WAVELET AND IFS COMPRESSION OF MULTIFRACTAL OBJECT BOUNDARIES

by
SHARJEEL A. SIDDIQUI

An M.Sc. Thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

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

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

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES
*****
COPYRIGHT PERMISSION

"Wavelet and IFS Compression of Multifractal Object Boundaries"

BY

Sharjeel A. Siddiqui

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

Of

MASTER OF SCIENCE

Sharjeel A. Siddiqui © **2005**

ABSTRACT

This thesis is concerned with compact modelling of multifractal object boundaries that are self-similar, rough (non-smooth) but crisp (one-pixel wide) and non occluded, for applications such as MPEG-7. Since traditional modelling techniques focus on smooth boundaries, this thesis uses techniques that are based on multiresolution, singularity detection, and affine transformations to model these boundaries compactly. This thesis presents two new approaches that are based on wavelets and iterated function systems (IFS).

The first approach uses wavelet-based techniques. In this approach, the boundary is represented compactly by two sets of descriptors and control points. The first set contains information about complexities present in the boundary, and is calculated using a fractal dimension analysis. The second set contains information about the shape of the boundary and is calculated by using the multiresolution and singularity detection properties of wavelet analysis. In addition, a wavelet based smoothing technique is applied before obtaining the second set of control points. The midpoint displacement algorithm is applied to the two sets of control points in order to reconstruct the boundaries with the required fractal dimension.

The second approach is based on the IFS. In this approach the object boundary is regarded as an attractor of an IFS having the same Rényi fractal dimension spectrum (RFDS) as the original boundary. The boundary is then reduced to a set of affine transformations and related edge points. The random iteration algorithm (RIA) is used to reconstruct the original boundary while preserving the original multifractal spectrum of the boundary.

The experimental results show that the multifractal object boundaries can be modelled compactly using both the approaches. The compression ratio achieved for wavelet-based approach was 285:1 while for IFS-based approach it was 175:1. The RFDS singularity measure was used to measure the quality of reconstruction. The results showed that the RFDS of the reconstructed boundary closely matches the RFDS of the original boundary, indicating good reconstruction quality. This result also highlights the significance of RFDS as a quality measure.

ACKNOWLEDGEMENTS

I would like to acknowledge and thank for the support of my advisor Dr. Witold Kinsner, for his guidance for the past four years, and for suggesting the topic of this thesis. His constant encouragement and suggestions guided me during the course of this research and enabled me to understand different aspects of this thesis. Dr. Kinsner was available when I needed him and I am very grateful for amount of time he spent with me. Also, I would like to acknowledge his financial support for conferences such as CCECE. I would also like to thank Dr. R. Mcleod and Jeff Anderson at Internet Innovation Center for their financial support during the course of this study. I would like to thank Dr. S. Balakrishnan, Dr. S. Hussain and Dr. R. Mcleod for agreeing to be on the committee and for their valuable comments.

I gratefully acknowledge the support of everybody in the Delta Research Group, past and present, including Kalen Brunham, Kevin Cannons, Jin Chen, Vincent Cheung, Dr. Richard Dansereau, Neil Gadhok, Dr. Bin Huang, Jay Kraut, Laila Safavian, Martin Stadler, Robert Berry and Hong Zhang. I would like to specially mention and thank Michael Potter, Stepehen Dueck and Aram Faghfouri for the many fruitful discussions we had over my thesis subject that helped me to understand and improve this research. Also, I would like to specially point out Dr. Hakim El-Boustani for his friendship, continuous help and encouragement during the course of this thesis. His guidance helped me to understand many important areas of this thesis.

Finally, I would like thank my parents for believing in me and for their continuous support. I would also like to thank my wife Erum for her love and encouragement. Her pleasant company during the later part of my thesis made my research work more interesting and delightful.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS, ACRONYMS, AND TRANSLATIONS

Bm	Brownian motion
CCECE	Canadian Conference on Electrical and Computer Engineering
CET	Combined Evidence Thresholding
Conf.	Conference
DA	Deterministic Algorithm
Dec.	December
ed.	edition
Ed.	Editors
Eq(s).	Equation(s)
et al.	et alia (Latin) ; and others
Feb.	February
Fig(s).	Figure(s)
fBm	fractional Brownian motion
FIF	Fractal Interpolation Functions
FIR	Finite Impulse Response
J.	Journal
Jan.	January

HCPs	High Curvature Points
HVS	Human Visual System
IBA	IFS based approach
IFS	Iterated Function Systems
ISDDR	Inter-Scale Difference Decay Rate
ISDR	Inter-Scale Decay Rate
MAT	Medial Axis (also called symmetric axis) Transform
MA	Multifractal Analysis
Mar.	March
MATLAB	Matrix Laboratory
MB	megabyte (or 1,024 kilobytes)
MDP	Midpoint Displacement Algorithm
MRA	Multi-Resolution Analysis
M.Sc.	Master of Science
MSE	Mean Square Error
no.	number
Oct.	October
PDF	Probability Density Function
P.Eng.	Professional Engineer
Ph.D.	Philosophiae Doctor (Latin) ; Doctor of Philosophy

p.	page
pp.	pages
Proc.	Proceedings
RIA	Random Iteration Algorithm
rms	*root mean square*
Rep.	Report
Rev.	Review
SDDR	Spatial Difference Decay Rate
sec	second(s)
Sep.	September
vel	volume element
vol.	volume
WBA	Wavelet based Approach
WTMM	Wavelet Transform Modulus Maxima

LIST OF SYMBOLS

D_S	Similarity Fractal Dimension
D_{BC}	Box-Counting Dimension
D_H	Hausdorff Dimension
E	Euclidean or Topological Dimension
H_q	Rényi Entropy
H	Hurst Exponent
$\mathbf{L^2(R)}$	Hilbert space of measurable, square-integrable 1-D functions
R	Set of Real numbers
W	Dyadic Wavelet Transform Operator
Z	Set of Integers

CHAPTER I

INTRODUCTION

1.1 Problem Definition

The recent rapid development of efficient and high quality image and video compression techniques has triggered much interest in identifying content in images and video that could help in obtaining better compression ratios. The proposed standard MPEG-7 includes Multimedia Content Description Interface [MPEG00] suggests image segmentation approaches based on colour, texture, shape and motion that can be used to specify a standard set of descriptors for content identification and representation. Content-based image compression schemes require accurate detection and compact modelling of objects present in the image. The process of image segmentation is often used to find these objects by locating their boundaries or regions of occupation that represent the shape of an object [Cast96][CoCe01][GoWo92][Russ98][ChBa98]. The resulting segmented image is compressed by modelling the boundaries and their interiors through compact representations.

The modelling of such boundaries can be considered as a curve-fitting problem through a set of optimum control points located along the path of the boundary, so that the curve would make an optimum fit to these control points. This can be achieved by applying curve-fitting methods like Bezier or splines, but problems arise if the boundary is either (i) rough (not smooth), or (ii) self-similar (fractal), or (i) blurred (more than one pixel wide).

1.2 Thesis Statement and Objectives

The objectives of this thesis is compact modelling of closed object boundaries that are (i) crisp (one pixel wide), (ii) rough (non smooth), (iii) self-similar (fractal), and (iv) non-occluded (not hidden by other object or boundary) using fractal and wavelet based techniques in order to achieve lossy compression, while preserving the key perceptual characteristics of the boundary.

This thesis presents two new approaches for compact modelling of object boundaries using wavelets and iterated function systems (IFS) techniques. Before applying these methods on the test images, it has been assumed that the object boundaries have already been segmented. The Rényi fractal dimension spectrum as well as the compression ratio are used as the quality measures for the reconstructed object boundaries.

1.3 Organization of the Thesis

This thesis is organized into nine chapters. Chapter 1 presents an introduction to the thesis subject. Chapter 2 provides background on image segmentation and object boundaries and the motivation behind this thesis. Chapter 3 provides the necessary background on fractals, fractal dimensions, multifractals, IFS and fractal curve generation techniques. Chapter 4 presents a background on multiresolution and wavelet techniques. Chapter 5 discusses how to find control points on smooth boundaries using wavelets, and the particular technique that was used in this thesis. Chapter 6 gives a background on wavelet based smoothing techniques and the description of the technique used in this thesis. Chapter 7 describes the design of the boundary modelling approachs presented in this thesis and their verification. Chapter 8 discusses the results and performance of these approachs. Finally, Chapter 9 reflects upon the thesis statement by presenting the conclusions and contributions of this thesis, as well as recommendations for future work. References and appendix follow after Ch. 9.

This thesis contains over 80 references. These references are arranged in alphabetical order. In this scheme there are at least four letters followed by last 2 digits of publication year. The letters are derived from the surname of the authors. The main advantage of this scheme is that a new reference code can be inserted any where in the document without rearranging the order of the previous codes.

The Matlab code used in this thesis is provided in appendix A. Each function contains proper header explaining the purpose of the function and its results.

CHAPTER II

IMAGE SEGMENTATION AND OBJECT BOUNDARIES

2.1 Introduction

This chapter presents a brief background on image segmentation, objects in digital images, and their boundaries. Since this thesis deals with object boundaries, more emphasis will be given on image segmentation related to these boundaries.

2.2 Objects, Shapes and Boundaries

The input to a typical image processing and analysis system is a gray-scale image of a scene containing the objects of interest. In order to understand the contents of the scene it is necessary to recognize the objects located in the scene. An object in an image can be any region of interest that can be differentiated from rest of the image based on properties like geometry, shape, colour, texture or illumination [Cast96] [CoCe01] [GoWo92] [Russ98]. The shape of the object is a binary image representing the extent of the object. In [CoCe01] shape is defined as any connected set of points. The shape can also be thought of as a silhouette of the object (e.g., obtained by illuminating the object by an infinitely distant light source).

The most defining feature of a shape is its *sketch* or *boundary* which is obtained using image segmentation techniques discussed in the later sections. There are many imaging applications where image analysis can be reduced to the analysis of shapes, (e.g., organs, cells, machine parts, characters).

2.2.1 Classification of Shape Analysis Methods

Shape analysis methods can be classified according to many different criteria. Pavlidis [Pavl78] has proposed the following classifications. The first classification is based on the use of shape defining boundary points as opposed to the interior of the shape. The two resulting classes of algorithms are known as boundary (also called external) and global (or internal), respectively. Examples of the former class are algorithms that parse the shape boundary [DuHa73][Gosh85][KaCh81][KaSa89][Neva82][TaSu89][YoWB74], and various Fourier transforms of the boundary [Gran72][PeFu77][RiHe74][WaWi80] [ZaRo72]. Examples of global methods include the medial axis (also called symmetric axis) transform (MAT) proposed by Blum and described in [Blum67][Smit70], moment based approaches [Hu62][Teag80], and methods of shape decomposition into other primitive shapes [FeSS80][KiPK87][RoBa79][SkSF80].

Another classification of shape analysis algorithms can be made on the basis of whether the result of the analysis is numeric or non-numeric. For example, the MAT produces another image (containing a symmetric axis) and is, therefore, called a space-domain technique. On the other hand, scalar transform techniques produce numbers (scalars or vectors) as results. Examples of later methods include various Fourier [Gran72]

[PeFu77] [RiHe74] [WaWi80] [ZaRo72] and moment-based [Hu62][Teag80] procedures for shape analysis.

A third classification of shape analysis methods can be made on the basis of information preservation. Methods which allow for the accurate (or at least sufficiently accurate) reconstruction of a shape from its descriptor are called *information preserving* methods, as opposed to methods only capable of partial reconstruction which are called *information non-preserving* techniques. These can also be called *lossy* and *lossless* techniques respectively. An example of an information non-preserving method is area to perimeter square ratio. Since many significantly different shapes can have the same area to perimeter square ratio, it is not possible to reconstruct the original shape knowing only its area to perimeter square ratio. Other simple shape descriptors suffer from the same problem.

This thesis will focus on the third type of classification for shape or boundary analysis methods.

2.2.2 Measures for Shape Description Methods

A common problem in shape description research is how to judge the quality of a shape description method. Not all methods are appropriate for every kind of shape and application; i.e., the method of choice depends on the properties of the shape to be described and the particular application. Furthermore, the presence of noise can also influence the choice of the method. A consistent measure for evaluation still does not exist for shape description methods. Several authors have proposed evaluation criteria in the form of lists of qualities that a good shape description method should have.

Marr and Nishihara [MaNi78] proposed a set of three criteria for the evaluation of shape description methods: i) accessibility, ii) scope and uniqueness and iii) stability and sensitivity. *Accessibility* describes how easy (or difficult) it is to compute a shape descriptor in terms of memory requirements and computational time. *Scope* demonstrates the class of shapes that can be described by the method. *Uniqueness* describes whether a one-to-one mapping exists between shapes and shape descriptors. *Stability* and *sensitivity* describe how sensitive a shape descriptor is to *small* changes in shape.

Brady [Brad83] proposed another set of criteria for the representation of shape:

- *Rich local support*: This criterion requires that a representation is information preserving (rich) and can be computed locally. Local support is important for the representation of occluded objects.
- Smooth extension and subsumption: This criterion ensures that local descriptions can easily produce global descriptions. This is a kind of continuity of representation.
- *Propagation*: This criterion adds a hierarchical property to representation in the sense that perceptual subparts are propagated from local to global levels of representation.

Brady illustrated these criteria on the Generalized Cylinder representation for three-dimensional objects and the Smoothed Local Symmetries representation for two-dimensional shapes [BrAs84].

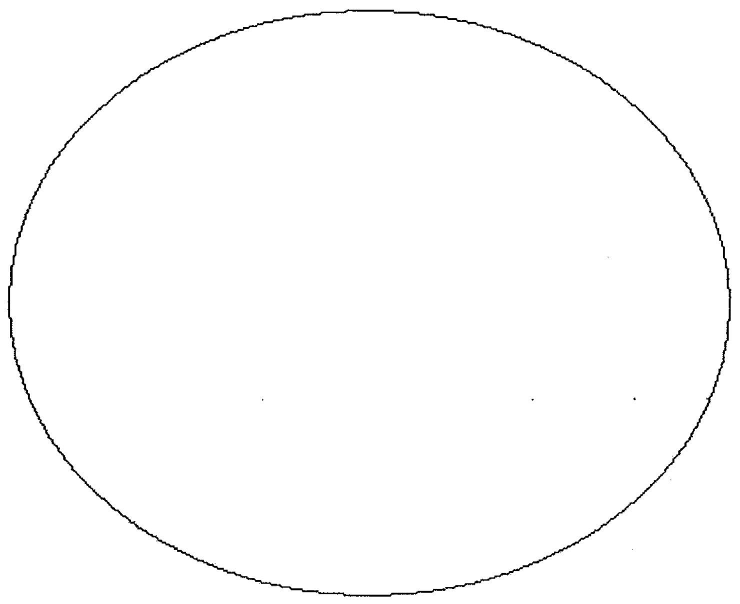
Both of the methods mentioned above define desired properties in terms of conceptual qualities that cannot be numerically expressed. This limitation makes the exact numerical comparison of shape description methods very difficult.

2.2.3 Types of Object Boundaries

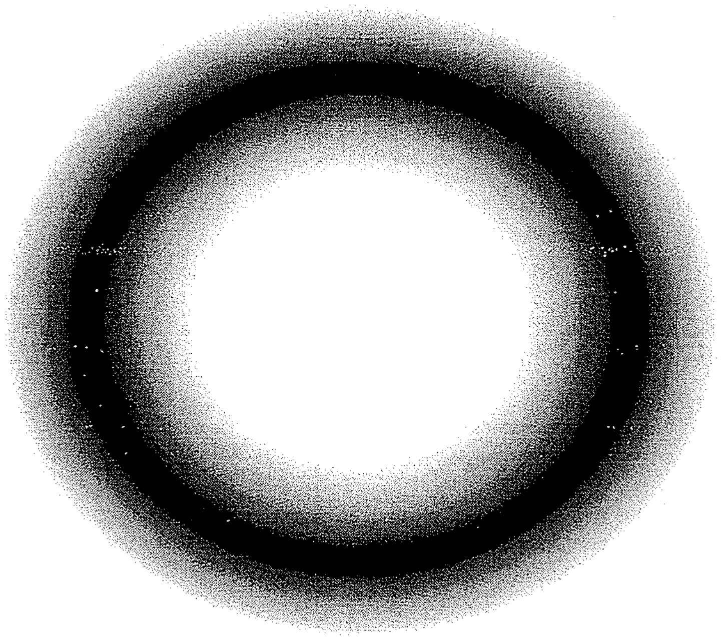
For the purpose of this thesis the boundary of an object or shape in an image is defined to be connected sets of points that outline the object completely, is closed and non occluded. In case of region or colour based segmentaion the outer most pixels of the region or object will be considered as the boundary. Different types of such object boundaries can be distinguished according to various types of geometrical (width or thickness) and statistical (self-similarity, texture, complexity) features present on it. Basically we can categorize a boundary in to two classes (i) smooth and (ii) fractal. Here 'complex' means that boundary cannot be characterized by standard tehniques that use Eucelidian geometry. These boundary types are as follows

2.2.3.1 Smooth and Crisp

This type of boundary is only one pixel-wide and is smooth. Such boundaries can be modelled using techniques like *spline* and *Bezier* curves [RoAd90]. Figure 2.1a shows an example of such a smooth and crisp boundary.



(a)



(b)

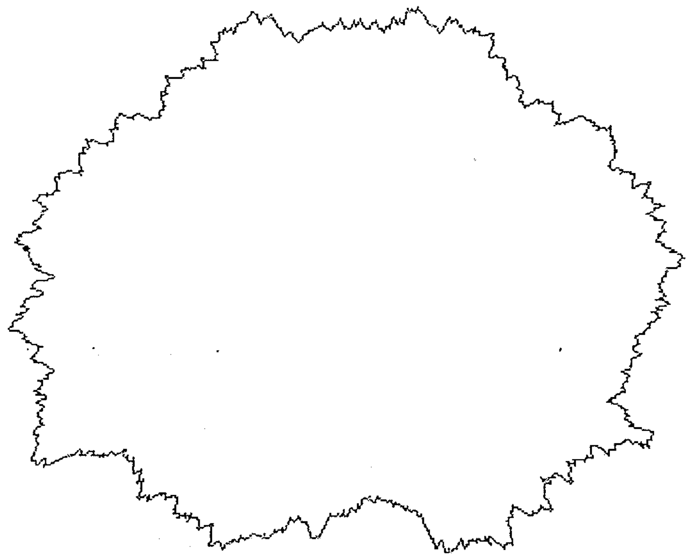
Fig. 2.1 Two types of smooth boundaries. (a) A smooth and crisp boundary.(b) A smooth and blurred boundary.

2.2.3.2 Smooth and Blurred

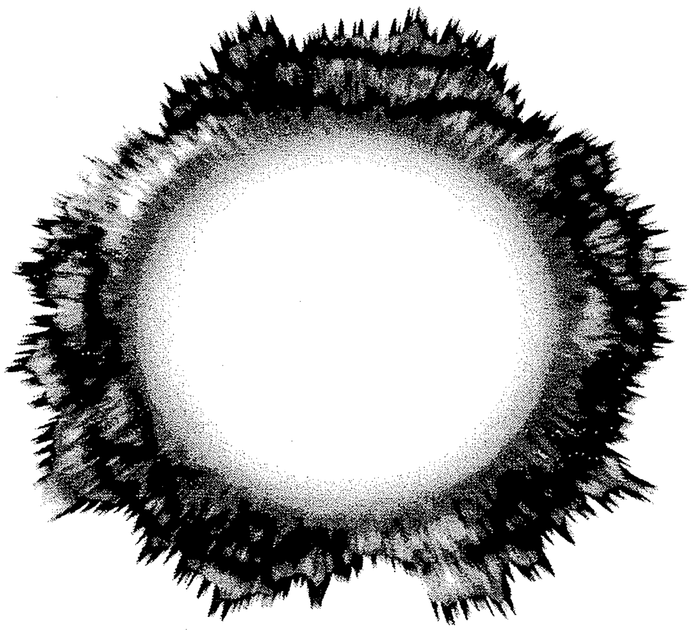
This type of boundary is more then one pixel-wide and is smooth. Such boundaries can be assumed to have some kind of texture with in them. These boundaries can be modelled using techniques like *texture* modelling that use measures such as variation of smoothness [RoAd90]. Figure 2.1b shows an example of such a smooth and blurred boundary.

2.2.3.3 Fractal and Crisp

This type of boundary is only one pixel-wide and is complex, with large amount of singularities. These boundaries can have self-similar or fractal characteristics. Such boundaries cannot be modelled using standard techniques like *spline* and *Bezier* curves [RoAd90]. Figure 2.2a shows an example of such a boundary. In some cases these boundaries can have local smooth characteristics, in variuos parts of the boundary while having fractal structures on other parts. Boundaries that have varying complexities and is self-similar can be considered to have multifractal charcteristics.



(a)



(b)

Fig. 2.2 Two types of complex boundaries. (a) A fractal and crisp boundary.(b) A fractal and blurred boundary.

2.2.3.4 Fractal and Blurred

This type of boundary is more then one pixel-wide and is complex. Such boundaries can be assumed to have some kind of texture with in them which can have self-similar or fractal characteristics. Such boundaries can be modelled using fractal or multifractal based texture modelling techniques [Kins94] if textures have self similar characteristics. Figure 2.2b shows an example of such a boundary.

In some cases, this type of boundary can be further subdivided in to two parts (i) interior, and (ii) exterior. Each part is then modelled separately to form the complete representation of the boundary.

The focus of this thesis is the complex and thin boundary that have self similar or multifractal characteristics. The topic of fractals and multifractals will be disussed in the next chapter.

2.3 Image Segmentation

The *human visual system* (HVS) appears to be one of the most sophisticated and versatile systems in nature [Attn54][Marr82][Zusn70]. Its ability to understand the organization of surrounding nature is unparalled in any artificially created reasoning systems. There has been a large amount of research activity concentrated on the study of the HVS. One of the purposes of these efforts is to provide a model for developing artificial systems for visual perception and cognition.

When a human observer views a scene, the visual system processing that takes place essentially segments the scene for him. This process is so efficient, that a person does not see a complex scene, but rather a collection of objects [Blum67]. With digital systems, holistic image analysis requires information to be extracted once the appropriate preprocessing steps of various filters are completed. In order to accomplish this, a digital image must be segmented, or partitioned into its constituent parts or objects. The segmentation process can also be viewed as separating the image into disjoint, or nonoverlapping regions. Here, a region is defined by a connected set of adjacent pixels. To be connected, a traceable path between any two pixels must be possible without ever leaving the set.

2.3.1 Different Techniques for Image Segmentation

Image segmentation can be approached from three different perspectives [Cast96][GoWo92][Russ98]. This first is a region approach where each pixel is assigned to a particular object or region. Next, there is a boundary approach where the location of boundaries existing between regions is desired. The final perspective is the edge approach, where identification of the edge pixels is followed by a linking process of these edges which will form the required boundaries. The level to which a digital image is segmented and processed is very subjective, meaning the segmentation process should stop when the objects of interest in an application have been identified.

The following sections present a brief overview of common image segmentation techniques.

2.3.1.1 Point Detetection

A point is the most basic and fundamental type of discontinuity in a digital image [Cast96][GoWo92][Russ98]. The most common approach to finding discontinuities is to run an (n x n) mask over each point in the image. The mask is best viewed as an (*n* x *n*) matrix which performs an averaging function according to the following equation

$$R = w_1z_1 + w_2z_2 + \ldots w_nz_n = \sum_{i=1}^{n} w_iz_i \tag{2.1}$$

where R is the response of the mask at any point in the image, z_i is the gray level of the pixel associated with the mask coefficient w_i. The response of the mask is then defined with respect to its center location. A general *3* x *3* mask is given by

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \tag{2.2}$$

Using the above description, we can easily detect isolated points in a digital image.

2.3.1.2 Line Detection

Line detection is the next level of complexity in the detection of image discontinuity [Cast96][GoWo92][Russ98]. Since a line in a digital image is a connected set of individual pixel points, the process is modelled after point detetection mentioned in the previous section. For any point in the image, a response can be calculated that will show which direction the point of a line is most associated with. For $n = 3$, the biased 3 x 3 matrices involved in calculating these responses are given by the following

For horizontal direction the matrice is given by

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.3}$$

For vertical direction the matrice is given by

$$\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \tag{2.4}$$

For diagonal (+45 degrees) direction the matrice is given by

$$\begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \tag{2.5}$$

For diagonal (-45 degrees) direction the matrice is given by

$$\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \tag{2.6}$$

The highest response calculation from these matrices will yield the direction of the given pixel.

2.3.1.3 Edge Detection

Since isolated points and lines of unitary pixel thickness are infrequent in most practical applications, edge detection is the most common approach in gray level discontinuity segmentation [Cast96][GoWo92][Russ98]. Edges can be defined in terms of singularity or non-differentiability at that location. Here the detection of discontinuity in a digital image is achieved by using the *gradient* and *Laplacian* operators. The gradient ∇f is the first order derivative operator and is given for an image *f(x,y)* as

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} \tag{2.7}$$

while the Laplacian $\nabla^2 f$ is the second order derivative operator and is given for an image *f(x,y)* as

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} \\ \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{2.8}$$

Since the gradient vector points in the direction of the maximum rate of change of *f(x,y)*, its magnitude and direction are important in edge detection. An example of a gradient operator is the *Sobel* and *Canny* operators [GoWo92]. The Canny operator differs from the other edge-detection methods in that it uses two different thresholds (to detect strong and weak edges), and includes the weak edges in the output only if they are connected to strong edges. The results from the Sobel and Canny edge detetectors are shown in Fig. 2.3.

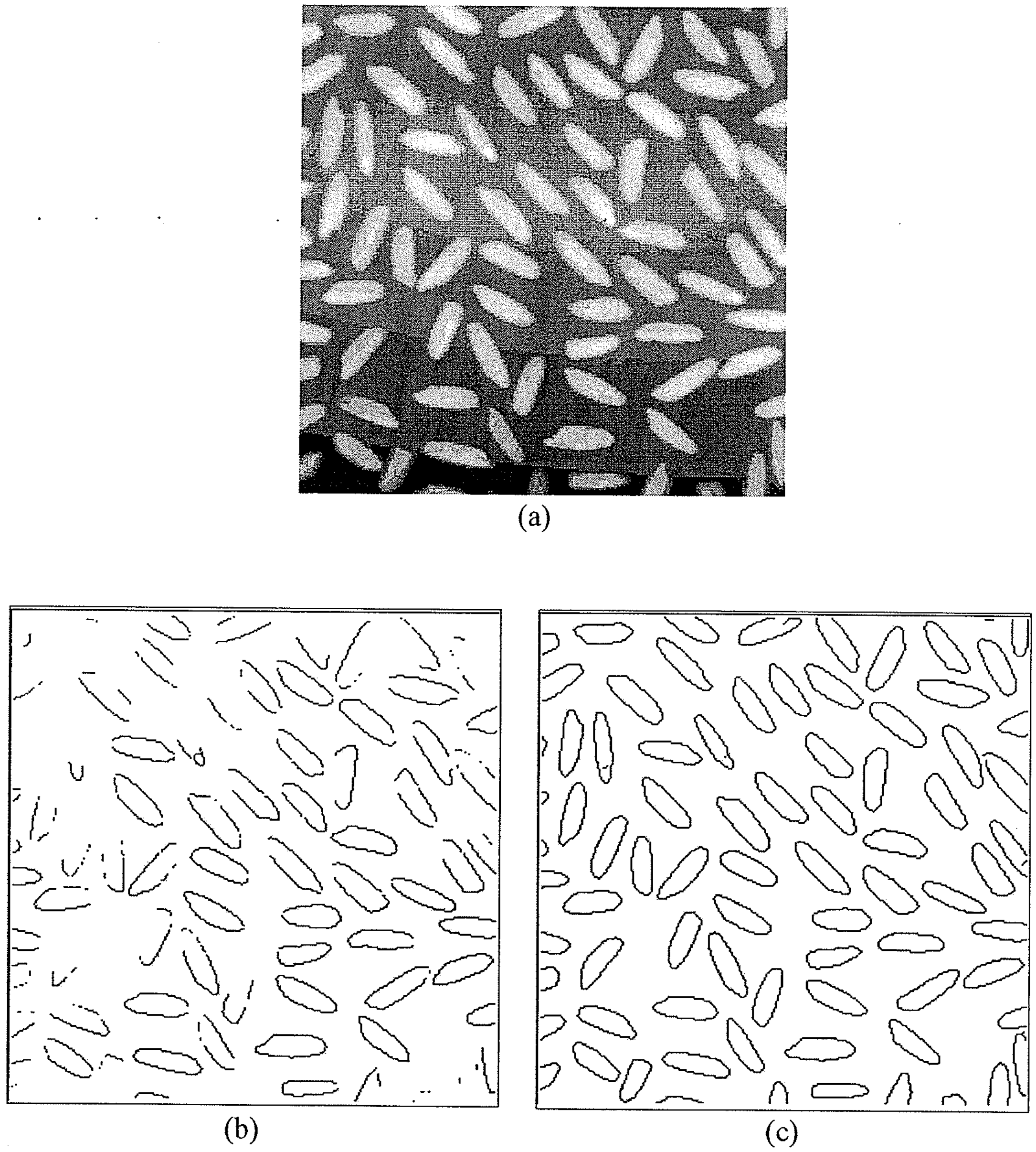


Fig. 2.3 Edge detection using *Sobel* and *Canny* operators. (a) The original *rice* image taken from Matlab.(b) Edge detection using the *Sobel* operator. (c) Edge detection using the *Canny* operator.

2.3.1.4 Edge Linking and Boundary Detection

In the above sections, the techniques associated with locating discontinuities in a digital image were discussed. In an ideal world of digital imaging, these techniques should distinguish the pixels lying only on the boundary between regions. In the real world, the sets of pixels determined by these techniques seldom characterize a boundary completely due to the effects of noise and nonuniform image illumination. Due to this reason, edge detection algorithms are often followed by linking or other boundary detection procedures designed to assemble edge pixels into meaningful regions [CoCe01]. The two common techniques of edge linking are as follows.

Gradient Based

This technique involves local processing of images; i.e., analysis of pixels in a small 3x3 or 5x5 neighborhood [Cast96][GoWo92][Russ98]. All points of the detected edge in an image can then be linked via these neighborhoods and formed into a boundary of pixels which share common properties. This linking is accomplished by an analysis of

- The strength of the response of the gradient operator used to produce the edge pixel, and
- The direction of the gradient.

