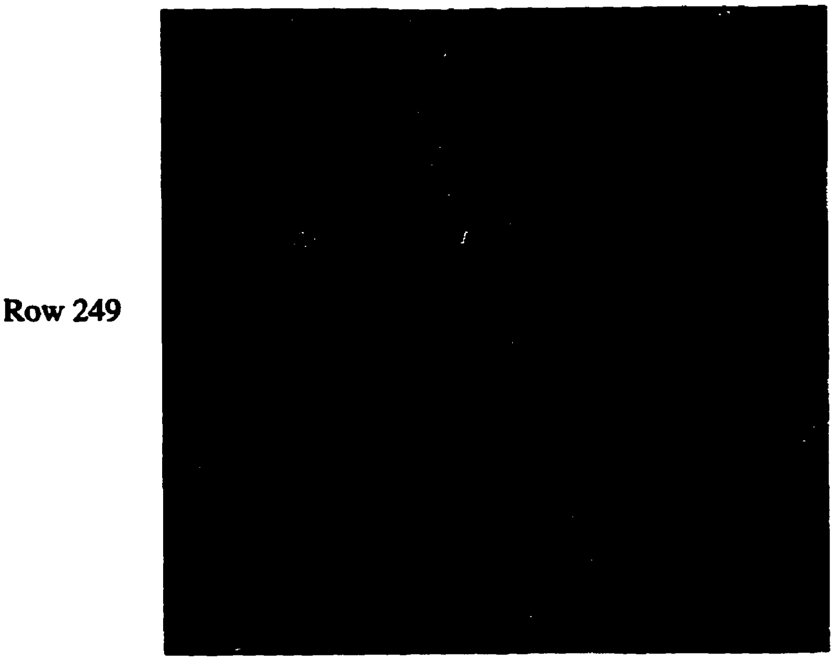


Fig. 5.21 An aerial image (512x512x8).

In this application, the rate of change of the singularity at $q = 1$ is used. The reason that we choose the rate of change of the singularity at $q = 1$ is that when q is too large or too small, the rate of change of the singularity is not very sensitive. This can be understood from the multifractal analysis of strange attractors.

The multifractal feature vector at row 249 is shown in Fig. 5.22. We can see many spikes in Fig. 5.22a, the spikes convey information of the multifractal features that the image contains. The larger spikes represent the greater differences of the singularity between the two neighbouring masks. The relationship between the two multifractal feature matrices can also be partially seen from the relationship of the two multifractal feature vector at row 249. With different density regions, the range of the rate of change of the singularity is different. From the multifractal feature vectors shown in Fig. 5.22, we can see the edges of this image are not difficult to detect. The density of the image can also be divided into several layers, with a threshold assigned to each layer.

A threshold for $\tilde{\alpha}_q'$ is set at 0.001 when the value of density function DF_q is between 7.2 and 7.4. When the value of density DF_q is between 6.8 and 7.2, a threshold for $\tilde{\alpha}_q'$ is set at 0.003. The threshold for $\tilde{\alpha}_q'$ is set at 0.008 when the value of density DF_q is between 6.1 and 6.8. The edge detection result for the aerial image is shown in Fig. 5.23.

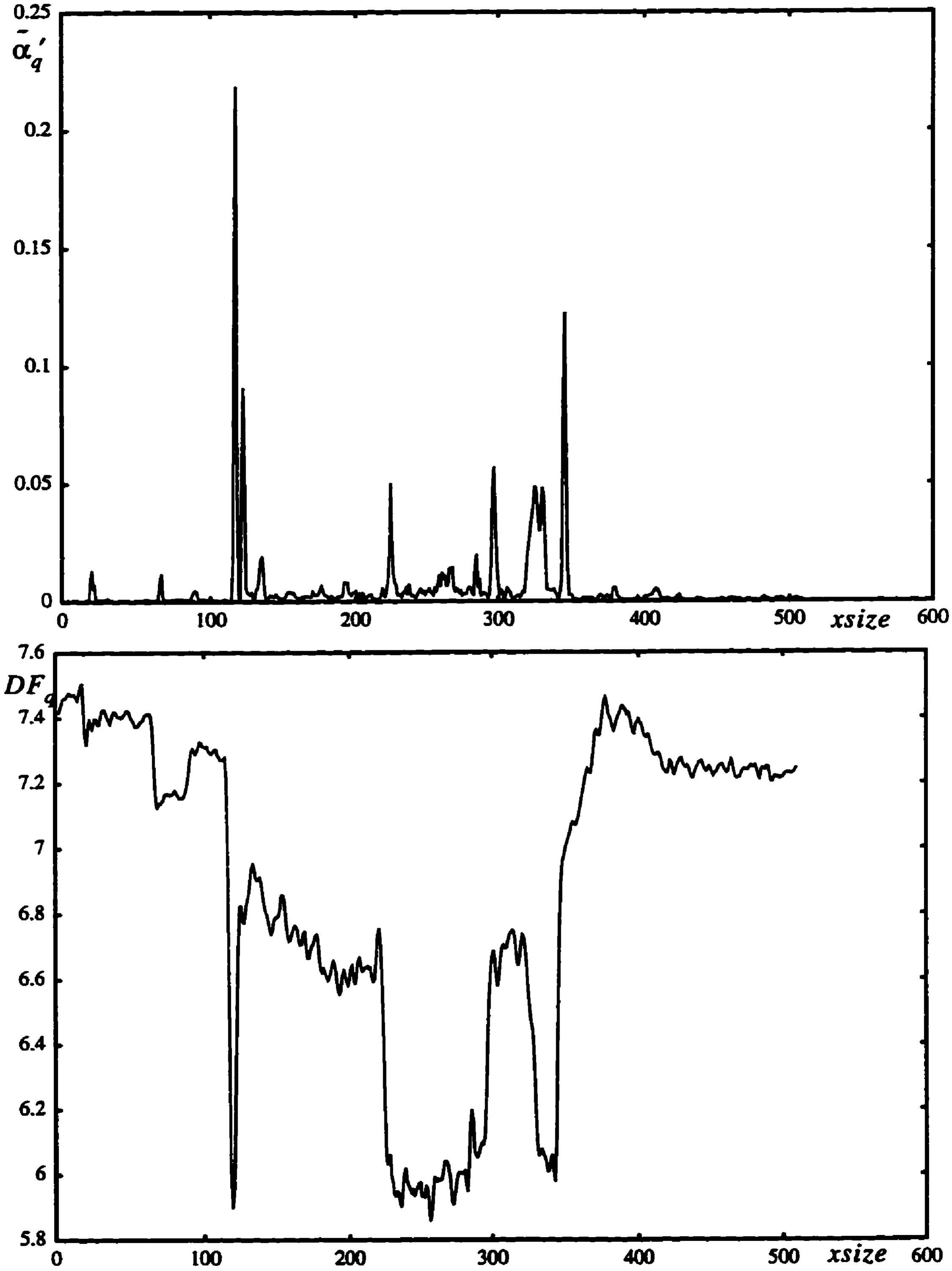


Fig. 5.22 The multifractal feature vector at row 249. (a) The rate of change of the singularity spectrum. (b) The density function DF_q.

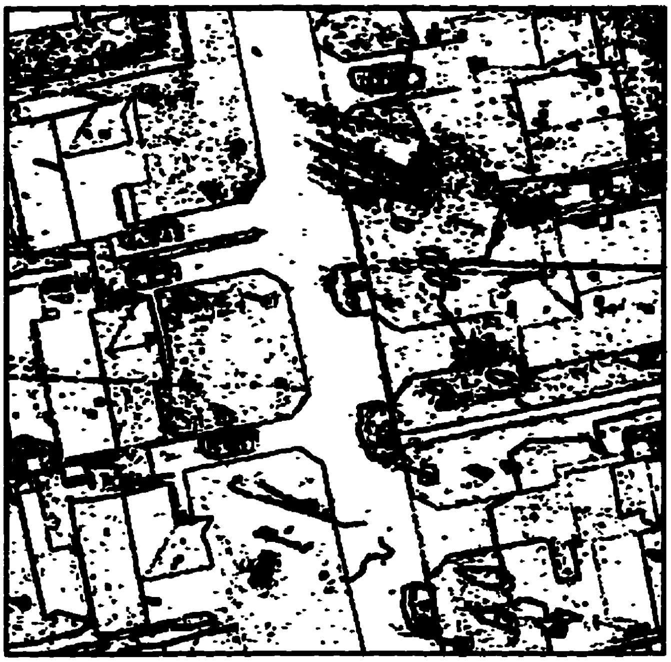


Fig. 5.23 Edge detection result for an aerial image using the multifractal method.

5.2.3.2 Classical Edge Detectors for the Aerial Image

In order to compare the results obtained in the previous section, some classical methods and their applications are presented in this section. The Sobel operator, the Prewitt operator, the Robert operator, and the Laplacian operator are used in edge detection.

Table 5.1: Classical Edge Detection Masks

Sobel Operator	Prewitt Operator	Robert Operator	Laplacian Operator
$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

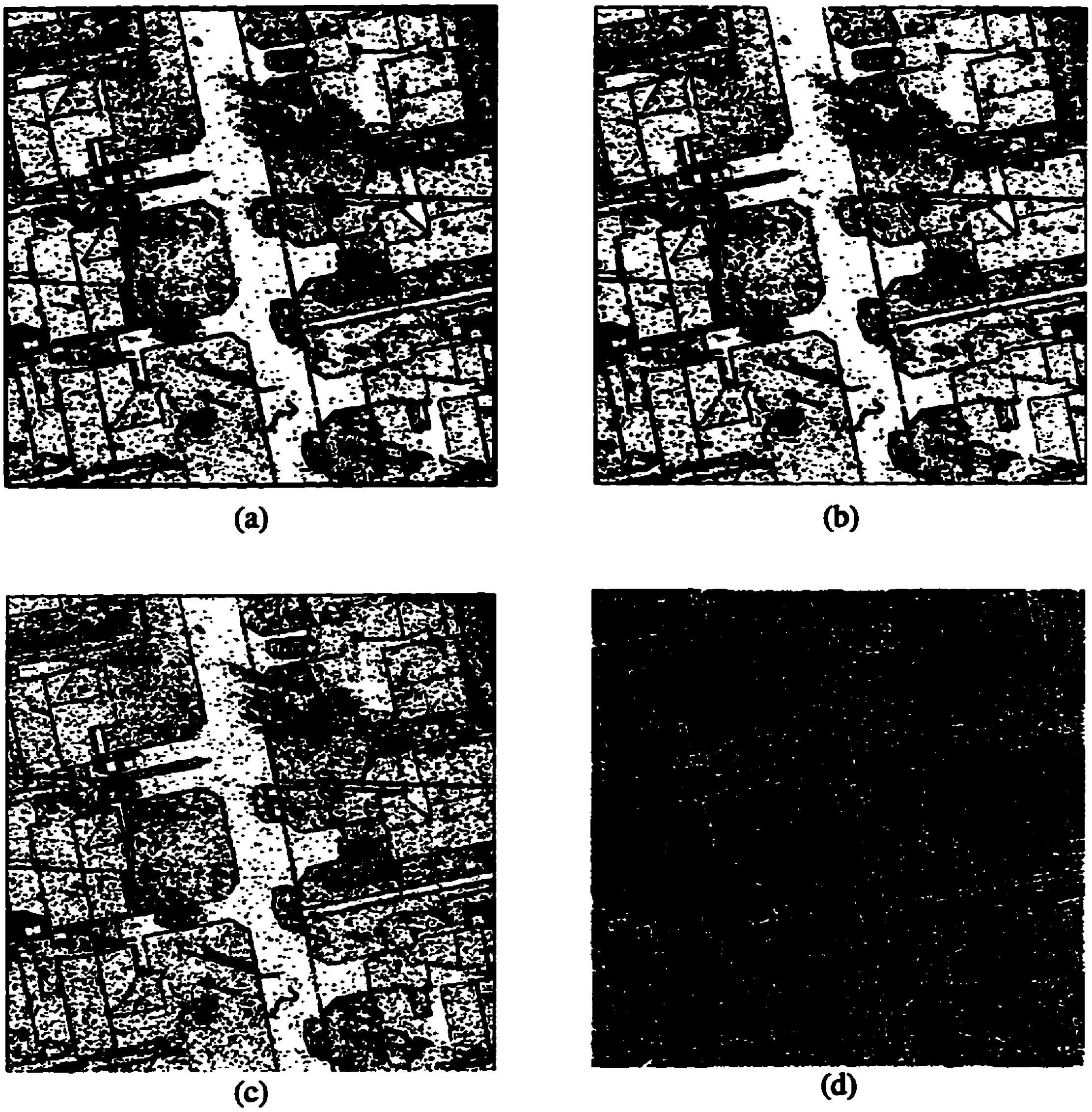


Fig. 5.24 Edge detection using classical operators. (a) Thresholding Sobel operator at T=77. (b) Thresholding Prewitt operator at T=77. (c) Thresholding Robert operator at T=19. (d) Thresholding Laplacian operator at T=140.

From the experimental results, we can see that the Sobel operator and the Prewitt operator can produce good results, but comparing to the result using multifractal method, more noise is produced.

5.2.4 Boundary Extraction For Cancer Cells

In this section, a benign cancer cells image and the a malignant cancer cells image are used as examples. Classifying pre-malignant mucosal leisons is one of the major difficulties for pathologists. The shape and irregularity of the cells are what we are interested in here.

5.2.4.1 Boundary Extraction for a Benign Cancer Cells Image

An image of benign breast cancer cells is shown in Fig. 5.25. The size of the original image is 256 by 256 pixels, with 8 bits representing each pixels. As we discussed earlier, based on the image, multifractal feature maps are chosen and with the chosen map the multifractal feature matrices should be produced.

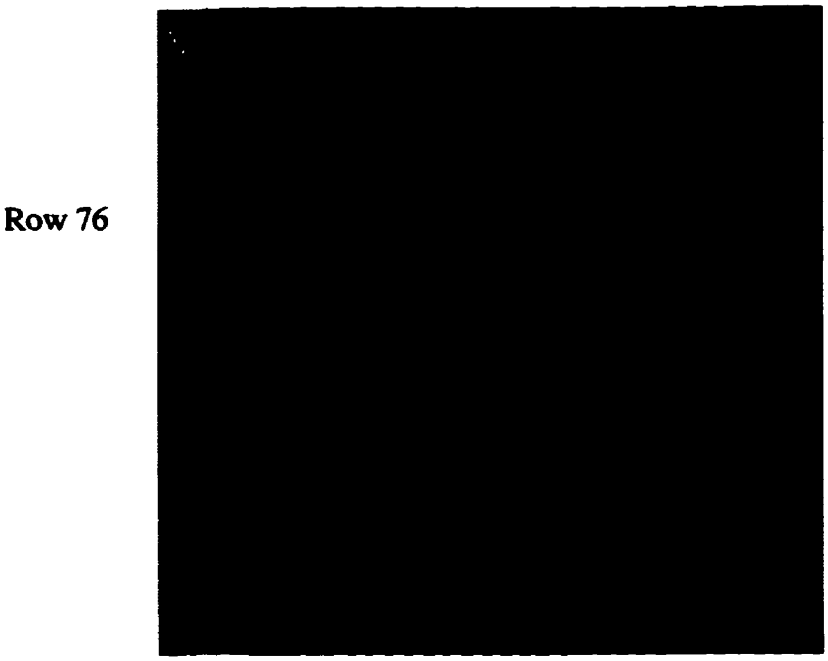


Fig. 5.25 An image of benign breast cancer cells.

In this application, the density function in Eq. (5.3) and the rate of change of the singularity in Eq. (5.2) at $q = 1$ are used again.

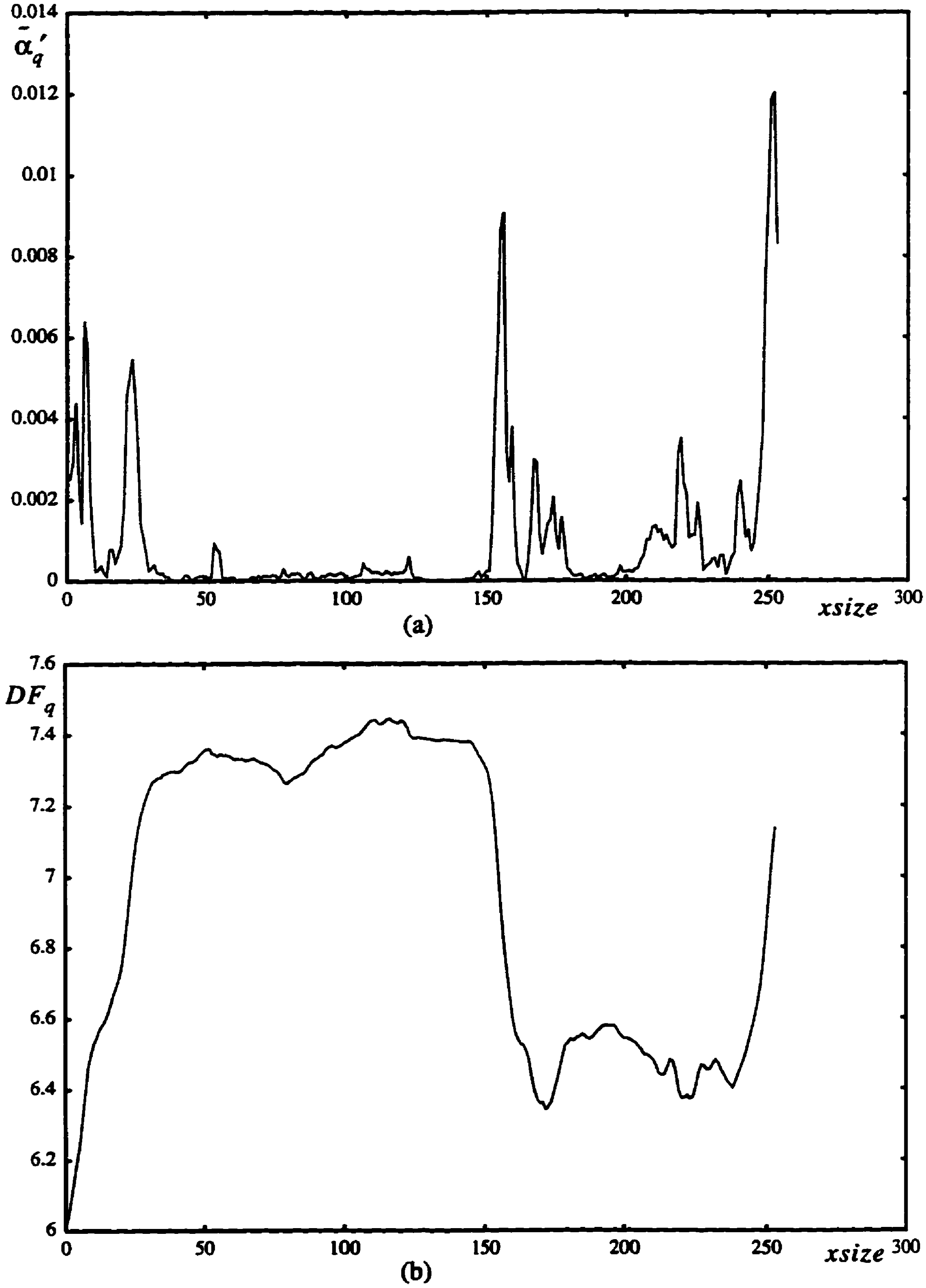


Fig. 5.26 The multifractal feature vector at row 76. (a) The rate of change of the singularity spectrum. (b) The density function DF_q.