Global Processing

This technique of edge linking involves global processing of the image [Cast96][GoWo92][Russ98]. This technique is subdivided into two major categories:

- Curve-fitting analysis, and
- Graph-theory techniques.

In the curve-fitting global analysis, points are linked by determining whether they lie on a specified curve. This means that for any *m* points in an image, subsets of these points that lie on straight lines must be selected. These subsets are calculated in a computationally efficient manner, using the Hough transform, while curve fitting is applied on the resulting points. Here, the concept of continuity is based upon a gap measurement between similar pixels to determine if the distance exceeds a certain threshold.

In global processing for the edge linking [Cast96][GoWo92][Russ98], edge segments are represented in the form of a graph which can be traversed and searched in order to the determine minimum cost path between significant edges. The determination of this minimum cost path is done with algorithms such as Djikstra's algorithm which is robust and handles noise well, but is more computationally intensive than the Hough transform approach [GoWo92].

2.3.2 Multilevel Thresholding of Histogram

If the histogram of an image *f(x,y)* were given, a threshold *T* could be specified that would separate the modes of the histogram [Cast96][GoWo92][Russ98]. This process could then be used to extract the objects of interest from those of the background. By choosing different values of *T*, the histogram could be used to pinpoint the class of objects in the image. This process is known as multilevel thresholding. If the number of histogram modes is large, then this technique becomes less reliable since the task of establishing multiple thresholds that isolate a region of interest is nontrivial.

If the image *f(x,y)* or region of interest in *f(x,y)* is noisy, then the associated histogram contains noise as well. In order to overcome this, the noisy histogram can be smoothed by convolution or by a curve-fitting procedure. This smoothing may shift the position of the minimum point on the graph, but the peaks are easy to locate and are stable. To make this process more reliable, a histogram of pixels having a gradient magnitude in the top ten percent can be created. This new histogram would eliminate the large number of interior and exterior pixels from consideration and therefore simplify the segmentation process.

2.3.3 Region Growing

Region growing is an approach to image segmentation in which neighboring pixels are examined and added to a region class if no edges are detected [Cast96][GoWo92][Russ98]. The simplest illustration of this process is called *pixel aggregation*, which starts with a set of seed points and these seeds grow in to regions. This process is iterated for each boundary pixel in the region. If adjacent regions are found, a region-merging algorithm is used in which weak edges are dissolved and strong edges are left intact.

There are several advantages of region growing over conventional segmentation techniques. Unlike the Gradient and Laplacian methods, the borders of regions found by region growing are perfectly thin since pixels are only added to the exterior of a region and are connected with each other. The algorithm is also stable in handling noise. A given region never contains too much of the background as long as the parameters are correctly defined. Other techniques that produce connected edges, like boundary tracking [CoCe01], are unstable in comparison. The most important aspect of the region growing approach is that membership in a region can be based on multiple criteria such as low gradient or gray level intensity value. The disadvantage to region growing is that the technique is computationally expensive.

2.3.4 Split and Merge

As an alternative to growing image regions from a set of seed points, a split and merge algorithm subdivides an image into an arbitrary set of disjointed regions and then merges or splits the regions depending on certain conditions [Cast96][GoWo92][Russ98]. The sub-division of the image is into four quadrants which are examined and compared to adjacent ones. If a region or subregion is similar (having same pixel value range), they are merged together. If adjacent regions are heterogeneous, then the splitting and comparing process is repeated accordingly. In the limit this procedure will continue until the level of the individual pixel is reached. This particular technique is computationally attractive since complete segmentation can be achieved in a finite number of iterations.

2.3.5 Clustering

This technique involves a search for pixel values in the given image that occur frequently and are separated by ranges of pixel values that occur less frequently [Cast96][GoWo92][Russ98]. Grouping these pixels together will cluster these pixels into natural classes which would show up as peaks on the histogram. If these peaks are then separated by a minimum Euclidean distance [RoAd90], the image can be segmented using thresholding techniques mentioned in the previous section 2.3.2. If two or more features are to be extracted from the image, the clusters can be created and their peaks separated by partitioning processes such as [FeSS80].

2.3.6 Morphology

Morphology is a tool for extracting image components that are useful in the representation and description of region shape, such as boundaries and skeletons [Cast96][CoCe01][GoWo92][Russ98]. This procedure is based upon the mathematical concept of set theory, and is a powerful approach for various image processing tasks. In this technique the sets in mathematical morphology represent the shapes of objects in an image.

2.3.7 Other Advanced Techniques

Some other advanced techniques also exist for image segmentation. These techniques mainly involve wavelets and fractals. Readers are encouraged to see [GoWo02][MaZh92] for wavelet-based techniques and [Barn00][Pent84][TuAB98] for a detailed discussion on fractal based techniques. The topic of fractals and wavelets will be discussed in the subsequent chapters.

2.4 Summary

This chapter presented a brief background on standard image segementation techniques and objects in images. Different type of object boundaries were discussed and the issue of quality measures for shape description methods was presented. It was also mentioned that for certain type of object boundaries standard modelling techniques do not work and they require more sophisticated techniques based on fractal and multifractal. A detailed background on fractal and multifractals will be presented in the next chapter.

CHAPTER III

BACKGROUND ON FRACTALS, MULTIFRACTALS, AND ITERATED FUNCTION SYSTEMS

3.1 Introduction

Fractal geometry was introduced by Benoit Mandelbrot who defined a *fractal* as a set for which the Hausdorff-Besicovitch dimension strictly exceeds the topological dimension [Mand82]. Since then the fractality was studied by many authors who proposed more accurate definitions. The fractal geometry deals with objects which cannot be handled by the standard goemetry, because standard geometry only involves topological dimensions. Fractal geometry dealing with spatial fractals, provides both descriptions and mathematical models for a variety of seemingly complex forms and patterns either existing in nature or artificially generated on computers. Fractal geometry plays an important role in modelling and animating natural phenomena and engineering process, and in music and art.

This thesis presents a new approach to model object boundaries using techniques involving fractal geometry. This chapter explains the essential topics for such techniques that are later used in the thesis for boundary analysis and reconstruction. This chapter first presents some well known fractal objects and overviews some of the many fractal dimensions characterizing them [Kins94a][Kins94b]. The chapter presents also an introduction to the multifractal analysis by defining the Rényi fractal dimension spectrum. Although the reconstruction of a fractal object can be done with many methods, this chapter focuses on the IFS and the *midpoint displacement* (MDP) algorithms.

3.2 Fractal Objects and Dimensions

3.2.1 Fractal Objects

Despite the complexity of a fractal object, it can usually be constructed, from simple rules. This can be seen easily from the mathematical fractals such as the Minkowski curve and the Koch curve in which only an initiator and a generator are necessary for their generation. These strict fractal objects are self-similar [Kins95] and are presented in the following description.

The Minkowski curve is produced according to the following rule:

(a) Begin with a unit straight line as initiator (although it may be anything else).

(b) Generate a curve containing eight scaled down initiators (the generator) with a scaling of 1/4 each, as shown in Fig. 3.1.

(c) Once the curve is constructed at step $k-1$, the object at step k is obtained by scaling it by 1/4 and replacing every segment in the object at step $k-1$.

The Minkowski curve is the limit curve, when $k \to \infty$. This limiting curve, although produced by simple rules has complicated mathematical characteristics as its length (at the limit) is infinite.

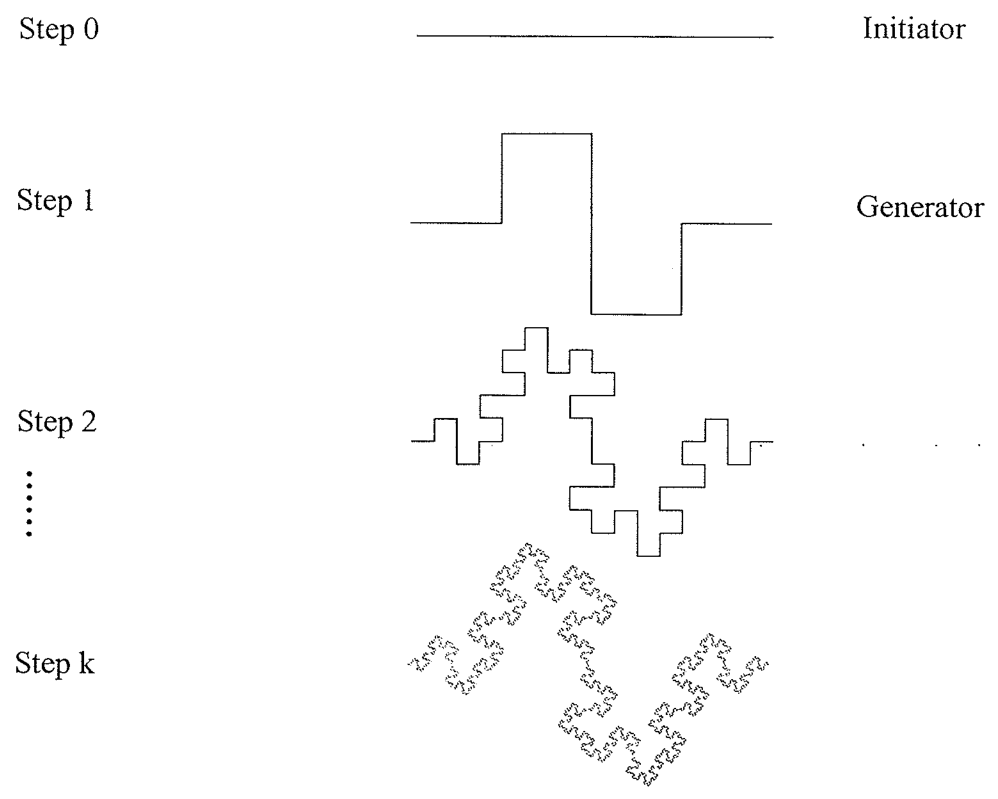


Fig. 3.1 Generation of the Minkowski curve (after [Kins96]) where k = 6.

The *Koch curve* is obtained with the same initiator as in the Minkowski curve production, but using the generator as illustrated in Step 1 of Fig. 3.2. Here, the initiator is scaled down by a factor $r = 1/3$ and replaced by 4 copies of its scaled version. At the *k*th iteration, the curve is the union of 4^k segments, each having a 3^{-k} length. Therefore, the length of the Koch curve is infinite. Notice also that the koch curve is continuous everywhere, but nowhere differentiable. The generator has only 3 points where the curve is not differentiable, but this number, increases to 15 at Step 2, then to 63 at step 3 and to infinity at the limit.

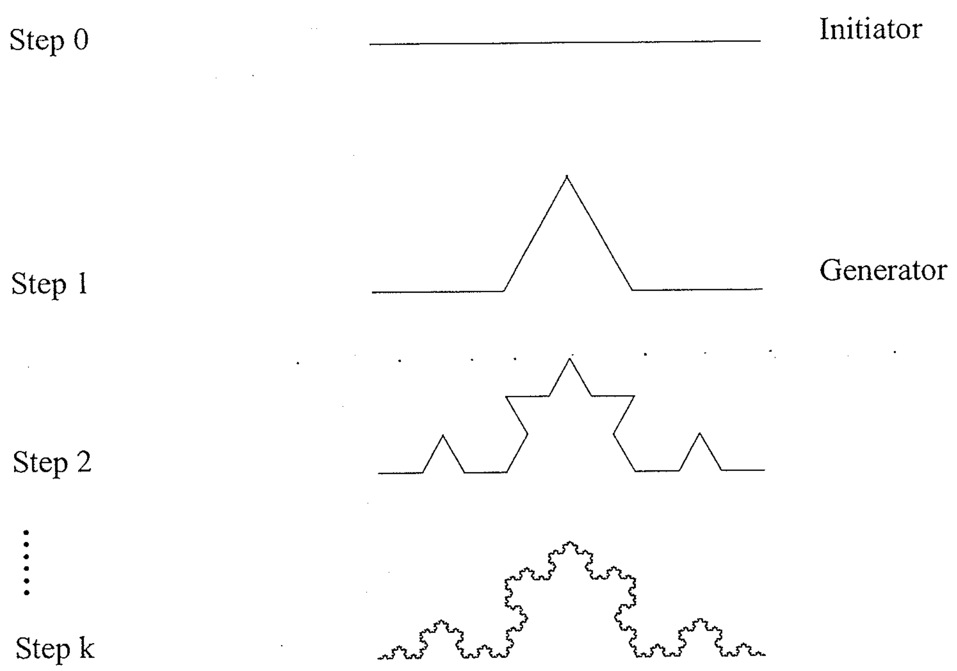


Fig. 3.2 Generation of the Koch curve (after [Kins96]) where k = 6.

This observed irregularity of both Minkowski and Koch curves can be characterized by a scalar called *fractal dimension* [Kins96]. In the following, a characterization of objects that are strict mono fractals and not a combination of fractals is presented.

3.2.2 Fractal Dimensions

There are many fractal dimensions. Kinsner has presented more than 21 fractal dimensions in a unified framework [Kins94a], where the dimensions are classified into morphological, entropy, variance, and spectral dimensions. In this section, the *Hausdorff* *dimension*, the *similarity dimension* and the *box-counting dimension,* the *variance fractal dimension,* and the *power spectrum dimension* are discussed.

Hausdorff Dimension

Let A be a subset of $\Re^d$. For a given $r > 0$, a^n **r-cover** of A is any countable collection of subsets $\{U_i\}$ such that $0 < |U_i| \le r$, where $|U_i| = \sup\{|x-y| : x, y \in |U_i|\}$ and $A \subset \bigcup_{i=1}^{\infty} U_i$. The Hausdorff (or Hausdorff-Besicovitch) dimension of A is defined as

$$D_H = \sup\{s:\mu^s(A) = \infty\} = \inf\{s:\mu^s(A) = 0\} \tag{3.1}$$

where

$$\mu^s(A) = \liminf_{r \to 0}\left\{\sum_{i=1}^{\infty} |U_i|^s : \{U_i\}\textbf{ is a r-cover of A}\right\} \tag{3.2}$$

Similarity Dimension

A self-similar set is made up with N(r) distinct copies of itself, each of which has been scaled down by a ratio *r* in all coordinates. The *similarity fractal dimension* D_S is given by

$$D_S = \frac{\log N(r)}{\log(1/r)} = \frac{\log(\textbf{ number of self-similar copies})}{\log(\textbf{ magnification factor})} \tag{3.3}$$

The Box-Counting Dimension

Let $N_r(A)$ be the minimum number of the covering set (volume elements or vels [Kins94]) intersected by the object A. The box-counting fractal dimension is defined by

$$D_{BC}(A) = \lim_{r \to 0} \frac{\log N_r(A)}{\log(1/r)} \tag{3.4}$$

Variance Fractal Dimension

It is known [Kins94a] that for a Brownian motion (or more generally, a fractional Brownian motion) $B(t)$, the increment $\Delta B = B(t_2) - B(t_1)$ exhibits the following power law

$$\mathrm{Var}[\Delta B] \propto |\Delta t|^{2H} \tag{3.5}$$

where $\Delta t = |t_2 - t_1|$, and H is called the *Hurst exponent* and is given by

$$H = \lim_{\Delta t \to 0} \frac{\log[\mathrm{Var}(\Delta B)]}{2\log(\Delta t)} \tag{3.6}$$

For an embedding Euclidean dimension E, the variance dimension is given by

$$D_\sigma = E + 1 - H \tag{3.7}$$

Power Spectrum Dimension

While the variance dimension can characterize the fractional Brownian motion, the power spectrum dimension is well adapted to a large class of signals having the following power spectrum density P

$$\hat{P} = \frac{c}{f^{\beta}} \tag{3.8}$$

where c is a constant and β is the spectral exponent.

For an embedding Euclidean dimension E, the spectral exponent β is related to the Hurst exponent by [Kins96]

$$\beta = 2H + E \tag{3.9}$$

The power spectrum dimension (also called the Fourier dimension) is given by [Kins96]

$$D_{\beta} = E + 1 - H = E + 1 - \frac{\beta - E}{2} = \frac{3E + 2 - \beta}{2} \tag{3.10}$$

In practice, an estimate for β can be obtained by minimizing the least square error between the power spectrum model in Eq. 3.9 and the actual power spectrum P_i of a given signal of size N (obtained by using the FFT); i.e.,

$$\beta = \frac{N\sum_i(\log P_i)(\log|f_i|) - \sum_i \log|f_i|\left(\sum_i \log P_i\right)}{\left(\sum_i \log|f_i|\right)^2 - N\sum_i(\log|f_i|)^2} \tag{3.11}$$

Table 3.1 shows the dimension values of the pure fractal objects described in Sec. 3.1.1. The similarity fractal dimensions are easily computed for the Koch and the Minkowski curves as they are self-similar, and are given by

$$D_S(\mathbf{Minkowski}) = \lim_{k\to\infty}\frac{\log 8^k}{\log 4^k} = 1.5 \tag{3.12}$$

$$D_S(\mathbf{Koch}) = \lim_{k\to\infty}\frac{\log 4^k}{\log 3^k} \approx 1.262 \tag{3.13}$$

Table 3.1: Fractal dimensions for the Koch curve and the Minkowski curve.

Dimension	Koch Curve	Minkowski Curve
Hausdorff D_H	1.262	1.500
Similarity D_S	1.262	1.500
Box-Counting D_{BC}	1.2663	1.5446

Usually, objects in nature are not single fractal objects, but mixture of monofractal objects. The single fractal dimensions are not adequate in charactesizing such object. For a complete description of such objects, the *multifractal analysis* (MA) presented in the following section has proven to be a very powerful tool.

3.3 Multifractals

While a fractal refers to a set (or an object), a multifractal is a combination of many single fractal objects. Multifractal dimensions are an extension of single fractal dimensions and can characterize complex multifractal structure.

The Rényi Fractal Dimension Spectrum

The Rényi entropy H_q is a generalization of the Shannon entropy and is given for any probability distribution $P = (p_j)$ by [CoTh91][Kins96]

$$H_q(P) = \frac{1}{1-q}\log\sum_{j=1}^{N(r)} p_j^{\,q} \qquad -\infty < q < \infty \tag{3.14}$$

where $N(r)$ is the number of non-empty vels with side length r intersected by the fractal object. Ths leads to the Rényi fractal dimension spectrum denoted D_q and is given by [Kins96]

$$D_q = \lim_{r\to 0}\frac{1}{1-q}\frac{\log\sum_{j=1}^{N(r)} p_j^{\,q}}{\log r} \tag{3.15}$$

The Rényi dimension spectrum is a bounded monotonically decreasing function of q. For a single fractal, D_q is constant for every $q > 0$. Also, for many particular values of q, D_q is reduced to some known fractal dimensions. More precisely, one can show that D_0 is simply the box-counting dimension, D_1 is called the *information dimension*, and D_2 is the *correlation dimension* [Kins94]. The bounds on D_q can also be found for $q = \pm\infty$ [Kins94]. Thus, the Rényi fractal dimension spectrum is always bounded.

3.4 Generating Fractals

This thesis is concerned with compression of the image boundaries using fractal geometry and multifractal measures. In the process of compression-decompression, the fractal curves are generated from the control points and the object's fractal dimension. This is done with two methods: The MPD algorithm and the IFS to be presented in the following section.

3.4.1 Midpoint Displacement

The *random midpoint displacement* (MPD) is the most popular way to produce Brownian motion [PeJS91][Kins96]. Let $X(t),\ \ t \in [0, 1]$ be the process to be produced such that

$$\mathrm{var}(X(t_2)-X(t_1)) = |t_2 - t_1|\sigma^2 \quad \textbf{for } (0 \le t_1 \le t_2 \le 1) \tag{3.16}$$

The MPD is described with the following steps.

(a) Set $X(0) = 0$ and select $X(1)$ as a sample of a Gaussian random variable with mean 0 and variance σ^2.

(b) $X\left(\frac{1}{2}\right)$ is taken to be the average of $X(0)$ **and** $X(1)$ plus an offset D_1; i.e., $X\left(\frac{1}{2}\right) = \frac{1}{2}(X(0)+X(1))+D_1$, where D_1 is a Gaussian random number with mean 0 and variance $\frac{1}{4}\sigma^2$.

(c) In the second step $X\left(\frac{1}{4}\right)$ **and** $X\left(\frac{3}{4}\right)$ are reconstructed from the averages of $X(0)$ **and** $X\left(\frac{1}{2}\right)$, and $X\left(\frac{1}{2}\right)$ **and** $X(1)$ respectively; i.e.,

$$\begin{cases} X\left(\frac{1}{4}\right) = \frac{1}{2}\left(X(0)+X\left(\frac{1}{2}\right)\right)+D_2 \\ X\left(\frac{3}{4}\right) = \frac{1}{2}\left(X\left(\frac{1}{2}\right)+X(1)\right)+D_2 \end{cases} \tag{3.17}$$

where D_2 is a Gaussian random number with mean 0 and variance $\frac{1}{8}\sigma^2$.

(d) The process is continued and in the *n*th step, $X\left(\frac{1}{2^n}\right), X\left(\frac{3}{2^n}\right), X\left(\frac{5}{2^n}\right), \ldots, X\left(1-\frac{1}{2^n}\right)$ are constructed by averaging and adding a Gaussian random number D_n with mean 0 and variance $\frac{1}{2^{n+1}}\sigma^2$.

The MPD algorithm has the advantage that it is simple to implement and can be generalized to higher dimensions. It is not, however, very efficient in producing fractal signals other than Brownian motion (BM) and any other fractional Brownian motion (fBm). The iterated function systems algorithm presented in the following section does not assume any particular distribution.

3.4.2 Iterated Function Systems

Iterated function system (IFS) is the name given by Barnsley and Demko to define a set of contractive transformations, $\{w_i, i = 1, \ldots, N\}$ on $\mathbf{R}^d$ [BaDe85]. For $d = 2$, the affine transformations allow all possible two dimensional transformations. Therefore, the study is restricted to the affine transformations (the IFS is then called *affine IFS*) given by

$$w_i\begin{pmatrix} t \\ x \end{pmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix}\begin{bmatrix} t \\ x \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix} \qquad (3.18)$$

The transformation of a compact subset $B \subset \mathbf{R}^d$ with all the maps is denoted by

$$\mathbf{W}(B) = \bigcup_{i=1}^{N} w_i(B) \qquad (3.19)$$

It can be shown that W is also a contractive transformation on the metric space of compact subsets equipped with the Hausdorff distance. (The Hausdorff metric between two compact subsets *A* and *B* is the maximum separation of the farthest elements of *A* or *B* from the opposite set). Furthermore, the fixed point theorem states that W has a unique fixed point A which can be reached from any initial compact subset $B \subset \mathbf{R}^d$ [Barn00] by

$$A = W(A) = \lim_{k \to \infty} W^k(B) \tag{3.20}$$

This fixed point A of W is called the *attractor* of the IFS.

In the present case, the interest is in approximating curves which are object boundaries from a collection of control points $\{(t_i, x_i), i = 0, 1, \ldots, N\}$, where $t_0 < t_1 < \ldots < t_N$. These curves are often fractals (continuous but nowhere differentiable) and can be modeled by an IFS whose attractor is the graph of a function f interpolating the control points; i.e., $f(t_i) = x_i$ for each i. Such functions are called *fractal interpolation functions* (FIF). It is shown [Barn00] that this problem has a solution given by

$$w_i\begin{pmatrix} t \\ x \end{pmatrix} = \begin{bmatrix} a_i & 0 \\ c_i & d_i \end{bmatrix}\begin{bmatrix} t \\ x \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix} \tag{3.21}$$

where the parameters $a_i, c_i, d_i, e_i,$ **and** f_i are determined by the following conditions

$$a_i = (t_i - t_{i-1})/(t_N - t_0) \tag{3.22}$$

$$e_i = (t_N t_{i-1} - t_0 t_i)/(t_N - t_0) \tag{3.23}$$

$$c_i = ((x_i - x_{i-1}) - d_i(x_N - x_0))/(t_N - t_0) \tag{3.24}$$

$$f_i = \frac{((t_N x_{i-1} - t_0 x_i) - d_i(t_N x_0 - t_0 x_N))}{(t_N - t_0)} \tag{3.25}$$

$$|d_i| < 1 \tag{3.26}$$

The free parameters d_i are called *vertical scaling factors* and are related to the fractal dimension D of the curve by

$$\sum_{i=1}^{N} a_i^{D-1}|d_i| = 1 \qquad (3.27)$$

It follows that given control points and the fractal dimension D, one can reconstruct the fractal object from the IFS given by Eq. 3.20. This operation is called IFS decompression (or decoding) and will be described after the next two subsections.

Hidden IFS

For more complicated (high curvature) boundaries, the generalized fractal interpolation functions which are attractors of *hidden* IFS are used. Here, the data points are 3-dimensional $\{(t_i, x_i, y_i), i = 0, 1, \ldots, N\}$ and the curve to be approximated has $\{(x_i, y_i), i = 0, 1, \ldots, N\}$ as control points. The IFS model is written by

$$w_i\begin{bmatrix} t \\ x \\ y \end{bmatrix} = \begin{bmatrix} a_i & 0 & 0 \\ c_i & d_i & 0 \\ c_{yi} & 0 & d_{yi} \end{bmatrix}\begin{bmatrix} t \\ x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ f_{yi} \end{bmatrix} \qquad (3.28)$$

where a_i, c_i, d_i, e_i, **and** f_i are the same as in Eqs. 3.21 to 3.25, c_{yi} **and** f_{yi} are given by replacing x_i **by** y_i respectively in Eq. 3.23 and Eq. 3.24, while d_{yi} are free parameters such that $|d_{yi}| < 1$.

Polar IFS

By transforming the control points $\{(t_i, x_i), i = 0, 1, \ldots, N\}$ into *polar coordinates* $\{(\theta_i, r_i), i = 0, 1, \ldots, N\}$ where

$$\begin{cases} t_i = r_i \cos\theta_i \\ x_i = r_i \sin\theta_i \end{cases} \tag{3.29}$$

then there exists a simple closed curve r $[\mathbf{0,2\pi}] \rightarrow \Re^2$ with $r(\theta_i) = (t_i, x_i)$ which arises as attractor of a certain IFS. The parameter *r* is called *polar fractal interpolation function* [DaDr99].

IFS Decompression

The IFS decoding process consists of generating the attractor A from the code given by the transformations w_i. For that purpose, two methods were investigated: the *deterministic algorithm* (DA) and *the random iteration algorithm* (RIA) [Barn00]. Starting from a non-empty compact set A_0, the deterministic algorithm computes the sequence $A_k = W^k(A_0)$ up to the attractor. While the convergence can be rapid, the DA results in a large amount of computation. The random algorithm applies at each iteration only one of the transformations $\{w_i\}$, randomly chosen with a predefined set probabilities $\{p_i\}$, often calculated using the formula (by default)

$$p_i = \frac{|a_i d_i - b_i c_i|}{\sum_{j=1}^{N} |a_j d_j - b_j c_j|} \tag{3.3}$$

Figure 3.3 displays the Sierpinski triangle for different iteration numbers, by applying both the deterministic algorithm and the random iteration algorithm with the default probabilities. The DA takes only 20 iterations to converge. Each iteration, however, takes many seconds to be performed because many points may be computed again and again. On the other hand, the RIA performs ten thousands more iterations in only a few seconds.

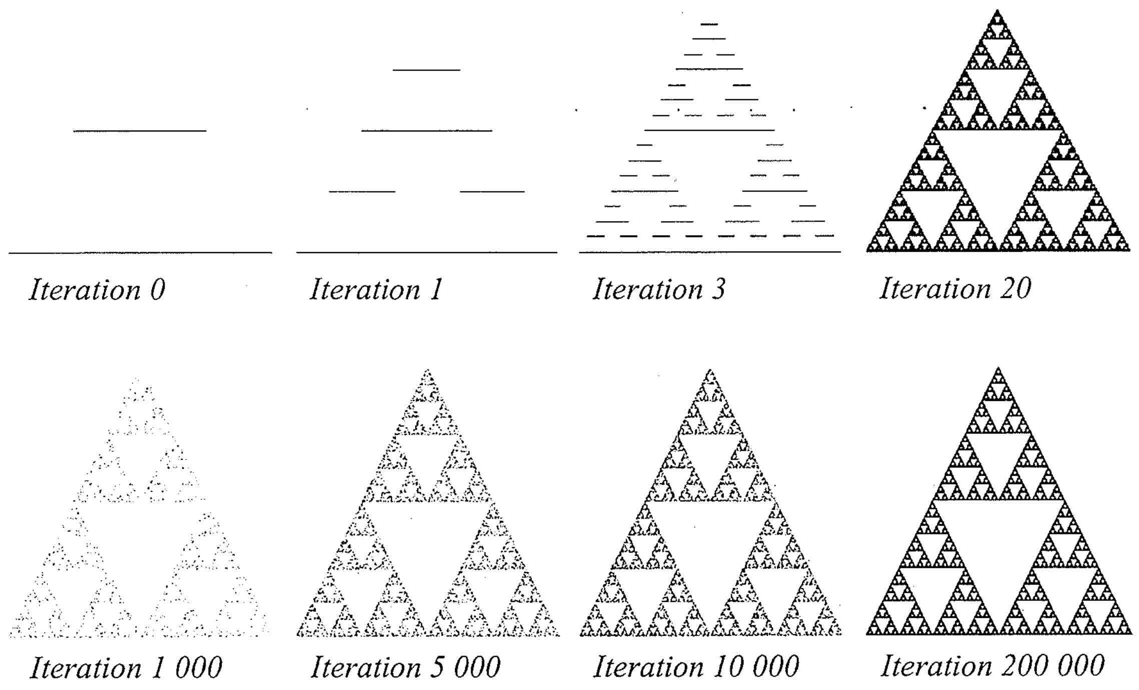


Fig. 3.30 Sierpinski triangle generated by IFS decoding. Top row: using Deterministic Algorithm, and down: using Iterated Random Algorithm.

3.5 Summary

This chapter presented some fractal objects and many definitions of fractal definitions. Although there exist many fractal dimension definitions, no single dimension can be, by itself, a characteristic of discrimination between complex objects. Instead, the Rényi fractal dimension spectrum can characterize the complexity of objects completely.

The MPD and the IFS algorithms were also presented, and it was shown how to produce fractal objects using these two techniques. While the MPD technique assumes a simple Brownian motion or a fractional Brownian motion, the IFS can be used with more degrees of freedom.

CHAPTER IV

MULTIRESOLUTION AND WAVELET TECHNIQUES

4.1 Introduction

In this chapter, a brief introduction to the concepts of multiresolution and wavelet analysis is provided. Multiresolution decomposition provides a scale-invariant interpretation of the image. The scale of an image varies with the distance between the scene and the optical center of the camera [Mall89]. When the image scale is changed, the interpretation of the scene should not change. A multiresolution representation can be partially scale-invariant if the sequence of resolution parameters $(r_j)_{j \in z}$ varies exponentially. Let us suppose that there exists a resolution step $\lambda \in \mathbf{R}$ such that for all integers j, $r_j = \lambda^j$. If the camera gets λ times closer to the scene, each object of the scene is projected on an area λ^2 times bigger in the focal plane of the camera. Hence, the object is measured at a resolution λ times bigger. Hence, the details of his new image at the resolution λ^j correspond to the details of the previous image at the resolution λ^{j+1}. Rescaling the image by λ translates the image details along the resolution axis. Here, resolution axis is the axis at which the scale changes [Mall89].

Multiresolution representation provides a simple hierarchical framework for interpretation of image information. At different resolutions, the details of an image generally characterize different physical structures of the scene. At a coarse resolution, these details correspond to the larger structures which provide the image 'context'. It is, therefore, natural to analyze the image details at a coarse resolution first and then gradually increase the resolution. Such a coarse-to-fine strategy is useful for pattern recognition algorithms [MaHi80].

Wavelets are mathematical functions that transform data into different frequency components, and then analyse each component with a resolution matched according to its scale. They have advantages over traditional Fourier methods in analyzing physical situations where the signal contains discontinuities or singularities and sharp spikes. Wavelets were developed independently in the fields of mathematics, quantum physics, electrical engineering, and seismic geology. Interchanges between these fields during the last 15 years have led to many new wavelet applications such as image compression, turbulence, human vision, radar, and earthquake prediction.Wavelets are one of the fundamental mathematical tools used in the field of multiresolution anlaysis.

This thesis presents a new approach to model object boundaries using techniques involving wavelets. This chapter explains the essential topics for such techniques that are later used in the thesis for boundary analysis and reconstruction. In the following, an introduction to wavelet transform is presented. Then a study of singularities using wavelet transform follows.

4.2 Notation and Basic Relations

This notation follows [Mall98]. The set of integers and real numbers are represented by **Z** and **R** respectively. $\mathbf{L}^2(\mathbf{R})$ denotes the Hilbert space of measurable, square-integrable 1-D functions $f(x)$. For $f(x) \in \mathbf{L}^2(\mathbf{R})$ and $g(x) \in \mathbf{L}^2(\mathbf{R})$, the inner product of $f(x)$ and $g(x)$ is written as

$$\langle g(u), f(u) \rangle = \int_{-\infty}^{\infty} g(u) f(u) du \tag{4.1}$$

The norm of $f(x)$ in $\mathbf{L}^2(\mathbf{R})$ is given by

$$\|f\|^2 = \int_{-\infty}^{\infty} |f(u)|^2 du \tag{4.2}$$

Convolution of two functions $f(x) \in \mathbf{L}^2(\mathbf{R})$ and $g(x) \in \mathbf{L}^2(\mathbf{R})$ is given as

$$f * g(x) = \int_{-\infty}^{\infty} f(u) g(x-u) du \tag{4.3}$$

Fourier transform of $f(x) \in \mathbf{L}^2(\mathbf{R})$ is written as

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(x) e^{-j\omega x} dx \tag{4.4}$$

$l^2(\mathbf{R})$ is the vector space of squarable-summable sequences; i.e.,

$$l^2(\mathbf{R}) = \{(\alpha_i)_{i \in Z} \in \mathbf{R} \ / \ \sum_{-\infty}^{\infty} |\alpha_i|^2 < \infty \} \tag{4.5}$$

Let $\mathbf{L}^2(\mathbf{R}^2)$ be the vector space of measurable, square-integrable two dimensional functions $f(x, y)$. For $f(x, y) \in \mathbf{L}^2(\mathbf{R}^2)$ and $g(x, y) \in \mathbf{L}^2(\mathbf{R}^2)$, the inner product of $f(x, y)$ with $g(x, y)$ is written as

$$\langle f(x, y), g(x, y) \rangle = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y), g(x, y) dx dy \tag{4.6}$$

The Fourier transform of $f(x, y) \in \mathbf{L}^2(\mathbf{R}^2)$ is written as

$$f(\omega_x, \omega_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j(\omega_x x + \omega_y y)} dx dy \tag{4.7}$$

4.3 Wavelet Transform

Grossmann and Morlet [GrMo84] defined the wavelet transform as a decomposition of a signal into a family of functions which are the translation and dilation of a unique function $\psi(x)$, called *mother wavelet.* The corresponding wavelet family is given by

$$(\sqrt{s}\psi(s(x - u)))_{(s, u) \in \mathbf{R}^+ \times \mathbf{R}} \tag{4.8}$$

where $\mathbf{R}^+$ is the set of positive real numbers. The wavelet transform of a function $f(x) \in \mathbf{L}^2(\mathbf{R})$ is then defined as

$$Wf(s, u) = \int_{-\infty}^{\infty} f(x)\sqrt{s}\psi(s(x-u))dx \tag{4.9}$$

Let the dilation of $\psi(x)$ by a factor s be denoted as

$$\psi_s(x) = \sqrt{s}\psi(sx) \tag{4.10}$$

Then the wavelet transform can be seen as an inner product in $\mathbf{L}^2(\mathbf{R})$; i.e.,

$$Wf(s, u) = \langle f(x), \psi_s(x-u)\rangle$$

Hence, wavelet transform corresponds to a decomposition of $f(x)$ on the family of functions $(\psi_s(x-u))_{(s,u)\in\mathbf{R}^+\times\mathbf{R}}$. In this thesis, both the signal $f(x)$ and the wavelet $\psi(x)$ are assumed to have real values. In order to reconstruct $f(x)$ from its wavelet transform, the Fourier transform $\hat{\psi}(\omega)$ of $\psi(x)$ must satisfy

$$0 < C_\psi = \int_0^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{\omega} d\omega < +\infty \tag{4.11}$$

This condition implies that $\hat{\psi}(0) = 0$, and that $\hat{\psi}(\omega)$ is small enough in the neighbourhood of ω= 0. Thus, the function $\psi(x)$ can be interpreted as the impulse response of a bandpass filter. The reconstruction of $f(x)$ from $Wf(s, u)$ is given by

$$f(x) = \frac{1}{C_\psi}\int_{-\infty}^{+\infty}\int_0^{+\infty} Wf(s, u)\psi_s(x-u)dsdu \tag{4.12}$$

For normalization purpose, it is assumed that the energy of $\psi(x)$ is equal to 1. Let $\tilde{\psi}_s(x) = \psi_s(-x)$. Hence, the wavelet transform at a point u and a scale s can be written as

$$Wf(s, u) = f_* \tilde{\psi}_s(u) \tag{4.13}$$

Therefore, a wavelet transform can be viewed as filtering $f(x)$ with a bandpass filter whose impulse response is $\tilde{\psi}_s(x)$. Now, the Fourier transform of $\psi(x)$ is given by

$$\hat{\psi}_s(\omega) = \frac{1}{\sqrt{s}}\hat{\psi}\left(\frac{\omega}{s}\right)$$

In contrast with the Gabor transform (or windowed Fourier transform), here the resolution of the wavelet transform varies with the scale (parameter s). Since $\psi(x)$ is real, $|\hat{\psi}(\omega)| = |\hat{\psi}(-\omega)|$. Let ω_0 be the center of the passband of $\hat{\psi}(\omega)$ hence,

$$\int_0^{+\infty}(\omega - \omega_0)|\hat{\psi}(\omega)|^2 dw = 0 \tag{4.14}$$

Let σ_ω be the rms bandwidth around ω_0, hence,

$$\sigma_\omega^2 = \int_0^{+\infty}(\omega - \omega_0)^2|\hat{\psi}(\omega)|^2 dw \tag{4.15}$$

The center of the passband of $\hat{\psi}_s(\omega)$ is $s\omega_0$ and its *root mean square* (rms) bandwidth is $s\sigma_\omega$. On the logarithmic scale, the rms bandwidth of $\hat{\psi}_s(\omega)$ is the same for all $s \in \mathbf{R}^+$

[Mall98]. In this way, a wavelet transform decomposes the signal into a set of frequency bands having a constant size on the logarithmic scale.

Let σ_u be the standard deviation of $|\psi(x)|^2$ around zero. It can be shown that the wavelet $\psi_s(x-u_0)$ has an energy concentrated around u_0 within a standard deviation σ_u/s. In the frequency domain, it can be seen that its energy is concentrated around $s\omega_0$ within a standard deviation $s\sigma_\omega$. In the phase-space, the resolution cell of this wavelet is therefore, equal to $[u_0-(\sigma_u/s), u_0+(\sigma_u/s)] \times [s\omega_0 - s\sigma_\omega, s\omega_0 + s\sigma_\omega]$ where the shape of the resolution cell varies with the scale s. When the scale s is small, the resolution is coarse in the spatial domain and fine in the frequency domain. If the scale s increases, the resolution increases in the spatial domain and decreases in the frequency domain

4.4 Continuous Dyadic Wavelet Transform

For most purposes, the wavelet model is not required to keep continuous scale parameters. To allow fast numerical implementations, it is imposed that the scale varies only along the dyadic sequence $(2^j)_{j \in Z}$. Here a review is presented of the main properties of a dyadic wavelet transform and the conditions under which it is complete and stable. A thorough presentation can be found in [Mall98]. We denote by $\psi_j(x)$ the dilation of $\psi(x)$ by a factor 2^{-j}; i.e.,

$$\psi_j(x) = \frac{1}{2^j}\psi\left(\frac{x}{2^j}\right) \tag{4.16}$$

The wavelet transform of $f(x)$ at scale 2^j and at position x is defined by the convolution product

$$W_j f(x) = f * \psi_j(x) \tag{4.17}$$

We refer to the dyadic wavelet transform as the sequence of functions

$$\mathbf{W}f = (W_j f(x))_{j \in \mathrm{Z}} \tag{4.18}$$

and **W** is the dyadic wavelet transform operator.

Now, a brief study of completeness and stability is provided. The Fourier transform of $W_j f(x)$ is

$$\hat{W_j f}(\omega) = \hat{f}(\omega)\hat{\psi}(2^j\omega) \tag{4.19}$$

By imposing that there exists two strictly positive constant A_1 and B_1 such that

$$\forall \omega \in R,\ A_1 \leq \sum_{-\infty}^{+\infty} \left|\hat{\psi}(2^j\omega)\right|^2 \leq B_1 \tag{4.20}$$

we ensure that the whole frequency axis is covered by dilation of $\hat{\psi}(\omega)$ by $(2^j)_{j \in \mathrm{Z}}$ so that $\hat{f}(\omega)$, and thus $f(x)$, can be recovered from its dyadic wavelet transform. The reconstruction wavelet $\chi(x)$ is any function whose Fourier transform satisfies

$$\sum_{j=-\infty}^{+\infty} \hat{\psi}(2^j\omega)\chi(2^j\omega) = 1 \tag{4.21}$$

If property (4.20) is valid, there exists an infinite number of functions $\hat{\chi}(\omega)$ that satisfy (4.21). The function $f(x)$ is recovered from its dyadic wavelet transform with the summation

$$f(x) = \sum_{j=-\infty}^{+\infty} W_j f * \chi_j(x) \tag{4.22}$$

This equation is proved by computing its Fourier transform and inserting (4.19) and (4.21). With the Parseval theorem, we derive a norm equivalence relation

$$A_1\|f\|^2 \leq \sum_{j=-\infty}^{+\infty} \|W_j f(x)\|^2 \leq B_1\|f\|^2 \tag{4.23}$$

This proves that the dyadic wavelet transform is more than complete, but still stable (as B_1/A_1 is closer to 1, it becomes more stable).

A dyadic wavelet transform is more than complete; it is redundant. Moreover, any sequence $(g_j(x))_{j \in \mathbf{Z}}$, with $g_j(x) \in \mathbf{L}^2(\mathbf{R})$, is not necessarily the dyadic wavelet transform of some function in $\mathbf{L}^2(\mathbf{R})$. It can be shown [Kais94], [Mall98] that $(g_j(x))_{j \in \mathbf{Z}}$ is the dyadic wavelet transform of some function in $\mathbf{L}^2(\mathbf{R})$, if and only if there exists a function $K_{l,j}(x)$ (known as reproducing kernel) such that

$$\sum_{l=-\infty}^{+\infty} g_l * K_{l,j}(x) = g_j(x) \qquad ,\forall j \in \mathbf{Z} \tag{4.24}$$

The energy of the kernel $K_{l,j}(x)$ measures the redundancy of the wavelet transform at the scales 2^j and 2^l.

4.5 Multiresolution Analysis

In the case of a *multiresolution analysis* (MRA), it can be shown [Mall89] that there exists a real function $\phi(x)$ whose Fourier transform is an aggregation of $\hat{\psi}(2^j\omega)$ and $\hat{\chi}(2^j\omega)$ at scales 2^j larger than 1; i.e.,

$$|\hat{\phi}(\omega)|^2 = \sum_{j=1}^{+\infty} \hat{\psi}(2^j\omega)\hat{\chi}(2^j\omega) \tag{4.25}$$

It is supposed here that the reconstructing wavelet $\chi(\omega)$ is such that $\hat{\psi}(\omega)\hat{\chi}(\omega)$ a positive, real, even function. It can be proved that property (4.25) implies that the integral of $\phi(x)$ is equal to 1 and, hence, it is a smoothing function (the function ϕ is called *scaling* function). Let S_j be the smoothing operator defined by

$$S_j f(x) = f * \phi_j(x) \tag{4.26}$$

where

$$\phi_j(x) = \frac{1}{2^j}\phi\left(\frac{x}{2^j}\right) \qquad (4.27)$$

If the scale 2^j is larger, then more details of $f(x)$ are removed by S_j. For any scale $2^J > 1$, (4.25) yields

$$|\hat{\phi}(\omega)|^2 - |\hat{\phi}(2^J\omega)| = \sum_{j=1}^{J} \hat{\psi}(2^j\omega)\hat{\chi}(2^j\omega) \qquad (4.28)$$

Hence, the higher frequencies of $S_1f(x)$, which have disappeared in $S_Jf(x)$, can be recovered from the dyadic wavelet transform $(W_Jf(x))_{1 \le j \le J}$ between the scales 2^1 and 2^J.

The importance of the MRA comes from the fact that wavelet analysis can be performed by *quadrature mirror filters* consisting of a low-pass filter associated with the scaling function ϕ, and a high-pass filter associated with the mother wavelet ψ. This makes the wavelet transform more practicable and easier to implement.

4.6 Discrete Dyadic Wavelet Transform

Input signal is measured at a finite scale in practical applications. Hence, the wavelet transform at an arbitrary fine scale cannot be computed. Let the finest resolution be normalized to 1, and assume that the original signal is a discrete sequence

$D = (d_n)_{n \in \mathbf{Z}}$ of finite energy. If there exist two constants $C_1 > 0$ and $C_2 > 0$ such that $\hat{\phi}(\omega)$ satisfies

$$\forall\omega \in R, \quad C_1 \le \sum_{n=-\infty}^{+\infty} \left|\hat{\phi}(\omega + 2n\pi)\right|^2 \le C_2 \tag{4.29}$$

then it can be proved [MaHi80] that there exists a function $f(x) \in \mathbf{L}^2(\mathbf{R})$ (not unique) such that

$$S_1 f(n) = d_n, \qquad \forall n \in \mathbf{Z} \tag{4.30}$$

The input signal can thus be rewritten as $(S_1 f(n))_{n \in \mathbf{Z}}$. Let us denote

$$W_j^d = (W_j f(n + \omega))_{n \in \mathbf{Z}} \tag{4.31}$$

and

$$S_j^d f = (S_j f(n + \omega))_{n \in \mathbf{Z}} \tag{4.32}$$

where ω is the sampling shift that depends only on the wavelet $\psi(x)$. For any coarse scale 2^J, the sequence of discrete signals

$$\{S_j^d(f), (W_{2^j}^d f), 1 \leq j \leq J\} \tag{4.33}$$

is called the discrete *dyadic wavelet transform* of $(S_1 f(n))_{n \in Z}$.

4.7 Wavelet Transform Modulus Maxima (WTMM)

From the discrete wavelet transform, at each scale 2^j, we detect the modulus maxima by finding the points where $|W_j f(n+\omega)|$ is larger than its two closest neighbor values and strictly larger than at least one of them. The abscissa $n+\omega$ and the value $W_j f(n+\omega)$ at the corresponding locations are recorded.

One signal sharp variation produces modulus maxima at different scales 2^j. It is known that the value of a modulus maximum at a scale 2^j measures the derivative of the signal smoothed at the scale 2^j [MaZh92]. Wavelet theory shows that the evolution across scales of the wavelet transform modulus depends on the local Lipschitz regularity of the signal.

4.8 Detection of Singularities using Wavelet Transform

A remarkable property of the wavelet transform is its ability to characterize the local regularity of functions [MaHw92]. In mathematics, local regularity is often measured with Lipschitz exponents.

Definitions:

- Let n be a positive integer and $n \le \zeta \le n+1$. A function $f(x)$ is said to be Lipschitz ζ, at x_0, if and only if there exists two constants A and $h_0 > 0$, and a polynomial of order n, $P_n(x)$, such that for $h < h_0$

$$|f(x_0 + h) - P_n(h)| \le A|h|^{\zeta} \tag{4.34}$$

- The function $f(x)$ is uniformly Lipschitz ζ over the interval $]a,b[$, if and only if there exists a constant A and for any $x \in]a, b[$ there exists a polynomial of order n, $P_n(h)$, such that Eq. (4.34) is satisfied for $x_0 + h \in]a, b[$.
- Lipschitz regularity of $f(x)$ at x_0 is the superior bound of all values ζ such that $f(x)$ is Lipschitz ζ at x_0.
- Function is singular at x_0, if it is not Lipschitz 1 at x_0.

A function $f(x)$ that is continuously differentiable at a point is Lipschitz 1 at this point. If the derivative of $f(x)$ is bounded but discontinuous at x_0, $f(x)$ is still Lipschitz 1 at x_0. It follows from the above definition that $f(x)$ is not singular at x_0. It can be proved that if $f(x)$ is Lipschitz ζ, for $\zeta > n$, then $f(x)$ is n times differentiable at x_0 and the polynomial $P_n(h)$ is the first $n + 1$ terms of the Taylor series of $f(x)$ at x_0. The Lipschitz regularity ζ_0 gives an indication of the differentiability of $f(x)$ but it is more precise. If the Lipschitz regularity ζ_0 of $f(x)$ satisfies $n \le \zeta_0 \le n+1$, then it is known that $f(x)$ is n times differentiable at x_0 but its nth derivative is singular at x_0 and ζ_0 characterizes this singularity.

Theorem (due to [MaZh92]): Let $0 < \zeta < 1$. A function $f(x)$ is uniformly Lipschitz ζ over $]a,b[$ if and only if there exists a constant $K > 0$ such that for all $x \in]a, b[$, the wavelet transform satisfies

$$|W_j f(x)| \le K(2^j)^{\zeta} \tag{4.35}$$

From the above equation we derive that

$$\log_2|W_j f(x)| \le \log_2(K) + \zeta j \tag{4.36}$$

Hence, if the uniform Lipschitz regularity is positive, the amplitude of the wavelet transform modulus maxima should decrease when the scale decreases. On the other hand, if the uniform Lipschitz regularity is negative, the wavelet transform modulus maxima increases. This means that such singularities are more singular than discontinuities. The signal is then viewed as tempered distribution. If the signal has uniform Lipschitz regularity ζ_0 equal to zero, then maxima values of $|W_j f(x)|$ remain constant over a large range of scales. At this stage one might wonder how to choose a particular wavelet to estimate uniform Lipschitz regularity. In the following, this issue is discussed.

4.9 Various Wavelets and their Properties

Here, we discuss various wavelets and their properties. Details regarding different types of wavelets are given in [Daub92]. In the following, we discuss various well known wavelets and their properties.

4.9.1 Haar Wavelet

The Haar wavelet is a simple compactly supported wavelet. It is a special case of Daubechies wavelets [Daub92] with one vanishing moment [MMOP96]. In other words this is the same as *db2*. The Haar wavelet function is shown in Fig. 4.1 and its main properties are illustrated in Table 4.1 [MMOP96].

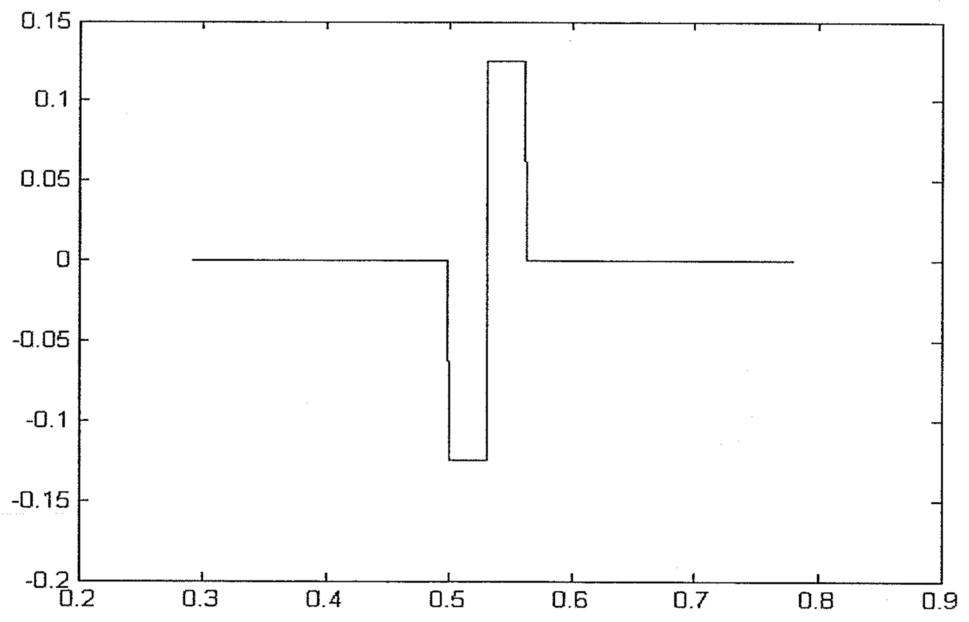


Fig. 4.1 Haar wavelet.

Table 4.1: Properties of Haar wavelet.

Items	Properties
Family	Haar
Short name	haar
Examples	haar is the same as dbl
Orthogonal	yes
Biorthogonal	yes
Compact support	yes
DWT	possible
CWT	possible
Support width	1
Filters length	2
Regularity	haar is not continuous
Symmetry	yes
Vanishing moments	1

4.9.2 Daubechies Wavelets

Daubechies wavelets are compactly supported wavelets with the extremal phase and the highest number of vanishing moments for a given support width. The associated scaling filters are minimum-phase filters. The wavelet function [Daub92 pp. 115, 132, 194, 242] for *db2, db4, db8 and db16* is show in Figs. 4.2a to 4.2d, respectively. These wavelets have vanishing moments 2, 4, 8 respectively [MMOP96]. Their main properties are illustrated in Table 4.2 [MMOP96].

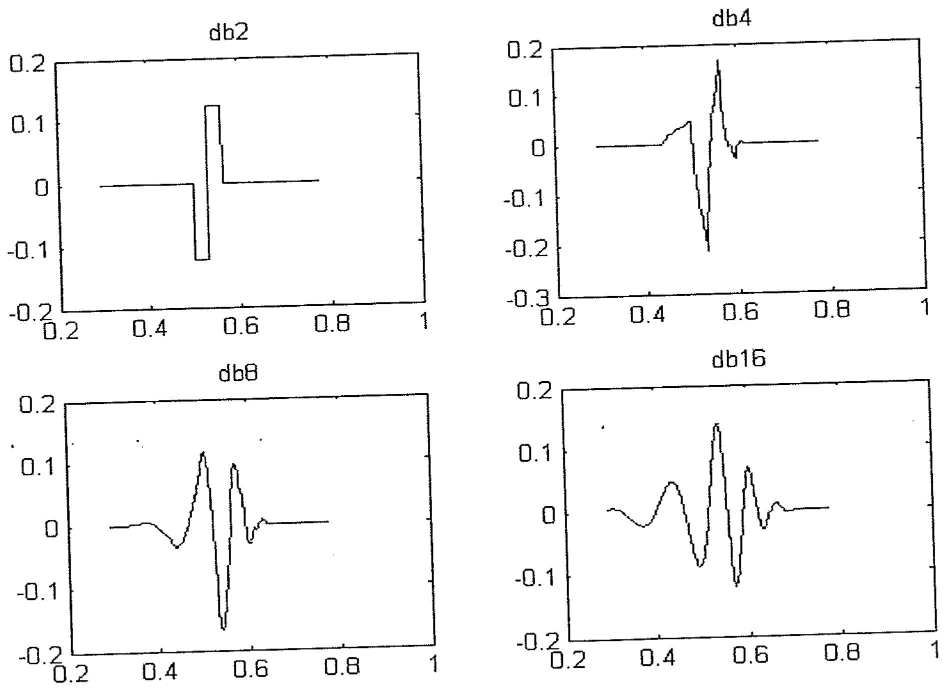


Fig. 4.2 Daubechies wavelet functions of db2 to db16.

Table 4.2: Properties of Daubechies wavelet.

Items	Properties
Family	Daubechies
Short name	db
Order	N (strictly positive integer)
Examples	dbl or haar, db4, db15
Orthogonal	yes
Biorthogonal	yes
Compact support	yes
DWT	possible
CWT	possible
Support width	2N-1
Filters length	2N
Regularity	about 0.2 N for large N
Symmetry	far from
Vanishing moments	N

4.9.3 Coiflet Wavelets

Coiflet wavelets (coiflets) are compactly supported wavelets with the highest number of vanishing moments for both ϕ and ψ for a given support width. These were constructed by Daubechies at the request of R. Coifman [Daub92]. The wavelet functions for *coif*1, *coif*2, *coif*3 and *coif4* are shown in Fig. 4.3a to 4.3d, respectively. These wavelets are more symmetrical than *dbN* family. Their main properties are illustrated in Table 4.3 [MMOP96].

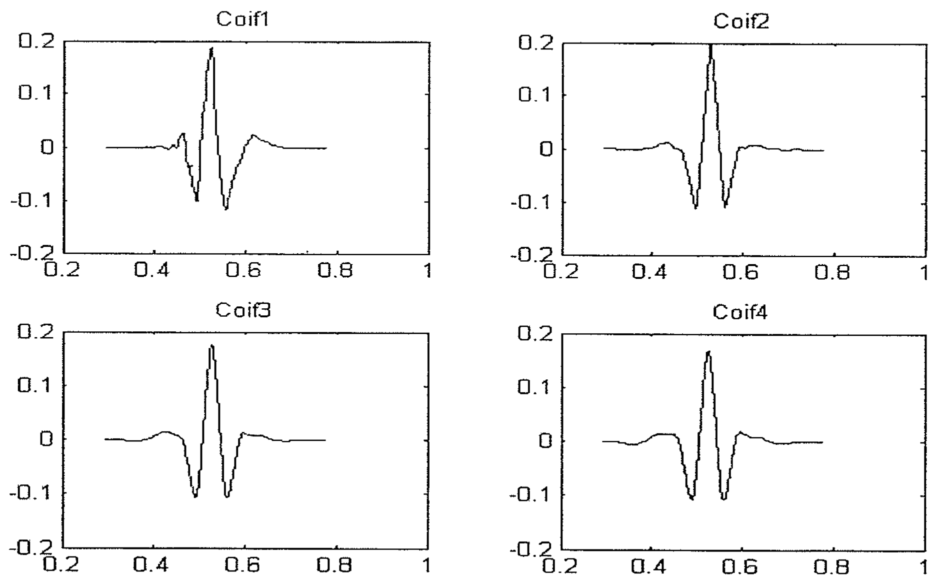


Fig. 4.3 Coiflets wavelet functions of *coif*1 to *coif*4.

Table 4.3: Properties of Coiflets.

Items	Properties
Family	Coiflets
Short name	coif
Order N	N = 1, 2, ..., 5
Examples	coif2, coif4
Orthogonal	yes
Biorthogonal	yes
Compact support	yes
DWT	possible
CWT	possible
Support width	6N-1
Filters length	6N
Symmetry	near from
Vanishing moments for psi	2N
Vanishing moments for phi	2N-1

4.9.4 Symlet Wavelets

Symlet wavelets are compactly supported wavelets with the least asymmetry and the highest number of vanishing moments for a given support width. Associated scaling filters are near linear-phase filters. These wavelets were built by Daubechies and are more symmetrical than the *dbN* [Daub92] family. These wavelets are implemented using the "minimum phase filter" [MMOP96]. The wavelet function for *sym8* is shown in Fig. 4.4. Their main properties are illustrated in Table 4.4 [MMOP96].

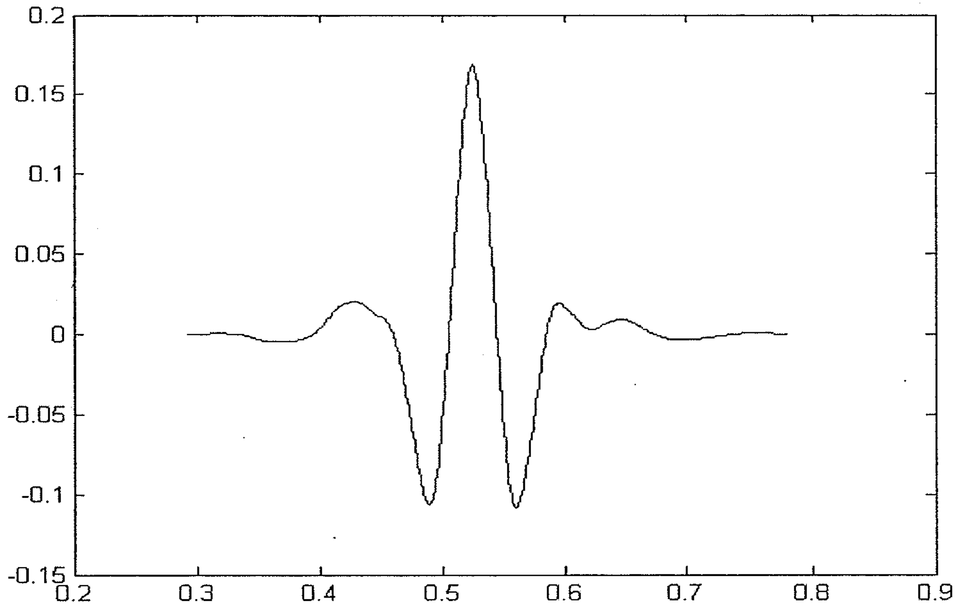


Fig. 4.4 Wavelet function for *sym8*.