The relationship between the two multifractal feature matrices can also be partially seen from the relationship of the two multifractal feature vectors at row 76. With different density region, the range of the rate of change of the singularity is different.

The boundary can be detected from just the first derivative of the singularity. A threshold for $\bar{\alpha}_q'$ is set at 0.0005, the boundary image can be obtained which is shown in Fig. 5.27.

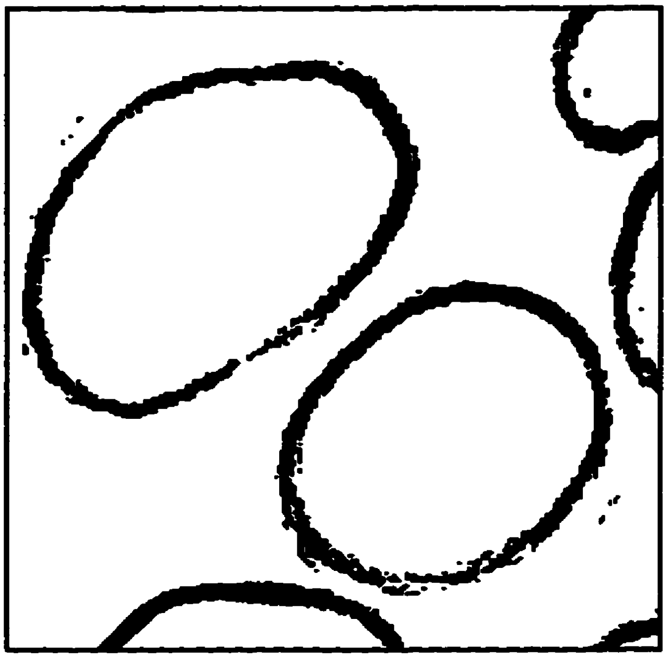


Fig. 5.27 The detected boundary from the benign cells image.

Classical methods can also be used to detect the edges of the benign cancer cells. The Sobel operator is used, then threshold is set. The result is shown in Fig. 5.28a. Thresholding the Prewitt operator and the Robert operator are also applied to the benign cancer cells image, and the results are shown in Fig. 5.28b and Fig. 5.28c, respectively. The result using the Laplacian operator is shown in Fig. 5.28d.

The results from thresholding the Sobel operator and the Prewitt operator are very close to the results using multifractals. The results from thresholding the Robert operator and the Laplacian operator introduce more noise. From Fig. 5.27 and Fig. 5.28, we still can see the advantage of the results using the multifractals over the classical methods.

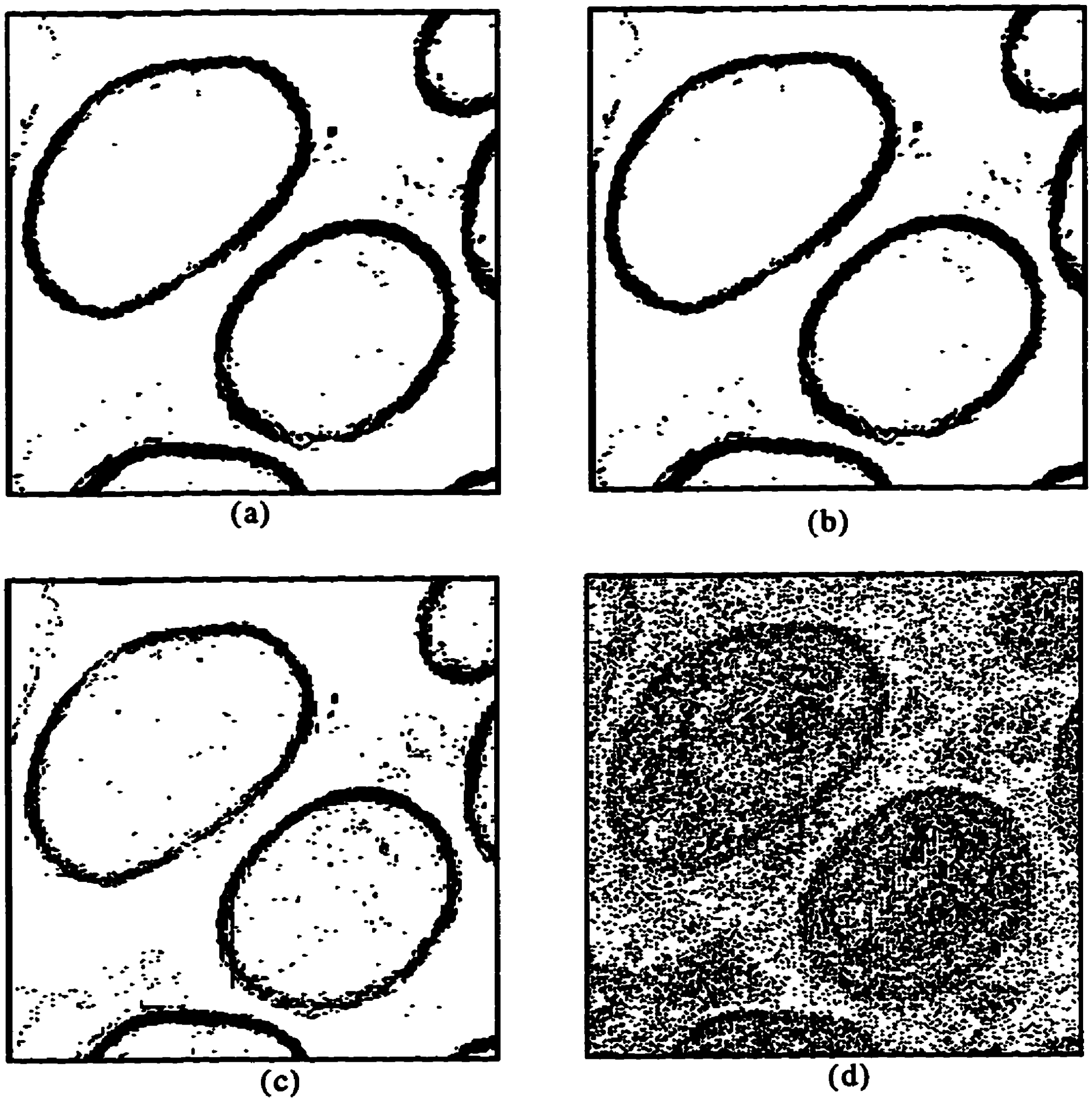


Fig. 5.28 Edge detection using classical operators. (a) Thresholding Sobel operator at T=77. (b) Thresholding Prewitt operator at T=77. (c) Thresholding Robert operator at T=19. (d) Thresholding Laplacian operator at T=140.

5.2.4.2 Boundary Extraction for a Malignant Cancer Cells Image

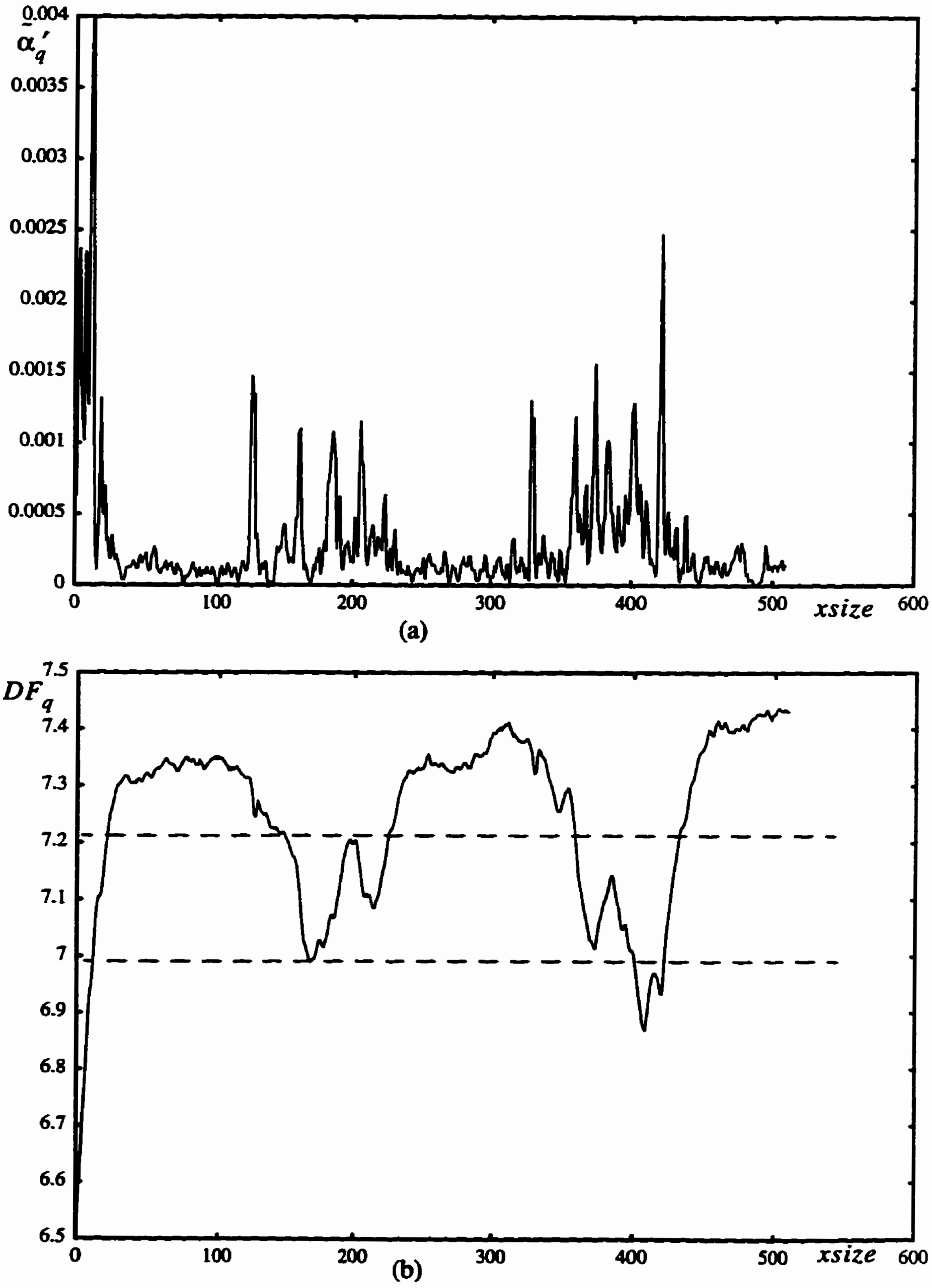
In this section, an image of malignant cells (Fig. 5.29) is used as an example. As we have observed, it is difficult to detect the boundary of this kind of image in cytology with the classical methods because of the mucosal leisons. The size of the original image is 512 by 512 pixels, with 8 bits representing each pixel. As we discussed earlier, based on the image, multifractal feature maps should be chosen and with the chosen map the multifractal feature matrices should be produced.

In this application, the density function in Eq. (5.3) and the change rate of the singularity in Eq. (5.2) at $q=1$ are still used.

Fig. 5.29 An image of malignant cancer cells.

The multifractal feature vectors at row 245 are shown in Fig. 5.30. We can see many spikes in Fig. 5.30a. The spikes also convey information of the multifractal features that the image contains. The larger spikes represent the greater differences of the singularity between the two neighbouring masks. The relationship between the two multifractal feature matrices can also be partially seen from the relationship of the two multifractal feature vectors at row 245. With different density region, the range of the rate of change of the singularity is different.

A threshold for $\tilde{\alpha}_q'$ is set at 0.00025 when the density DF_q is larger than 7.2. When the density DF_q is between 7.1 and 7.25, a threshold $\tilde{\alpha}_q'$ is set at 0.0003. The edge detection result for the aerial image is shown in Fig. 5.31.



(a)

(b)

Fig. 5.30 The multifractal feature vector at row 245. (a) The rate of change of the singularity spectrum. (b) The density function DF_q.

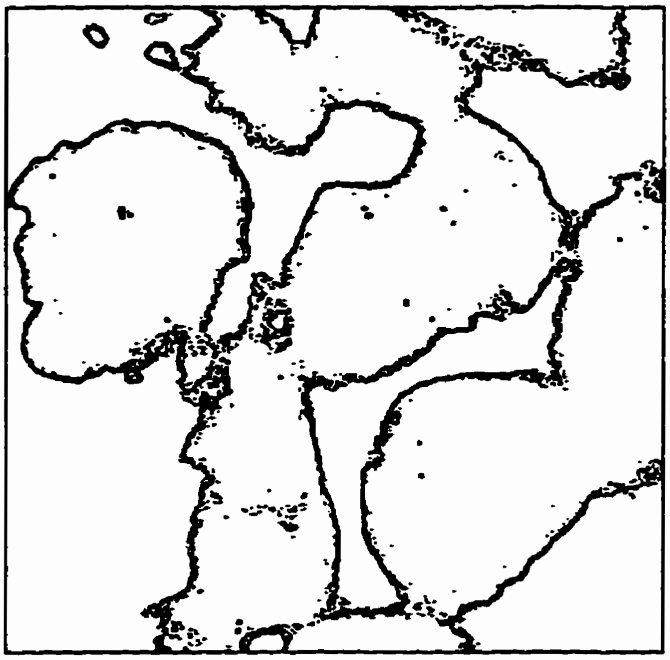
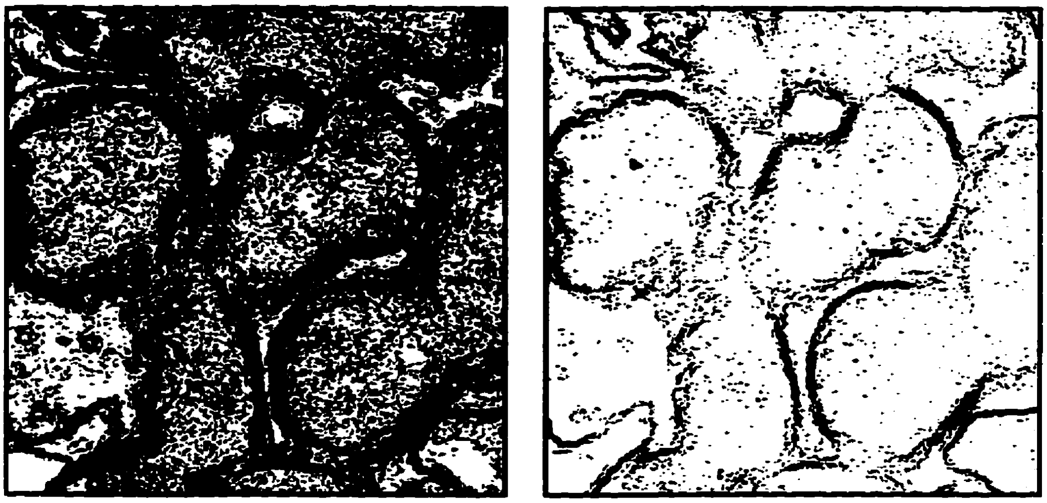


Fig. 5.31 The detected boundary from the malignant cancer cells.

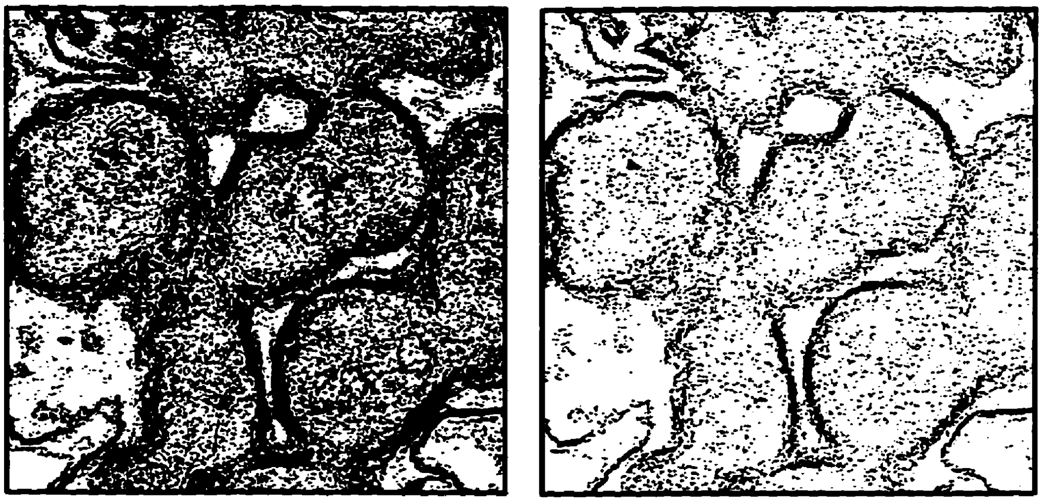
Comparing the rate of change of the singularity $\bar{\alpha}_q'$ which is shown in Figs. 5.22, 5.26 and 5.30, it can be seen that $\bar{\alpha}_q'$ for the malignant cancer cells is much smaller in most of the regions. Although the malignant cancer cells image has smaller $\bar{\alpha}_q'$, there are still striking differences among regions with nonuniform property. With the multifractal feature map, this nonuniform property is reflected in the singularities. But for the classical mask edge detectors, this nonuniform property cannot be reflected well.

Classical methods are used to detect the edges of the malignant cancer cells. For the Sobel operator, thresholds are set, and the results are shown in Fig. 5.32. Thresholding the Prewitt operator and the Robert operator are also applied to the malignant cancer cells image, with the results shown in Figs. 5.33 and 5.34, respectively.



(a) (b)

Fig. 5.32 Sobel operator is used in the malignant cancer cell image. (a) T=27. (b) T=54.



(a) (b)

Fig. 5.33 Thresholding Prewitt operator is used in the malignant cancer cell image. (a) T=7. (b) T=15.

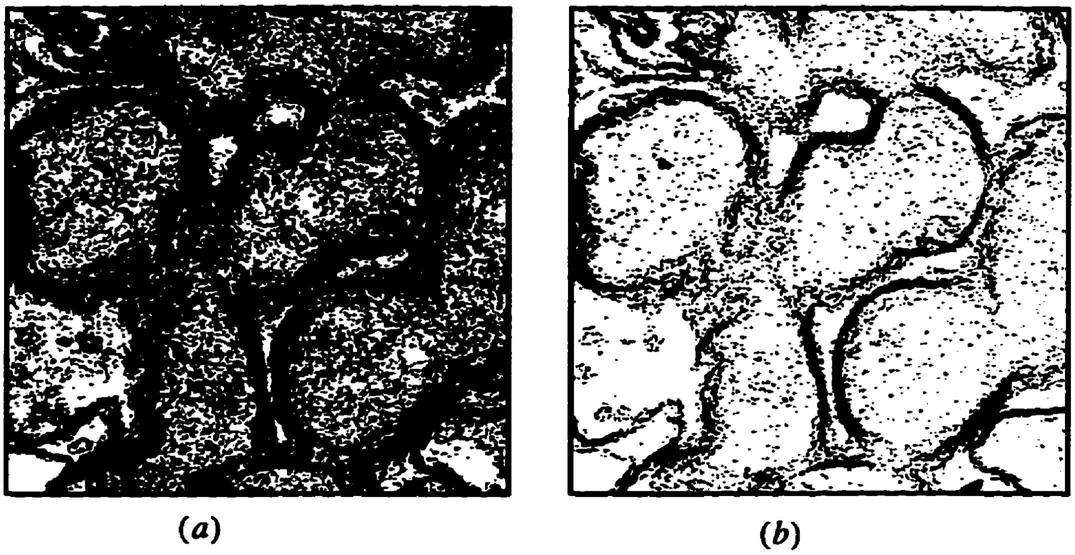


Fig. 5.34 Thresholding Robert operator is used in the malignant cancer cell image. (a) T=25. (b) T=45.