Table 4.4: Properties of symlets.

Items	Properties
Family	Symlets
Short name	sym
Order N	N = 2, 3, ..., 8
Examples	sym2, sym8
Orthogonal	yes
Biorthogonal	yes
Compact support	yes
DWT	possible
CWT	possible
Support width	2N-1
Filters length	2N
Symmetry	near from
Vanishing moments	N

4.9.5 Biorthogonal Wavelets

Biorthogonal wavelets are compactly supported biorthogonal spline wavelets for which the symmetry and the exact reconstruction are possible with finite impulse response (FIR) filters, whereas in the orthogonal case this is not possible. They were developed by Cohen *etal.* [CoDF92]. It is well known that the symmetry and the exact reconstruction are incompatible, if the same FIR filters are used for both decomposition and reconstruction [MMOP96]. Hence, they used different FIR filters for reconstruction and decomposition. This wavelet family is labeled as *biorNr.Nd*, where *Nr* is the filter lengths for reconstruction and *Nd* is the filter length for decomposition. The wavelet functions for *bior*3.7 are shown in Figs. 4.5a and 4.5b, while the wavelet functions for *bior*7.3 are shown in 4.5c and 4.5d respectively. Their main properties are illustrated in Table 4.5 [MMOP96].

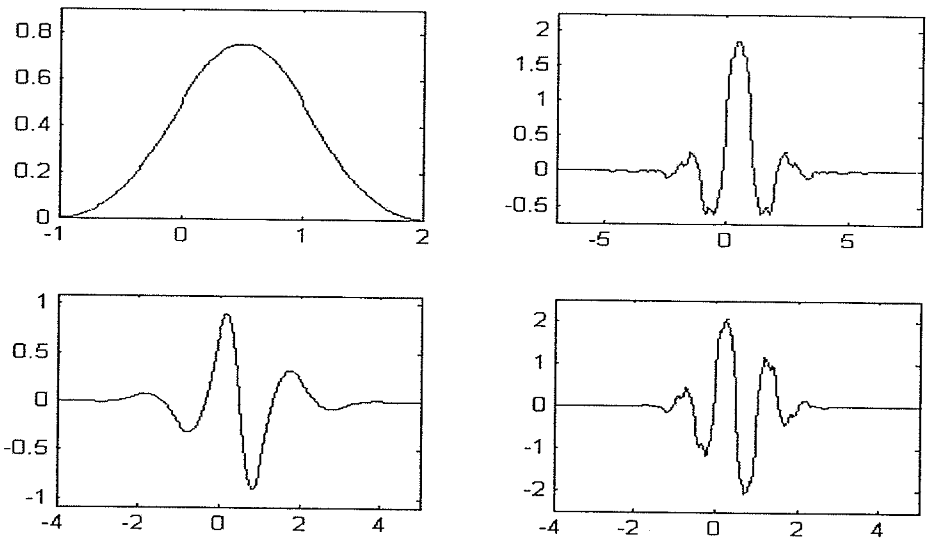


Fig. 4.5 Biorthogonal wavelet functions, (a) and (b) are *bior*3.7, and (c) and (d) are *bior*7.3.

Table 4.5: Properties of Biorthogonal wavelet

Items	Properties
Family	Biorthogonals
Short name	bior
Order Nr,Nd	Nr = 1 , Nd = 1, 3, 5
Examples	bior3.1, bior5.5
Orthogonal	no
Biorthogonal	yes
Compact support	yes
DWT	possible
CWT	possible
Support width	2Nr+1 for rec., 2Nd+1 for dec.
Symmetry	yes
Vanishing moments for psi	Nr-1

4.9.6 Meyer Wavelet

The Meyer wavelet is infinitely regular orthogonal wavelet. This wavelet and scaling functions are defined in the frequency domain. The Meyer wavelet function is shown in Fig 4.6. and its main properties are illustrated in Table 4.6 [MMOP96].

Table 4.6: Properties of Meyer wavelet

Items	Properties
Family	Meyer
Short name	meyr
Orthogonal	yes
Biorthogonal	yes
Compact support	no
DWT	possible but without FWT
CWT	possible
Support width	infinite
Effective support	[-8 8]
Regularity	indefinitely derivable
Symmetry	yes

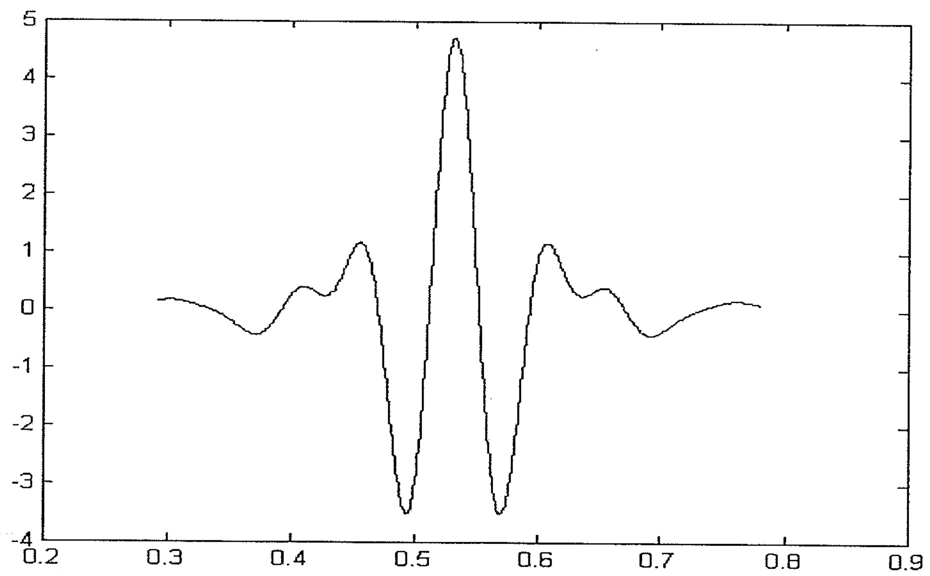


Fig. 4.6 Meyer wavelet function of degree 2.

4.9.7 Mexican Hat Wavelet

The Mexican hat wavelet is a second derivative of the Gaussian probability density function. This wavelet has no scaling function [Daub92]. Mathematically, it has a closed form analytical expression. The Mexican hat wavelet function is shown in Fig 4.7 and its main properties are illustrated in Table 4.7 [MMOP96].

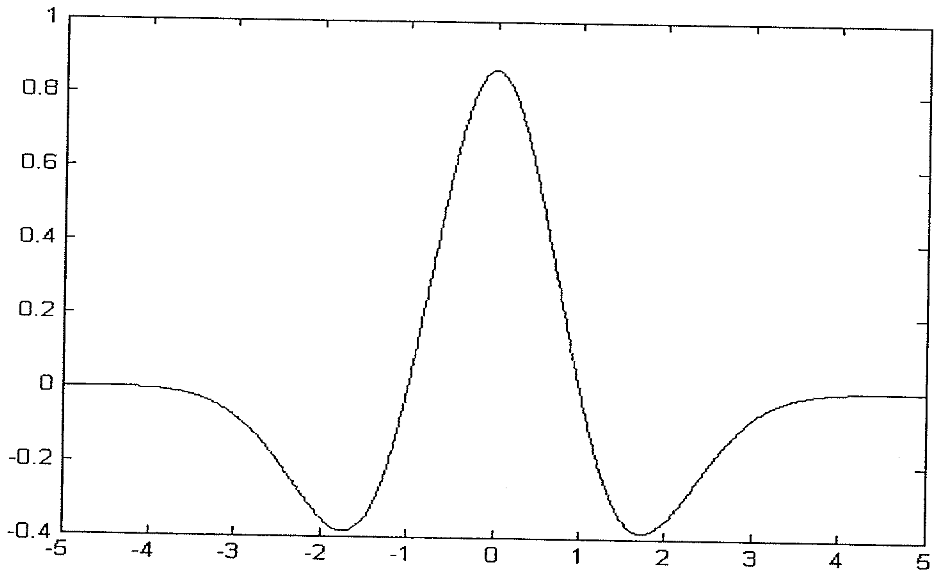


Fig. 4.7 Maxican Hat wavelet function.

Table 4.7: Properties of Mexican hat wavelet.

Items	**Properties**
Family	Mexican hat
Short name	mexh
Orthogonal	no
Biorthogonal	no
Compact support	no
DWT	no
CWT	possible
Support width	infinite
Effective support	[-5 5]
Symmetry	yes

4.9.8 Morlet Wavelet

The Morlet wavelet is a symmetrical wavelet and has no scaling function [Daub92]. Mathematically, it has a closed form analytical expression. The Morlet wavelet function is shown in Fig 4.8 and its main properties are illustrated in Table 4.8 [MMOP96].

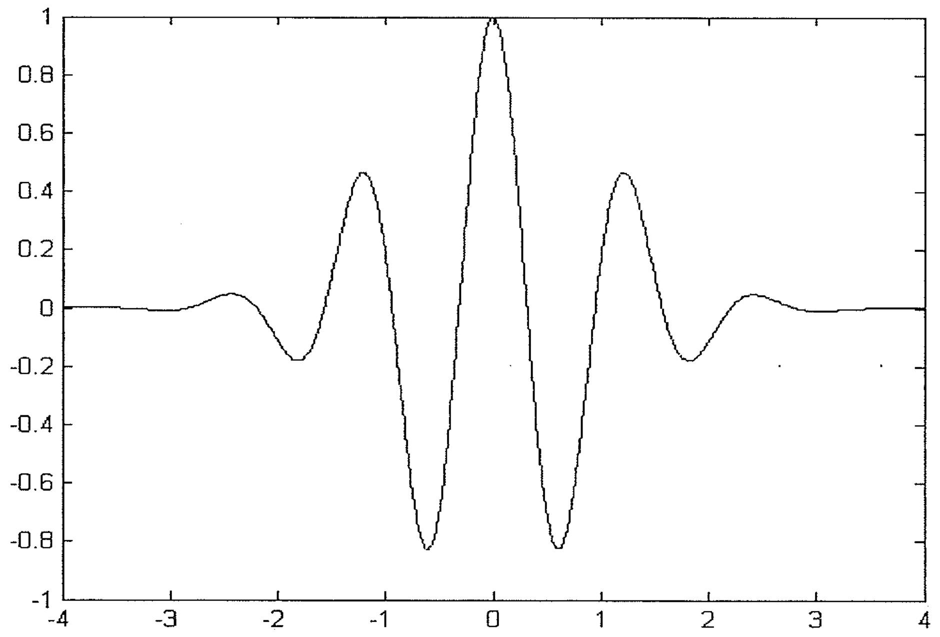


Fig. 4.8 Morlet wavelet function.

Table 4.8: Properties of Morlet wavelet.

Items	**Properties**
Family	Morlet
Short name	morl
Orthogonal	no
Biorthogonal	no
Compact support	no
DWT	no
CWT	possible
Support width	infinite
Effective support	[-4 4]
Symmetry	yes

4.10 Selection of Proper Wavelet

It was found [MaHw92] that the number of vanishing moments of the wavelet plays a major role in detecting the required singularity from wavelet transform modulus maxima. Where a wavelet is said to have n vanishing moments, if and only if for all positive integer $k < n$, it satisfies

$$\int_{-\infty}^{+\infty} x^k \psi(x) dx = 0 \tag{4.37}$$

If it is needed to estimate Lipschitz exponents up to a maximum value n, a wavelet with at least n vanishing moments is needed [MaHw92]. Using wavelets with more than one vanishing moments has the advantage of being able to measure the Lipschitz regularity up to a higher order, but it also increases the number of maxima lines. The number of maxima at a given scale often increases linearly with the number of moments of the wavelet. In order to minimize the amount of computation, the minimum number of maxima necessary to detect the irregular behavior of the signal should be known [MaHw92]. This means that a wavelet with the fewest vanishing moments should be selected, capable of detecting the Lipschitz exponents of highest order of interest. Another related property that influences the number of modulus maxima is the number of oscillations of the wavelet $\psi(x)$. For most types of singularities, the number of maxima lines converging to the singularity depends upon the number of local extrema of the wavelet itself. A wavelet with n vanishing moments has at least $n + 1$ local extrema. For numerical computations, it is better to choose a wavelet with exactly $n + 1$ local extrema [MaHw92].

It has been proven [MaZh92] that if a signal is singular at a point x_0, there exists a sequence of wavelet transform modulus maxima that converge to x_0 when the scale decreases. Hence, the detection of all the singularities from the positions of the wavelet transform modulus maxima is possible. Moreover, the decay of the wavelet transform is bounded by the decay of these modulus maxima, and we can thus measure the local uniform Lipschitz regularity from this decay [MaZh92].

In image processing applications, it is often required to detect discontinuities and peaks that have Lipschitz exponents smaller than 1 [MaZh92]. It is, therefore, sufficient to use a wavelet with only one vanishing moment. Moreover, edges in the images sometimes have a ramp profile. Hence we can model a smooth variation at x_0 as a singularity convolved with a Gaussian. Mallat and Zhong [MaZh92] proved that Lipschitz regularity in such cases can be obtained if the wavelet is a derivative of the Gaussian. However, a derivative of a Gaussian is not orthogonal. Hence, they developed a new wavelet which is very close to the derivative of a Gaussian. They reported an error of less than 10 percent due to the approximation. This wavelet has one vanishing moment, and is a quadratic spline of compact support and is continuously differentiable. Figure. 4.9 shows graphical representation of the wavelet $\psi(x)$ which is a quadratic spline and its integral $\theta(x)$ which is cubic spline.

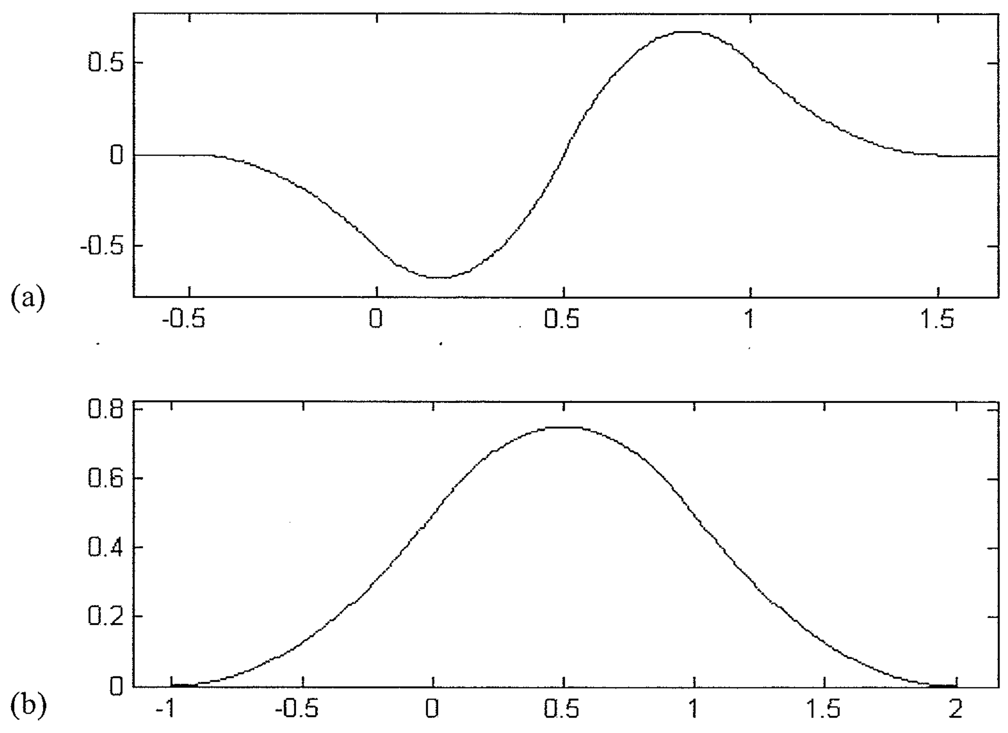


Fig. 4.9 Quadratic spline wavelet. (a) Wavelet $\psi(x)$. (b) smoothing function $\theta(x)$.

4.11 Summary

This chapter presented a detailed overview of multiresolution analysis and wavelets. A thorough mathematical illustration of wavelet transform was shown and the important topic of WTMM was presented. This chapter also discussed different well known wavelet functions with their sailent properties. The topic of selecting a wavelet function for edge detection and boundary analysis was also discussed with emphasis on quadratic spline wavelet.

CHAPTER V

DETECTION OF CORNERS AND HIGH CURVATURES USING WAVELET TRANSFORMS

5.1 Introduction

A multiresolution representation of an image provides a simple hierarchical framework for interpreting the input image information [Koen84]. In the literature, scale-space based techniques are used for developing multiscale corner detection algorithms [RaCh92] [FeKr94] [AsBr86] [RaRa97]. Wavelet theory provides a unified framework for a number of techniques that had been developed independently for various signal processing applications. In addition, there exist efficient algorithms to compute wavelet transforms using subband filtering techniques. Despite their advantages, they have yet to be fully explored in the area of computer vision. Recently, some work has been reported on boundary and surface representation using wavelet transforms [TiBo97a] [TiBo97b] [Reis96]. Few results have also been reported regarding the use of wavelets for corner detection such as [LSCT93] and [LeSC95]. There have been attempts to solve this problem using linear approximation [ChVC96]. These techniques, however, lack the advantages of multiresolution analysis. This chapter presents a summary of an efficient multiresolution technique for detecting corners and high curvature points on smooth curves. This technique was first introduced by Quddus [CQGa00a][CQGa00b][QCGa99][QCGa00][QuGa00] [QuGa02].

5.2 Assumptions Regarding Preprocessing

All techniques for feature detection from the shape of the object boundary require some preprocessing of the input image. This preprocessing includes:

- Extraction of object silhouettes,
- Edge detection,
- Thresholding,
- Separating objects (if there are multiple objects), and
- Boundary tracking.

In addition to the considerable computation involved, these steps require selection of appropriate algorithms. This selection depends upon the expected background noise, the operating environment (lighting arrangements and nature of the objects etc.), the quality and setting of camera (mono/stereo, contrast, viewing angles etc.) and the computational overhead, which is again constrained by the overall speed requirement of the system. The speed requirement again heavily depends upon the available computing platform which along with the other subsystems determine a significant part of the cost of the overall system.

Here it is assumed, as has been done by almost all the authors, that these preprocessing steps have been done properly and with the least possible cost. Hence, after tracking the boundary of the planar object, the boundary information is represented in parametric form as

$$C = \{x(t), y(t), t = 1, \ldots, n\} \tag{5.1}$$

where t is the index of the boundary pixels.

5.3 Computation of Curvature Information

The boundary information in a parametric form is useful only when proper information of the boundary curvature is obtained. In the literature, there are mainly three types of measures that are employed for the curvature analysis of planar 2-D objects. These representations reduce the dimensionality of the problem from a 2-D object contour to a 1-D signal that has the rotation invariance property.

5.3.1 Curvature Function

The curvature function is defined as the derivative of the slope with respect to the arc length (t). It can be defined in terms of the derivative of the functions $x(t)$ and $y(t)$ as [RaCh92]

$$CF(x,y) = \frac{\frac{\partial^2 y}{\partial x^2}}{\left[1+\left(\frac{\partial^2 y}{\partial x^2}\right)^2\right]^{3/2}} \tag{5.2}$$

Furthermore, the functions $x(t)$ and $y(t)$ must be related by

$$\begin{cases} \frac{\partial x}{\partial t} = \cos\Gamma \\ \frac{\partial y}{\partial t} = \sin\Gamma \end{cases} \tag{5.3}$$

where Γ indicates the orientation of the tangent along the curvature. Hence, after few mathematical manipulations, it yields the curvature expression given by

$$CF(x,y) = \left(\frac{\partial x}{\partial t}\right)\left(\frac{\partial^2 y}{\partial t^2}\right) - \left(\frac{\partial y}{\partial t}\right)\left(\frac{\partial^2 x}{\partial t^2}\right) \tag{5.4}$$

Since the curvature function depends on the first and second derivatives of the parametric curve $C(x, y)$, it is highly sensitive to boundary noise. Approximate digital computation of the derivatives of Eq. 5.3 is described in [RaCh92].

5.3.2 Radial Function

The boundary is approximated by an ordered sequence of angularly trespassed vectors projected between an arbitrary reference point (such as centroid) and the boundary points. This is often called the radial representation or the radial function. This way, the boundary of an object is described in a polar form as $r(\phi)$, where r represents the length of a line joining a point on the boundary with the reference point, and ϕ is the angle that this line makes with the reference axis. To make the representation invariant to translation and independent of the object contour starting point, the centroid of the object is selected as the reference point.

However, if any of the radial vectors intersects the object more than once, the function $r(\phi)$ is multivalued and cannot be directly used to represent the contour. To overcome this restriction, Tieng and Boles [TiBo97a] proposed a modification to the radial function. The main feature of this representation is that it directly facilitates the reconstruction of the object contour. Its main drawback is that the representation is very sensitive to the occlusion because occlusion changes the location of the centroid and hence the overall representation.

5.3.3 Orientation Space

This function relates the orientation of the tangent to the curve to the arc length t along the curve. The orientation is defined as [LiSr90]

$$\phi(t) = \tan^{-1}\frac{\Delta y}{\Delta x} \tag{5.5}$$

where $\Delta y = \partial y/\partial t \approx dy/dt$ and $\Delta x = \partial x/\partial t \approx dx/dt$. Since the arctangent function only returns an angle in the range of $(-\pi,\pi)$, any angle direction outside this range is wrapped around, thus resulting in artificial discontinuity in the edge gradient direction. The normalization step traces the function and searches for local discontinuity greater than π or less than $-\pi$. An offset of $\pm 2\pi$ is added to the function at the following point to correct the wraparound. This will result in a continuous function along the entire contour, with the exception of the initial and final points of the contour. For a closed contour, these points will always have an artificial discontinuity of 2π [LiSr90].

If the orientation at some point is defined by simply replacing the derivative above by the first difference, the orientation resolution, is only $\pi/4$. To improve the orientation resolution, the orientation at a point P_i is defined as

$$\phi(t) = \tan^{-1}\{(y_{i+q} - y_{i-q})/(x_{i+q} - x_{i-q})\} \tag{5.6}$$

for some $q > 1$ to obtain a smoothed version of orientation. The parameter q depends on two conflicting factors, namely, the orientation resolution and the corner discrimination capability. The larger q is, the higher is the orientation resolution and the less is the corner discrimination ability. This is because two corners may be merged if they are separated with less than the smoothing length. The proper choice is to select the smallest q that can provide acceptable orientation resolution. It has been determined in [LeSC95] that $q = 3$ is a reasonable choice. This choice causes the orientation profile of a corner to become a ramp-like profile instead of a step with a variation interval equal to the smoothing length ($2q+1 = 7$). Figures 5.1b to 5.1c show the orientation profiles of the image shown in Fig. 5.1a for $q = 1$ and $q = 3$, respectively. This thesis considers the orientation space function with $q = 3$ for curvature analysis of a smooth object boundary.

5.4 Detection of Corner Points

In this section, the derivation of some corner indicators based on the evolution of the wavelet transform magnitudes across scales at the corner positions is presented. The results from [LeSC92] and [CQGa00a][CQGa00b][QCGa99][QCGa00][QuGa00] [QuGa02] are summarized and will be used in detection of control points for smooth boundaries. Consider the wavelet function $\psi(t) = \frac{d\theta(t)}{dt}$ where, $\psi(t)$ and $\theta(t)$ are shown in Figs. 4.9a and 4.9b. For simplicity, $\theta(t)$ is considered to be the Gaussian function. Hence

$$\theta(t) = \frac{1}{\sqrt{2\pi}}\exp\left(\frac{t^2}{2\sigma^2}\right) \tag{5.7}$$

Lee et al. [LeSC92] and Quddus [Qudu99] presented an analysis of the behavior of wavelet transform modulus maxima with different corner models. A summary of this analysis is presented.

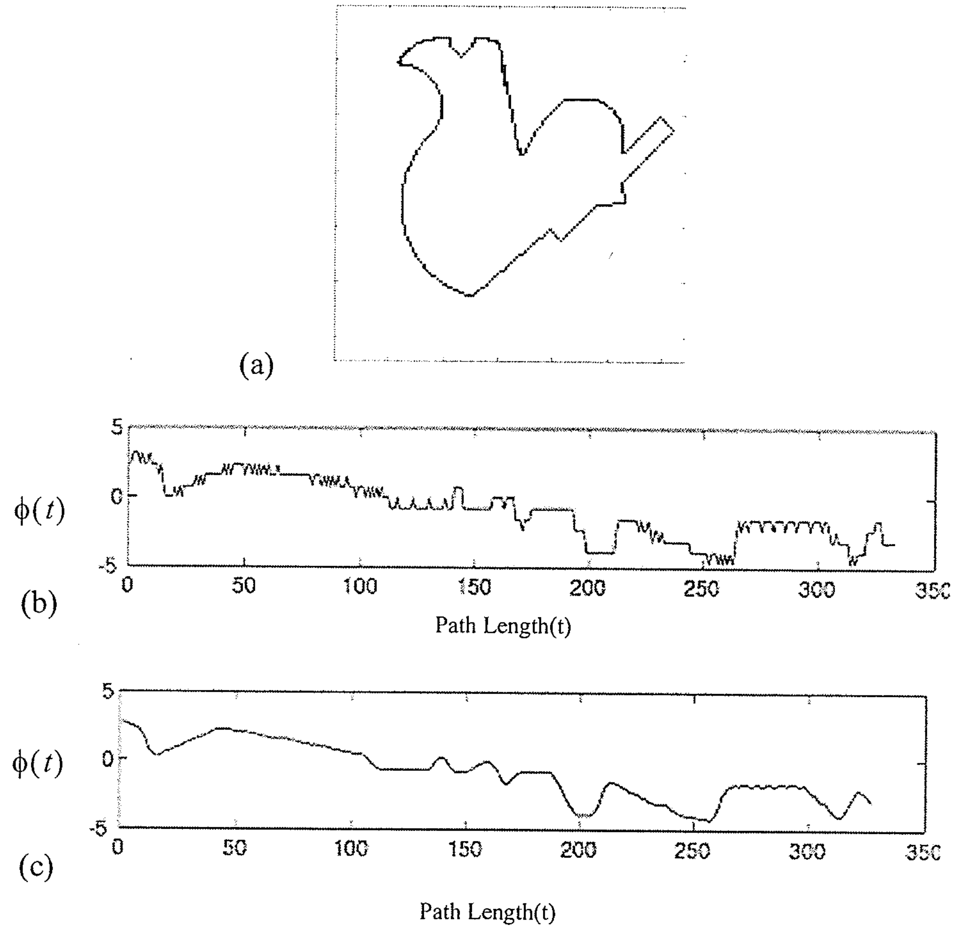


Fig. 5.1 Orientation-space representation: (a) Input image 128 x 128, (b) Orientation profile with q = 1, and (c) Orientation profile with q = 3.

5.4.1 The Generalized Single Corner Model

Figure 5.2a shows a generalized single corner, consisting of two arcs having curvatures k_1 and k_2, enclosing an angular discontinuity of ϕ. Figure 5.2(b) shows smoothed version of its orientation profile where $d = 2q+ 1$.

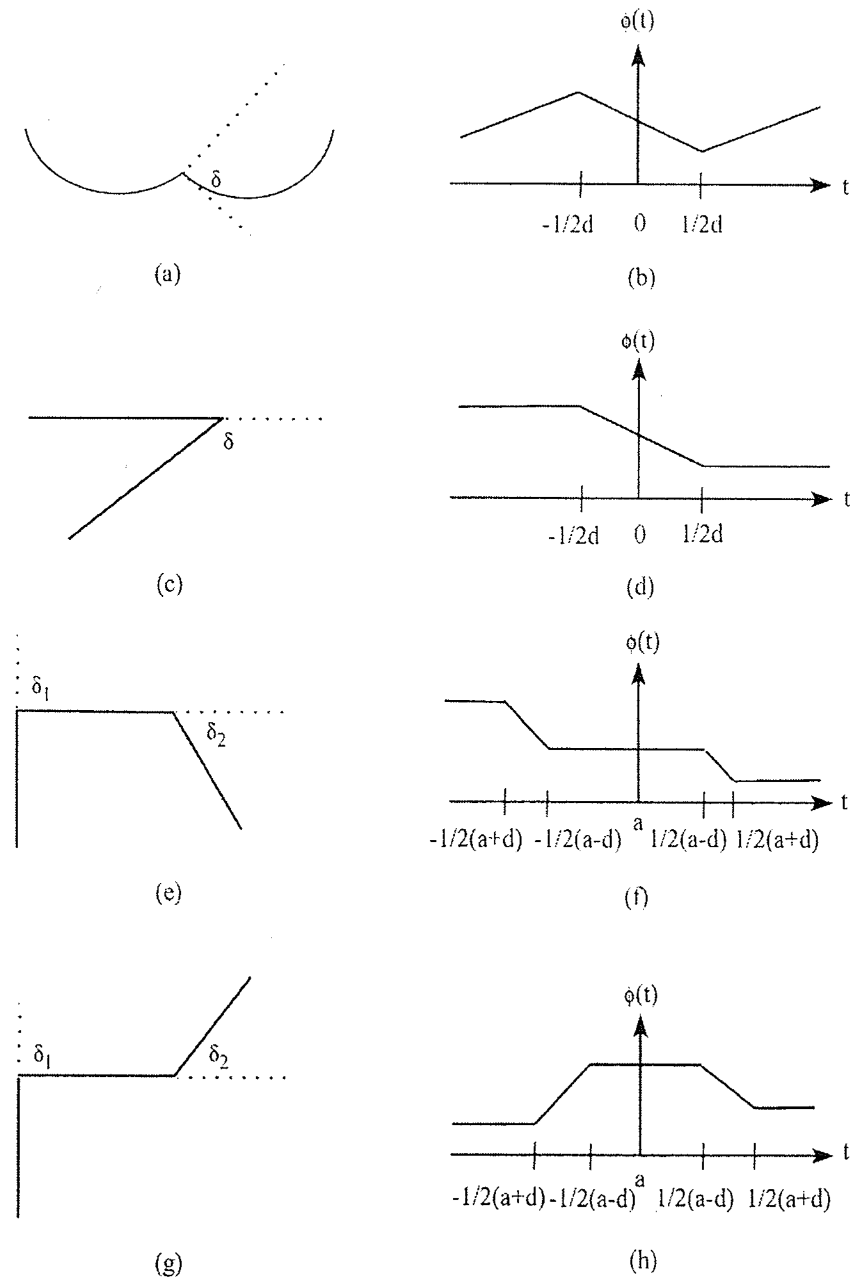


Fig. 5.2 Corner models: (a) Generalized corner model, (b) Orientation profile of generalized corner model, (c) A Γ type corner, (d) Orientation profile of Γ type corner, (e) An *END* type corner, (f) Orientation profile of *END* type corner, (g) *STAIR* type corner, and (h) Orientation profile of *STAIR* type corner.