In the above results for each operator, two thresholds are set. We can see that when the threshold *T* is small, although the boundary of the cells are visible, too much noise is introduced, and the boundary is very thick. When we adjust the threshold to a higher position, the noise is reduced greatly, but the boundary in the image becomes dimmer, or even lost in some places.

The result of thresholding the Laplacian operator (shown in Fig. 5.35) is even worse than the Sobel operator, the Prewitt operator and the Robert operator.

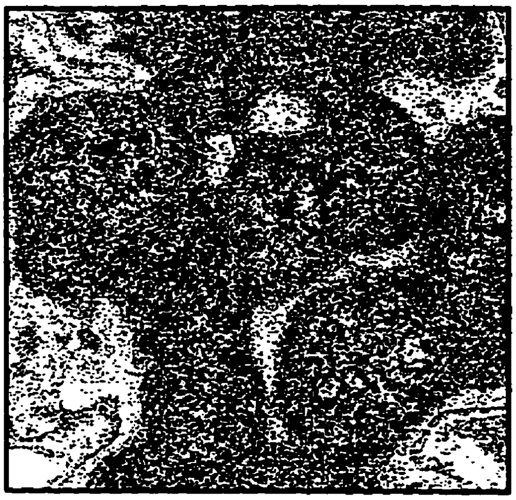


Fig. 5.35 Thresholding of Laplacian is used in the malignant cancer cells image (T=153).

From the above images, we can see that the classical methods do not produce better results than the multifractal method. The multifractal method demonstrates that it is more effective in processing this kind of difficult image.

5.2.5 Texture Extraction Using Multifractals

In this section, we use the malignant cells image (shown in Fig. 5.29) again. By suppressing the relatively high density and low density, the textures of the mucosal leisons of the cancer cells can be extracted. This can be noticed through the vector at row 245. So the layer with the density between the two dot lines at 6.98 and 7.21 is selected which is shown in Fig. 5.30. A threshold is set at 0.00025, the image of the extracted textures is shown in Fig. 5.36.

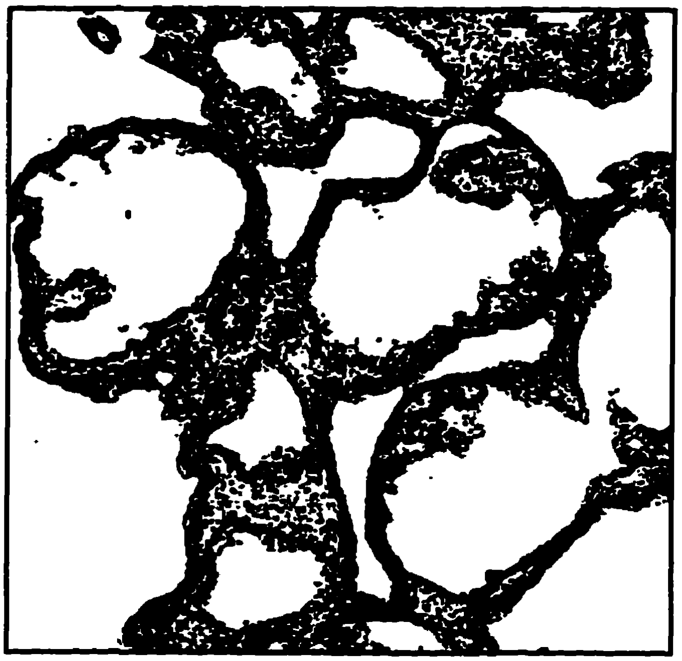


Fig. 5.36 The extracted textures from the malignant cancer cells.

In order to extract the texture in the relatively dense part, a high density layer should be selected, and then a threshold set. We select a density between 7.3 and 7.5, and the threshold is set at 0.001. The selected texture in the dense part is shown in Fig. 5.37.

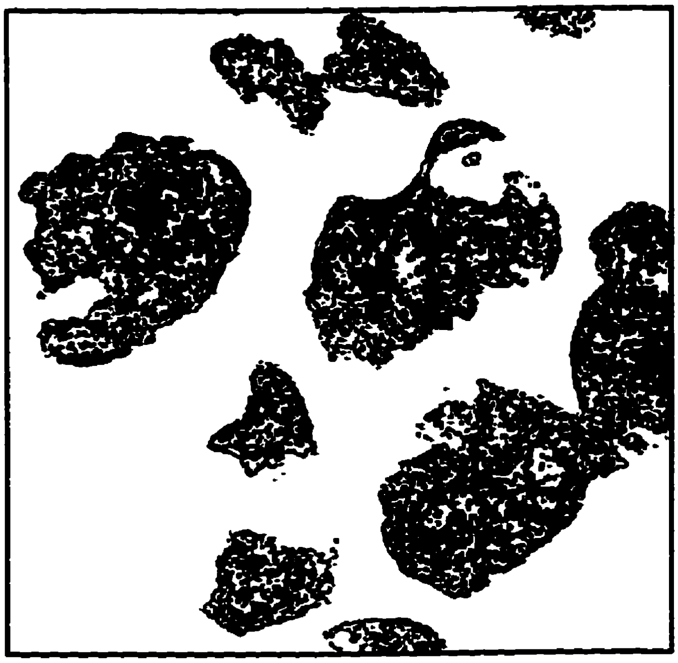


Fig. 5.37 The selected texture image.

5.3 Decomposition of the Mandelbrot Spectrum

In Chapter 4, we have found the approximation of the singularity measure as

$$\bar{\alpha}_q = \frac{\sum_{j=1}^{s^2} (255-G_j)^q \left(\log(255-G_j) - \log\left(255s^2 - \sum_{j=1}^{s^2} G_j \right) \right)}{\left(\log\frac{1}{s} \right) \sum_{j=1}^{s^2} (255-G_j)^q} \qquad (5.4)$$

where G_j ($j = 1, 2, 3, \ldots s^2$) is the grey level value of an image of $s \times s$ pixels. The range of the singularity spectrum is very important in the decomposition of the Mandelbrot spectrum. The extreme values of the approximation are given as follows

$$\lim_{q \to -\infty} \bar{\alpha}_q = \frac{\log(255-G_{max}) - \log\left(255s^2 - \sum_{j=1}^{s^2} G_j \right)}{\log\frac{1}{s}} = \bar{\alpha}_{\max} \qquad (5.5)$$

$$\lim_{q \to +\infty} \tilde{\alpha}_q = \frac{\log(255 - G_{min}) - \log\left(255s^2 - \sum_{j=1}^{s} G_j\right)}{\log\frac{1}{s}} = \tilde{\alpha}_{\min} \tag{5.6}$$

So the range of the singularity spectrum is

$$\tilde{\alpha}_{\max} - \tilde{\alpha}_{\min} = \frac{\log(255 - G_{max}) - \log(255 - G_{min})}{\log\frac{1}{s}} \tag{5.7}$$

where $G_{\max}$ and $G_{\min}$ denote the maximum and minimum of the grey level from $\{G_j\}$ j=1, 2, ..., s^2. This range is only related to the maximum value and minimum value of the grey levels.

The Mandelbrot dimension, according to the multifractal formalism, reflects the intensity of each singularity spectral component

$$\tilde{f}_q = \frac{\sum_{j=1}^{s^2} (255 - G_j)^q \log(255 - G_j) - \sum_{j=1}^{s^2} (255 - G_j)^q \log \sum_{j=1}^{s^2} (255 - G_j)^q}{\sum_{j=1}^{N\varepsilon} (255 - G_j)^q \log\frac{1}{s}} \tag{5.8}$$

The values at both ends are also rewritten as follows:

$$\lim_{q \to +\infty} \tilde{f}_q = \frac{\log \kappa_{G_{min}}}{\log\frac{1}{s}} \tag{5.9}$$

$$\lim_{q \to -\infty} \tilde{f}_q = \frac{\log \kappa_{G_{max}}}{\log\frac{1}{s}} \tag{5.10}$$

where $\kappa_{G_{\min}}$, $\kappa_{G_{max}}$ denote the numbers of pixels which have grey level values $G_{\min}$, G_{max} in the images, respectively.

In this section, we use an image shown in Fig. 5.38. The image is 256 by 256 pixels, each pixel is represented using 8 bits. In the dark parts of this image, there are many hidden textures. This can be unveiled with the help of the singularity in the image. So, the multifractal technique will be utilized to extract the texture. The Mandelbrot spectrum of this image is given in Fig. 5.39. The range of the singularity spectrum is from 1.873931 to 2.827305.

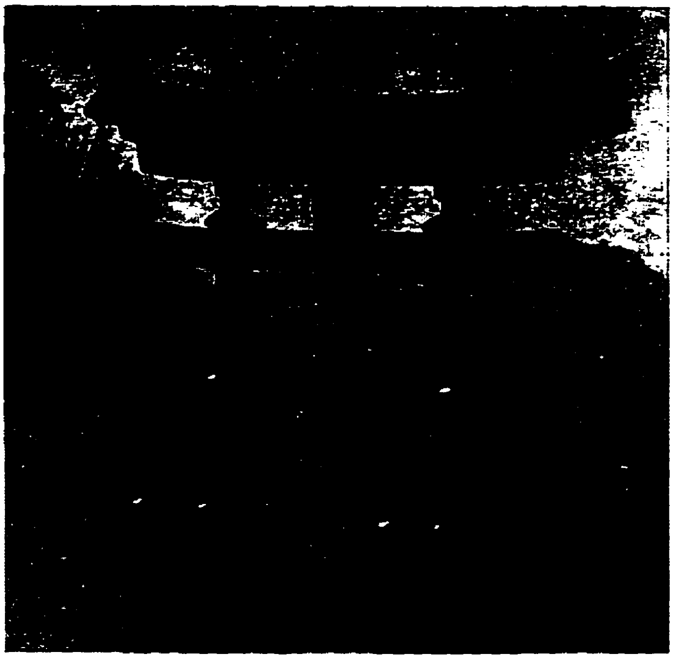


Fig. 5.38 An image of 256 by 256 pixels with 8 bits representing each pixel.

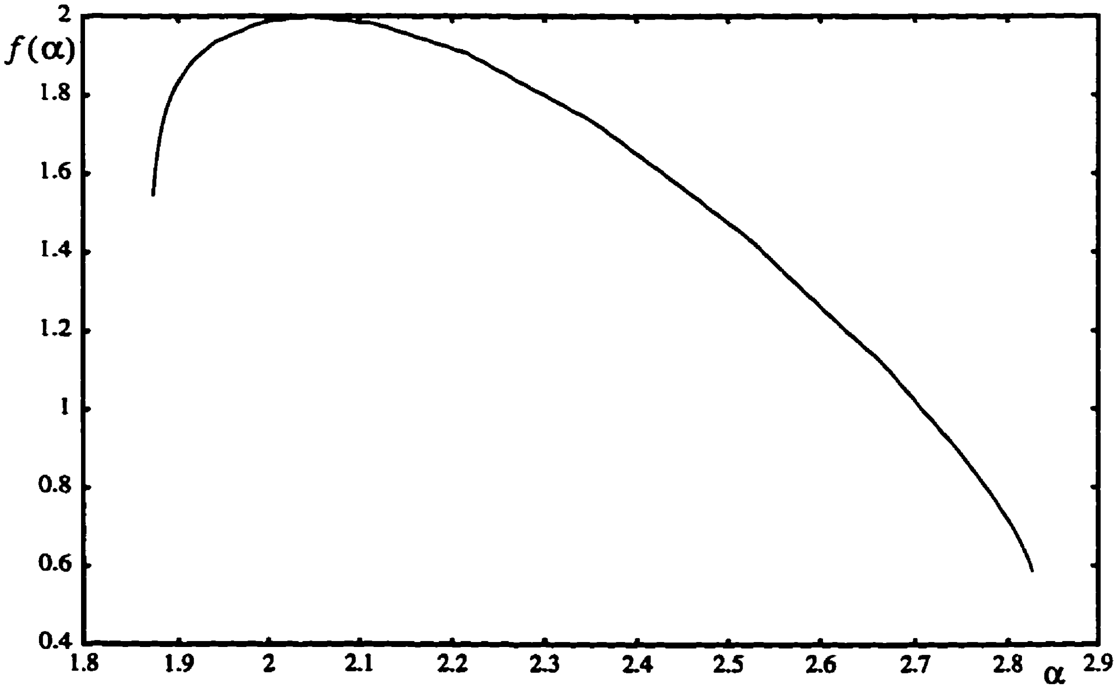
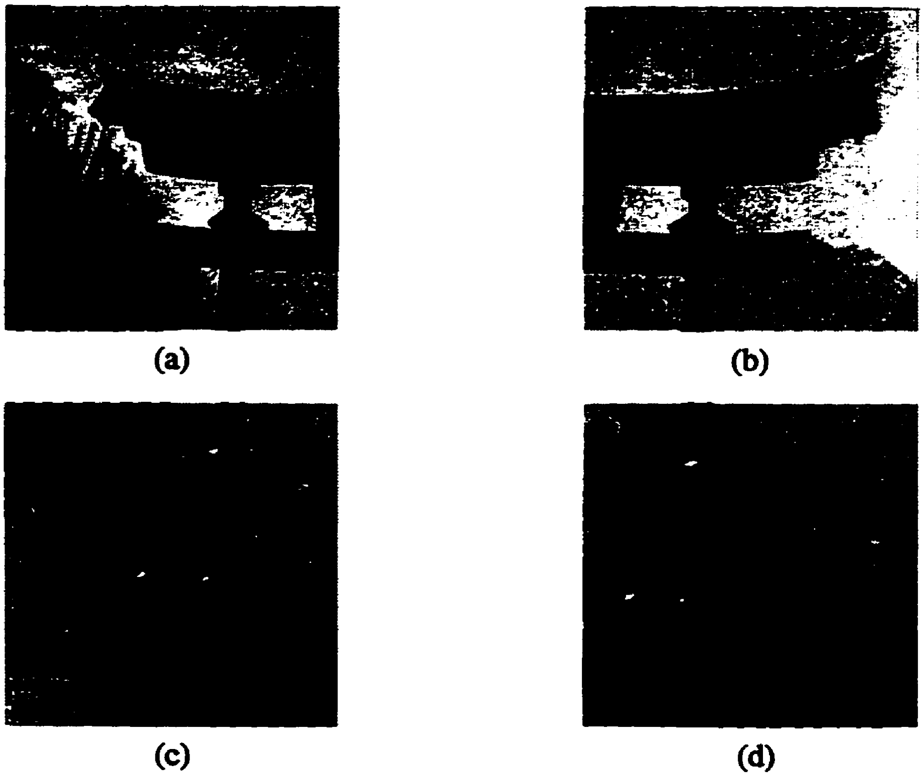


Fig. 5.39 The Mandelbrot spectrum of the image shown in Fig. 5.38.

5.3.1 The Mandelbrot Spectra of Sub-Images

Before we decompose the Mandelbrot spectrum of the image shown in Fig. 5.38, it is necessary to make some tests first to get some understanding of the relationship of the multifractal measures between an image and a portion thereof. We cut the image into four non-overlapped sub-images which are shown in Fig. 5.40. The Mandelbrot spectra of the four sub-images are then calculated. The comparison diagram of the spectra among the original image and the four sub-images is shown in Fig. 5.41.

From Fig. 5.41, the ranges of singularity of the four sub-images are all within the range of the singularity of the original image. We also can see that the single hump of the Mandelbrot spectrum of the original image is higher than of Mandelbrot spectra of the four sub-images; *i.e.*, for each singularity point, the corresponding Mandelbrot dimensions of the four sub-images are all smaller than the Mandelbrot dimension of the original image at the same singularity point.


(a)

(b)

(c)

(d)

Fig. 5.40 The four images cut from the original image. (a) Upper left part. (b) Upper right part. (c) Lower left part. (d) Lower right part.

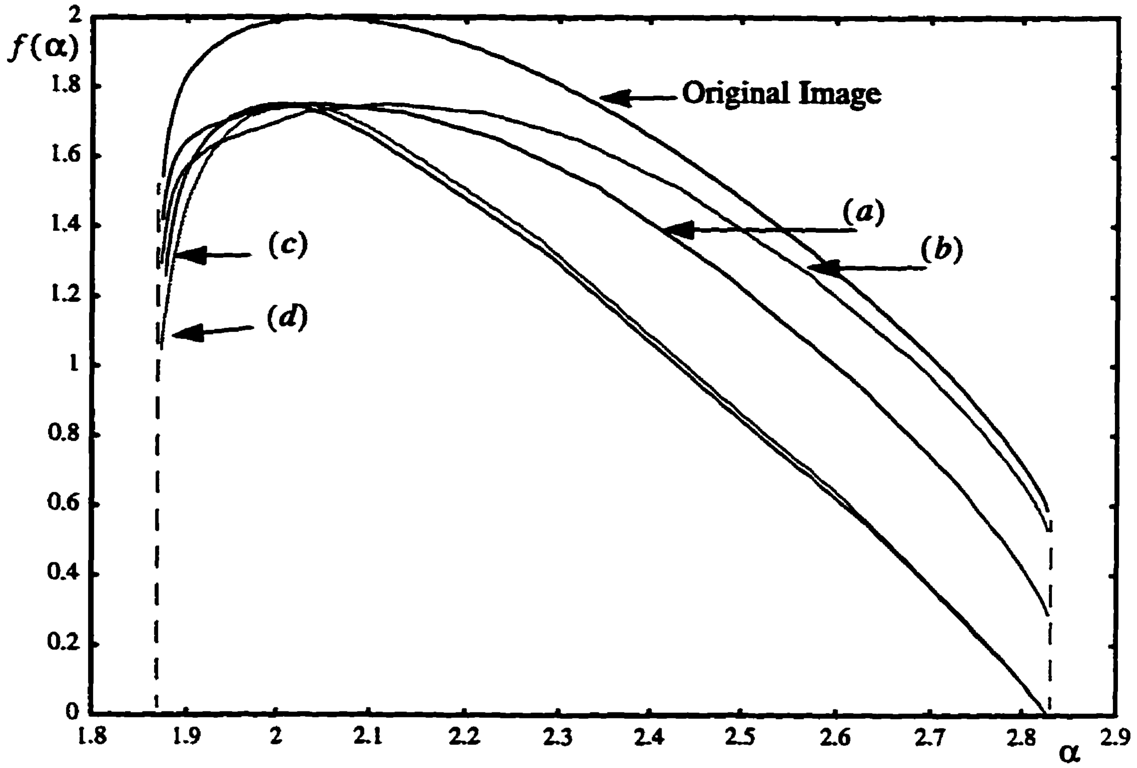


Fig. 5.41 The Mandelbrot spectra of the four images (Fig. 5.40a to 5.40d) and the original image.

This is so because no multifractal information is added to them; what we have done is to simply split it into four sub-images. To be more exact, the maximum grey level $G'_{\max}$ in each sub-image is not larger that the maximum number $G_{\max}$ in the original image, and the minimum number $G'_{\min}$ in each sub-image cannot be smaller than the minimum number $G_{\min}$ in the original image. According to Eq. (5.5) and Eq. (5.6), the ranges of the four sub-images should not exceed the range of the original image.

From Eqs. (5.8), (5.9), and (5.10), we can also conclude that the f_q of each sub-image is smaller than f_q of the original image. It is quite straightforward especially at both ends of the singularity spectrum.

Since the four sub-images are not selected according to singularity, we can see that each sub-image consists of a very broad range of the singularity that the original image has. This is reflected by the histogram of the four sub-images shown in Fig. 5.42. We can see the range of the distribution of the grey level for each sub-image is also quite close.

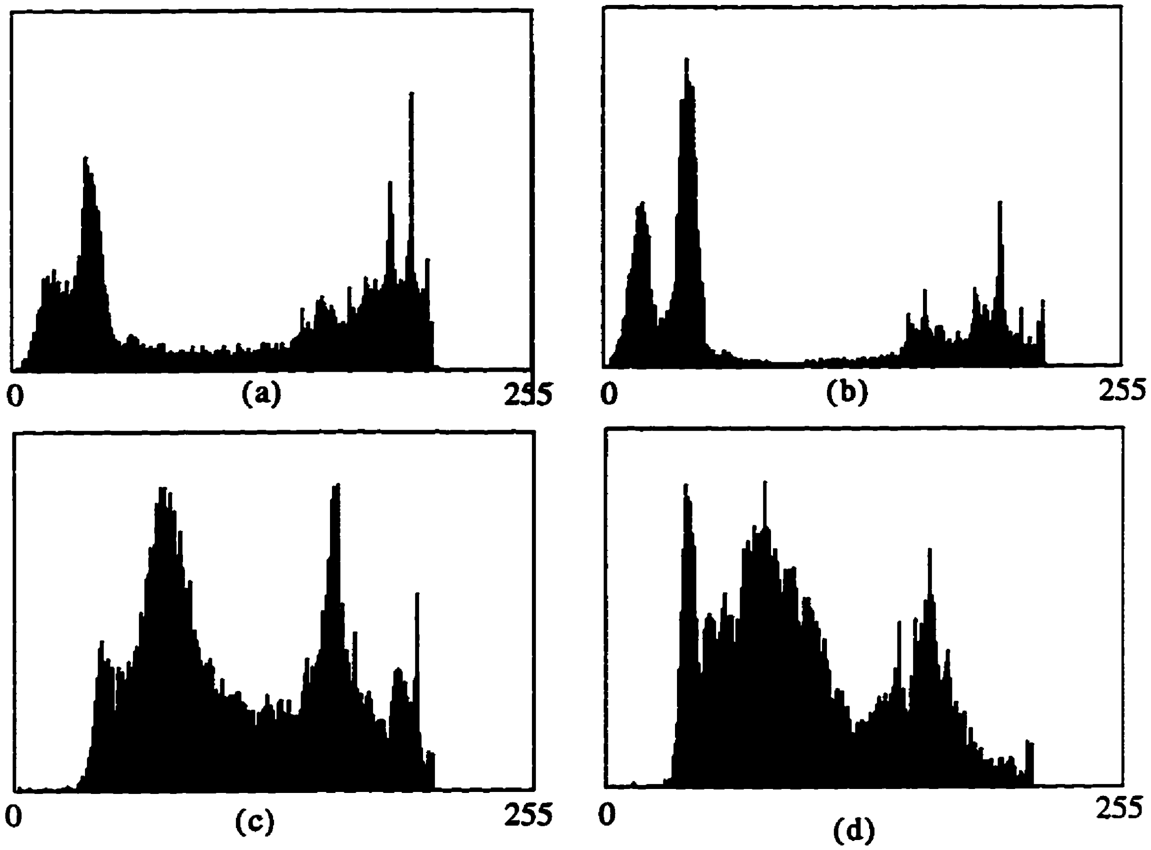


Fig. 5.42 The histograms of the four sub-images in Fig. 5.40. (a) Upper left part. (b) Upper right part. (c) Lower left part. (d) Lower right part.

5.3.2 The Decomposition of the Mandelbrot Spectrum

5.3.2.1 Set-valued Multifractal Feature Map

In the previous section, the rate of change of the singularity and density of images are used as multifractal feature maps. The map is a single-valued map (shown in Fig. 5.18). In this section, another multifractal feature map will be used. This multifractal feature map is very different from the former maps: it is not single-valued map but a set-valued map. This map is an extension of the single-valued multifractal feature map. With a sub-image or image mask, the Mandelbrot spectrum can be obtained. We know that the Mandelbrot spectrum includes not a single value, but an infinite number of values. The map will match any mask image to its corresponding Mandelbrot spectrum. Other kinds of maps can also be chosen for different application. Comparing with the single-valued feature map, the difference is that at the bottom of Fig. 5.18, the feature set contains the set $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ as its *value*, rather than the real value *a*, *b*, and *c* as its value.

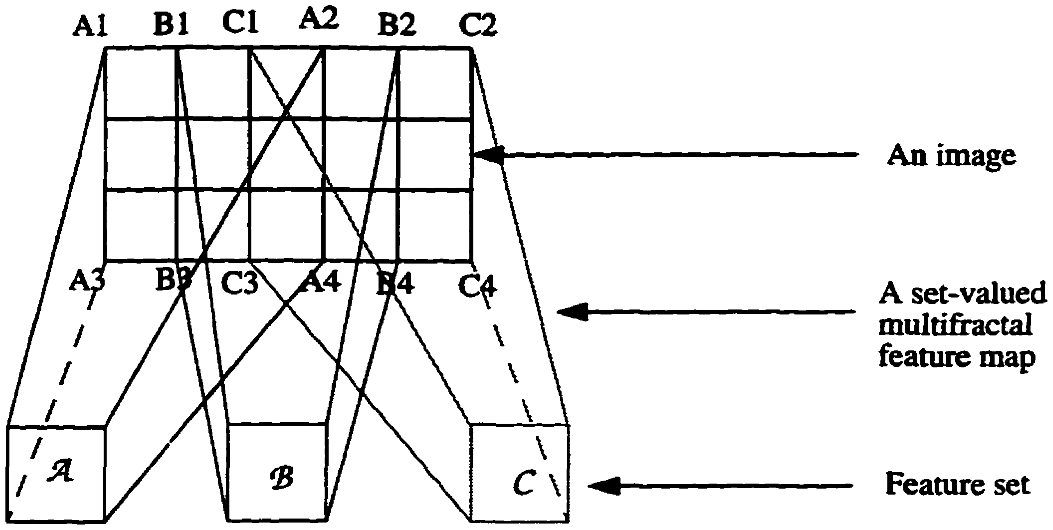


Fig. 5.43 The mapping from an image to a set-valued multifractal feature set

5.3.2.2 Decomposition of the Mandelbrot Spectrum through Singularity

In this section, the singularity spectrum is chosen as the element of the feature set; *i.e.*, the set-valued multifractal feature map will map the elements (masks) of the image into a feature set which consists of the singularity spectrum as its elements. After our mask goes over all the position in an image of size *n* by *n* pixels, there will be $(n\text{-}2)^2$ set values and their corresponding the Mandelbrot dimension returned. These $(n\text{-}2)^2$ Mandelbrot spectra returned are the decomposition elements of the Mandelbrot spectrum of the original image. The decomposed singularity spectra may overlap. Since the mask size can be any number not greater than the image size, the set-valued multifractal feature map may map the image into a different feature set. So the decomposition of the Mandelbrot spectrum of an image is not unique.

As an example, the following shows an incomplete non-overlapped decomposition of the Mandelbrot spectrum of the image shown in Fig. 5.38. In this example, a mask of size 3 by 3 pixels is chosen. The image is 256 by 256 pixels. With the set-valued multifractal feature map, 254^2= 64516 Mandelbrot spectra will be returned, all these contribute to the Mandelbrot spectrum of the image which has been shown in Fig. 5.39. It is difficult to show all the 64516 decomposition elements.

The singularity of the image is from 1.873931 to 2.827305. This range is segmented into two non-overlapping intervals $\mathcal{A}_1$=(1.873931, 2.350618) and $\mathcal{A}_2$=(2.350618, 2.827305). $\mathcal{A}_1$ can also be incompletely divided into two non-overlapping sets: $\mathcal{B}_1$ and $\mathcal{B}_2$. $\mathcal{B}_1$ and $\mathcal{B}_2$ can be further divided into $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{F}_1$, $\mathcal{F}_2$, respectively. $\mathcal{C}_1$ can be divided into $\mathcal{D}_1$ and $\mathcal{D}_2$, $\mathcal{D}_1$ can be divided into $\mathcal{E}_1$ and $\mathcal{E}_2$. The incomplete decomposition is shown in Fig. 5.44.

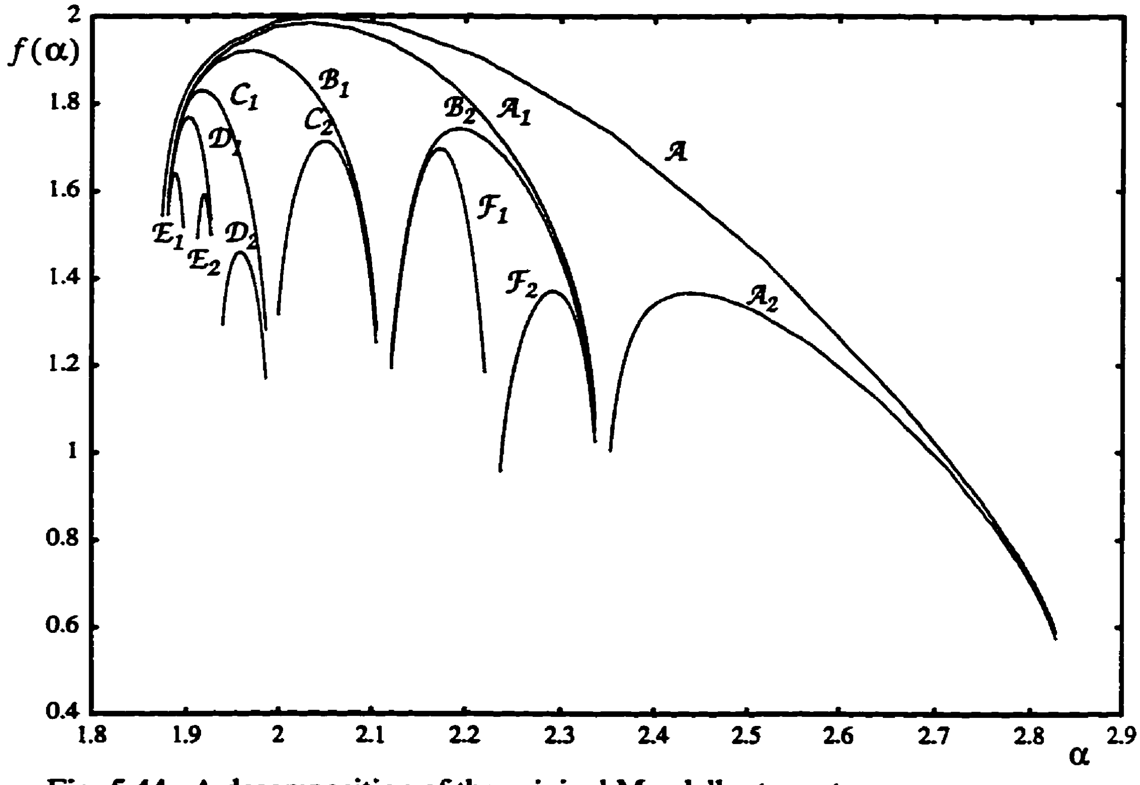


Fig. 5.44 A decomposition of the original Mandelbrot spectrum.

Another example of the non-overlapping incomplete decomposition of the Mandelbrot spectrum is shown in Fig. 5.45. The non-overlapped intervals of the singularity spectrum are (1.873931 1.903724), (1.903724 1.933517), (1.933517 1.963310), (1.963310 1.993103), (1.993103 2.022896), (2.022896 2.052689), (2.052689 2.082482), (2.082482 2.112274), (2.112274 2.171860), (2.171860 2.231446), (2.231446 2.291032), (2.291032 2.350618), (2.350618 2.469790), (2.469790 2.588961), (2.588961 2.827305). With these non-overlapping singularity sets, the corresponding Mandelbrot spectra can be found.

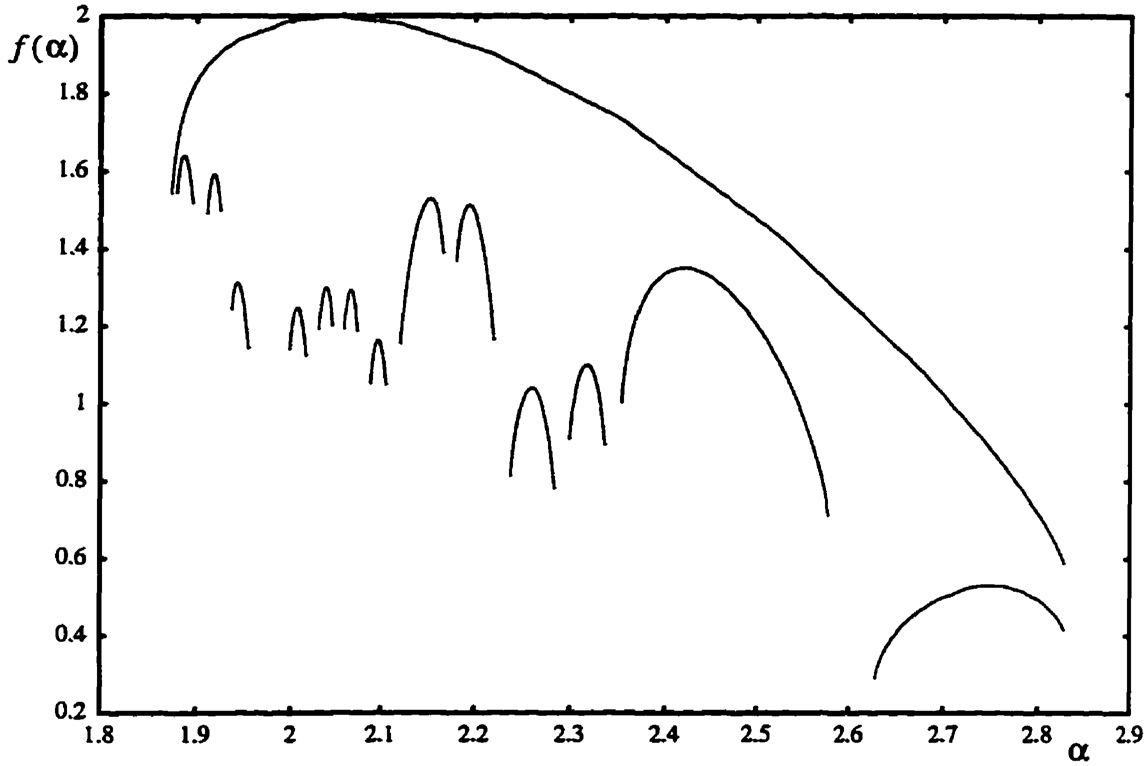


Fig. 5.45 A non-intersected decomposition of the original Mandelbrot spectrum.

5.3.2.3 An Application of the Decomposition of the Mandelbrot Spectrum

As an application of the decomposition of the Mandelbrot spectrum, we again use the image shown in Fig. 5.38 as an example. In the relatively dark parts of the image, the textures cannot be perceived by the human eyes. The dark parts indicate high-density regions of points, if the image is modelled as a strange attractor. In the multifractal model, the area with denser points correspond to the smaller singularity.

In order to extract the textures in the dark parts, the singularity range we choose is from 0.005 to 0.01. From the multifractal feature set, all the elements whose singularity spectrum is within this range are selected. The corresponding textures are shown in Fig. 5.46.

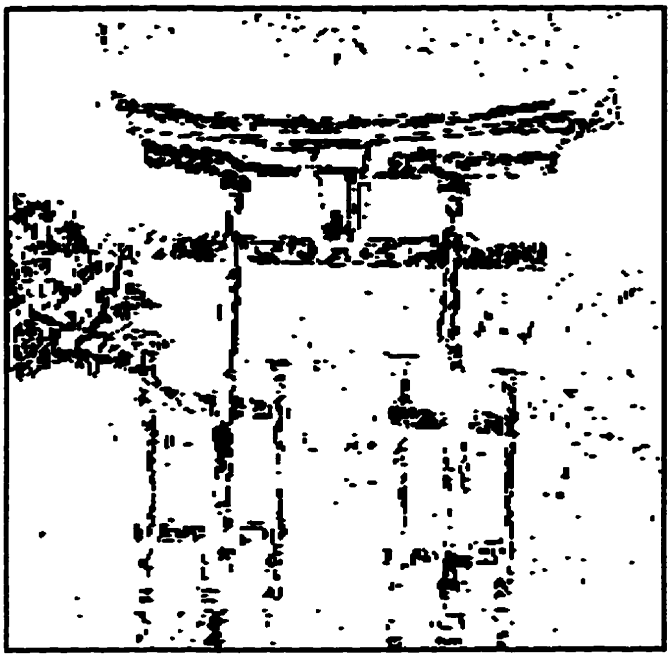


Fig. 5.46 The textures extracted using the decomposition elements of the original image.

5.4 Overview of Chapter 5

This chapter presents all the experimental results and discussion. In this chapter, the experiments can be classified into two types, one is the approximation of the multifractal measures for strange attractors, the other is the application of the multifractal measures for image analysis. In Sec. 5.1, we have shown the number of points of a strange attractor and vel size are the two factors that affect the approximation of the multifractal measures. Based on this, a more accurate estimation of multifractal measures is obtained. In Secs 5.2 and 5.3, multifractal measures applied to grey level images lead to edge detection, boundary extraction and texture extraction results that are better than those obtained with traditional techniques. This demonstrates the potential of the multifractal measures as a new image processing technique.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

This thesis has presented a study of the approximation of multifractal measures on strange attractors in chaotic dynamical systems and on non-stationary spatial signals (*e.g.*, images). The work described in this thesis was motivated by the need for new and better image segmentation techniques.