This smoothed corner model is defined as

$$\phi(t) = \begin{cases} k_1 t + c & -d/2 < t \\ c - d\frac{k_1}{2} + [(k_1 + k_2)/2 + \delta/d](t + d/2) & -d/2 < t < d/2 \\ k_2 t + c + \delta & d/2 < t \end{cases} \qquad (5.8)$$

The wavelet transform of $\phi(t)$ at scale 2^j is the convolution of $\phi(t)$ with the wavelet function $\psi(t)$ and is given by

$$\begin{aligned} W_j\phi(t) = {} & 1/(2^j\sqrt{2\pi}\sigma)\left\{ k_2\int_{-\infty}^{t-(d/2)} \exp(-s^2/(2^{2j+1}\sigma^2))ds \right. \\ & + \quad k_1\int_{t-d/2}^{\infty} \exp(-s^2/(2^{2j+1}\sigma^2))ds \\ & + \quad \left. [(k_1 + k_2)/2 + \delta/d]\int_{t-d/2}^{t+d/2} \exp(-s^2/(2^{2j+1}\sigma^2))ds \right\} \end{aligned} \qquad (5.9)$$

Differentiating (5.8) with respect to t with the limit $d \rightarrow 0$ we get

$$\begin{aligned} \frac{dW_j\phi(t)}{dt} = {} & (1/(2^j\sqrt{2\pi}\sigma))\{(3/2)[k_1 + k_2]\exp(-t^2/(2^{2j+1}\sigma^2)) \\ & + \quad [\delta/d]\exp(-t^2/(2^{2j+1}\sigma^2))\} \end{aligned} \qquad (5.10)$$

From Eq.(5.9) one can observe that the local maximum of $W_j\phi(t)$ exists at $t = 0$ and this maximum position is independent of the corner angle δ, the curvature k and the scale factor j. Hence, the local extremum will produce a consistent peak in the wavelet transform. In [LeSC92][CQGa00a][CQGa00b][QCGa99][QCGa00][QuGa00][QuGa02], it is shown that *Inter-Scale Difference Decay Rate* (ISDDR) is given by

$$ISDDR_j = \frac{(1-1/\sqrt{2})-\{d^2/(96)(2)^2(j)(\sigma^2)\}(2-1/\sqrt{2})}{(\sqrt{2}-1/2)-\{d^2/(48)(2)^2(j)(\sigma^2)\}(2\sqrt{2}-1/2)} \tag{5.11}$$

From the above relation it can be observed that ISDDR does not change much across the scales j. Hence a corner survives through most of the scales.

5.4.2 The Γ Type Corner Model

We now consider another corner model as shown in Fig. 5.2(c). Clearly this corner model is just a special case of the generalized single corner model (if $k_1 = k_2 = 0$). This corner model is important since it is encountered frequently in practical applications, e.g. robot vision and industrial inspection. Figure. 5.2d shows the smoothed corner model in the orientation space. This smoothed corner model is defined by

$$\phi(t) = \begin{cases} c & -d/2 < t \\ c + (\delta/d)(t + d/2) & -d/2 < t < d/2 \\ c + \delta & d/2 < t \end{cases} \tag{5.12}$$

Following [LeSC92][CQGa00a][CQGa00b][QCGa99][QCGa00][QuGa00][QuGa02], one can obtain the indicator for this corner model, *Inter-Scale Decay Rate* (ISDR), which is defined as

$$\begin{aligned} ISDR_j &= (W_{j+1/2}\phi(0)/W_j\phi(0)) \\ &= \frac{(24)(2)^2(j)(\sigma^2)-d^2/2}{(24)(\sqrt{2})(2)^2(j)(\sigma^2)-\sqrt{2}d^2} \end{aligned} \tag{5.13}$$

Again, it is observed that ISDR does not change much with the change of the scales j. Hence, a Γ type corner survives through most of the scales.

5.4.3 The *END* Type Corner Model

The END type corner model consists of two corners whose angle changes are with the same sign and separated by a width of d, as shown in Fig. 5.2e. The profiles of this type of corner model in the orientation space are sketched in Fig 5.2f. The corresponding definition of Fig. 5.2e is given as

$$\phi(t) = \begin{cases} c & -(a+d)/2 > t \\ c + (\delta_1/d)(t + (a+d)/2) & -(a+d)/2 < t < -(a-d)/2 \\ c + \delta_1 & -(a-d)/2 < t < (a-d)/2 \\ c + \delta_1 + (\delta_2/d)(t - (a-d)/2) & (a-d)/2 < t < (a+d)/2 \\ c + \delta_1 + \delta_2 & (a+d)/2 < t \end{cases} \qquad (5.14)$$

The wavelet transform of $\phi(t)$ is given as

$$W_j\phi(t) = 1/(2^j\sqrt{2\pi}\sigma)\left\{(\varsigma_2/d)\int_{t-(a+d)/2}^{t-(a-d)/2} \exp(-s^2/(2^{2j+1}\sigma^2))ds \right. \qquad (5.15)$$
$$\left. +(\varsigma_1/d)\int_{t+(a-d)/2}^{t+(a+d)/2} \exp(-s^2/(2^{2j+1}\sigma^2))ds \right.$$

Following [LeSC92][CQGa00a][CQGa00b][QCGa99][QCGa00][QuGa00][QuGa02], one can obtain the indicator of this corner model, *Spatial Difference Decay Rate* (SDDR), which is defined as $\varsigma_1 \neq \varsigma_2$

$$SDDR_j = (1/\sqrt{2})\frac{IT_{j+1/2}-OT_{j+1/2}}{IT_j-OT_j} \tag{5.16}$$

where

$$IT_j = \int_{-d/2}^{+d/2}(\exp-s^2)/(2^{2j+1}\sigma^2))ds \tag{5.17}$$

and

$$OT_j = \int_{a-d/2}^{a+d/2}(\exp-s^2)/(2^{2j+1}\sigma^2))ds \tag{5.18}$$

Again, observe that ISDR does not change much with the scales j. Hence, an END type corner survives through most of the scales. Now if $j >> d$ then $IT_j \approx OT_j$ and hence END type corner converges to a single corner.

5.4.4 The STAIR Type Corner Model

The STAIR type corner is like END type corner except that the changes in the corner angles are of opposite signs as shown in Fig. 5.2g. Its corresponding orientation profile is shown in Fig. 5.2h. Mathematically, it is given as

$$\phi(t) = \begin{cases} c & -(a+d)/2 > t \\ c+(\delta_1/d)(t+(a+d)/2) & -(a+d)/2 < t < -(a-d)/2 \\ c+\delta_1 & -(a-d)/2 < t < (a-d)/2 \\ c+\delta_1-(\delta_2/d)(t-(a-d)/2) & (a-d)/2 < t < (a+d)/2 \\ c+\delta_1-\delta_2 & (a+d)/2 < t \end{cases} \tag{5.19}$$

Following [LeSC92][CQGa00a][CQGa00b][QCGa99][QCGa00][QuGa00][QuGa02] one can find that for *STAIR* type corner model.

$$SDDR_j = (1/\sqrt{2})\frac{IT_{j+1/2}-OT_{j+1/2}}{IT_j-OT_j} \tag{5.20}$$

which is the same as the result of END type corner but here ITS and OT, are of the opposite sign. Hence, when the scale increases the two extrema move away from each other.

5.4.5 Properties of Corners

The analysis presented in the previous subsections results in few properties which can be summarized as follows:

Property 1: If the local variation is the result of an isolated single corner, then the corresponding extremum survives at every scale. In case of a double corner, at least one will persist at higher scales in the wavelet transform domain. (From the analysis of *Generalized* and Γ type corner model.)

Property 2: For an END type corner, the two extrema have the same signs and move towards each other as the scale increases and get merged into one. As to a STAIR type corner, the two extrema are with opposite sign and move away from each other when the scale increases. (From the analysis of END and STAIR type corner model.)

Property 3: For an END type corner, the interaction between the two corners is constructive, hence the wavelet transform magnitudes at the two corner positions are larger than that of a single corner. On the other hand, for STAIR type corner, the interaction between the two corners is destructive. Hence, the wavelet transform magnitudes at the two corner positions are smaller than that of a single corner. (From the analysis of END and STAIR type corner model.)

Property 4: In the wavelet transform domain, any important event is detected easily at higher scales, while good localization of the events is obtained at lower scales [RaCh92].

5.4.6 Corner Detection Algorithm

In this subsection, a summary of a robust corner detection algorithm using spline wavelet as described in [LeSC92][Qudu99] is presented. Corners and arcs are relative terms and largely depend upon the shape of the object under consideration. For example, if there are sharp corners at the boundary, small curvature changes will not be recognized as corners. On the other hand, if the shape consists only of small curvature changes then these curvature changes will be recognized as corner points. Hence, in order to provide such robustness, this algorithm starts with the normalization of wavelet transform modulus maxima. This normalization is obtained at each level regarding the global maxima of that level. This normalization also makes the algorithm adaptive when a different wavelet is used for the decomposition. This is done in Step 1 of the algorithm.

Step 1: Orientation profile $\phi(t)$ is decomposed using wavelet transform at scales $2^1, 2^2, 2^3$ and 2^4.Wavelet transform modulus maxima (WTMM) are computed at each of these levels. These WTMMs at each level are normalized with respect to the maximum peak at that level.

Step 2: The events (valid corners) are observed at the highest scale (2^4). The peaks higher than some threshold τ_1 are recognized as valid events. These events are successively tracked at lower scales (2^3, 2^2, 2^1) to find their exact locations. (Properties 1 and 4.)

Step 3: Those events which are increasing at decreasing scales and are greater than some threshold $\tau_2 < \tau_1$ at scale 2^4 are also taken as valid events. This step is to take care of *STAIR* type corners. These events are also successively tracked at lower scales (2^3, 2^2, 2^1) to find their exact locations. (Properties 3 and 4.)

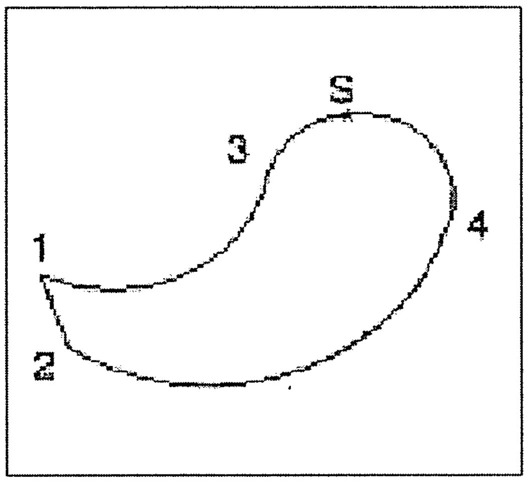
Step 4: Find a large (greater than some threshold τ_3) events in the vicinity of already detected events in the previous steps from the lowest scale (i.e., 2^1). This step is to find any *END* type corner model. (Properties 1 and 2.)

5.5 Smooth Joints using Wavelet Transform

In the previous section a summary of corner detection technique presented in [LeSC92][CQGa00a][CQGa00b][QCGa99][QCGa00][QuGa00] and [QuGa02] was discussed. In this section we present the summary of modifications that were done in the above mentioned technique by Quddus in [CQGa00a][CQGa00b][QCGa99][QCGa00][QuGa00][QuGa02] to detected high curvatures and smooth joints. As already discussed, corners, inflections and smooth joints are important features for shape representation and analysis. Given the multiresolutional nature of wavelets along with the efficient computation algorithms, it is a natural candidate for such a task. Arcs or curvatures in the orientation space, however, cannot be detected by directly using wavelets with only one vanishing moment. At the same time, wavelets with more than one vanishing moments will not be able to locate properly the sharp changes in the orientation space. To solve this problem, discontinuities are introduced deliberately. These discontinuities are formed by an operation similar to Sample and *Hold*. Hence ramp profiles of the arcs in the orientation space become stair type as shown in Fig. 5.3b. Now the wavelet with one vanishing moment can be used to detect these artificial discontinuities and thereby providing detection of arcs along the boundary in the orientation space.

Figures 5.3 and 5.4 show the detection process of smooth joints. In Fig. 5.3a the starting point on the object boundary for the orientation profile is marked as ‘S. The boundary is analysed in a anti-clockwise fashion. The orientation profile for Fig. 5.3a is shown in Fig. 5.3b. Figure 5.4 shows the WTMM for three levels for the orientation profile. Figure. 5.4a to 5.4c show the WTMM for level 2^1, 2^2 and 2^3, respectively. It is observed that the inflection point can be detected by observing the sign of WTMM which is marked as label 3. The smooth joints can be detected by observing a discontinuity in the amplitude of WTMM which is marked as label 4. Fig. 5.3a shows all the corners and smooth joints detected using this technique. This various steps of this algorithm is described in the next section.

(a)



(b)

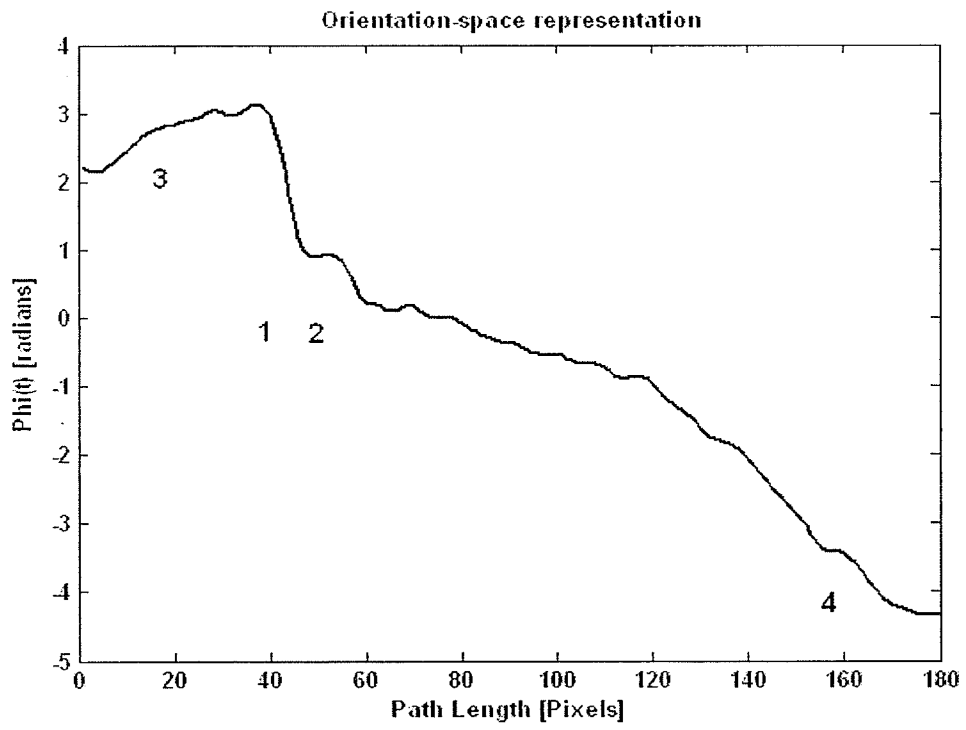


Fig. 5.3 Profiles of smooth joints (a) Original image 256 x 256 with corners(1,2), inflection (3) and smooth joint (4) 'S' indicates starting point of boundary tracking. (b) Orientation profile.

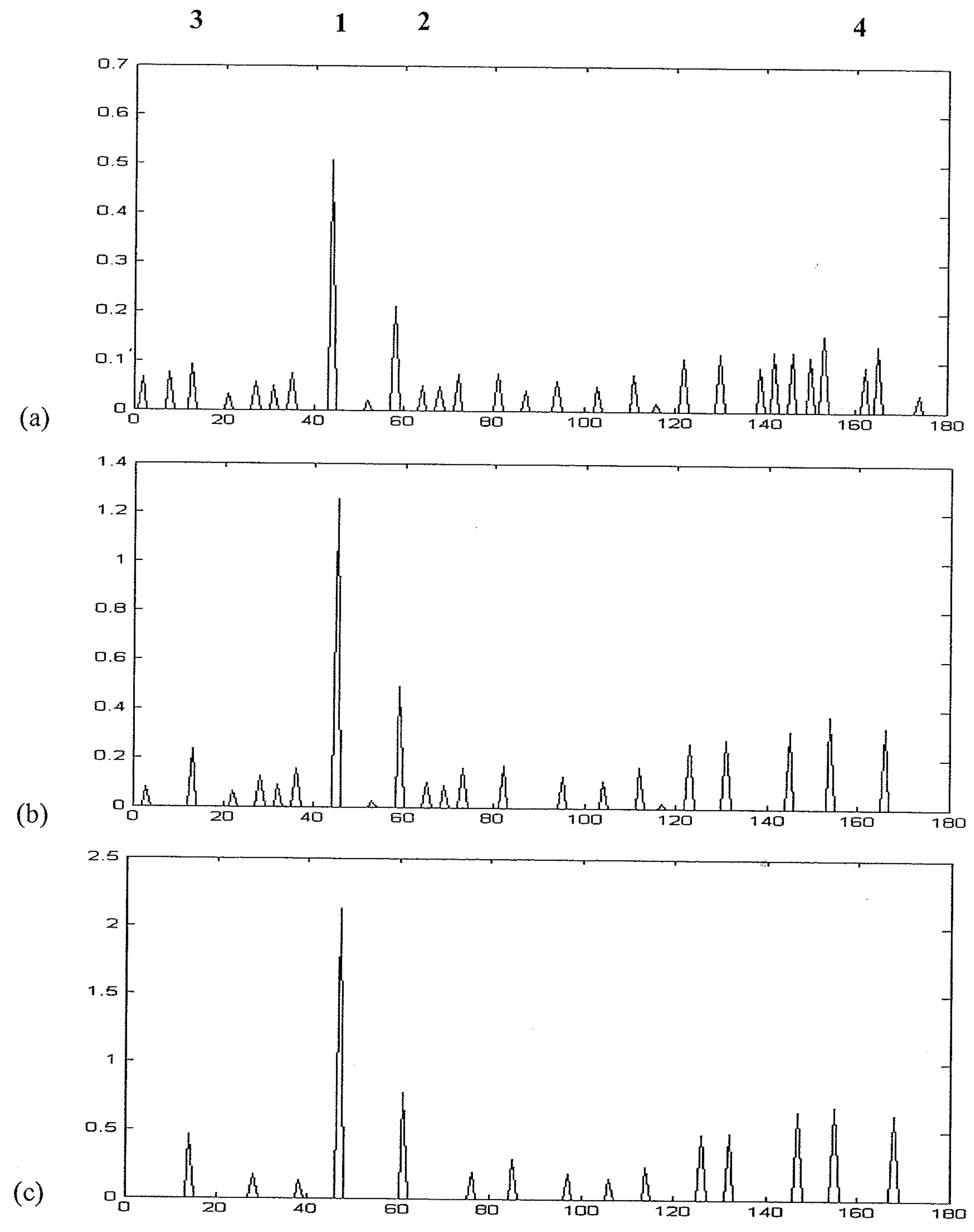


Fig. 5.4 Three levels of WTMM for orientation profile of Fig. 5.3b. (a) Level 1 or 2^1, (b) Level 2 or 2^2, and, (c) Level 3 or 2^3. The corresponding points in the Fig. 5.3 are labelled at the top.

The *Sample* and *Hold* operation (for the ith segment) is done at the interval η_i is given by

$$\eta_i = \frac{t_i}{C} \tag{5.21}$$

where t_i is the length of a valid ith segment, and C is a constant. It can be seen that increasing this constant would increase the susceptibility to the noise. On the other hand, decreasing this constant would lead to more error in the detection of smooth joints. Here, $C = 5$ is found to be satisfactory by [CQGa00a][CQGa00b][QCGa99][QCGa00][QuGa00] [QuGa02].

5.5.1 Algorithm for the Detection of Smooth Joints

This algorithm is invoked only after the corner points have been detected as explained in subsection 5.4.6.

Step 1: Segment the orientation profile, from start to end, with break points at corner points already detected. WTMM is computed at scale 2^3 for each segment (see Fig. 5.4).

Step 2: The arcs are extracted by observing those impulses which have approximately the same amplitude and are appearing consecutively. Here, three or more consecutive impulses, of approximately the same heights, are taken as an indication of a valid arc.

Step 3: A point of inflection is obtained at the mid point between the sign change of the two valid and consecutive arcs. This point is labeled as (3) in Fig. 5.4a.

Step 4: A smooth joint is detected by observing the sudden change in amplitude of consecutive impulses belonging to the two valid and consecutive arcs. The smooth joint is labeled as (4) on the bottom of Fig. 5.4c.

This technique is computationally efficient because it requires decomposing the orientation profile only once using very fast wavelet decomposition algorithms [Mall98]. Moreover, corner and smooth joint detection steps are simple to implement efficiently on computers and does not require segmenting the orientation profile.

5.6 Summary

In this chapter, detection of control points on a smooth boundary using wavelets was discussed. The detection of control points is a very important step in the boundary modelling approaches presented in this thesis. This chapter presented different types of measures that are employed for the curvature analysis of an object boundary. A detailed discussion of the different corner models and their important properties was presented. The particular algorithm that is used in this thesis for the detection of control points using wavelets was explained in detail.

In the next chapter a discussion about wavelet based smoothing will be presented. Smoothing is required because the detection of control points discussed in this chapter requires a smooth boundary for processing.

CHAPTER VI

WAVELET BASED SMOOTHING

6.1 Introduction

As discussed in Ch.1, the rapid development of computerized scientific instruments has produced a wide variety of interesting problems in data analysis and signal processing. In fields ranging from extra-galactic astronomy to molecular spectroscopy to medical imaging to computer vision, one must recover a signal, curve, image, spectrum, or density from incomplete, indirect, and noisy data. As discussed in the previous chapters, wavelet analysis has been recognized as a tool with important applications in areas of statistical concern such as time series, function estimation and image analysis. Wavelet-based methods for signal and image denoising have been developed over recent years, and are now well established. These techniques have been applied to both one dimensional signals and to images. Typically, they proceed in three stages: (i) a wavelet transform operation, (ii) some form of processing operation on the wavelet coefficients, and (iii) an inverse wavelet transform.

A signal is often seen as containing noise which obscures the information of interest. This may be simply a measurement error, or it could be fluctuation details which are a nuisance when the underlying trends or discontinuities are being investigated. Many methods have been developed for smoothing a signal, in the hope that the noise can be suppressed and the significant patterns retained and revealed. These have ranged from simple moving averages or moving medians to methods of considerable mathematical complexity. Wavelets seem to offer a smoothing approach which is relatively simple to use, while adapting well, and automatically, to the form of the signal being smoothed. In this chapter a brief overview of popular wavelet based smoothing techniques is presented. We also discuss the particular technique that was applied in this thesis.

6.2 Smoothing via Wavelet Shrinkage or Thresholding

Traditional linear methods of smoothing such as [WoWa75] achieve noise suppression by broadening the features significantly. These state-of-the-art methods use adaptive linear smoothers based on fitting splines under tension with adaptively chosen tension parameter or truncating the empirical Fourier series with adaptively chosen truncation. The adaptive spline under tension suppresses noise, but at the expense of significant broadening of sharp edges that results in the loss of certain features. The adaptive Fourier Series estimate leaves features sharp, but does not really suppress the noise. It has been shown by Donoho and his collaborators such as Iain Johnstone [DoJo94a][DoJo94b][DoJo98], Gerard Kerkyacharian and Dominique Picard [DoKP95] that shrinking noisy wavelet coefficients via soft thresholding offers very attractive alternatives to existing methods of recovering signals from noisy data. These methods have theoretical properties of adaptive minimaxity that far surpass anything previously known. It has been found [Mall98] that wavelet thresholding methods work well in problems ranging from photographic image restoration [Laki96] to medical imaging.

6.2.1 Wavelet Transform of a Noisy Signal

Consider a signal $f(t)$ on the interval $t \in [2000, 4000]$ in Fig. 6.1 that has been contaminated by white noise z. The noisy signal is shown in Fig. 6.1a and given by

$$d_i = f(t_i) + \nu z_i \tag{6.1}$$

where $t_i = i/n$, $z_i \sim N(0, 1)$ is a Gaussian white noise, and ν is a noise level. The wavelet decomposition of the signal d at level $j=N$ is given by

$$w = W_j d \tag{6.2}$$

The plot of wavelet coefficients in Fig. 6.1 suggests that small coefficients are dominated by noise, while coefficients with a large absolute value carry more signal information than noise. Replacing the smallest, noisy coefficients by zero and a backwards wavelet transform on the approximation scale may lead to a reconstruction with the essential signal characteristics and with less noise. More precisely, this idea is motivated by three observations and assumptions:

1. The decorrelating property of a wavelet transform creates a sparse signal: most untouched coefficients are zero or close to zero.
2. Noise is spread out equally over all coefficients.
3. The noise level is not *too* high, so that we can recognize the signal and the signal wavelet coefficients.

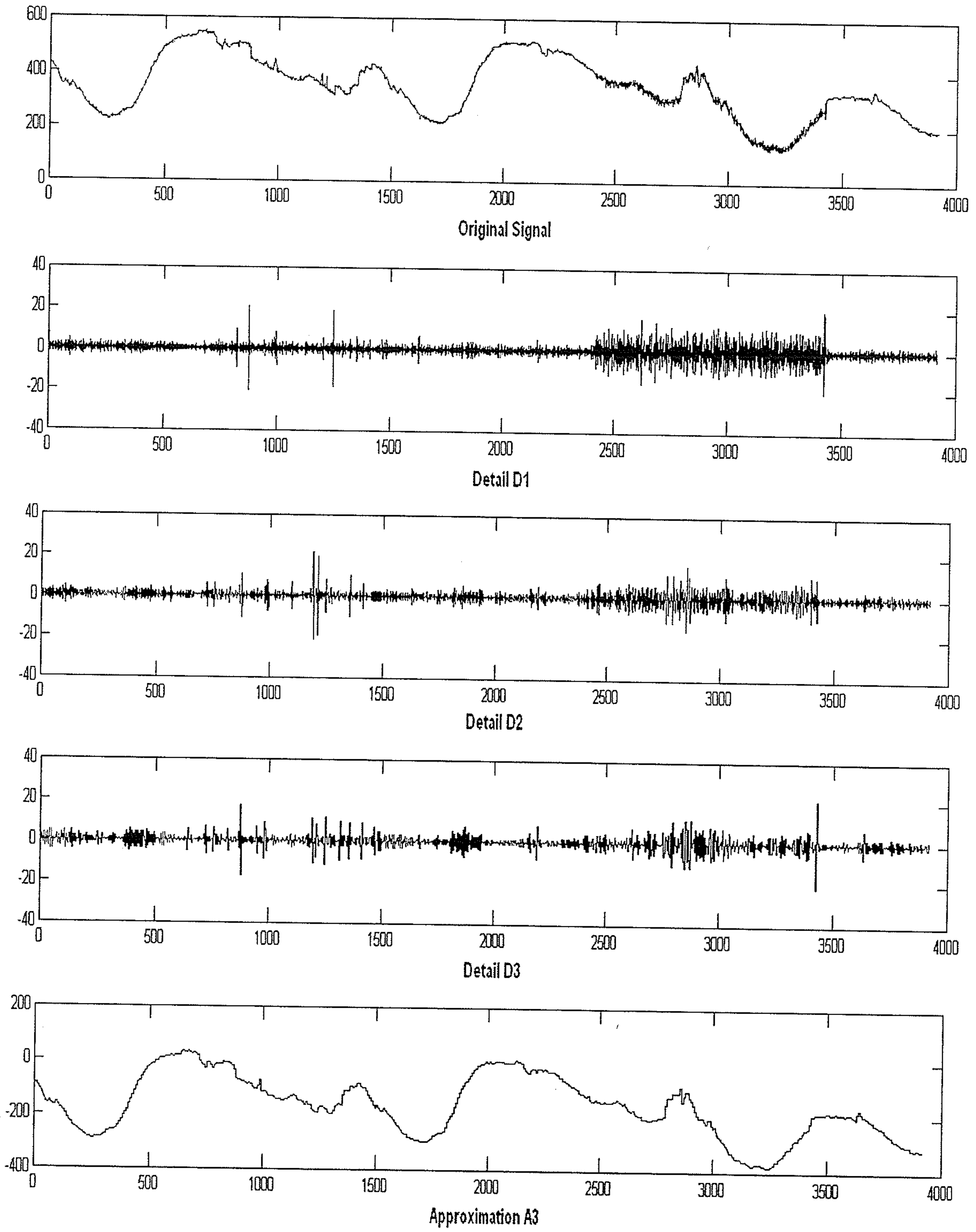


Fig. 6.1 Wavelet transform of a noisy signal with Daubechies wavelets *Db1* at level 3.