Multifractals are used to characterize very complicated objects. But in practice, getting an accurate approximation of the multifractal measures is not easy. The multifractal measures are defined through the Rényi generalized entropy and the Rényi dimension. Since the complete strange attractor consists of an infinite number of points, we cannot obtain the theoretical value of the probability; instead, we consider a finite number of points in the volume elements. Therefore, this study reduces to a finite number of points and finite size of vels. We have shown that, for a given vel size, the Rényi dimension and the Mandelbrot dimension are sensitive to the number of points used in the attractor, and that for a given number of points in the strange attractor, they are also sensitive to the vel size. We also find that for a given vel size, there is a minimum bound on the number of points required. The smaller the vel size, the larger the minimum bound. Furthermore, when the number of points which is above the minimum bound increases, we can see the convergence property of the Rényi dimension and the Mandelbrot spectrum. For example (see Fig. 5.9), when N increases from 500 to 5,000, the Rényi dimension of the Hénon attractor at $q = -30$ increases by 0.330732, but this difference diminishes monotonically to 0.014259 for N from 90,000 to 100,000. When N increases from 1,000 to 10,000, the Rényi dimension of the Poincaré section for the Rössler attractor at $q = -30$ increases by 0.322862, but this difference diminishes monotonically to 0.014477 for N from 90,000 to 100,000 (see Fig. 5.11). When N increases from 5,000 to 50,000, the Rényi dimension of the Ikeda attractor at $q = -30$ increases by 0.331553, but this difference diminishes monotonically to 0.014568 for N from 900,000 to 1,000,000 (see Fig. 5.13). The convergence can be a guideline to determine the number of points required to compute the dimension.

Since multifractals can be used to describe very complicated objects, we use them for characterization of images which consist of different grey level pixels distributed in the image. If images are modelled as strange attractors, the concept of the multifractal measures (such as singularity and Mandelbrot spectrum) can be consider as a new approach to image processing, and complicated non-stationary signals.

The experimental results in Chapter 5 show that the segmentation by multifractal feature map is better than the segmentation using traditional techniques. The advantage of this new approach can be seen fully when it is applied to some complicated images such as the malignant cancer cells image shown in Fig. 5.31. The traditional segmentation techniques either produce much more noise or fuzzy boundaries or both.

6.2 Contributions

We believe that this thesis and the work described have provided the following contributions:

a. A study of two factors: the number of the points from a strange attractor and the size of vels that cover the strange attractor, which affect the approximation of the multifractal measures for the strange attractor (see Secs. 5.1.1 and 5.1.2).

b. A study of the compromise of the two factor above in the approximation of the multifractal measures for the strange attractor (Secs. 3.3.2 and 5.1.3).

c. A study of the extreme values of the multifractal measures (Sec. 3.3).

d. Extending the study of lower bounds from single-fractal dimension to multifractal measures.(Sec. 3.3.2)

e. Software implementation of the approximation of the multifractal measure for strange attractors (Appendix A and Appendix B).

f. A framework for modelling non-stationary spatial signals as strange attractors (Chapter 4).

g. A study of multifractal measures for images (Chapter 4).

h. A study of the relationship between the extreme values of the mulifractal measures for images and grey levels (Sec. 4.4).

i. Software implementation of the approximation of the multifractal measures for images (Appendix A).

j. A study of the single-valued multifractal feature map and image segmentation (Sec. 5.2).

k. Software implementation of the single-valued multifractal feature map for image segmentation (Sec. 5.2).

l. A study of the set-valued multifractal feature map and decomposition of the Mandelbrot spectrum (Sec. 5.3).

m. Software implementation of the set-valued multifractal feature map for decomposition of the Mandelbrot spectrum (Appendix B).

n. Application of the decomposition of the Mandelbrot spectrum for texture extraction (Sec. 5.3).

o. Software implementation of the decomposition of the Mandelbrot spectrum for texture extraction (Appendix B).

6.3 Recommendations

Based on the work of this thesis, we recommend the following.

a. Development of a more formal theory of the approximation of multifractal measures.

b. Development of a more formal theory of the multifractal model for non-stationary spatial signals. It is believed that this theory will facilitate many applications in image processing.

c. Development of neural-network theory for processing the multifractal feature sets (both single-valued and set-valued).

d. Development of a formal theory of the decomposition of the Mandelbrot spectrum.

e. Development of an application of the decomposition of the Mandelbrot spectrum.

REFERENCES

[AADG86]N. B. Abraham, A. M. Albano, B. Das, G. De Guzman, S. Yong, R. S. Gioggia, G. P. Puccioni, and J. R. Tredicce, "Calculating the dimension of attractors from small data sets," *Physics Letters*, vol. 114A, no. 5, pp. 217-221, Feb. 1986.

[ACEG87]D. Auerbach, P. Cvitanovic, J. P. Eckmann, and G. Gunaratne, "Exploring chaotic motion through periodic orbits," *Phys. Rev. Lett.*, vol. 58, pp. 2387-2389, 1987.

[ArFG91] F. Arduini, S. Fioravanti, and D. D. Guisto, "A multifractal-based approach to natural scene analysis," *Proc. IEEE Intern. Conf. Acoustics, Speech & Signal Processing, ICASSP'91*, IEEE Cat. No. CH2977-7/91, vol. 4, M8.4, pp. 2681-2684,1991.

[AFGI91] F. Arduini, S. Fioravanti, D. D. Guisto, and F. Inzirillo, "Multifractals and texture classification," *Proc. IEEE Intern. Conf. Acoustics, Speech & Signal Processing, ICASSP'91,* IEEE Cat. No. CH2977-7/91, vol. 4, M84, pp. 454-457,1991.

[BaBN92]A. Bath, G. Baumann, and T. F. Nonnenmacher, "Measuring Renyi dimensions by a modified box algorithm," *J. Phys. A: math. gen.* vol. 23, pp. 381-391, 1992.

[Barn88]M. Barnsley, *Fractal Everywhere*. Boston, MA: Academic, pp. 396, 1988.

{ QA614.86.B17 1988; ISBN 0-12-079062-9 }

[BaHr92] M. Barnsley and L. Hurd, *Fractal Image Compression*. Boston, MA: Academic, pp. 396, 1988.

{ QA614.86.B17 1988; ISBN 0-12-079062-9 }

[Beck90] C. Beck, "Upper and lower bounds on the Renyi dimensions and the uniformity of multifractals," *Physica D*, Vol. 41, pp. 67-78, 1990.

[BeSz92] J. Bene and P. Szepfalusy, "Bounds for the Renyi entropies and dynamical phase transitions," *Physical Review A*, Vol. 46, no. 2 pp. 801-808, 1992.

[CaLu78] T. Caeli, and B. Julez, "On perceptual analysis underlying visual texture discrimination : Part I," *Biol. Cybern,* 28, 167-176, 1978.

[ChSK93] B. B. Chaudhuri, N. Sarkar, and Kundu, "Improved fractal geometry based texture segmentation technique," *IEE Proceedings-E,* Vol. 140, no. 5, pp. 233-241, 1993.

[EcPr86] B. Eckhardt, "Irregular scattering," *Physica D,* vol. 33, pp. 89-98, 1988.

[EcPr86] J.-P. Eckmann and I. Procaccia, "Fluctuations of dynamical scaling indices in nonlinear systems," *Phys. Rev. A,* vol. 34, no. 1, pp. 185-187, 1992.

[EcRu92] J.-P. Eckmann and D. Ruelle, "Ergodic theory of chaos and strange attractors," *Review Modern Physics,* vol. 57, no. 3, pp 617-656, 1985.

[EcRu92] J.-P. Eckmann and D. Ruelle, "Fundamental limitations for estimating dimensions and Lyapunov exponents in dynamical systems," *Phys. D,* vol. 56, pp. 185-187, 1992.

[Feit78] S. D. Feit, "Characteristic exponents and strange attractors," *Commun. Math. Phys.* vol. 61, pp. 249-260, 1978.

[FiGi95] S. Fioravanti and D. Giusto, "Texture representation through multifractal analysis of optical mass distributions," pp. 2463-2467, 1995.

[Davi80] L. Davis, "Image texture analysis techniques-A survey, in Digital Image Processing," *Proceedings NATO Advanced Study Inst.,* Bonas, France, June 23-July 4, 1980, 189-201.

[Deva92] Robert L. Devaney, *A First Course in Chaotic Dynamical Systems: Theory and Experiment.* Reading, MA: Addison-Wesley, 1992, 302pp.

[GOTV93] J.-M. Ghez, E. Orlandini, M.-C. Tesi, and S. Vaienti, "Dynamical integral transform of entropy," *Physica D.* vol. 63, pp. 282-298, 1993.

[GOWo92] R. C. Gonzalez, R. E. Wood, *Digital Image Processing,* Addison-Wesley Publishing pp. 663-667, 1992.

[Gras83] P. Grassberger, "Generalized dimensions of strange attractors," *Phys. Lett.,* vol.

97A, no. 6, pp. 227-230, 1983.

[GrOY88]C. Grebogi, E. Ott, and J. A. Yorke, "Unstable periodic orbits and the dimensions of multifractal chaotic attractors," *Science,* vol. 238, pp. 632-638, October 1988.

[GrOY87]C. Grebogi, E. Ott, and J. A. Yorke, "Chaos, strange attractors, and fractal basin boundaries in nonlinear dynamics," *Phys. Rev. A,* vol. 37, no. 5, pp. 1711-1724, 1988.

[GrPr83] P. Grassberger and I. Procaccia, "Measuring the strangeness of attractors," *Physica*, vol. 9D, pp. 189-208, 1983.

[GrPr83] P. Grassberger and I. Procaccia, "Characterization of strange attractor," *Phys. Rev. Lett.*, vol. 50, no. 5, pp. 189-208, 1983.

[GrPr83] P. Grassberger and I. Procaccia, "Dimensions and entropies of strange attractor from a fluctuating dynamical approach," *Physica*, vol. 13D, pp. 34-54, 1984.

[HaJK86] T. C. Halsey, M. H. Jensen, L. P. Kadanoff, I. Procaccia and B. I. Shraimanand, "Fractal measure and their singularities: The characterization of strange sets" *Physical Review A*, vol. 33, no. 2, pp. 1141-1150, 1986.

[HaJo85] S. M. Hammel and C. K. R. T. Jones, "Global dynamical behavior of the optical field in a ring cavity," *J. Opt. Soc. Am. B*, vol. 2, no. 4, pp. 552-564, 1985.

[Hara79] R. M. Haralick, "Statistical and structural approaches to texture," *Proc. IEEE* 67, 1979, 786-804.

[HePr83] H. G. E. Hentschel and I. Procaccia, "The infinite number of generalized dimensions of fractals and strange attractors," *Physica*, vol. 8D, pp. 435-444, 1983

[Hilb91] D. Hilbert, "Über die stetige Abbildung einer Linie auf ein Flächenstück," *Mathematische Annalen 38*, pp. 459-460, 1891.

[JuBu75] B. Julesz, "Experiments in the visual perception of texture," *Sci. Amer. 232*, pp. 34- 43, 1975.

[Kauf91] Z. Kaufmann, "Characteristic quantities of multifractals to the Feigenbaum attractor," *Physica D*, Vol. 54, pp. 75-84, 1991.

[Kins94a] W. Kinsner, "A unified approach to fractal and multifractal dimensions," *Technical Report*, DEL94-4. Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, Manitoba, Canada, May 1994, pp. 140.

[Kins94b] W. Kinsner, Fractal and Chaos Engineering, 24.721 Course Notes, University of Manitoba, 1994.

[Kins96] W. Kinsner, "A unified approach to morphological, entropy, spectral, and variance fractal dimensions," *Mathematical and Computer Modelling J.*, 1996 (in print).

[KeCC87] J. M. Keller, R. M. Crownover, and R. Y. Chen, "Characteristics of natural scenes related to the fractal dimension," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, no. 5, pp. 621-627, 1987.

[KeCh89] J. M. Keller, and S. Chen, "Texture description and segmentation through fractal geometry," *Computer Vision, Graphics and Image Processing*, Vol. 45, pp. 150-166, 1989.

[LaRi92] G. Landini and J. W. Rippin, "Fractal dimension of the epithelial-connective tissue interfaces in premalignant and malignant epithelial lesions of the floor of the mouth," *Analytical and Quantitative Cytology and Histology.*

[Mand83] B. Mandelbrot, *The Fractal Geometry of Nature*. New York, NY: W.H. Freeman, 1983, pp. 468.

{QA447.M357 1982; ISBN 0-7167-1186-9}

[MiPr84] A. Ben-Mizrachi and Itamar Procaccia, "Characterization of experimental (niosy) strange attractors," Phys. Rev. A, Vol 29, no. 2, pp. 29-31, 1984.

[NiNi87] C. Nicolis and G. Nicolis, "Fundamental limitations for estimating dimensions and Lyapunov exponents in dynamical systems" *Nature*, vol. 326, pp 609-612, 1986.

[Ott93] E. Otto, *Chaos in dynamical systems*. Cambridge, University Press. 1993.

{TA1632.P54 1992; ISBN 0-521-43799-7}

[PaRi77] S. R. Parks and W. Richards, "Visual texture discrimination using random dot patterns," *J. Opt. Soc. Amer. 67, 765-771*, 1977.

[PaSc87] K. Pawelzik and H. G. Schuster, "Generalized dimensions and entropies from a measured time series," *Phys. Rev.* vol. 35, no. 1, pp. 481-485, 1987.

[PCFS80] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a time series," *Phys. Rev. Lett.* vol. 45, pp. 712-716, 1980.

[Phuv94] S. Phuvan, "Texture characterization using polyfractal measure," *World Wide Congress on Artificial Network.* pp. 2394-2467, 1994.

[Peit84] H.-O.Peitgen, H. Jurgens, and Dietmar Saupe, "*Chaos and Fractals: New Frotiers of Science.*" New York, NY: Springer Verlag.

{TA1632.P45 1992; ISBN 3-540-97903-4}

[Pent84] A. P. Pentland, "Shading into texture," *Proceeding National Conf. on AI,* Austin, Texas, Aug. 1984, pp. 269-273.

[Pent93] A. P. Pentland, "Fractal-based description of natural scenes," *IEEE Trans. on PAMI,* Vol. 6, No. 6, pp. 661-674, 1984.

[PNHA84] S. Peleg, J. Naor, R. Hartley, and D. Avnir, "Multiple resolution texture analysis and classification." *IEEE, PAMI-6(4),* July 1984.

[Rény55] A. Rényi, "On a new axiomatic theory of probability," *Acta Mathematica Hungarica* 6, pp. 285-335, 1955.

[SaCh92] N. Sarkar and B. B. Chaudhuri, "An efficient approach to estimate fractal dimension of textural images," *Pattern Recognition,* Vol. 25, no. 9 pp. 1035-1041, 1992.

[Schr91] M. Schroeder, *Fractals, Chaos, Power Laws.* New York, NY: W.H. Freeman, 1991, pp. 429.

{QC174.17.S9S38 1990; ISBN 0-521-42632-4}

[Smit88] L. A. Smith, "Intrinsic limits on dimension calculations," *Phys. Lett.*, vol. 133A, no 6, pp. 2387-2389, 1988.

[SMLS89] T. G. Smith Jr., W. B. Marks, G. D. Lange, W.H. Sheriff Jr., and E. A. Neale, "A fractal analysis of cell images," *Journal of Neuroscience Methods*, 27, pp. 173-180.

[StLN95] M. A. Stoksik, R. G. Lane, and D. T. Nguyen, "Practical synthesis of accurate fractal images," *Graghical Model and Image Processing*, vol. 57, no. 3, pp. 206-219, 1995.

[StMe88] H. E. Stanley and P. Meakin, "Multifractal phenomena in physics and chemistry," *Nature*, vol. 233, no. 9, pp. 545-547, 1988.

[TsEl88] A. A. Tsonis and J. B. Elsner, "The weather attractor over very short timescales," *Nature*, vol. 235, no. 29, pp. 405-409, 1988.

[VeMB92] J. Levy Vehel, P. Mignot, and J. P. Berroir, "*Multifracatals, texture, and image analysis,*" In CVPR. 1992.

[Visc92] T. Vicsek, *Fractal Growth Phenomena*, Singapore: World Scientific, 1992, pp. 488.

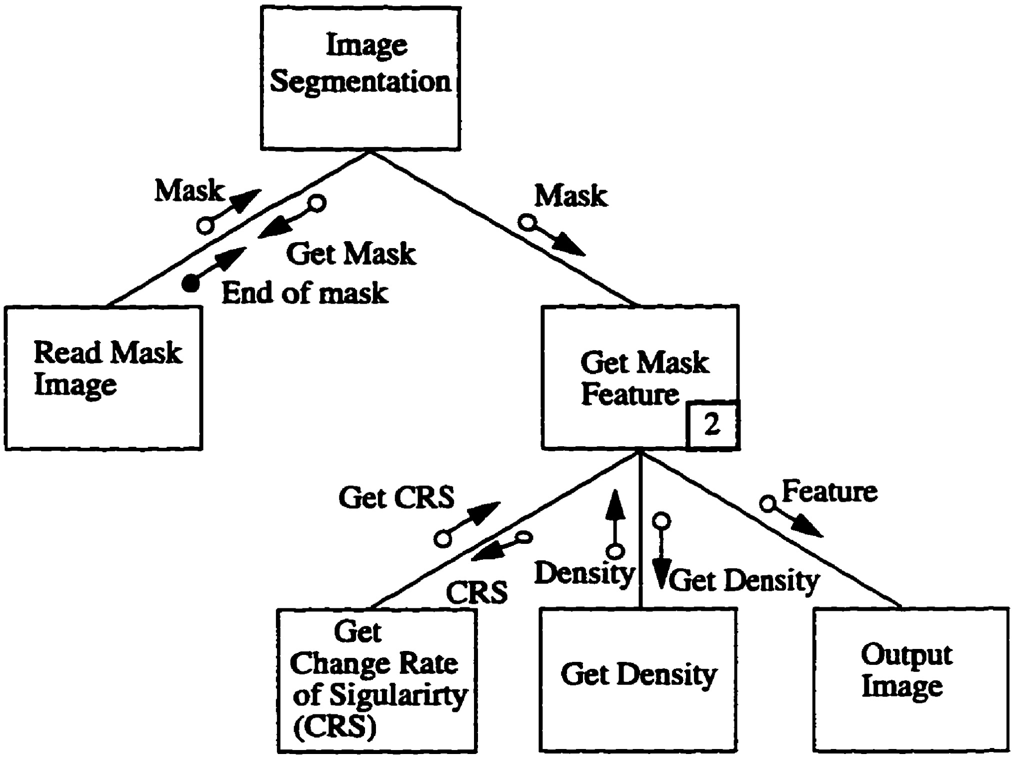
{QA614.86.V53 1992; ISBN 9-810206-69-0}

APPENDIX A

Structure of Program and Source code in C

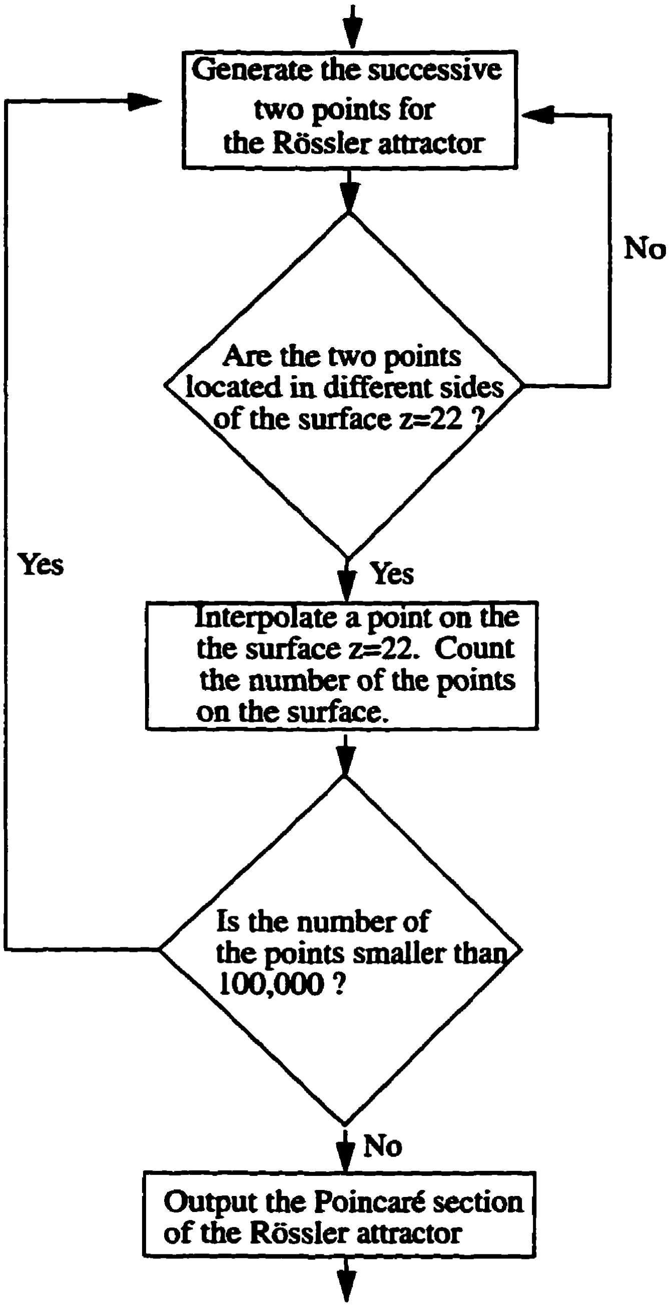
A. 1 Structured Charts

Segmentation of an image using multifractal feature map

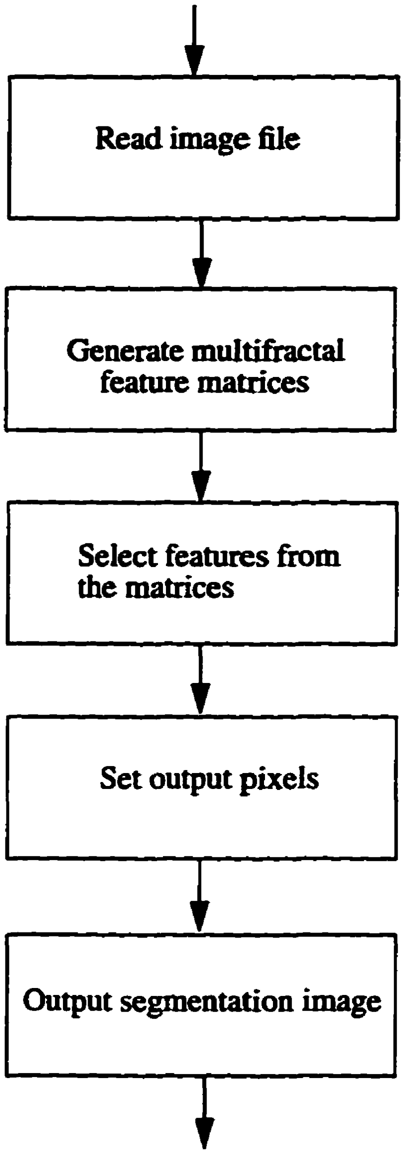


A. 2 Flow Diagrams

1. Generation of the Poincaré section of the Rössler attractor


Generate the successive two points for the Rössler attractor
Are the two points located in different sides of the surface z=22 ?
No
Yes
Interpolate a point on the the surface z=22. Count the number of the points on the surface.
Is the number of the points smaller than 100,000 ?
Yes
No
Output the Poincaré section of the Rössler attractor

2. Segmentation of an image using a multifractal feature map


Read image file
Generate multifractal feature matrices
Select features from the matrices
Set output pixels
Output segmentation image

A. 3 Source Code in C

```
/*##########################################################################################################

Program:        Generation of the Poicare section of Rosler attractor

Programmer:     Hongjing Chen
                Department of Electrical & Computer Engineering
                University of manitoba
                Winnipeg, Manitoba
                hongjin@ee.umanitoba.ca

Program name:   psr.c
Program description:This program is used to generate the the Poicare section of the
                Rosler attractor

version:        1.0

Last Update:    Mar. 12, 1995

##########################################################################################################*/



#include<stdio.h>
#include<math.h>
#include<sys/time.h>

#define a 0.2
#define b 0.2
#define c 5.7
```

```
#define max(a,b) ( (a>b) ? a:b )
#define min(a,b) ( (a<b) ? a:b )
void RK( double x, double y, double z, double *xnew,  double *ynew, double *znew);
void PS(void);
unsigned long int N=42949600;
main()
{
      time_t started,finished;
      double x,y,z,xnew,ynew,znew;
      long int i, ctr,ctrN;
      double m,px,py,pz,temp;
      double XMAX,XMIN,YMAX,YMIN,ZMAX,ZMIN;
      FILE *fp;
      FILE *fpr;
      fpr=fopen("sector","w");
      fp=fopen("r2","w");
      printf("N=%ld\n",N);

      started=time(NULL);
      x=1.161129;
      y=-1.190744;
      z=0.041733;
      XMAX=x; XMIN=x; YMAX=y; YMIN=y; ZMAX=z; ZMIN=z;
      m=22.0;
      ctr=0,ctrN=0;
      /*for(i=0;i<50000;i++)*/
      for(;;)
      {
            printf("%lf %lf %lf\n",x,y,z);
            ctrN++;
RK(x,y,z,&xnew,&ynew,&znew);
fprintf(fpr,"%lf %lf %lf\n",x,y,z);

XMAX = max(XMAX,xnew);
XMIN = min(XMIN,xnew);
YMAX = max(YMAX,ynew);
YMIN = min(YMIN,ynew);
ZMAX = max(ZMAX,znew);
ZMIN = min(ZMIN,znew);

if( (z-m) * (znew-m) < 0 )
{
        temp=(z-m)/(m-znew);
        pz=m;
        px= (x+temp*xnew) / (1+temp);
        py= (y+temp*ynew) / (1+temp);

        fprintf(fp,"%lf %lf\n",px,py);
        printf("%lf %lf\n",px,py);
        ctr++;
}
x=xnew,y=ynew,z=znew;
if(ctr==100000) break;
}
printf("XMAX=%lf XMIN=%lf\n",XMAX,XMIN);
printf("YMAX=%lf YMIN=%lf\n",YMAX,YMIN);
printf("ZMAX=%lf ZMIN=%lf\n",ZMAX,ZMIN);
printf("# poincare section points %d\n",ctr);
printf("# Rossler attractor points now %d\n",ctrN);
fclose(fpr);
```

```
        finished=time(NULL);
        printf("\n Run time = %d sec [%d min] \n",
        (int)(difftime(finished,started)),(int)(difftime(finished,started)/60));
         (int)(difftime(finished,started)),(int)(difftime(finished,started)/60));
        return;
}

void RK(double x,double y,double z,double *xnew,double *ynew,double *znew)
{
    double d0_x,d0_y,d0_z,d1_x,d1_y,d1_z,d2_x,
        d2_y,d2_z,d3_x,d3_y,d3_z;
    double xt,yt,zt;
    double dt,dt2,third=0.3333333333;
    dt=0.01;
    dt2=dt/2;
    d0_x = -( y + z ) * dt2;
    d0_y= ( x+ a * y ) * dt2;
    d0_z= ( b + x * z - c * z ) * dt2;

    xt = x + d0_x;
    yt = y + d0_y;
    zt = z + d0_z;
    d1_x = -( yt + zt ) * dt2;
    d1_y =  ( xt + a * yt ) * dt2;
    d1_z = ( b + xt * zt - c * zt ) * dt2;

    xt=x+d1_x;
    yt=y+d1_y;
    zt=z+d1_z;

    d2_x = -(yt+zt)*dt;
```

```
        d2_y=( xt+a*yt)*dt;
        d2_z=(b+xt*zt-c*zt)*dt;

        xt=x+d2_x;
        yt=y+d2_y;
        zt=z+d2_z;

        d3_x = -(yt+zt)*dt2;
        d3_y=( xt+a*yt)*dt2;
        d3_z=(b+xt*zt-c*zt)*dt2;

        *xnew=x+( d0_x + d1_x + d1_x + d2_x + d3_x)*third;
        *ynew=y+( d0_y + d1_y + d1_y + d2_y + d3_y)*third;
        *znew=z+( d0_z + d1_z + d1_z + d2_z + d3_z)*third;

}
```

```
/*#############################################################################

Program:        Generation of the Henon attractor

Programmer:     Hongjing Chen
                Department of Electrical & Computer Engineering
                University of manitoba
                Winnipeg, Manitoba
                hongjin@ee.umanitoba.ca

Program name:   hedata.c
Program description:This program is used to generate the Henon attractor

version:        1.0

Last Update:    Mar. 10, 1995

#############################################################################*/



#include<stdio.h>
#include<math.h>
#include<time.h>
#define a 1.4
#define b 0.3
#define N 100000

unsigned short int Num[4097][4097];

main()
```

```
{
        int k;
        double x[N+1],y[N+1];
        double xmax,xmin,ymax,ymin;
        FILE *fp;

        started=time(NULL);
        fp= fopen("h100000","w");

        x[0]=0.514978;
        y[0]=-0.228282;
        xmax=0;
        xmin=0;
        ymax=0;
        ymin=0;
        for(k=0;k<N;k++)
        {
                x[k+1]=y[k]+1.0-a*x[k]*x[k];
                y[k+1]=b*x[k];

                fprintf(fp,"%lf %lf\n",x[k],y[k]);

                if(x[k]>xmax)
                {
                        xmax=x[k];
                }
                if(x[k]<xmin)
                {
                        xmin=x[k];
                }
                if(y[k]>ymax)
```

```
            {
                ymax=y[k];
             }
            if(y[k]<ymin)
            {
                ymin=y[k];
            }

        }
        fclose(fp);
        printf("\n xmax=%lf xmin=%lf",xmax,xmin);
        printf("\n ymax=%lf ymin=%lf\n",ymax,ymin);
        return;
    }
```

```
/*##########################################################################

Program:        Generation of the Ikeda attractor

Programmer:     Hongjing Chen
                Department of Electrical & Computer Engineering
                University of manitoba
                Winnipeg, Manitoba
                hongjin@ee.umanitoba.ca
Program name:   opdata.c
Programm description:This program is used to generate the Ikoda attractor
version:        1.0

Last Update:    July 10, 1995

##########################################################################*/


#include<stdio.h>
#include<math.h>
#define a 5.5
#define r 0.85
#define R 0.9
#define K 0.4

#define N 1000000
double x[N+1],y[N+1];

main()
{
        int i,k,z;
```

```
    double thita;
    FILE *fp1;

    fp1= fopen("HUGE","w");
    x[0]=0.541597;
    y[0]=0.270511;
    for(k=0;k<N;k++)
    {
        thita=K-a/(1+x[k]*x[k]+y[k]*y[k]);
        x[k+1]=r+R*( x[k]*cos(thita) - y[k]*sin(thita) );
        y[k+1]=R*( x[k]*sin(thita) + y[k]*cos(thita) );
        if(x[k]>=-0.661 && x[k]<=1.558 && y[k]>=-1.411 && y[k]<=0.809)
        {
            fprintf(fp1,"%lf %lf\n",x[k],y[k]);
        }

    }
    fclose(fp1);
    return;
}
```

```
/*########################################################################

Program:        Estimation of multifratal measures for a strange attractor

Programmer:     Hongjing Chen
                Department of Electrical & Computer Engineering
                University of manitoba
                Winnipeg, Manitoba
                hongjin@ee.umanitoba.ca

Program name:   disp.c
Program description:This Program is used to estimate the multifractal
                measures for strange attractors
version:        1.4

Last Update:    Sept. 20, 1995

########################################################################*/

#include<stdio.h>
#include<math.h>
#define max(a,b) ( (a>b) ? a:b )
#define min(a,b) ( (a<b) ? a:b )

unsign long int N ;
int Num[1800][2300];

double *px,*py;
double xmax,xmin,ymax,ymin;
void RenyiDim();
```

```
main()
{
    char filename[25];
    int k;
    long int s;
    char tempDq[20];
    char tempAq[20];
    char tempFA[20];
    char str[10];
    FILE *fileptr;

    px=(double *)malloc(sizeof(double)*N);
    py=(double *)malloc(sizeof(double)*N);

    printf("\n Enter the filename:");
    fflush(stdin);
    gets(filename);

    for(k=1; k<= 10; k++)
    {
        N=10000*k;
        sprintf(str,"%d",N);
        if( (fileptr=fopen(filename,"r") )== NULL )
        {
            printf("Disk Error.");
            exit(0);
        }

        for( s=0;s<N;s++)
        {
            fscanf(fileptr,"%lf %lf", px+s, py+s);
```

```
			/*printf("px[%d]=%lf py[%d]=%lf\n",s,*(px+s),s,*(py+s));*/
		}

		xmax=*px; xmin=*px; ymax=*py; ymin=*py;
		for( s=0;s<N;s++)
		{
			xmax = max(xmax, *(px+s) );
			xmin = min(xmin, *(px+s) );
			ymax = max(ymax, *(py+s) );
			ymin = min(ymin, *(py+s) );
		}
		printf("xmax=%lf xmin=%lf\n",xmax,xmin);
		printf("ymax=%lf ymin=%lf\n",ymax,ymin);

		RenyiDim();
		return;
	}

}

void RenyiDim()
{
	double sum,nl,na,ns;
	int s;

	double grid,r,q,slope,fq,aq,dDq,ddDq,temp1;
	int d,m,z;
	long int k,j,i;
	FILE *fpdq;

	FILE *fpfq;
FILE *fpaq;
FILE *fpfa;
FILE *fpdDq;
FILE *fpddDq;
fpdq=fopen("1hthu.dq","w");
fpfq=fopen("1hthu.fq","w");
fpaq=fopen("1hthu.aq","w");
fpfa=fopen("1hthu.fa","w");
fpdDq=fopen("1hthu.dDq","w");
fpddDq=fopen("1hthu.ddDq","w");

s=10;
d=(long int)pow( 2.0,(double)s );
grid=1.0/(double)d;
printf("ok\n");
for(i=0;i<N;i++)
{
    Num[(int)( (px[i]-xmin) / grid )][(int) ((py[i]-ymin) / grid) ]++;
}

for(z=-30*10; z<=30*10; z++)
{
    q=(double)z/10;
    nl=0.0;na=0.0;ns=0.0;sum=0.0;

    for(k=0;k<=(long int)( (xmax-xmin)/grid );k++)
    {
        for(j=0;j<=(long int)( (ymax-ymin)/grid );j++)
        {

            if(Num[k][j]!=0)
```

```
                {
                                        sum=sum+pow( (double)Num[k][j],q);


                                            nl=pow((double)Num[k][j],q)*q*
                        log((double)Num[k][j])+nl;
                                        na=pow((double)Num[k][j],q)*
                        log((double)Num[k][j])+ na;
                                        ns=pow((double)Num[k][j],q)*
                        (log((double)Num[k][j]))*
                        (log((double)Num[k][j])) +ns;



                        }
                }
            }
        r=log(grid);
        if(q!=1)
        {
            slope=( log(sum)-q*log((double)N) )/((q-1.0)*r);
            fq=( nl - sum*log(sum) )/(sum*r);
            aq=(na-log(N)*sum)/(sum*r);
            dDq=( (q-1.0)*na + sum*log((double)N) - sum*
            log(sum) )/((q-1.0)*(q-1.0)*r*sum);

            temp1=(q-1.0)*(q-1.0)*sum*ns-2*(q-1.0)*
            sum*na-2*sum*sum*log((double)N)
                            -(q-1.0)*(q-1.0)*na*na+
                        2*sum*sum*log(sum);
            ddDq=temp1/( (q-1.0)*(q-1.0)*(q-1.0)*r*sum*sum );

            printf("%lf %lf\n",q,slope);
```

```
                fprintf(fpdq,"%lf %lf\n",q,slope);
                fprintf(fpfq,"%lf %lf\n",q,fq);
                fprintf(fpaq,"%lf %lf\n",q,aq);
                fprintf(fpfa,"%lf %lf\n",aq,fq);
                fprintf(fpdDq,"%lf %lf\n",q,dDq);
                fprintf(fpddDq,"%lf %lf\n",q,ddDq);
            }
            fclose(fpdq);
            fclose(fpaq);
            fclose(fpfa);
            fclose(fpdDq);
            fclose(fpddDq);
        }
}
```

```
/*##############################################################################################

Program:        Estimation of multifratal measures for images

Programmer:     Hongjing Chen
                Department of Electrical & Computer Engineering
                University of manitoba
                Winnipeg, Manitoba
                hongjin@ee.umanitoba.ca
Program package name: ImgMulFracMeasure
Program inlcluded:Makefile readpgm.c renyidimen.c
Program description:This package is used to estimate the multifractal
                measures for images
version:     1.0

Last Update: Sept. 30, 1995

###############################################################################################*/
```