6.2.2 Hard and Soft Thresholding

In the above section we have mentioned a procedure in which small coefficients are removed, while the others are left untouched. This keep-or-kill procedure is called *hard thresholding,* as shown in Fig. 6.2b that plots the output coefficient versus the input. An alternative for this scheme is *soft-thresholding*, as illustrated in Fig. 6.2c in which the coefficients above the threshold are shrunk in absolute value. The amount of shrinking equals the threshold value, so that the input-output plot becomes continuous. This scheme was first presented by Donoho [Dono95], and has the following 3 steps:

Step 1. Decompose: Choose a wavelet, and select a level *N*. Compute the wavelet decomposition of the signal *d* at level *N*, yielding noisy wavelet coefficients.

Step 2. Threshold Detail Coefficients: For each level from 1 to N, select a threshold and apply the soft-threshold to the detail coefficients.

$$w' = \begin{cases} w+a & w<-a \\ 0 & -a<w<a \\ w-a & w \geq a \end{cases} \tag{6.3}$$

where w is an original wavelet coefficient, w' is the thresholded coefficient, and a is threshold value given by

$$a = \sqrt{2\log n}\sigma \tag{6.4}$$

where σ is the noise standard deviation and n is number of points.

Step 3. Reconstruct: Compute wavelet reconstruction using the original approximation coefficients of level *N* and the modified detail coefficients of levels from 1 to *N*.

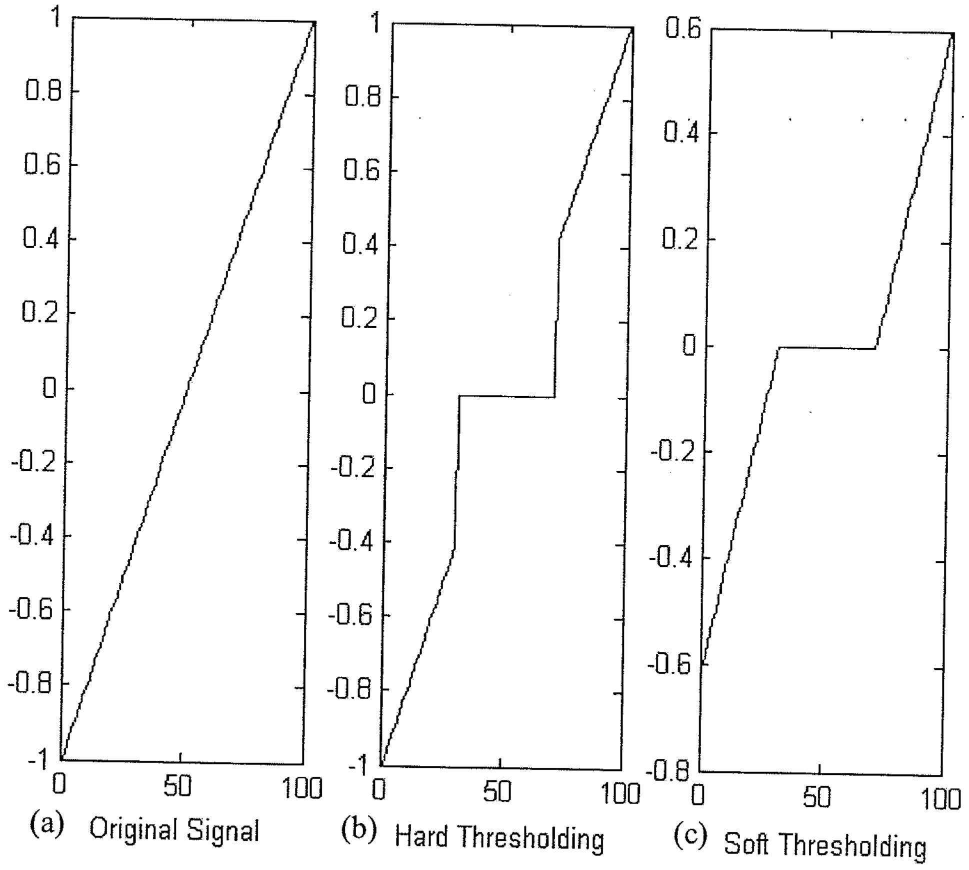
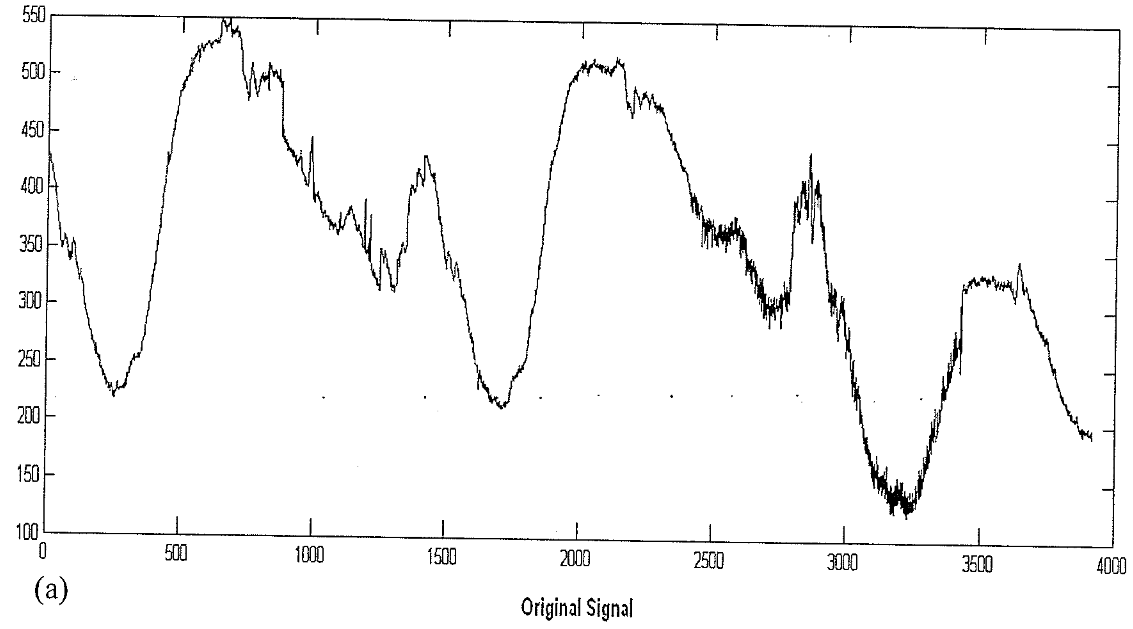


Fig. 6.2 Hard and soft thresholding.



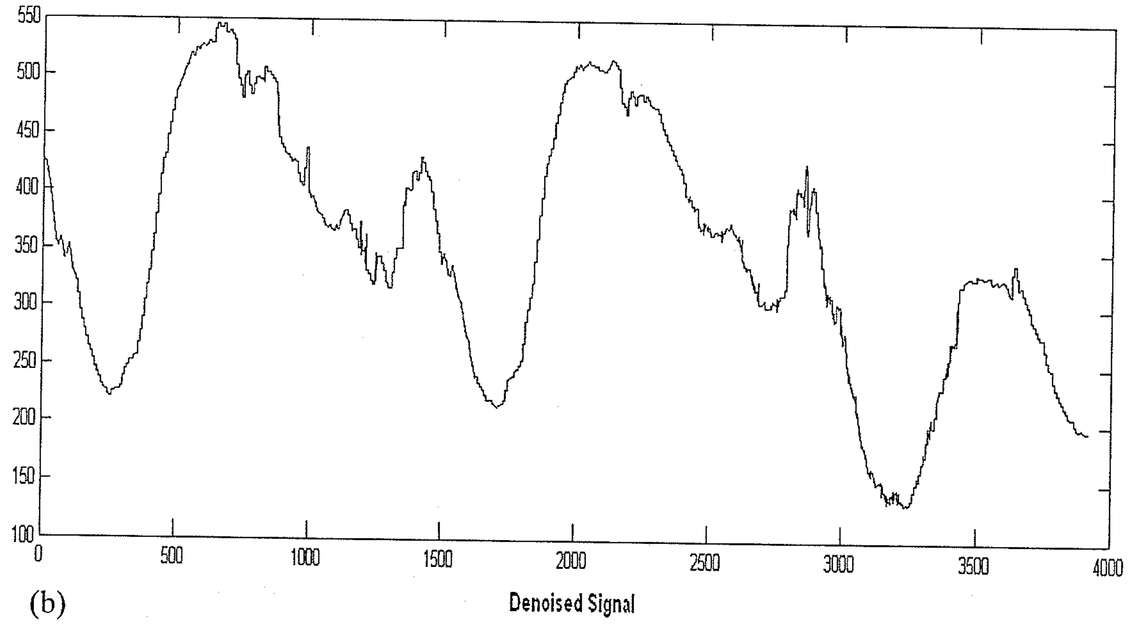


Fig. 6.3 Denoising or smoothing using Donoho technique.

6.2.2.1 Threshold Selection and Smoothing

Although the hard-thresholding may seem to be a more natural approach from the computational point of view, the continuity of the soft-thresholding operation has important advantages. In some cases pure noise coefficients may pass through a hard threshold and appear in the output as annoying, spurious spikes. Soft-thresholding shrinks these false structures. A central question in many threshold procedures is how to choose the threshold. According to [Dono95] a threshold is a trade-off between closeness of fit and smoothness. In order to denoise or smooth the signal, it is required to optimize the mean-squared error (from Eq. 6.1)

$$n^{-1}E\|\hat{f}-f\|^2_{l^2_n} = n^{-1}\sum_{i=0}^{n-1} E(\hat{f}(i/n)-f(i/n))^2 \tag{6.5}$$

subject to the side condition that with high probability $\hat{f}$ is at least as smooth as f.

A small threshold yields a result close to the input, but this result may still be noisy. A large threshold on the other hand, produces a signal with a lot of zero wavelet coefficients. This sparsity can be regarded as smoothness where the output has a simple, smooth representation in the chosen basis. Giving extra emphasis to smoothness, however, destroys some of the signal singularities. In image processing, it may cause blur and artifacts. Significant amount of work has been done in literature to solve the problem of threshold selection and readers are encouraged to see [Mall98], [LiSr90].

There is a formal way to demonstrate that thresholding is a particular example of a more general class of smoothing algorithms. These algorithms typically look for a compromise between closeness of fit and smoothness. Smoothness, or sparsity, can be expressed by some measure of *entropy*, which should be minimized. On the other hand, for closeness of fit, the algorithms use an error *energy* term that is the norm of the difference between input and output (described above). A smoothing parameter λ takes care of the compromise between these two and the algorithm minimizes:

$$\|w_\lambda - w\|^2 + \lambda\xi(w_\lambda) \tag{6.6}$$

where $\xi(w_\lambda)$ is the entropy of the output which is given by the following norm in l^2 at level j

$$\xi(w_\lambda) = \sum_{j=L}^{J-1} \gamma_j \|w_{\lambda,j}\|^2 \tag{6.7}$$

The resulting coefficient after linear shrinkage is given by

$$w_{\lambda,j} = \frac{1}{1+\lambda_j} w_j \tag{6.8}$$

The smoothing parameters could be chosen to optimize the mean square error of the result, although in practical applications, this error cannot be computed exactly. We could limit the possibilities *a priori* to

$$\lambda_j = \begin{cases} 0 & j = L \le K-1 \\ \infty & j = K \le J-1 \end{cases} \tag{6.9}$$

where K acts as smoothing parameter. In the above case, the coefficients at low resolution levels are kept, while the ones at finer scales are thrown away. In order to keep the largest coefficients, we need to minimize

$$\sum_{i=1}^{N} \left(|w_{\lambda i} - w_i|^k + \lambda^k x(w_{\lambda i})\right) \tag{6.10}$$

where

$$x(w) = \begin{cases} 0 & w_{\lambda i} = 0 \\ 1 & w_{\lambda i} \ne 0 \end{cases} \tag{6.11}$$

Here the entropy is now given in l^1

$$\xi(w_\lambda) = N \tag{6.12}$$

where $N = \{i = 1,\dots,N | w_{\lambda i} \ne 0\}$. If we use the l^1 norm as measure of sparsity. We minimize

$$\sum_{i=1}^{N} \left((w_{\lambda i} - w_i)^2 + 2\lambda|w_{\lambda i}|\right) \tag{6.13}$$

and entropy is now given by

$$\xi(w_\lambda) = 2\sum_{i=1}^{N}|w_{\lambda,j}| \tag{6.14}$$

The following section presenets an overview of the wavelet smoothing technique that was applied in this thesis.

6.3 Combined Evidence Thresholding

In early wavelet based denoising or smoothing techniques, maximally-decimated wavelet transforms were used, but now commonly redundant wavelet transforms are used that result in wavelet coefficient set representation, corresponding to different resolution scales, along with a final smoothed residual. In other words, this is known as *multiresolution analysis*. A variety of processing methods have been employed to process the wavelet coefficients at the intermediate stage. The most widely known methods are based on thresholding techniques as described above and stem largely from the early work of Donoho and Johnstone [DoJo94a]. In this approach, a threshold (soft or hard) is used to discriminate between wavelet coefficients that are assumed to arise from noise, and those thought to arise from features of the signal. A compromise thresholding technique, (often referred to as semi-soft wavelet shrinkage), was introduced by Gao and Bruce [GaBr95] and involves the use of two thresholds T1 and T2. Wavelet coefficients of magnitude below T1 (the lower) are suppressed, while those above T2 are left alone. Coefficients of intermediate magnitude are attenuated in proportion to their magnitude.

All of these techniques effectively partition the wavelet coefficient sets on the basis of their absolute value relative to a threshold. An alternative approach was introduced by

Healy et al. [XuHe94], and uses the correlation between wavelet planes as the basis for discrimination between coefficients due to noise and coefficients arising from signal features. This approach exploits the fact that noise is normally spatially localized and can, therefore, be expected to be weakly correlated across resolution scales. Correlation with its successors is calculated for all coefficients within each wavelet plane. Coefficients which exhibit low correlation with those on successor wavelet planes are suppressed while those exhibiting strong correlation are retained.

These approaches have one thing in common: they involve making a binary decision for each wavelet coefficient. In one case, the decision is made on the basis of the wavelet coefficient's absolute magnitude, while in the other it is made on the basis of a scaled correlation value. In both cases, the wavelet coefficients are effectively partitioned on the basis of just one characteristic. There is some recognition in the scientific community that this approach is too simple and considerable research is being employed on methods such as Bayesian techniques [Vida94].

Brown has introduced [Brow00] a new technique called *combined evidence thresholding* (CET). In the decision process, this technique uses both the magnitude of wavelet coefficients and their correlation with successor wavelet planes. The absolute magnitude of wavelet coefficients is treated as one source of *evidence* on whether or not a wavelet coefficient is attributable to noise. The correlation with corresponding wavelet values on successor wavelet planes is treated as another. Numerical evidence values are computed from both sources, and then combined to give a *combined evidence value*. For the combined evidence values below a lower threshold, the wavelet coefficients are deemed not to be attributable to noise, and are left alone. For values above an upper threshold, they are suppressed, while for intermediate values they are scaled in proportion to the strength of the evidence.

In this thesis, the CET technique was used for boundary smoothing purposes. This technique was chosen because it is simple to implement, computationally fast and gives better results then other standard techniques like the Donoho soft thresholding [DoJo94a].

6.3.1 The CET Algorithm

The CET technique does not depend on any particular wavelet transform algorithm and could be combined with any algorithm for generating a redundant multi-resolution transform. This thesis uses the *atrous* wavelet transform algorithm [Brow00] [RiDu92]. This algorithm has been described in detail in [StMB94], [MuSB95], [Shen92] and [StMB98]. According to Brown [Brow00] the CET algorithm has the following steps:

Step 1. Implement a redundant wavelet transform of the source signal. This yields a number of wavelet coefficient sets and a smoothed residual signal.

Step 2. Considering each wavelet coefficient set, do the following tasks for all the coefficients within each set.

- Calculate an evidence value from the absolute wavelet magnitude considering the proposition that wavelet coefficient arises from noise in the signal.

- Calculate an evidence value from the correlation with successor wavelet values.
- Combine the two evidence values.
- Process the wavelet coefficient on the basis of the combined evidence value.

Step 3. Implement an inverse wavelet transform.

The sub steps of Step 2 in the above algorithm need further explanation. A brief description for each of these intermediate steps is given in the following sections.

6.3.1.1 Calculation of Evidence Value from Wavelet Coefficient Magnitudes

In conventional hard or soft thresholding methods like [Dono94], the wavelet magnitude threshold is used. This threshold can be either provided as a user adjustable parameter, or estimated automatically by different statistical methods. The best known statistical estimator is probably the universal threshold estimator proposed by Donoho and Johnstone [DoJo94a]. In CET, a magnitude threshold is used along with each wavelet coefficient value. This is done in order to generate a numerical evidence estimate for the proposition that the wavelet coefficient is attributable to noise. In [Brow00], this evidence estimate E_1 was given as follows for a wavelet coefficient

$$E_1 = \begin{cases} 0 & |w| > T \\ 1 & w = 0 \\ Q & otherwise \end{cases} \tag{6.15}$$

where

$$Q = \frac{e^{\frac{|W|}{T}} - e}{1 - e} \tag{6.16}$$

The above mapping reflects accurately the expected probability distribution of wavelet coefficients, which is known to have the general form

$$h(v) = N\exp\left(-\frac{|v|}{\alpha}\right)^{\beta} \tag{6.17}$$

where N is a normalization constant and α and β are parameters. Application of the function to the array of wavelet coefficients, generates an array of evidence values in the range 0 to 1.

6.3.1.2 Calculation of Evidence Value from the Wavelet Correlation

The second sub step is to derive a numerical estimate of evidence, for noise contamination, from the correlation between wavelet planes. In this approach the correlation between wavelet planes is treated as providing one source of information from which we can derive a numerical evidence value.

Given a wavelet coefficient set at scale j, the first step is to compute its correlation with wavelet coefficients at the next highest scale (j + 1). This is done simply by multiplying the two wavelet coefficient sets on a point by point basis. For each position we get

$$Corr_j = W_j * W_{j+1} \tag{6.18}$$

Now, we scale the correlation values to take into account that, on average, wavelet coefficient magnitudes decline with increasing resolution level. This is achieved by computing the total power P_{corr} in the correlation values and the total power P_W in the j wavelet coefficient set

$$P_{corr} = \sum Corr_j * Corr_j \tag{6.19}$$

$$P_w = \sum (W_j * W_j) \tag{6.20}$$

The summations are carried out over the full set of wavelet coefficients. The original correlation values are then scaled to give a new set of scaled correlation values as follows

$$Corr'_j = Corr_j * \sqrt{P_w / P_{corr}} \tag{6.21}$$

This scaling step ensures that the total power in the correlation $Corr'_j$, is the same as the total power in the wavelet coefficient set W_j. The minimum absolute value within the set of scaled correlation values is found to be

$$Corr'_{min} = min(|Corr'_j|) \tag{6.22}$$

This value is treated as indicating definite noise, and all positions where $|Corr'_j|$ equals $Corr'_{min}$ are assigned an evidence value E_2 of unity. Any position where $|Corr'_j|$ is greater than $|W_j|$ is treated as indicating that the wavelet coefficient is definitely due to signal features, and an evidence value of 0 is assigned. For the remaining situations where

$|Corr'_j|$ is less than $|W_j|$ but greater than $Corr'_{min}$, an evidence value between 0 and 1 is assigned. In this latter case the value to be assigned is a ratio of the distance between $|W_j|$ and $|Corr'_j|$, and the distance between $|W_j|$ and $Corr'_{min}$. The overall process for assigning evidence values E_2 is as follows

$$E_2 = \begin{cases} 0 & |Corr'_j| \geq |W_j| \\ 1 & |Corr'_j| = Corr'_{min} \\ U & otherwise \end{cases} \tag{6.23}$$

where

$$U = \frac{|W_j| - |Corr'_j|}{|W_j| - Corr'_{min}} \tag{6.24}$$

6.3.1.3 Combining Evidence Values

In the above sections we have obtained two *evidence* values, for each wavelet coefficient at level j. These are then combined to give a single evidence value E for each wavelet coefficient, in the following way

$$E = \sqrt{E_1 * E_2} \tag{6.25}$$

This is the value used subsequently to determine the next step.

6.3.1.4 Smoothing using the Combined Evidence Values

The next stage in the algorithm maps the combined evidence value E to a scaling factor F which takes values from 0 to 1. The process requires two further user-defined evidence thresholds which are parameters of the method, E_{lower} and E_{upper}. Using these values the scaling factor is computed in the following way, for each wavelet coefficient

$$F = \begin{cases} 0 & E \geq E_{upper} \\ 1 & E \leq E_{lower} \\ (E_{upper} - E)/(E_{upper} - E_{lower}) & otherwise \end{cases} \tag{6.26}$$

Once F has been determined, a new wavelet coefficient value W'_j is determined as

$$W'_j = F * W_j \tag{6.27}$$

In [Brow00], it was determined that an upper evidence value of 0.6 was appropriate in most circumstances. To apply the algorithm to a given wavelet plane, one must still provide two thresholds. The first is a wavelet magnitude threshold, which is used in the first stage of processing to compute the E_1 evidence values. The second is the lower evidence value, E_{lower}, which is used in the final step to map the combined evidence value E to the scaling factor F. In the case of the wavelet magnitude threshold, one could use statistical estimators to determine a threshold automatically, but for the purposes of this thesis and as determined in [Brow00] through experiments, fixed thresholds were used. In the case of the lower evidence threshold, E_{lower}, the value 0.1 has been found most appropriate in almost all cases, although a value of 0.05 has sometimes been used when processing the highest resolution wavelet plane.

6.3.2 Wavelet Filter for CET

In [Brow00], CET applied the a`trous algorithm as a basis for generating wavelet transforms. This algorithm is normally implemented using a linear low-pass FIR filter, such as a 5 by 5 B3 spline filter [StMB94], [MuSB95], [Shen92] and [StMB98]. A more sophisticated version of this algorithm uses 5 b 5 B3 spline filter along with two modern detail-preserving non-linear filters. These are a FIR-median hybrid filter [HeNe87], and the more recently introduced Median Rational Hybrid filter [KhGa98]. The linear FIR sub-filters are applied within the sample window, and the output is a median of the sub-filter outputs. The filter used in this thesis is fully described in [KhGa98]. At the first stage the filter is applied to the source image and the resulting smoothed output is subtracted from the source image to give the first wavelet coefficient set. The filter is then dilated by interleaving zeroes (in the case of linear filters), and the enlarged filter is applied to the smoothed output from the first stage. Subtraction then yields the second wavelet coefficient set. The process is repeated for as many resolution levels as are required. The actual filter parameter values that are used in this thesis are discussed in Ch.7.

6.4 Summary

In this chapter, a brief overview of wavelets and their application towards signal and image denoising and smoothing was discussed. Standard wavelet based smoothing techniques with their applications was presented. A detailed decription of the CET algorithm that was used in this thesis to smooth the fractal boundaries was also presented.

In the next chapter the two approaches for modelling object boundaries will be presented where one of the approache employ the wavelet based smoothing technique that was discussed in this chapter.

CHAPTER VII

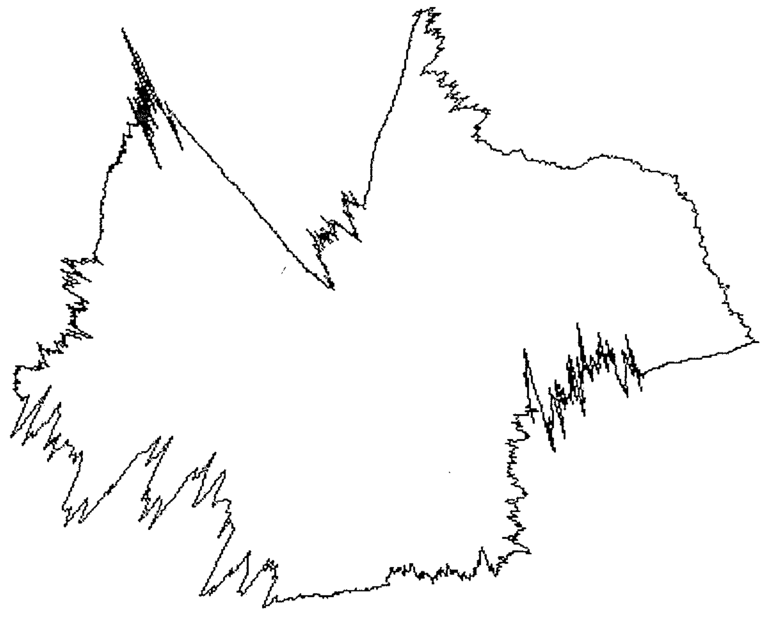
EXPERIMENT DESIGN AND VERIFICATION

7.1 Introduction

In this chapter, the system design for object boundary modelling is presented. First a description of the test images is presented, followed by a description of the modelling approaches. Basically, two approaches are presented in order to compactly represent object boundaries. Each approach is a collection of methods and techniques that are based on wavelets or fractals or both. The first approach is called *wavelet based approach* (WBA) while the second approach is called *IFS based approach* (IBA).

7.2 Test Images

There are two test images used in the thesis. Each test image contains simulated boundary of an object. These boundaries are segmented multifractal in nature, and were constructed using MPD method described in Ch. 3. The fractal dimension for each segment was known beforehand. The first image is called Boundary-1 and is shown in Fig. 7.1a, while the second image is called Boundary-2 and is shown in Fig 7.1b. These boundaries were simulated in Matlab 6.5.


(a)

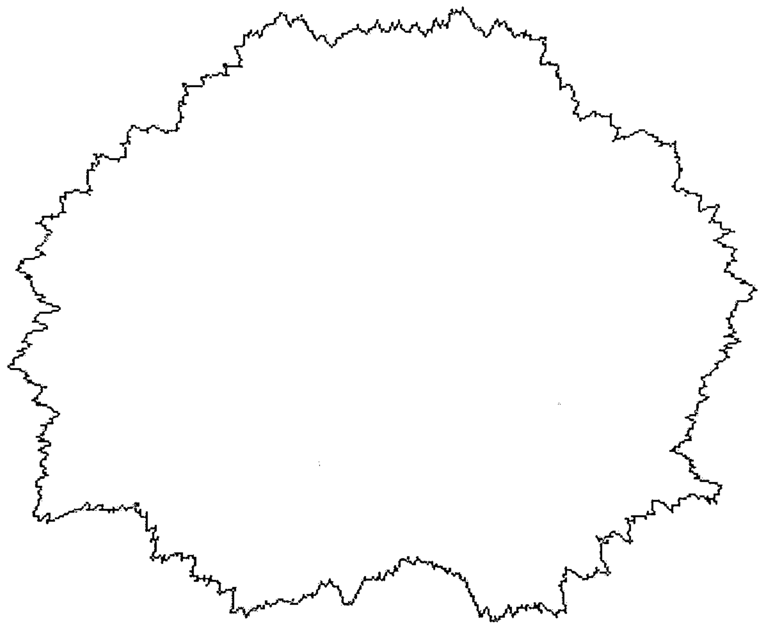

(b)

Fig. 7.1 Two test images used in the thesis.(a) Boundary-1. (b) Boundary-2.

The first image (Boundary-1) contains a complex form of a fractal object boundary. This object boundary was constructed using different fractal dimension for each segment on the boundary. It was constructed in this way because usually natural objects do not have boundary with one fractal dimension and are multifractal. The object boundary of Boundary-1 contains segments that have fractal dimensions ranging from 1 to 1.9, where fractal dimension of 1 depicts a smooth boundary segment and 1.9 as a very complex one. In comparison, the second image (Boundary-2) contains a simpler object boundary that was constructed using a single fractal dimension for each segment.

The two object boundary modelling approaches presented in this thesis can be thoroughly tested and evaluated on the two test images. This is because the two test images contain boundary segments that represent most of the object boundary types that are crisp, rough, fractal, and non-occluded.

7.3 Wavelet-Based Approach

This section presents a detailed description of the wavelet-based approach. Figure. 7.2 shows the various steps for analysis and synthesis of modelling object boundaries used in this approach. This approach is divided into analysis and synthesis, as explained next.

This section explains the analysis part of the wavelet based approach where the object boundary is reduced to sets of control points and edge points with fractal dimension

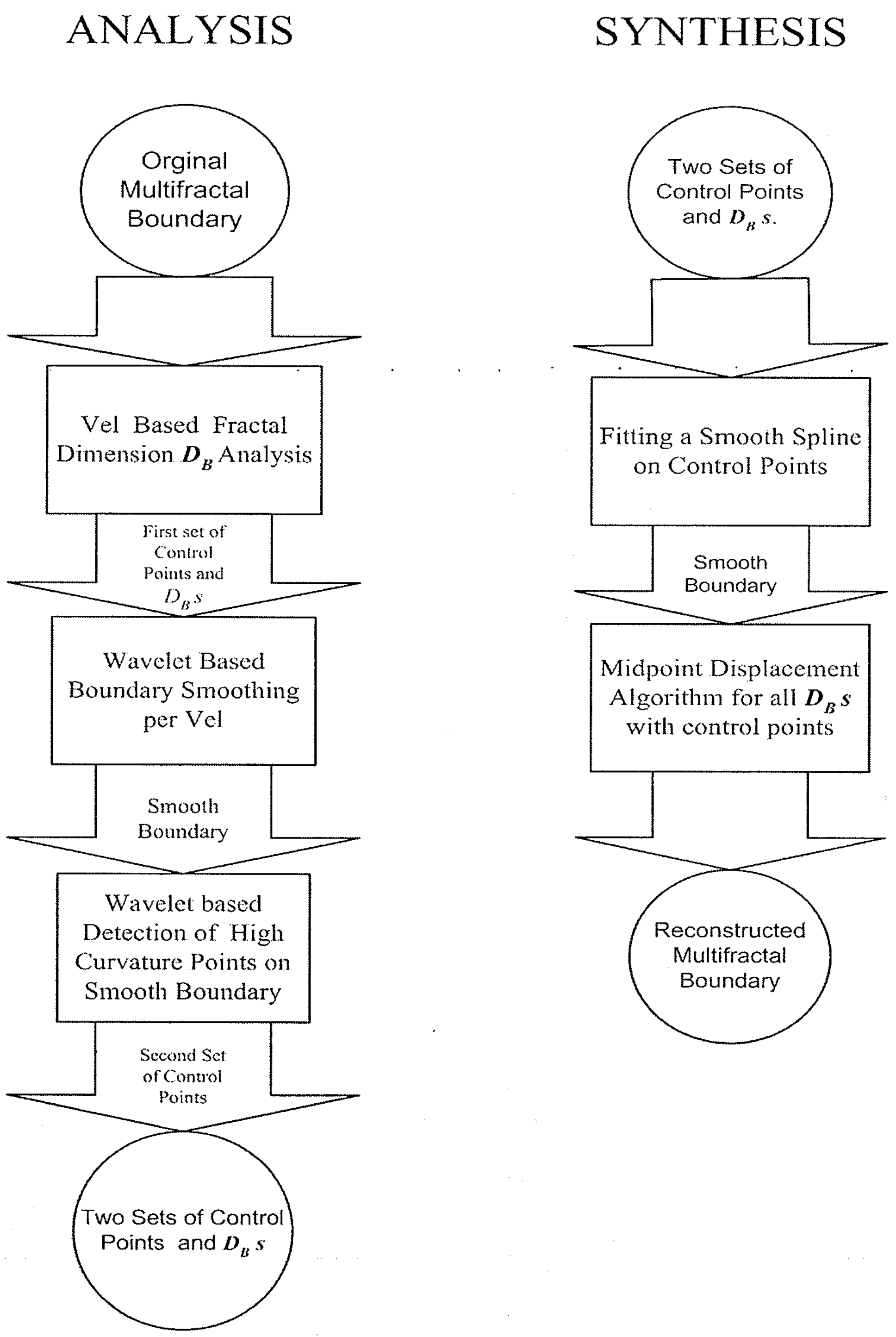

ANALYSIS
SYNTHESIS
Orginal Multifractal Boundary
Vel Based Fractal Dimension D_B Analysis
First set of Control Points and $D_B s$
Wavelet Based Boundary Smoothing per Vel
Smooth Boundary
Wavelet based Detection of High Curvature Points on Smooth Boundary
Second Set of Control Points
Two Sets of Control Points and $D_B s$
Two Sets of Control Points and $D_B s$.
Fitting a Smooth Spline on Control Points
Smooth Boundary
Midpoint Displacement Algorithm for all $D_B s$ with control points
Reconstructed Multifractal Boundary

Fig. 7.2 Wavelet-based approach.

values. First, the number of points on the boundary is reduced by representing the local roughness with compact fractal descriptors, along with related edge points. Finally, a set of salient control points can be found on the smoothed version of the rough crisp boundary that will preserve the underlying shape of the object.