Makefile

```
CC = gcc
RM = rm -f

INCLUDE_DIRS =

# Extra compiling definitions for debuging purposes

COMPILE_DEFINES =
CPPFLAGS = $(INCLUDE_DIRS) $(COMPILE_DEFINES)

CFLAGS = $(CPPFLAGS)

LD_DIRS =

LDLIBS = -lm

LDFLAGS = $(LD_DIRS)

TARGET = ohh

CODE_OBJS = readpgm.o renyidimen.o
CODE_SRCS = readpgm.c renyidimen.c
SRCS = $(CODE_SRCS)

.SUFFIXES: .o .c

.c.o:
	$(CC) $(CFLAGS) -c $<

all:	$(TARGET)

$(TARGET): $(CODE_OBJS)
	$(CC) $(CFLAGS) $(LDFLAGS) $(CODE_OBJS) $(LDLIBS) -o $@

clean:
	$(RM) $(TARGET) $(CODE_OBJS)

strip:
	strip $(TARGET)
```

readpgm.c

```
#include <stdio.h>
#include <malloc.h>
int main(argc, argv)
int argc;
char **argv;
{
        FILE *fh;
        char str[255];
        int ysize, xsize;
        int i, j;
        char ch;
        int **image;
        char filename[255];
        printf("Please input the image filename\n");
        scanf("%s", filename);

        if ( (fh = fopen(filename, "r")) == NULL )
        {
                fprintf(stderr, "Error opening file.\n");
                exit(-1);
        }

        while ( (getc(fh) != '\n') ) { }
        while ( (getc(fh) != '\n') ) { }
        fscanf(fh, "%d %d\n", &ysize, &xsize);
        printf("ysize = %d, xsize = %d\n", ysize, xsize);
        fscanf(fh, "%s\n", str);
        printf("ookk %s\n",str);
```

```
    image = (int **) calloc(ysize, sizeof(int *));
    for (i = 0; i < ysize; i++)
        image[i] = (int *) calloc(xsize, sizeof(int));

    for (j = 0; j < xsize; j++)
    {
        for (i = 0; i < ysize; i++)
        {
            image[i][j] = 255-(int)getc(fh);
        }
    }

    Count_Psudo_Points(image,xsize,ysize);

    return 0;
}
```

renyidimen.c

```
#include<math.h>
#include<stdio.h>
Count_Psudo_Points(image,xsize,ysize)
int **image;
int xsize,ysize;
{
    double N,temp;
    int i,j;
    N=0.0;
    for(i=0;i<xsize;i++)
    {
```

```
                for(j=0;j<ysize;j++)
                {
                        N=(double)image[i][j]+N;

                }
        }

        printf("N=%lf",N);

        RenyiDimen(image,xsize,ysize,N);

}
RenyiDimen(image,xsize,ysize,N)
int **image;
double N;
{
        int z,i,j;
        double q,nl,na,ns,sum,r,slope,fq,aq;
        double grid;
        FILE *fpfa;
        FILE *fpaq;
        FILE *fpdq;
        FILE *fpfq;

        char filename[255];

        char filenamedq[255];
        char filenameaq[255];
        char filenamefq[255];
        char filenamefa[255];
```

```
printf("Input Output File Name\n");
scanf("%s", filename);


sprintf(filenamedq, "%s_dq", filename);
sprintf(filenameaq, "%s_aq", filename);
sprintf(filenamefq, "%s_fq", filename);
sprintf(filenamefa, "%s_fa", filename);


fpdq=fopen(filenamedq,"w");
fpfq=fopen(filenamefq,"w");
fpfa=fopen(filenamefa,"w");
fpaq=fopen(filenameaq,"w");

printf("xsize=%d",xsize);
grid=1.0/xsize;


for(z=-30*2; z<=30*2; z++)
{
        q=(double)z/2;
        nl=0.0;na=0.0;ns=0.0;sum=0.0;

        for(i=0;i<xsize;i++)
        {
                for(j=0;j<ysize;j++)
                {
                        if(image[i][j]!=0)
                        {
                                sum=sum+pow( (double)image[i][j],q);
```

```
                    nl=pow((double)image[i][j],q)*q*
                        log((double)image[i][j])+nl;
                    na=pow((double)image[i][j],q)*
                        log((double)image[i][j])+ na;
                                    ns=pow((double)image[i][j],q)*
                    (log((double)image[i][j]))*
                    (log((double)image[i][j])) +ns;
                }
            }
        }
        r=log(grid);
        if(q!=1)
        {
            slope=( log(sum)-q*log((double)N) )/((q-1.0)*r);
            fq=( nl - sum*log(sum) )/(sum*r);
            aq=(na-log(N)*sum)/(sum*r);
            fprintf(fpdq,"%lf %lf\n",q,slope);
            fprintf(fpfq,"%lf %lf\n",q,fq);
            fprintf(fpfa,"%lf %lf\n",aq,fq);
            fprintf(fpaq,"%lf %lf\n",q,aq);
            printf("%lf %2.30f\n",q,aq);
        }
        fclose(fpdq);
        fclose(fpfq);
        fclose(fpfa);
        fclose(fpaq);
    }
}
```

```
/*################################################################################

Package:        Segmentation using a multifractal feature map

Programmer:     Hongjing Chen
                Department of Electrical & Computer Engineering
                University of manitoba
                Winnipeg, Manitoba
                hongjin@ee.umanitoba.ca

Package name:    MulFracSeg
Program included:Makefile edge.c readpgm.c renyidimen.c renyidimen.h round.h
Package description:This package is used to segment the image
                using a multifractal feature map.
version:        1.0

Last Update:    Dec. 12, 1995

################################################################################*/
```

Makefile

```
CC = gcc
RM = rm -f

INCLUDE_DIRS =

# Extra compiling definitions for debuging purposes
COMPILE_DEFINES =

CPPFLAGS = $(INCLUDE_DIRS) $(COMPILE_DEFINES)
```

```
CFLAGS = $(CPPFLAGS)

LD_DIRS =

LDLIBS = -lm

LDFLAGS = $(LD_DIRS)

TARGET = oh

CODE_OBJS = readpgm.o renyidimen.o edge.o

CODE_SRCS = readpgm.c renyidimen.c edge.c

SRCS = $(CODE_SRCS)

.SUFFIXES: .o .c

.c.o:
	$(CC) $(CFLAGS) -c $<

all:	$(TARGET)

$(TARGET): $(CODE_OBJS)
	$(CC) $(CFLAGS) $(LDFLAGS) $(CODE_OBJS) $(LDLIBS) -o $@

clean:
	$(RM) $(TARGET) $(CODE_OBJS)

strip:
        strip $(TARGET)
fred:
        ls -al $(CODE_OBJS)
```

edge.c

```
#include <stdio.h>
#include <math.h>
EdgeImage(edge,xsize,ysize)
int **edge;
int xsize,ysize;
{
        int i,j;
        unsigned char Pixel;
        FILE *fp;

        if( (fp = fopen("edge.pgm","wb") ) != NULL)
        {
                fprintf( fp, "P5\n# Hongjin's file\n%d %d\n255\n", xsize, ysize);
                for (j=0;j<ysize;j++)
                {
                        for(i=0;i<xsize;i++)
                        {
                                Pixel = (unsigned char)edge[i][j];
                                fwrite( &Pixel, sizeof(char), 1, fp);
                        }
                }
         }
```

```
    else
    {
        perror("Error opening file.");
        exit(0);
    }
    fclose(fp);
}
```

readpgm.c

```
#include <stdio.h>
#include <malloc.h>
#include <math.h>
#include "renyidimen.h"
#include "round.h"

int **blockExtract(image, x, y, width, height)
int **image;
int x, y;
int width, height;
{
    int **returnBlock;
    int i, j;

    returnBlock = (int **) calloc(width, sizeof(int *));
    for (i = 0; i < width; i++)
        returnBlock[i] = (int *) calloc(height, sizeof(int));

    for (j = y; j < height + y; j++) {
```

```
                for (i = x; i < width + x; i++) {
                        returnBlock[i - x][j - y] = image[i][j];
                }
        }

        return returnBlock;
}

feature Extract_feature(int **, int, int, int, int);

int main(argc, argv)
int argc;
char **argv;
{
        int loop;
        int S;
        FILE *fh;
        char str[255];
        int ysize, xsize;
        int i, j,m,n,k;
        char ch;
        int **image;
        int **imageBlock;
        int xBlockSize, yBlockSize;
        double a;
        int **edge;
        feature Cstx;
        FILE *fpf;
        FILE *fpda;

        fpf=fopen("desity_urban512","w");
```

```
fpda=fopen("dalpha_urban512","w");
if ( (fh = fopen("/home/ee/u34/hongjin/Image_LIB/mcell512.pgm", "r"))
                                        == NULL )
{
        fprintf(stderr, "Error opening file.\n");
        exit(-1);
}

while ( (fgetc(fh) != '\n') ) { }
 while ( (fgetc(fh) != '\n') ) { }
while ( (fgetc(fh) != '\n') ) { }
fscanf(fh, "%d %d\n", &ysize, &xsize);
printf("ysize = %d, xsize = %d\n", ysize, xsize);
fscanf(fh, "%s\n", str);
xBlockSize = yBlockSize = 3;

edge= (int **) calloc(ysize - yBlockSize+1, sizeof(int *));
for (i = 0; i < ysize - yBlockSize+1; i++)
        edge[i] = (int *) calloc(xsize - xBlockSize+1, sizeof(int));

image = (int **) calloc(ysize, sizeof(int *));
for (i = 0; i < ysize; i++)
        image[i] = (int *) calloc(xsize, sizeof(int));

for (j = 0; j < xsize; j++)
{
        for (i = 0; i < ysize; i++)
        {
                image[i][j] = 255-(int)fgetc(fh);
        }
}
```

```
    for (n = 0; n < ysize - yBlockSize + 1; n++)
    {
        for (m = 0; m < xsize - xBlockSize + 1; m++)
        {
            imageBlock = blockExtract(image, m, n, xBlockSize,
             yBlockSize);
            Cstx = Extract_feature(imageBlock, xBlockSize,
             yBlockSize,m,n);
            if(fabs(Cstx.man)>=7.5||fabs(Cstx.man)>=7.3&&
            fabs(Cstx.man)<7.5&&fabs(Cstx.dalpha)<0.0001||
            fabs(Cstx.man)<7.3)
                edge[m][n]=255;
            else
            {
                edge[m][n]=0;
            }

            fprintf(fpf,"%d %lf\n",m, Cstx.man);
            fprintf(fpda,"%lf\n", fabs(Cstx.dalpha));
            free(imageBlock);
        }
    }

    EdgeImage(edge,xsize - xBlockSize+1,ysize - yBlockSize+1);
    return 0;
}
```

renyidimen.c

```
#include<math.h>
#include<stdio.h>
#include "renyidimen.h"

feature RenyiDimen(int **, int, int, double, int, int);

feature Extract_feature(image,xsize,ysize,m,n)
int **image;
int xsize,ysize;
int m, n;
{
        double N,temp;
        int i,j;
        N=0.0;
        for(i=0;i<xsize;i++)
        {
                for(j=0;j<ysize;j++)
                {
                        N=(double)image[i][j]+N;/*the grey has been reversed*/

                }
        }

        return RenyiDimen(image,xsize,ysize,N,m,n);

}


feature RenyiDimen(image,xsize,ysize,N,m,n)
```

```
int **image;
int xsize, ysize;
double N;
int m, n;
{
    int z,i,j;
    double q,nl,na,ns,sum,r,man,sing;
    double dDq,ddDq;
    double grid;
    double slope,temp1;
    feature fred;

    grid=0.33333333333333333;

    q=(double)1.1;
    nl=0.0;na=0.0;ns=0.0;sum=0.0;

    for(i=0;i<xsize;i++)
    {
        for(j=0;j<ysize;j++)
        {
            if(image[i][j]!=0)
            {
                sum=sum+pow( (double)image[i][j],q);
                na=pow((double)image[i][j],q)*
                log((double)image[i][j])+ na;
                ns=pow((double)image[i][j],q)*
                (log((double)image[i][j]))*
                (log((double)image[i][j])) +ns;
            }
        }
        }
        r=log(grid);

        if(q!=1)
        {
                fred.sing=(na-log(N)*sum)/(sum*r);
                fred.dalpha=(sum*ns-na*na)/(sum*sum*r);
                fred.slope=( log(sum)-q*log((double)N) )/((q-1.0)*r);
                fred.man=( nl - sum*log(sum) )/(sum*r);
                dDq=( (q-1.0)*na + sum*log((double)N)  -
                sum*log(sum))/((q-1.0)*(q-1.0)*r*sum);
                fred.dman=(1-q)*dDq-slope+q*fred.dalpha+sing;
        }
        return(fred);
}
```

renyidimen.h

```
struct _feature {
        double slope;
        double dalpha;
        double dman;
        double man;
        double sing;
};

typedef struct _feature feature;
```

round.h

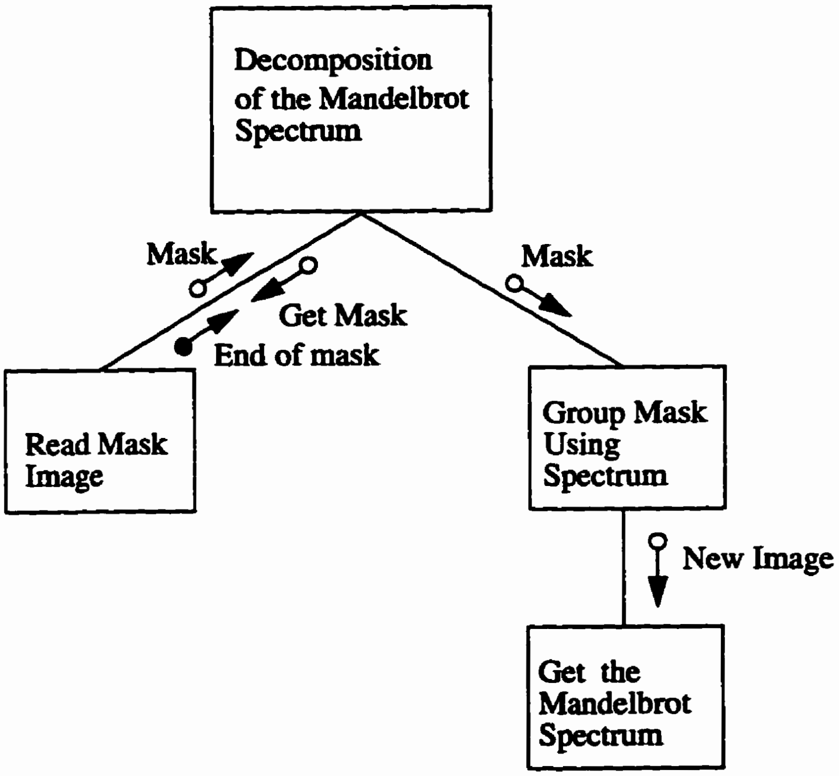
```
int round(num)
double num;
{
        if(num>=0)
        {
                num=num+.5;
        }
        else
        {
                num=num-.5;
        }
        return (int)num;
}
```

APPENDIX B

Structure of Program and Source code in C++

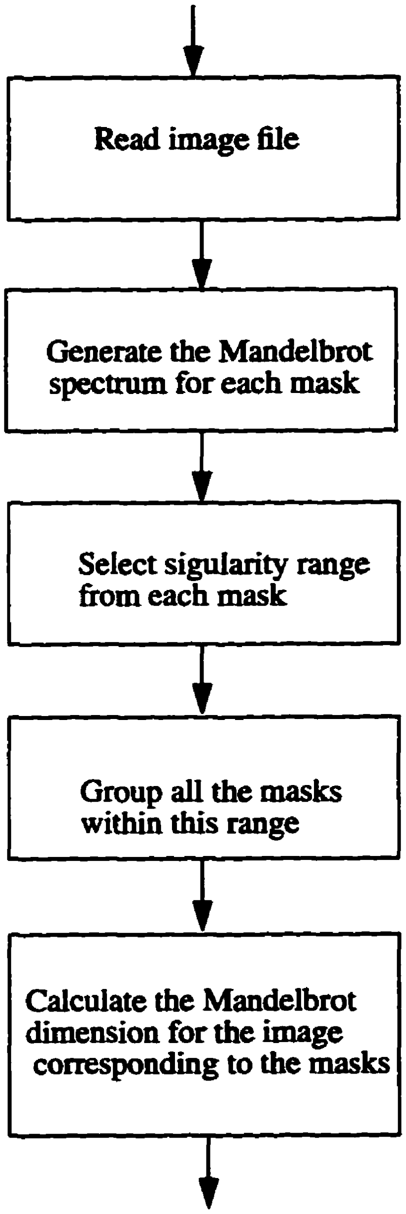
B. 1 Structured Chart

The decomposition of the Mandelbrot spectrum

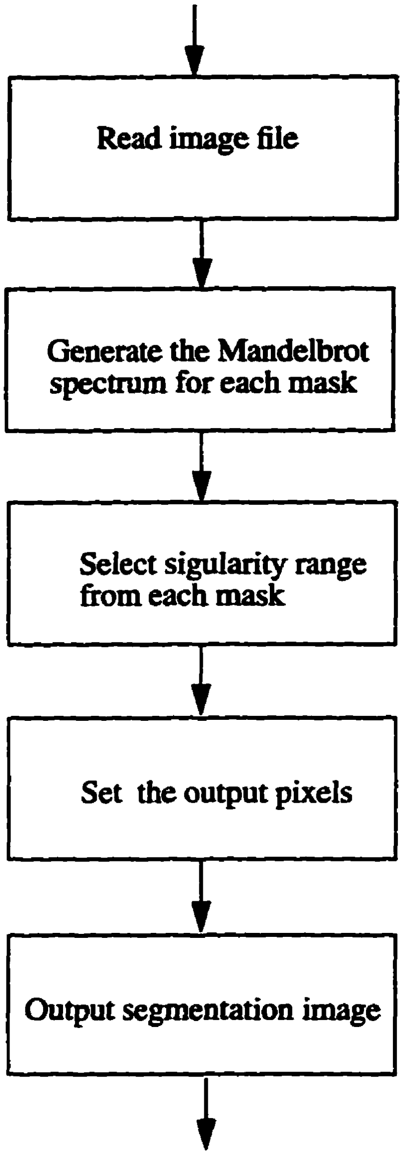

Decomposition of the Mandelbrot Spectrum
Mask
Get Mask
End of mask
Mask
Read Mask Image
Group Mask Using Spectrum
New Image
Get the Mandelbrot Spectrum

B. 2 Flow Diagrams

1. The decomposition of the Mandelbrot spectrum


Read image file
Generate the Mandelbrot spectrum for each mask
Select sigularity range from each mask
Group all the masks within this range
Calculate the Mandelbrot dimension for the image corresponding to the masks

2. The application of the decomposition of the Mandelbrot spectrum


Read image file
Generate the Mandelbrot spectrum for each mask
Select sigularity range from each mask
Set the output pixels
Output segmentation image

B. 3 Source Code in C++

```
/*################################################################################

Program Package:Segmentation using multifractal feature maps

Programmer:     Hongjing Chen
                Department of Electrical & Computer Engineering
                University of manitoba
                Winnipeg, Manitoba
                hongjin@ee.umanitoba.ca

Package names:  DecomMand
Program names:  Makefile main.cc InfoOfSeg.h InfoOfSeg.cc MulMaxMin.cc
                MulMaxMin.h InfoOfSeg.h
Package description:This package is used to segment the image
                using multifractal feature maps.
version:        1.5

Last Update:    April 10, 1996

################################################################################*/
```

Makefile

```
CC = g++

RM = rm -f
```

```
LDLIBS = -lm

TARGET = r

MY_OBJS = main.o MulMaxMin.o InfoOfSeg.o
MY_SRCS = main.cc MulMaxMin.cc  InfoOfSeg.cc
SRCS = $(MY_SRCS)

.SUFFIXES: .o .cc

.cc.o:
        $(CC) $(CFLAGS) -c $<

all:    $(TARGET)

$(TARGET):$(MY_OBJS)
        $(CC) $(MY_OBJS) $(LDLIBS) -o $@

clean:
        $(RM) $(MY_OBJS) 00* Seg*

backup: Makefile *.cc *.h
        tar cf $(TARGET)`date +%y%m%d` Makefile *.cc *.h
        gzip -S .tgz $(TARGET)`date +%y%n%d`
```

InfoOfSeg.h

```
typedef struct
```

```
{
    int xsize, ysize;
    int **image;
    double N;
}Data1;



/**************************PUBLICOPERATIONS*********************
*           void LoadImage(char* filename);
*           int Mean(char* filename);
*           void MaxMinGS(char* filename);
*           void PsuedPoints();
*           void MandelbrotSpectrum(char* filename);
*
******************************************************************/


class InfoOfSegImage
{
    private:
            Data1 Image;

    public:
            InfoOfSegImage();
            ~InfoOfSegImage();
            void LoadImage(char* filename);
            int Mean(double a, double b);
            //void MaxMinGS(char* filename);
            void MaxMinGS(double a, double b);
```

```
                void PsuedPoints();
                void MandelbrotSpectrum(char* filename);



};
```