7.3.0.1 Local Fractal Dimension Analysis

This is the first step in the analysis of the object boundary. In this thesis, two approaches were tested for this step (i) *static vel covering* and, (ii) *moving vel.*

Static Vel Covering

The boundary of an object is first covered by a set of vels. The D_β is then calculated for each fractal structure present in specific vels. The boundary of an object is first covered by a set of uniform vels. The motivation for this technique comes from methods that use similar morphological geometrical arrangement of the vels along the boundary is represented by chain codes [GoWo92]. Since the chain codes can only work on one vel wide boundary representation, the thinning algorithm [GoWo92] is first applied on vels. This technique can be implemented easily and is computationally fast for large number of boundary points.

The accuracy of D_β, like all other fractal dimensions, depends upon the number of points contained in a vel. Therefore, 5% of the total number of boundary points was found experimentally as adequate for accurate calculation in each vel. It was also determined experimentally that the minimum number of points that are required for accurate calculation is 512. The main idea behind this algorithm is to find the D_β for minimum number of vels that would have sufficient number of points. This algorithm is implemented using the following six steps:

Step 1. A set of vels is used to cover the fractal boundary, where the size of each vel was chosen to be at most 1/10 of the maximum height and width of the boundary. These vels (not boundary points) are the fundamental units for further processing.

Step 2. The number of boundary points in each vel is calculated and vels are marked according to the following rules:

(a) Empty **E** vels are eliminated from further calculations.

(b) The vels that have fewer than 0.3% of the total number of the boundary points are designated as sparsely populated S vels. The set of all S_j vels is denoted by **S**.

(c) If a vel has two or more geometrically independent boundary segments then this vel is designated as a multiple segment M_j vel. The set of all M_j vels is denoted by **M**.

(d) A thinning algorithm is then applied on all remaining vels except **S**, and the information about the discarded D_j vels is stored. The set of all D_j vels is denoted by **D**.

(e) The 8-connected chain code algorithm [GoWo92] is applied on the remaining vels (non **S** and non **D**) to obtain a one-dimensional representation of the boundary. The vels forming the chain codes are called C_j vels.

Step 3. The D_β is calculated for all the **C** vels starting from the first vel in the chain code. At least 512 points per vel are required for calculation accuracy, otherwise vels are merged (as described below) to form larger C vels to have more points.

Step 4. The D_β is calculated for all the **C** vels.

Step 5. The D_β in the C_j vel is compared with the previous C_{j-1} vel, and if the difference between their values is less than a certain threshold K, then the two C vels are considered to have the same D_β.

Step 6. The coordinates of the edge points of the fractal boundary in each **C** vel are also saved.

The vels are eventually merged according to the following general rules:

(a) If two vels have to be merged then the edge boundary points of one vel should be directly linked with the edge boundary points in the other vel.

(b) Merging vels should not be part of any previous merger (not applicable for **M** vels).

Each C_j vel is merged with another vel according to the above rules:

- If an adjoining D_j or D_jM_j vel exists then it is merged with the C_j vel along with any Sj or S_jM_j vel as mentioned above.
- If a C_j vel under consideration has fewer than 5% of total boundary points, then it is merged with the next C_{j+1} vel in the chain code, along with any adjoining D_j or S_j vels.

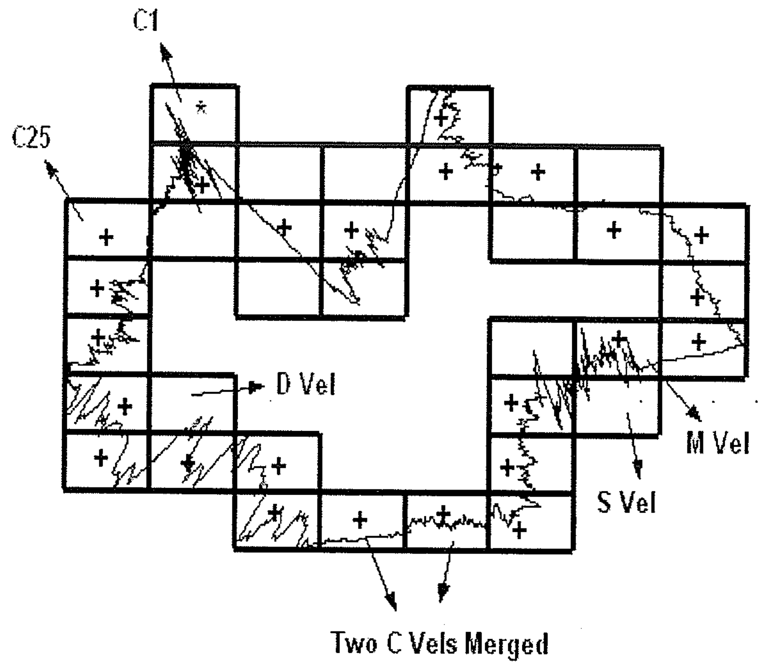
After each merger, the new larger vel is designated as C_j while the information about any M_j vel in it is retained for next calculations.

The above fractal analysis technique using static vel covering is shown in Figs. 7.3a and 7.3b.

Moving Vel

This technique for boundary analysis uses a moving vel in order to traverse the boundary and calculate local fractal dimension D_β. The size of the vel is chosen in such a way that there are atleast 512 points in the vel for the accurate calculation of D_β. First, a starting location on the boundary is chosen which can be and random point on the boundary. Then the vel traverses the boundary in clock wise or counter clock wise fashion without any overlapping. The traversing is stopped when the vel reaches the starting point. For every vel step the boundary edge points included in that vel need to be determined. This is done by doing a search in the neighbourhood of 3 pixels around the previously determined point. If no point is found then the neighbourhood size is increased by 3 pixels

(a)



(b)

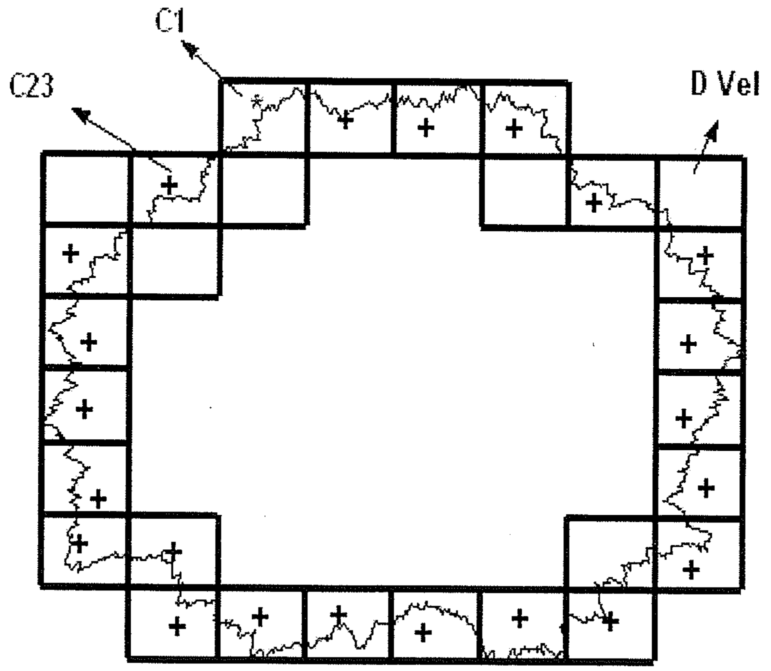


Fig. 7.3 Static vel covering for both the boundaries. (a) Vel covering for Boundary-1.(b) Vel covering for Boundary-2.

and so on. The boundary edge points and the value of D_β are stored just like in the static vel covering method for each vel step. This technique is computationally intensive and only work when there is one pixel wide boundary and no occlusion. The moving vel technique is shown in Figs. 7.4a and 7.4b.

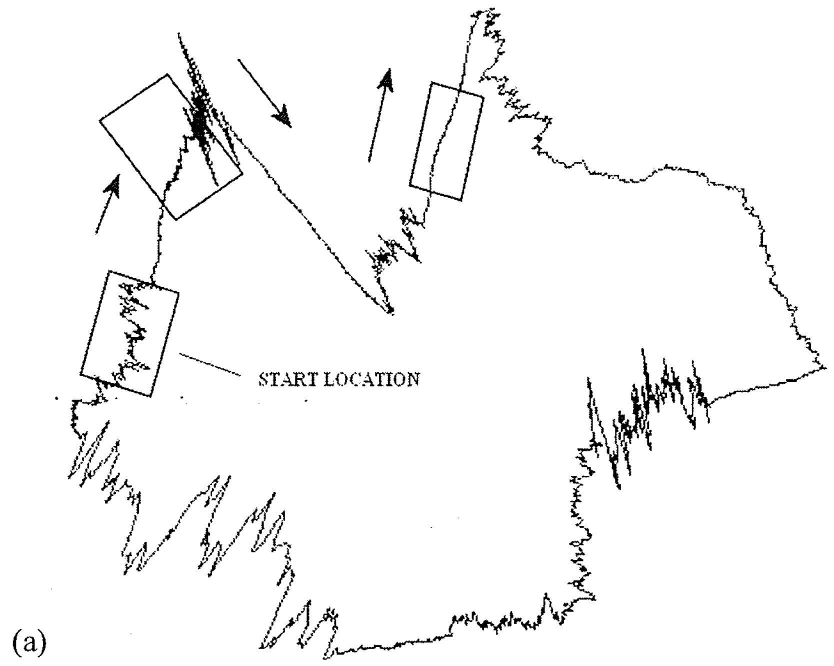
For complex shaped boundaries such as Boundary-1, the static vel technique is recommended. This is because moving vel technique can include boundary points that are not part of the boundary segment inside the given vel during its search process. Due to this reason more emphasis is given on static vel technique for this thesis.

7.3.0.2 Boundary Smoothing

The second step in the fractal boundary analysis is the smoothing of the boundary. The fractal boundary requires smoothing in order to locate shape defining control points. For this purpose the wavelet based smoothing technique called CET discussed in Ch. 6 is applied on each segment of the boundary covered by the vel.

7.3.0.3 Detection of Control Points on a Smooth Boundary

The last step in the fractal boundary analysis is the detection of control points on the smoothed boundary. This technique is described in Ch. 5. Before applying this technique we have assumed that rough boundary is mostly smoothed in the earlier step. Since the dominant information regarding shape is usually available at the *high curvature points* (HCPs), which become corners (singular points) in the limit, they provide important



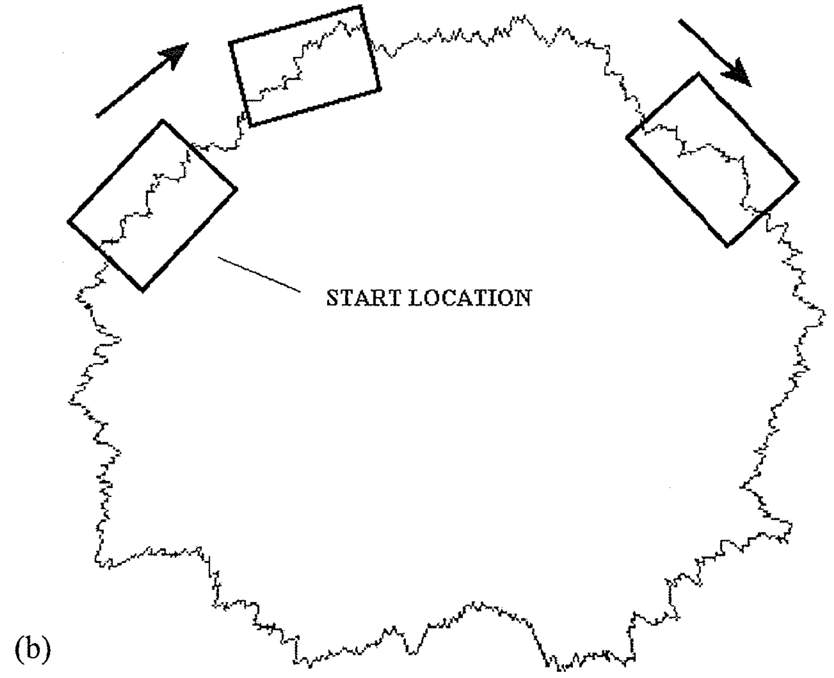


Fig. 7.4 Moving vel covering for both the boundaries. (a) Vel movement for Boundary-1.(b) Vel movement for Boundary-2.

features for object recognition, shape representation, image interpretation and motion analysis. These HCPs form the control points on smooth boundaries and fractal curves.

As discussed in Ch. 5, the detection of singularities can be done using wavelet transform. Wavelet decomposition provides a natural setting for the multi-level image contour analysis, as it is able to detect singularities efficiently [MaZh92]. Since WTMM provide precise and useful information for curvature analysis [CQGa00a][CQGa00b][QCGa99][QCGa00][QuGa00][QuGa02], it is used in this paper to extract control points on smoothed boundaries. As mentioned in Ch.5, biquadratic wavelets perform better than other wavelets for corner detection applications [MaZh92][QuGa02].

In this step, the boundary is tracked and its orientation profile is computed as discussed in Ch.5 and in [QuGa02][SiKi02]. The wavelet transform of the orientation profile is computed for dyadic scales from 2^1 to 2^6. The WTMM are then computed, and only those WTMMs larger than a certain threshold are considered to be the important curvatures. The result of the corner detection scheme where $\tau_1 = 0.4$, $\tau_2 = 0.1$ and $\tau_3 = 0.65$ are discussed in the next chapter.

The HCPs detected as a result of the above technique form the second set of points that are the smoothed boundary control points. In case, more then one control point is detected for a small segment of the boundary such as in the neighbourhood of 10 pixels, then only one control point is chosen from the cluster of control points. The control point at the nearest distance from the mean of all control point positions is the chosen control point.

7.3.1 Synthesis

In this section we discuss the method of reconstructing the fractal boundary by given sets of edge points with their D_β s and control points. In order to reconstruct the fractal boundary of similar roughness as the original, we first draw a smooth spline curve [RoAd90] [SiKi02] on the second set of control points in order to determine the underlying shape of the complex object boundary. After obtaining a smooth boundary we superimpose a fractal curve on it. We use the MPD technique [Kins95] to construct the fractal curve of a given fractal dimension. The first set of edge points with their D_β s are used for this purpose.

7.4 IFS-Based approach

This section provides a detailed description of the IFS-based approach. Figure 7.5 shows the various steps for analysis and synthesis of modelling object boundaries for IFS -based approach. This approach like the first approach is divided into analysis and synthesis, as explained next.

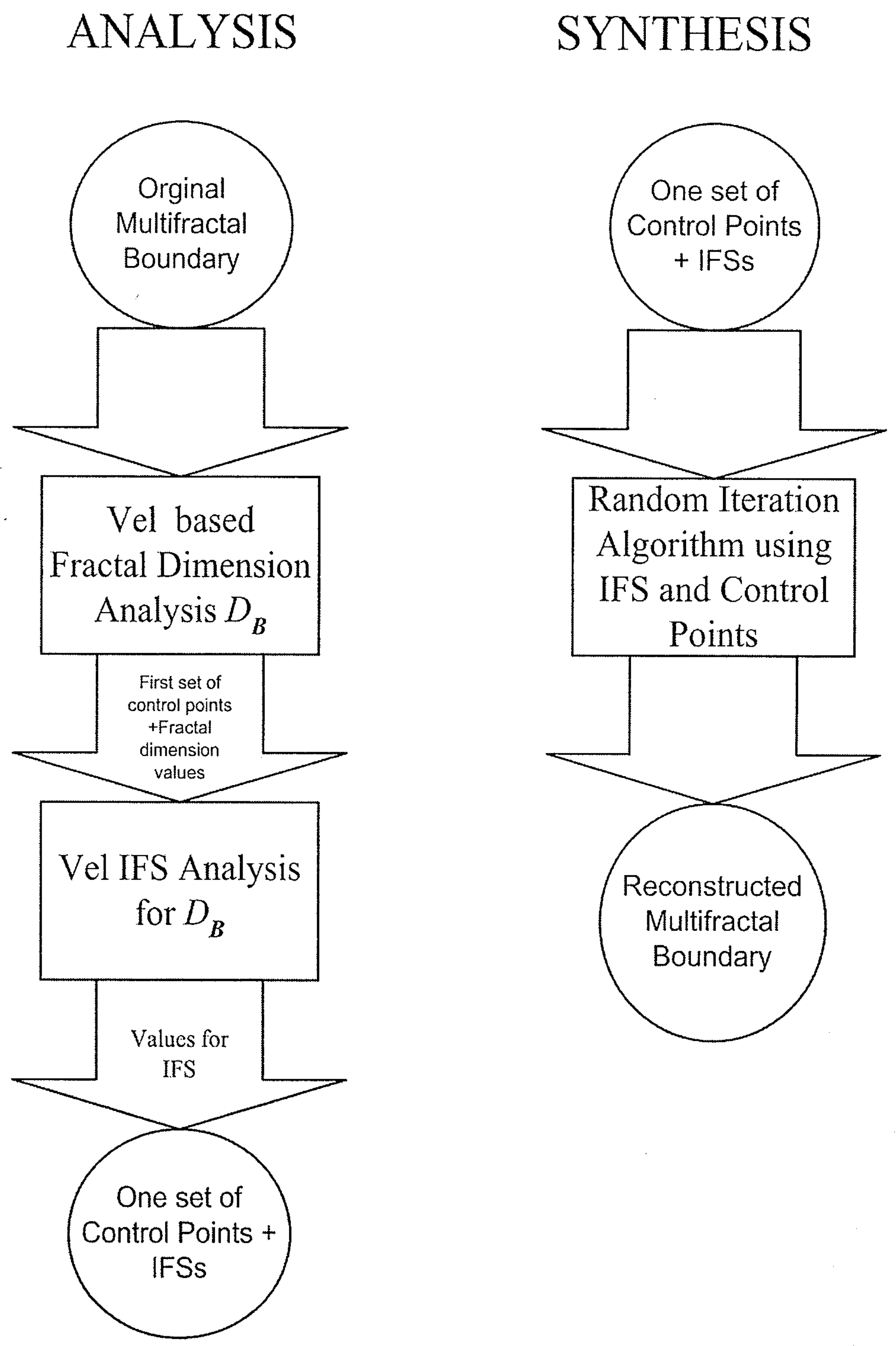

ANALYSIS
SYNTHESIS
Orginal Multifractal Boundary
Vel based Fractal Dimension Analysis D_B
First set of control points +Fractal dimension values
Vel IFS Analysis for D_B
Values for IFS
One set of Control Points + IFSs
One set of Control Points + IFSs
Random Iteration Algorithm using IFS and Control Points
Reconstructed Multifractal Boundary

Fig. 7.5 IFS-based approach.

7.4.1 Analysis

This section explains the analysis part of the IFS-based approach where the object boundary is reduced to sets of IFS values and edge points. In this approach, the number of points on the boundary is reduced by representing the local roughness with IFS descriptors, along with related edge points. Unlike the first approach there is no smoothing involved in the IFS based approach.

7.4.1.1 Local Fractal Dimension Analysis

Similar to the first approach, this approach uses one of the two vel based methods (discussed in sec. 7.3.1.1) for the fractal analysis of the object boundary. The calculation of D_β for each fractal structure present in a specific vel is also done the same way as in WBA.

7.4.1.2 Calculation of Local IFS

The IFS for each fractal structure in the vel is obtained according to D_β calculated in the earlier step. It was found that, in general, IFS obtained using at least four control points for each vel fractal structure on the boundaries of Fig. 7.1 was adequate for accurate description. Thses control points are equally spaced on the fractal structure. We use linear IFS (LIFS) for simple fractal structures and hidden-variable IFS (HVIFS) for more complex (having curvatures) ones. The number of control points for complex fractal structures can be increased. Also, it is important to keep in mind that these control points are only used for calculating the IFS for each fractal structure. Some of them can form the edge points for the given fractal structure.

At the end of this analysis step the boundary is reduced to edge points for each vel fractal structure and the related IFS values.

7.4.2 Synthesis

This section explains the synthesis part of the IFS based approach where the object boundary is reconstructed from the sets of IFS values and edge points obtained from analysis. Unlike the first approach the synthesis part of this approach is straight forward and only contains one step.

7.4.2.1 Boundary Reconstruction using RIA.

In order to reconstruct the fractal boundary of similar roughness as the original, we use RIA technique discussed in Ch. 3 to construct the graph of the attractor of the given IFS within each vel and then joining them by using the edge points. This graph forms the reconstructed boundary. In case, there were multiple boundary segments in a vel then multiple graphs are constructed.

7.5 Summary

In this chapter the two approaches for multifractal object boundary modelling were presented. A brief overview of the test images used for the approaches was also presented. In the following chapter the results obtained from the two approaches will be discussed and the two approaches will be compared in terms of performance and compression.

CHAPTER VIII

EXPERIMENTAL RESULTS AND DISCUSSION

8.1 Introduction

This chapter presents the results obtained by applying both the object boundary modelling approaches discussed in Ch.7. The results for the calculation of fractal dimension D_β within the vels for both approaches are presented first. In the subsequent sections, the results from analysis and synthesis steps for each approach are discussed. An overall comparison between the result from the two approaches completes this chapter.

8.2 Spectral Fractal Dimension D_β Results

The calculation of D_β is the first and fundamental step for both approaches discussed in previous chapter. Since this step is common for both approaches therefore results from this step are described first.

8.2.1 Static Vel Covering

The algorithm described in Ch.7 for the covering of the fractal boundary by set of uniform vels was implemented in Matlab 6.5 and its code is described in Appendix A by function vel_covering.m. Figures 8.1 and 8.2 show the fractal boundary of Figs.7.1a and 7.1b respectively, covered by a set of 25 vels initially. As it can be seen, few of the ***C*** vels were merged according to the rule stated in Sec. 7.2.3, and the spectral fractal dimension

D_β was calculated after the merging of these vels. There are some empty ***E*** vels and discarded ***D*** vels.

Result of the spectral fractal dimension D_β in the vels are shown in Tables 8.1 and 8.2, respectively. As it can be seen, the values for fractal dimension D_β are less then 2 which is expected since the boundary segment inside the vel is embedded in a space with topological dimension $E = 2$. The different values of D_β directly correspond with local complexities of the boundary. For example, it can be seen in Table 8.1 that D_β for vel **C1**, which is more complex, is higher than vels **C2** and **C3**. It can also be seen that the values of D_β in both the Tables are within the error of 0.1

Since the power spectrum density is symmetric, the algorithm for calculating D_β is simple to implement, and gives accurate results. The calculation of D_β was discussed in Ch. 3. These calculated values for D_β were used for both approaches discussed in Ch. 7. Additional corresponding results are presented in the subsequent two sections.

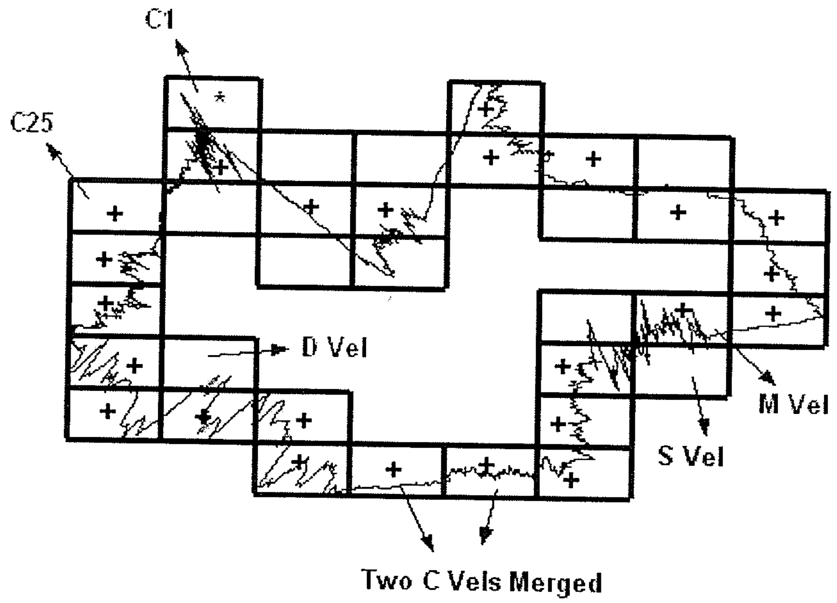


Fig. 8.1 An analysis of a multifractal boundary, where symbols '+' represent 'C' vels and '*' starting C vel. Symbols C1, C2, C3 to C23 show the order of 'C' vels in the chain code. No 'S' vels are shown here.

Table 8.1: Comparison between calculated D_β and actual D_β for different *C* vels. Some vels were merged before calculation.

	C VELS						
	C1	**C2&C3**	**C4**	**C5**	**C6&C7**	**C8&C9**	**C20**
Calculated	1.57	1.32	1.376	1.43	1.23	1.69	1.19
Actual	1.6	1.35	1.4	1.45	1.2	1.65	1.2

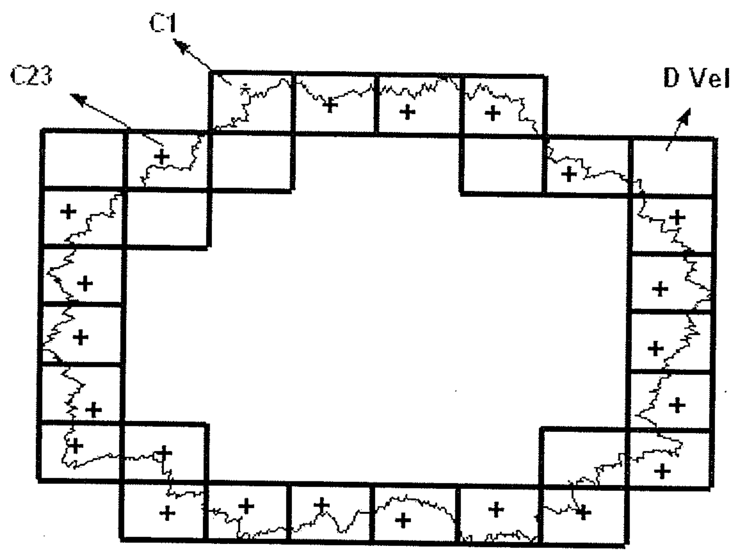


Fig. 8.2 An analysis of a multifractal boundary, where symbols '+' represent 'C' vels and '*' represent starting C vels. Symbols C1, C2, C3 to C23 show the order of 'C' vels in the chain code. No other vels beside D are shown here.

Table 8.2: Comparison between calculated D_β and actual D_β for different C vels. Some vels were merged before calculation.

	C VELS						
	C1	**C2&C3**	**C4**	**C5**	**C6&C7**	**C8&C9**	**C20**
Calculated	1.7	1.68	1.81	1.74	1.79	1.73	1.76
Actual	1.8	1.8	1.8	1.8	1.8	1.8	1.8

8.3 Wavelet-Based Approach

8.3.1 Analysis Results

The experiments were conducted to analyze a simulated multifractal boundary and then to reconstruct the boundary that is visually similar to the original. The HCPs were detected as a result of the wavelet based technique discussed in Ch. 5 and they form the set of the smoothed boundary control points. These control points and smoothed boundaries are shown in Figs. 8.3a and 8.3b respectively. Table 8.3 and 8.4 show the total number of boundary points for Boundary-1 and Boundary-2, respectively, after the analysis step. Here it is assumed that it takes one byte to store the value of each position co-ordinate of the control point. Similarly each value of D_β is also stored in one byte. It can be observed that some control points were not included in the final result as those points were omitted due to close proximity with each other. These extra points are also visible in Fig. 8.3. As discussed in previous chapter only one control point was chosen from a group of closely situated points.

It can be seen that each boundary is now represented compactly by the two sets (Edge Points with their D_β s and Control Pts). The total number of points in the original boundary are also shown.