InfoOfSeg.cc

```
#include<malloc.h>
#define max(a,b) ( (a>b) ? a:b )
#define min(a,b) ( (a<b) ? a:b )

void InfoOfSegImage::PsuedPoints()
{
        double N=0.0;
        Image.N=6439153.0;
        printf("N=%lf\n", Image.N);

}


void InfoOfSegImage::MandelbrotSpectrum(char* filename)
{
        char outputneme1[25];
        FILE *fpfa;
        sprintf(outputneme1,"fa%s",filename);
        fpfa=fopen(outputneme1,"w");
```

```
printf("Do not be panic, No problem right now,
        just make sure xsize = %d is correct ?\n", Image.xsize);

double r=log(1.0 / Image.xsize);

for (int j = 0; j < Image.xsize; j++)
{
        for (int i = 0; i < Image.ysize; i++)
        {
                Image.image[i][j] = 255 - Image.image[i][j];
        }
}

for(int z=-30*2; z<=30*2; z++)
{
        double q=(double)z/2;
        double nl = 0.0; double na = 0.0;
        double ns = 0.0; double sum = 0.0;

        for(int i = 0; i < Image.xsize; i++)
        {
                for(int j = 0; j < Image.ysize; j++)
                {
                        if(Image.image[i][j] != 0)
                        {
                                sum = sum +
                                pow((double)Image.image[i][j],q);
                                nl = pow((double)Image.image[i][j],q)*q*
                                        log((double)Image.image[i][j]) + nl;
                                na = pow((double)Image.image[i][j],q)*
                        log((double)Image.image[i][j]) + na;
                    ns = pow((double)Image.image[i][j],q)*
                        (log((double)Image.image[i][j]))*
                        (log((double)Image.image[i][j])) + ns;
                }
            }
        }

        if(q!=1)
        {

            double slope=( log(sum)-q*log(Image.N) )/((q-1.0)*r);
            double fq=( nl - sum*log(sum) )/(sum*r);
            double aq=(na-log(Image.N)*sum)/(sum*r);
            fprintf(fpfa,"%lf %lf\n",aq,fq);
            //printf("%lf %lf\n",q,slope);

            //fprintf(fpdq,"%lf %lf\n",q,slope);
            //fprintf(fpfq,"%lf %lf\n",q,fq);
            //fprintf(fpaq,"%lf %lf\n",q,aq);
            //printf("%lf %2.30f\n",q,aq);
        }
    }
    fclose(fpfa);
}

InfoOfSegImage::InfoOfSegImage()
{
    Image.xsize = 0;
    Image.ysize = 0;
    Image.image = NULL;
```

```
}

InfoOfSegImage::~InfoOfSegImage()
{
        free(Image.image);
}


void InfoOfSegImage::LoadImage(char* filename)
{
        char str[255];
        FILE *fh;
        if ( (fh = fopen(filename, "r")) == NULL )

        {
                fprintf(stderr, "Error opening file.\n");
                exit(-1);
        }

        char PPMType[3];
        PPMType[0] = fgetc(fh);
        PPMType[1] = fgetc(fh);
        PPMType[2] = '\0';
        printf("PPMType = '%s'\n", PPMType);
        while ( (fgetc(fh) != '\n') ) { }
        if ( strcmp(PPMType, "P5") != 0 )
        {
                fprintf(stderr, "Not a PGM (P5) file.\n");
                return;
        }
        do {
            fgets(str, 255, fh);
            printf("In here str = '%s'\n", str);
        } while ( str[0] == '#' );

        printf("str = '%s'\n", str);

        sscanf(str, "%d %d\n", &Image.ysize, &Image.xsize);
        printf("ysize = %d, xsize = %d\n", Image.ysize, Image.xsize);
        fscanf(fh, "%s\n", str);



        Image.image = (int **)calloc(Image.ysize, sizeof(int *));
        for(int i=0; i<Image.ysize; i++)
        {
            Image.image[i]= (int *) calloc(Image.xsize, sizeof(int));
        }

        for (int j = 0; j < Image.xsize; j++)
        {
            for (int i = 0; i < Image.ysize; i++)
            {
                Image.image[i][j] = (int)fgetc(fh);
                //printf("(%d,%d) %d\n",i, j,Image.image[i][j]);

            }
        }
        fclose(fh);
}

int InfoOfSegImage::Mean(double a, double b)
{
```

```
FILE *fp_mean;
FILE *fp_index;
FILE *fp_var;
FILE *fp_svar;
fp_mean = fopen("00mean","a");
fp_index = fopen("00index","a");
fp_var = fopen("00var","a");
fp_svar = fopen("00svar","a");
int temp=0;
int index=0;
for (int j = 0; j < Image.xsize; j++)
{
        for (int i = 0; i < Image.ysize; i++)
        {
                if(Image.image[i][j]!=255)
                {
                        temp+=Image.image[i][j];
                        index ++;
                }
        }
}

int meanvalue = temp /index;


fprintf(fp_index,"%lf %d\n%lf %d\n",a,index,b,index);
fprintf(fp_mean,"%lf %d\n%lf %d\n",a,meanvalue,b,meanvalue);
//fprintf(fp_mean,"%s %d\n",filename,meanvalue);
fclose(fp_index);
fclose(fp_mean);
```

```
    int var_tmp = 0;
    for (int j = 0; j < Image.xsize; j++)
    {
        for (int i = 0; i < Image.ysize; i++)
        {
            if(Image.image[i][j]!=255)
            {
                int dif = Image.image[i][j] - meanvalue;
                var_tmp += dif * dif;
            }
        }
    }
    int var = var_tmp / (index);
    int svar = (int) (sqrt(  (double) var_tmp / (index-1))) ;

    printf("index=%d\n",index);
    printf("meanvalue=%d\n",meanvalue);
    printf("var = %d\n",var);
    printf("svar = %d\n",svar);
    fprintf(fp_var,"%lf %d\n%lf %d\n",a,var,b,var);
    fprintf(fp_svar,"%lf %d\n%lf %d\n",a,svar,b,svar);
    fclose(fp_var);
    fclose(fp_svar);

    return temp;
}


void InfoOfSegImage::MaxMinGS(double a, double b)
{
    FILE *fp_max;
    FILE *fp_min;
    fp_max = fopen("00max","a");
    fp_min = fopen("00min","a");
    int maxtemp=0;
    int mintemp=255;

    for (int j = 0; j < Image.xsize; j++)
    {
        for (int i = 0; i < Image.ysize; i++)
        {
            if(Image.image[i][j]!=255)
            {
                maxtemp = max(maxtemp, Image.image[i][j]);
                mintemp = min(mintemp, Image.image[i][j]);


            }
        }
    }
    printf("maxtemp=%d\n",maxtemp);
    printf("mintemp=%d\n",mintemp);
    fprintf(fp_max,"%lf %d\n%lf %d\n",a,maxtemp,b,maxtemp);
    fprintf(fp_min,"%lf %d\n%lf %d\n",a,mintemp,b,mintemp);
    fclose(fp_max);
    fclose(fp_min);
}
```

MulMaxMin.c

```
#include<stdio.h>
```

```
#include<iostream.h>
#include<malloc.h>
#include<math.h>
#include<fstream.h>
#include<string.h>
#include<assert.h>
#include "MulMaxMin.h"

#define max(a,b) ( (a>b) ? a:b )
#define min(a,b) ( (a<b) ? a:b )

/*************************PUBLIC OPERATIONS*********************
*
*       void SetMaskSize()
*       void SegImageOrigin()
*       void SegImageput(char *putname)
*       void SegCondition(double LBound, double HBound, double tolerance)
*       void GetImageFeature()
*       void SegOutput(char *putname)
*       void Mask(int x, int y)
*       void FreeMask()
*       void GetMaskFeature()
*       void LoadImage(char* filename)
*
**********************************************************************/

void Multifractals::SetMaskSize()
{
        nbyn.width = 3;
        nbyn.height = 3;
}
```

```
void Multifractals::SegImageOrigin()
{
        SegImage = (int **) malloc(Image.ysize * sizeof(int *));
        for (int k = 0; k < Image.ysize; k++)
        {
                SegImage[k] = (int *) malloc(Image.xsize * sizeof(int));
        }

        int tempYsize = Image.ysize - nbyn.height + 1;
        int tempXsize = Image.xsize - nbyn.width + 1;
        for (int n = 0; n < Image.ysize; n++)
        {
                for (int m = 0; m < Image.xsize; m++)
                {
                        SegImage[m][n]=255;
                }
        }

        for (int n = 0; n < tempYsize; n++)
        {
                for (int m = 0; m < tempXsize; m++)
                {
                        if( edge[m][n]==0 )
                        {
                                for(int j = 0; j < nbyn.height; j++)
                                        for(int i=0; i < nbyn.width; i++)
                                                SegImage[m+j][n+i]=255-
                                                Image.image[m+j][n+i];
                        }
                }
        }

}

void Multifractals::SegImageput(char *putname)
{
        FILE *fp;
        char filename[25];
        unsigned char Pixel;
        strcpy(filename, putname);

        if( (fp = fopen(filename,"wb") ) != NULL)
        {
                fprintf( fp,"P5\n# Hongjin's file\n%d %d\n255\n", Image.xsize,
                                                        Image.ysize);
                for (int j=0;j<Image.ysize;j++)
                {
                        for(int i=0;i<Image.ysize;i++)
                        {
                                Pixel = (unsigned char)SegImage[i][j];
                                fwrite( &Pixel, sizeof(char), 1, fp);
                        }
                }
        }
        else
        {
                perror("Error opening file.");
        exit(0);
        }
        fclose(fp);
```

```
}

void Multifractals::SegCondition(double LBound, double HBound, double tolerance)
{

        Features **A1;
        int tempYsize = Image.ysize - nbyn.height + 1;
        int tempXsize = Image.xsize - nbyn.width + 1;

        printf("\nThe image size is %d  in SegCondition\n", Image.xsize);
        printf("The mask size we use is %d  in SegCondition\n", nbyn.width);
        A1 = GetImageFeature();
        edge = (int **) malloc(tempYsize * sizeof(int *));
        assert(edge!=NULL);
        for (int k = 0; k < tempYsize; k++)
        {
                edge[k] = (int *) malloc(tempXsize * sizeof(int));
                //assert(edge[k]!=NULL);
        }

        for (int n = 0; n < tempYsize; n++)
        {
                for (int m = 0; m < tempXsize; m++)
                {
                        //int a1 = (int) fabs( A1[m][n].maxG-HBound);
                        //int a2 = (int) fabs( A1[m][n].minG-LBound);
                        //printf("%d %d\n",A1[m][n].maxG, A1[m][n].minG);

                        double a1 = HBound - A1[m][n].maxG;
                        double a2 = A1[m][n].minG - LBound;
```

```
            if((a1 >= 0.0) && (a2 >= 0.0))
            {
                edge[m][n]=0;
            }
            else
                edge[m][n]=255;

        }
    }

    for (int k = 0; k < tempYsize; k++)
        free(A1[k]);
    free(A1);

}


Features** Multifractals::GetImageFeature()
{
    Features **AA;

    printf("Image size=%d\n",Image.ysize);
    printf("Mask size=%d\n",nbyn.height);

    int tempYsize = Image.ysize - nbyn.height + 1;
    int tempXsize = Image.xsize - nbyn.width + 1;

    AA = (Features **) malloc(tempYsize * sizeof(Features *));
    assert(AA!=NULL);
    for (int k = 0; k < tempYsize; k++){
        AA[k] = (Features *) malloc(tempXsize * sizeof(Features));
        assert(AA[k]!=NULL);
        }

        for (int n = 0; n < tempYsize; n++)
        {
                for (int m = 0; m < tempXsize; m++)
                {
                        Mask(m,n);
                        AA[m][n] = GetMaskFeature();
                }
        }
        FreeMask();

        return AA;
}

void Multifractals::SegOutput(char *putname)
{
        FILE *fp;
        char filename[25];
        unsigned char Pixel;
        strcpy(filename, putname);

        int tempYsize = Image.ysize - nbyn.height + 1;
        int tempXsize = Image.xsize - nbyn.width + 1;

        if( (fp = fopen(filename,"wb") ) != NULL)
        {
                fprintf( fp,"P5\n# Hongjin's file\n%d %d\n255\n", tempXsize,
                                                          tempYsize);
                for (int j=0;j<tempYsize;j++)
                {
                        for(int i=0;i<tempXsize;i++)
                        {
                                Pixel = (unsigned char)edge[i][j];
                                fwrite( &Pixel, sizeof(char), 1, fp);
                        }
                }
         }
        else
        {
                perror("Error opening file.");
        exit(0);
        }
        fclose(fp);
}


void Multifractals::FreeMask()
{

        for (int k = 0; k < nbyn.width; k++)
                free(returnBlock[k]);
        free(returnBlock);
        returnBlock=NULL;
        //SetMaskSize();

}


void Multifractals::Mask(int x, int y)
```

```
{

        returnBlock = (int **) malloc(nbyn.width * sizeof(int *));
        for (int k = 0; k < nbyn.width; k++)
                returnBlock[k] = (int *) malloc(nbyn.height * sizeof(int));

        for (int j = y; j < nbyn.height + y; j++)
        {
                for (int i = x; i < nbyn.width + x; i++)
                {
                        returnBlock[i - x][j - y] = Image.image[i][j];
                }
        }
}


Multifractals::Multifractals()
{
        Image.xsize = 0;
        Image.ysize = 0;
        Image.N = 0.0;
        Image.image = NULL;
        Image.fh = NULL;
        returnBlock = NULL;
        SegImage=NULL;
        nbyn.width = 0;
        nbyn.height = 0;
        prevW = 0;
        prevH = 0;
        DAF.minG = 0.0;
        DAF.maxG = 0.0;
```

```
}


Multifractals::~Multifractals()
{
        free(Image.image);
        free(returnBlock);
        free(Image.fh);
}


/*void Multifractals::Display()
{
        for (int n = 0; n < Image.ysize - nbyn.height + 1; n++)
        {
                for (int m = 0; m < Image.xsize - nbyn.width + 1; m++)
                {
                        printf("%lf\n",AA[m][n].slope); ;
                }
        }
}
*/

Features Multifractals::GetMaskFeature()
{
        int maxtemp=0;
        int mintemp=255;

        for(int i = 0; i < nbyn.width; i++)
        {
                for(int j = 0; j < nbyn.height; j++)
```

```
            {
                        maxtemp = max(maxtemp, returnBlock[i][j]);
                        if( returnBlock[i][j] != 0 )
                        mintemp = min(mintemp, returnBlock[i][j]);
            }
      }
      if(mintemp==0) printf("Please report bugs to hongjin@ee.umanitoba.ca");

      DAF.maxG =( log((double)mintemp) - log(Image.N) )/log(1.0/Image.xsize);
      DAF.minG =( log((double)maxtemp) - log(Image.N) )/log(1.0/Image.xsize);
      //printf("%lf %lf\n", DAF.maxG, DAF.minG);



      return DAF;
}

void Multifractals::LoadImage(char* filename)
{
      char str[255];
      if ( (Image.fh = fopen(filename, "r")) == NULL )

      {
            fprintf(stderr, "Error opening file.\n");
            exit(-1);
      }

      while ( (fgetc(Image.fh) != '\n') ) { }
      while ( (fgetc(Image.fh) != '\n') ) { }
      while ( (fgetc(Image.fh) != '\n') ) { }
      fscanf(Image.fh, "%d %d\n", &Image.ysize, &Image.xsize);
      printf("ysize = %d, xsize = %d\n", Image.ysize, Image.xsize);
```

```
    fscanf(Image.fh, "%s\n", str);
}

void Multifractals::ImageMatrix()
{
    double NTemp = 0.0;
    Image.image = (int **)calloc(Image.ysize, sizeof(int *));
    for(int i=0; i<Image.ysize; i++)
    {
        Image.image[i]= (int *) calloc(Image.xsize, sizeof(int));
    }

    for (int j = 0; j < Image.xsize; j++)
    {
        for (int i = 0; i < Image.ysize; i++)
        {
            Image.image[i][j] = 255-(int)fgetc(Image.fh);
            NTemp += (double)Image.image[i][j];
        }
    }
    Image.N = NTemp;
}
```

MulMaxMin.h

```
#include<stdio.h>
#include "Typedef.h"
```

```
class Multifractals
{
    private:
        Data Image;
        int **returnBlock;
        int **SegImage;
        Features DAF;
        MaskSize nbyn;
        short int prevW;
        short int prevH;
        int **edge;
        Features GetMaskFeature();//Calculate the feature based on Mask(x,y)
    public:
        Multifractals();
        ~Multifractals();
        void LoadImage(char* filename);//Read a image file
        void ImageMatrix();//Put coodinate to each pixel read
        void SetMaskSize();//Set size of mask
        void Mask(int x, int y);//Read Data of the mask beginning from (x,Y)
        void FreeMask();        //Clean the mem of previous mask
        Features** GetImageFeature();//Expand GetMaskFeature()
        void SegCondition(double LBound, double HBound, double tolerance);
                                              //Select the feature, mark them with
white
        void SegOutput(char *putname); //Output the marked pixels with white
        //void Output(char *filename); //Put values from GetImageFeature()
        //void Display();
        void SegImageOrigin();
        void SegImageput(char *putname);


};
```

Typedef.h

```
typedef struct
{
        double maxG;
        double minG;
        //double fq;
}Features;

typedef struct
{
        int xsize, ysize;
        int **image;
        FILE *fh;
        double N;
}Data;

typedef struct
{
        int width;
        int height;
}MaskSize;
```

main.cc

```
#include "MulMaxMin.h"
```

```
#include<math.h>
#include<time.h>
#include "InfoOfSeg.h"


main(int argc, char** argv)
{
        time_t started, finished;
        started = time(NULL);
        int counter;

        for (counter = 1; counter < argc; counter += 2)
        {
                Multifractals urban;
                urban.LoadImage("/home/ee/u34/hongjin/cNewFolder/Levy
                        /CPPLEVI/mulfeature/Press
                        /UsingAlphaSpectrum/original.pgm");
                urban.ImageMatrix();
                //char outname1[25];
                //sprintf(outname1, "MaskSlope%d", q);
                //urban.Output(outname1);
                urban.SetMaskSize();


                urban.SegCondition( atof(argv[counter]), atof(argv[counter + 1]), 0.0);

                char outname2[25];
                sprintf(outname2, "Seg%s_%s.pgm", argv[counter], argv[counter +1]);
                urban.SegOutput(outname2);
                urban.SegImageOrigin();
```

```
            char outname3[25];
            sprintf(outname3, "SegImage%s_%s.pgm",
                    argv[counter], argv[counter+1]);
            urban.SegImageput(outname3);




            char outname4[25];
            sprintf(outname4, "%s_%s",argv[counter], argv[counter+1]);
            InfoOfSegImage Seg;
            Seg.LoadImage(outname3);
            Seg.Mean( atof( argv[counter]), atof( argv[counter+1]) );
            //Seg.MaxMinGS(outname4);
            Seg.MaxMinGS( atof( argv[counter]), atof( argv[counter+1]) );
            Seg.PsuedPoints();
            Seg.MandelbrotSpectrum(outname4);

        }
        finished = time(NULL);
        printf("\n Run time = %d sec [%d min %d sec] \n",
        (int)(difftime(finished,started)),
        (int)(difftime(finished,started)/60),
        (int)(difftime(finished,started))-
        (int)(difftime(finished,started)/60)*60);
}
```