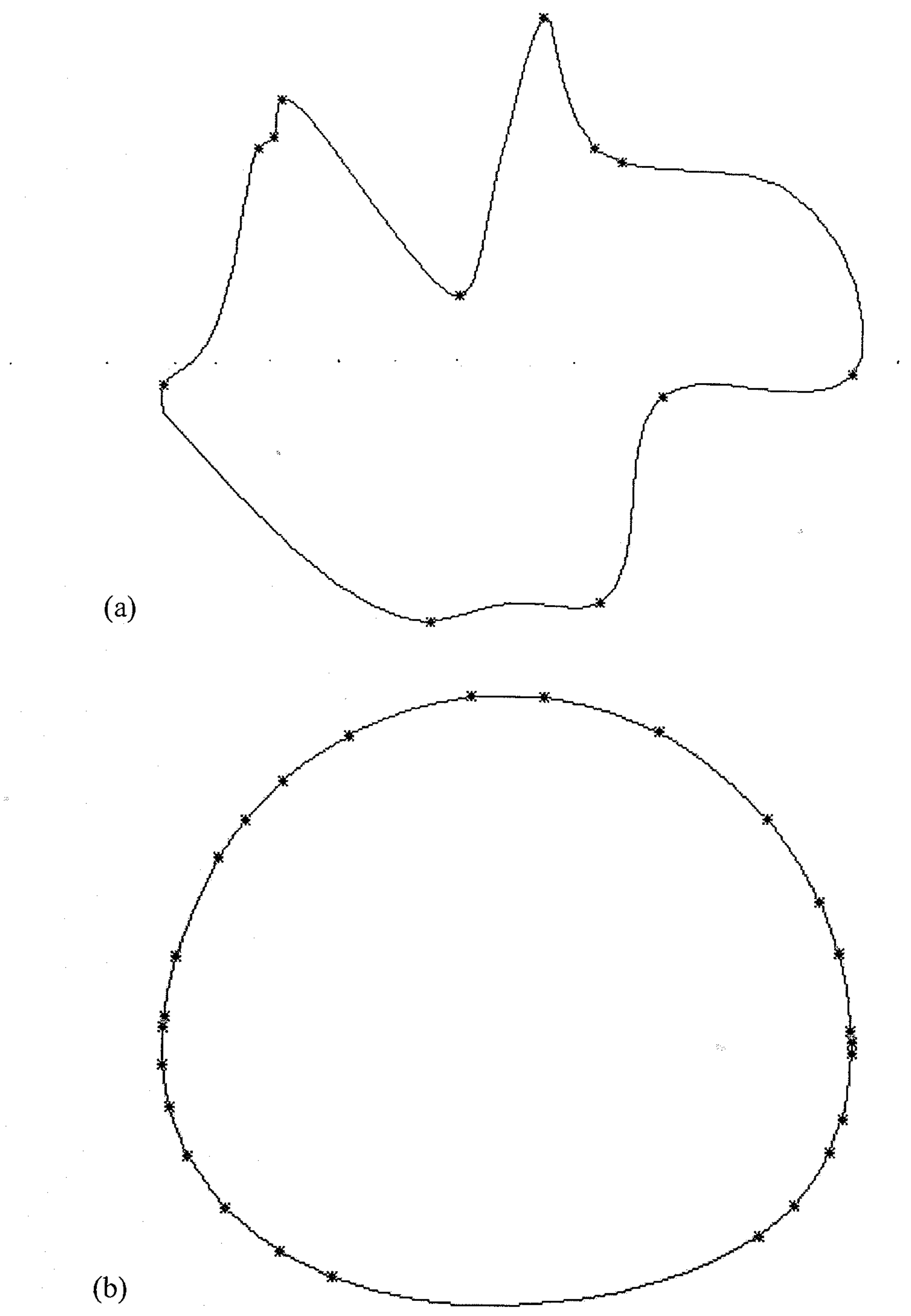


Fig. 8.3 Wavelet detection of control points on smoothed boundary. '*' shows originally detected control points. (a) Boundary-1. (b) Boundary-2.

Table 8.3: Result after analysis step for Boundary-1 using wavelet-based approach.

Number of Boundary Points (Bytes)			
	Edge Points with their D_β s and Control Points		
Original	**Set 1 (C vel)**	**Set2**	**Total**
14759x2 = 29518	23 (D_β s) +27x2	13x2	103

Table 8.4: Result after analysis step for Boundary-2 wavelet-based approach.

Number of Boundary Points (Bytes)			
	Edge Points with their D_β s and Control Points		
Original	**Set 1 (C vel)**	**Set2**	**Total**
6000x2 = 12000	18 (D_β s) +18x2	7x2	68

8.3.2 Rényi Dimension

A Rényi fractal dimension spectrum was obtained for the reconstructed fractal boundaries in order to evaluate the performance of the wavelet-based approach. Figures 8.4 and 8.7 show the original boundaries for the two test images and their respectively Rényi spectrum. The reconstructed boundaries and their Rényi spectrum are shown Figs. 8.5 and 8.8 respectively. Figure 8.6 and 8.9 show the comparison between the Rényi spectrum of the original boundary and the reconstructed boundary for Boundary-1 and Boundary-2, respectively. It can be seen from the comparison that the Rényi spectra of the original multifractal boundary and the reconstructed boundary are almost identical. However, it can be seen from Fig. 8.9 that the reconstructed Rényi spectra for Boundary-2 differs from the original for the negative values of q. Notice that a single fractal boundary is represented by a line at $Dq = 1.2$. This result shows that the reconstructed fractal boundaries are similar in complexities to the original boundary for positive values of q and they look very similar to each other when observed by the human eye.

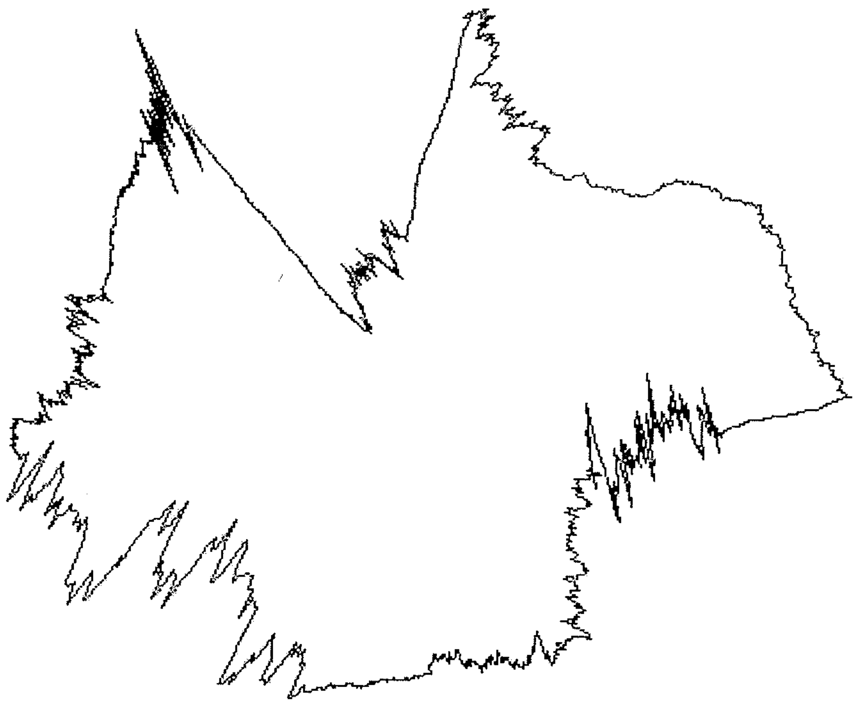
The Rényi spectrum can also be used to determine the optimum size of the vels (Sec. 7.2) for accurate calculation. This can be achieved by minimizing the least squared error between the Rényi spectra of the original boundary and the reconstructed one. The error minimization is done to a certain threshold value while the size of the vels is reduced accordingly. This technique can be further improved by calculating for each vel and plotting them with Rényi spectrum of the original boundary. The size of the vel is then reduced by minimizing the least squared error between the vel and the calculated by the Rényi spectrum. This technique would lead to a covering with variable vel sizes.

The Rényi spectra of Boundary-1 in Fig. 8.6 are almost identical because the number of monofractal segments and related D_β with control points of the original boundary is known beforehand, therefore, the size of the vel was chosen accordingly. However, this was not the case for Boundary-2.

8.3.3 Compression Ratio

Table 8.1 and 8.2 shows that multifractal boundaries can be represented very compactly using this technique. The original multifractal boundary of Fig. 7.1a had 14759 points while the one of Fig 7.1b had 6000 points with each point having xy co-ordinates. After the analysis, two sets of edge points with their s and control points were obtained. The first set contains values in C vels with related boundary edge points (xy coordinates). The second set contains control points obtained after wavelet analysis of the smoothed

(a)



(b)

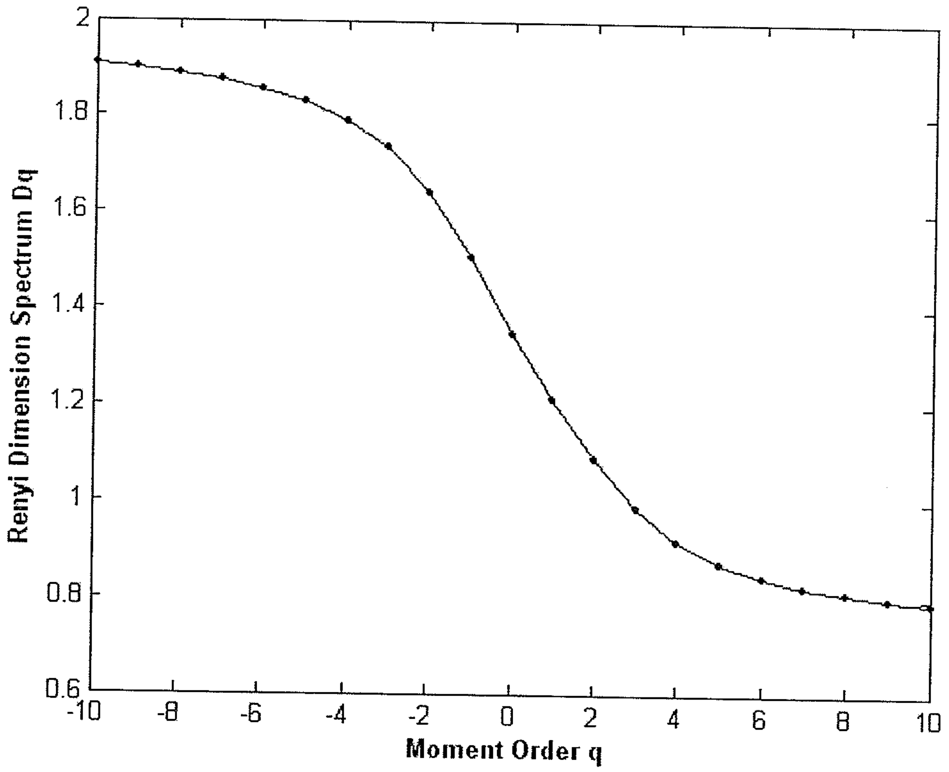
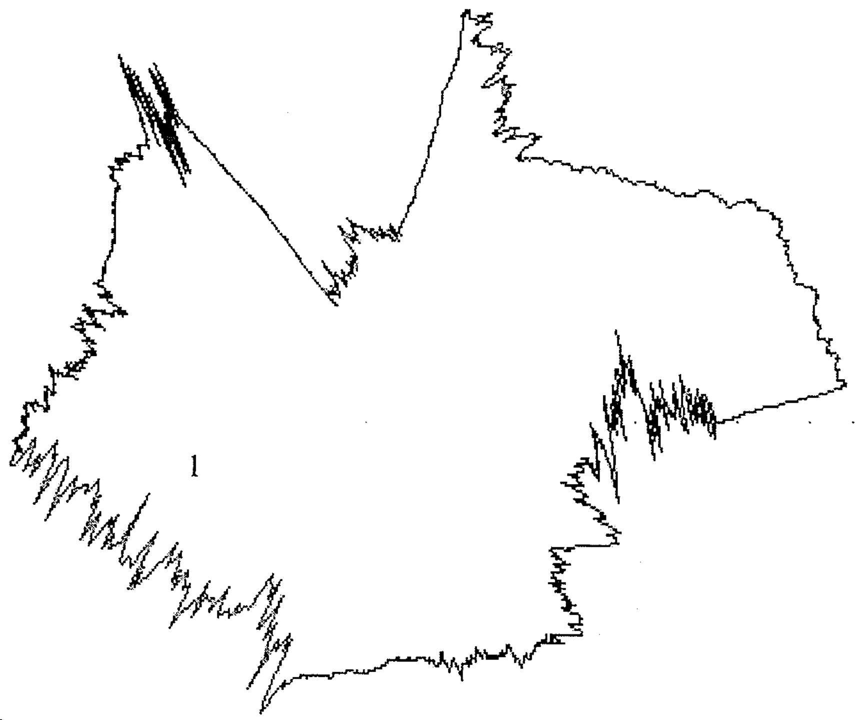


Fig. 8.4 (a) Original boundary (Boundary-1) and, (b) its Rényi dimension spectrum.

(a)



(b)

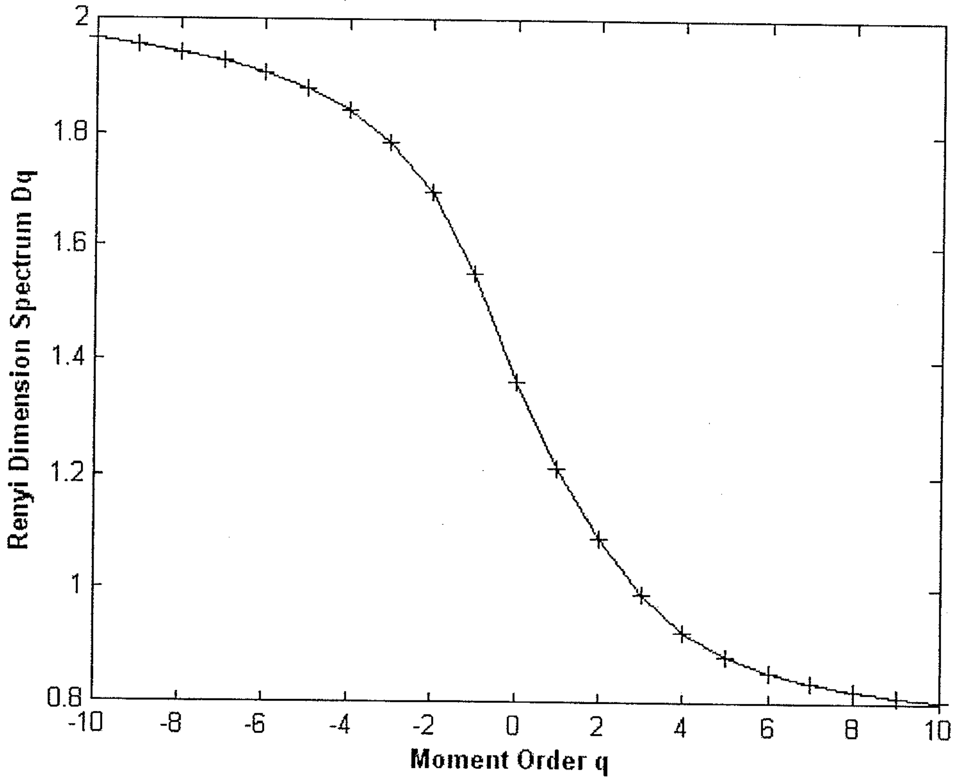


Fig. 8.5 (a) Reconstructed boundary (Boundary-1) using wavelet based approach and, (b) its Rényi dimension spectrum.

boundary. The compression ratio obtained for the current resolution of Boundary-1 is equal to **286.6:1**. while the one for Boundary-2 is **176.4:1**.

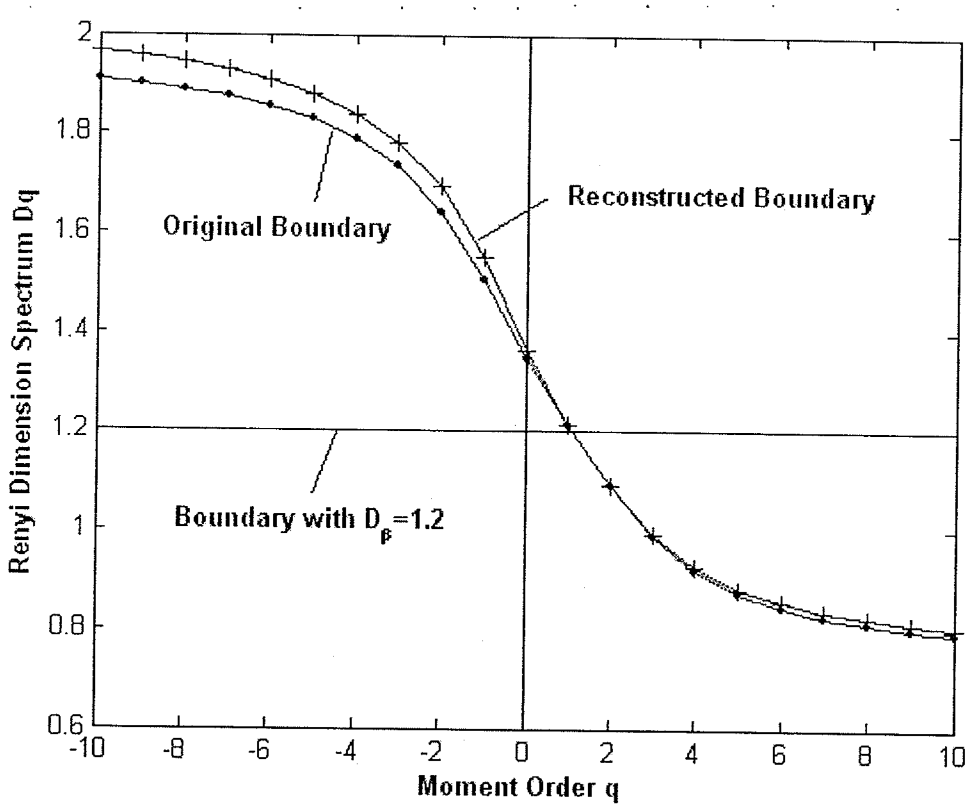
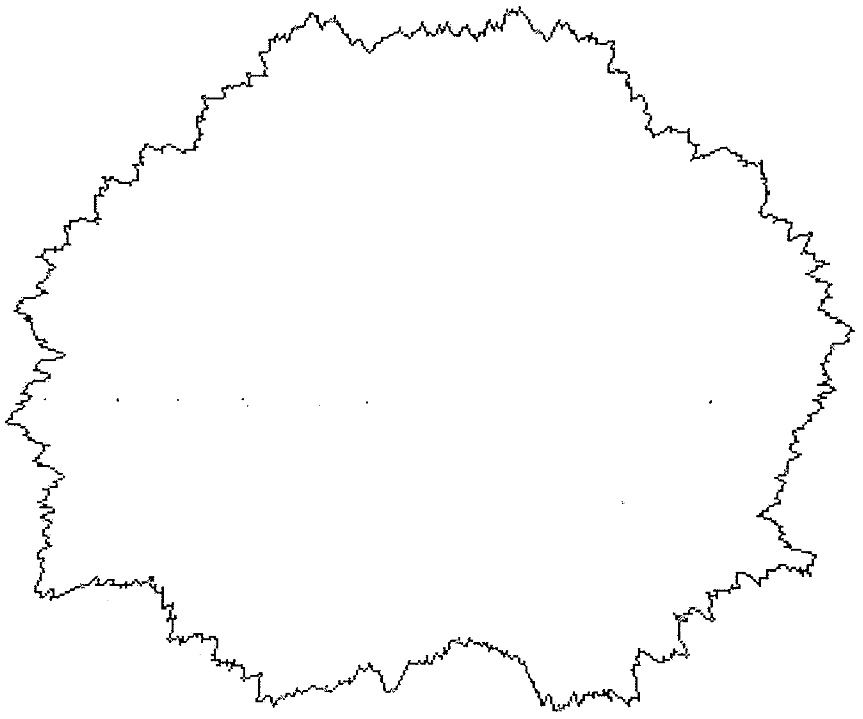


Fig. 8.6 Comparison of Rényi dimension spectrums of original boundary and reconstructed boundary for Boundary-1 using wavelet approach.

(a)



(b)

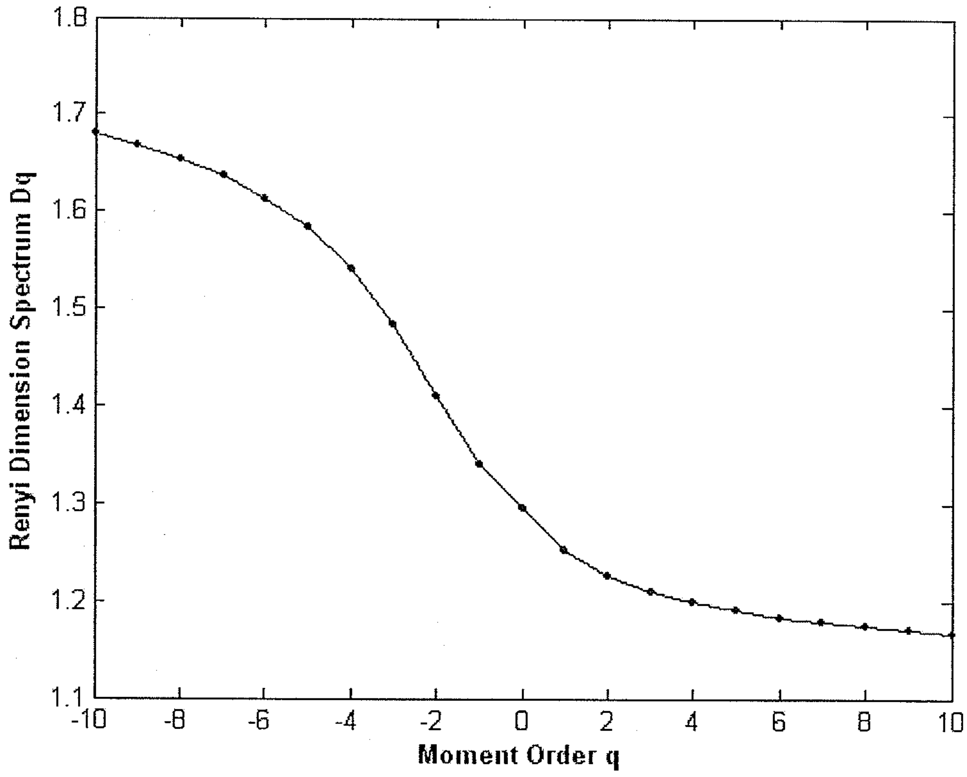


Fig. 8.7 (a) Original boundary (Boundary-2) and, (b) its Rényi dimension spectrum.

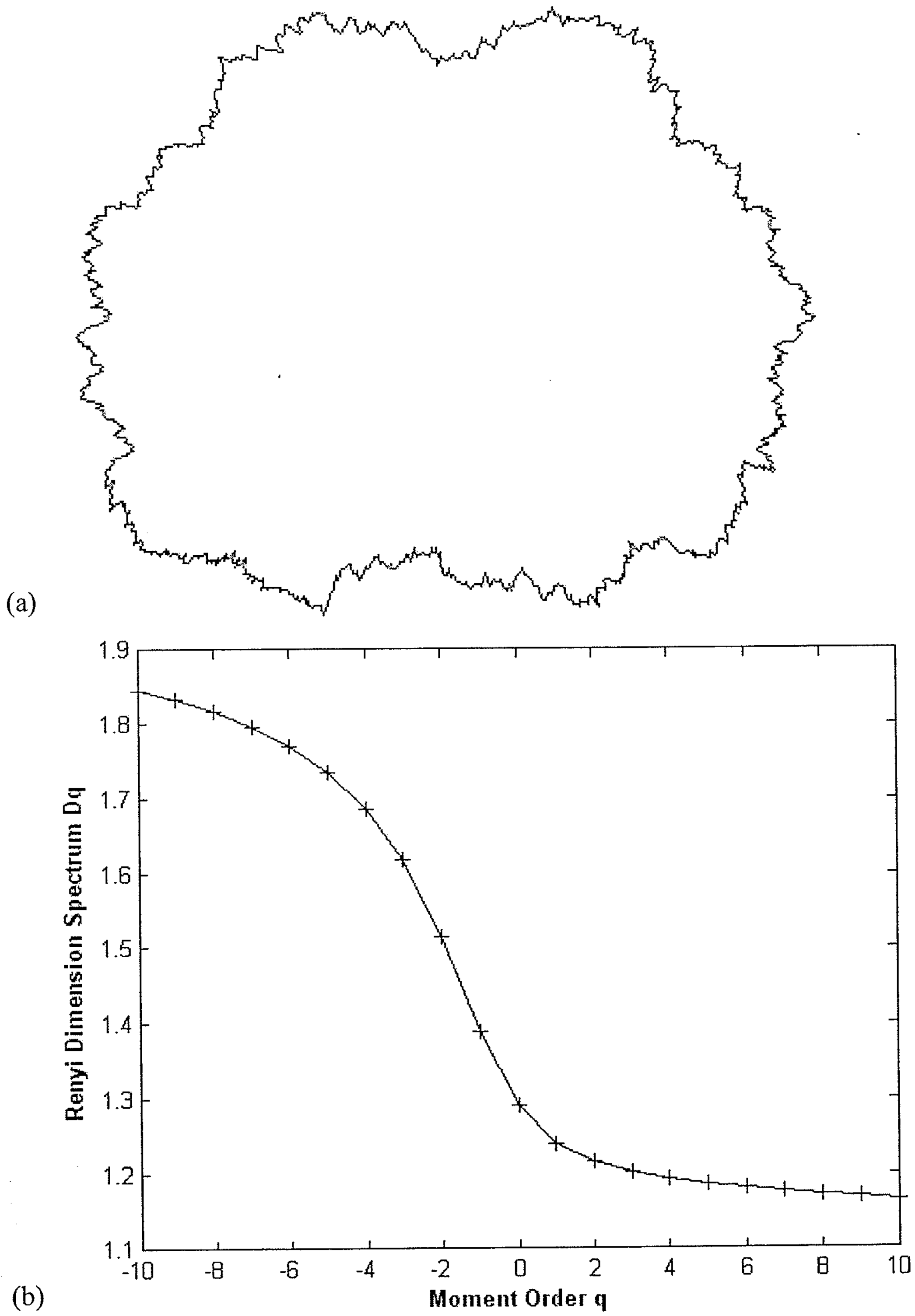


Fig. 8.8 (a) Reconstructed boundary (Boundary-2) using wavelet based approach and, (b) its Rényi dimension spectrum.

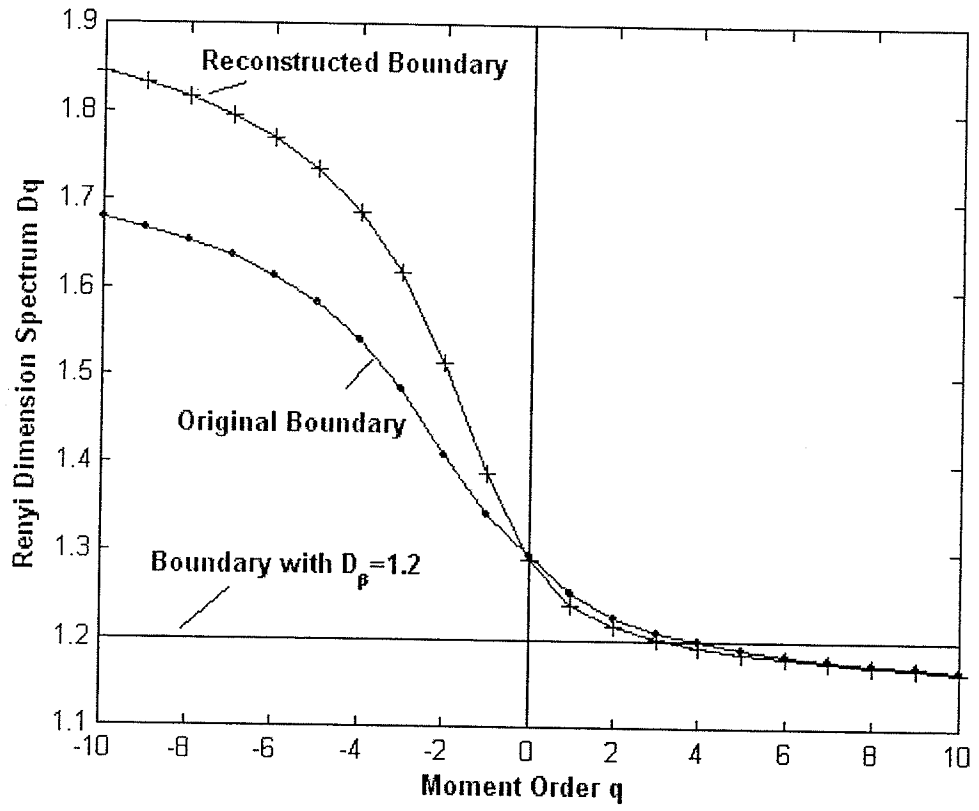


Fig. 8.9 Comparison of Rényi dimension spectrums of original boundary and reconstructed boundary for Boundary-2 using wavelet approach.

8.4 IFS-Based Approach

8.4.1 Analysis Results

The experiments were conducted to analyze a simulated multifractal boundary and then to reconstruct the boundary that is visually similar to the original one using the IFS based frame work The motivation for this technique comes from methods that use similar morphological covering to calculate the box counting fractal dimension [TuAB98]. The significance of our technique is that we are able to partition the boundary for calculation of local D_βs and IFS. The arrangement of vels is described in detail in Sec.7.2, and is identical to first step of the wavelet based approach. The IFS for each fractal structure in the vel is obtained according to D_β calculated earlier. It was found that, in general, IFS obtained using at least four control points for each vel fractal structure on the boundaries of Figs. 7.1a and 7.1b, was adequate for accurate description. We use linear IFS (LIFS) for simple fractal structures and hidden-variable IFS (HVIFS) for more complex (having curvatures) ones. The number of control points for complex fractal structures can be increased. The geometrical arrangement of the vels along the boundaries of Figs. 7.1a and 7.2b is shown in Figs.8.10 and 8.11, respectively, and LIFS calculated for three of the vels is shown in Tables 8.5 and 8.6, respectively.

Tables 8.7 and 8.8 show the total number of boundary points for Boundary-1 and Boundary-2 respectively after the analysis step. It can be seen that each boundary is now compactly represented by the edge points with linear IFS parameters. In this the IFS-based approach there is no need to store the values of D_β s. The total number of points in the original boundary are also shown.

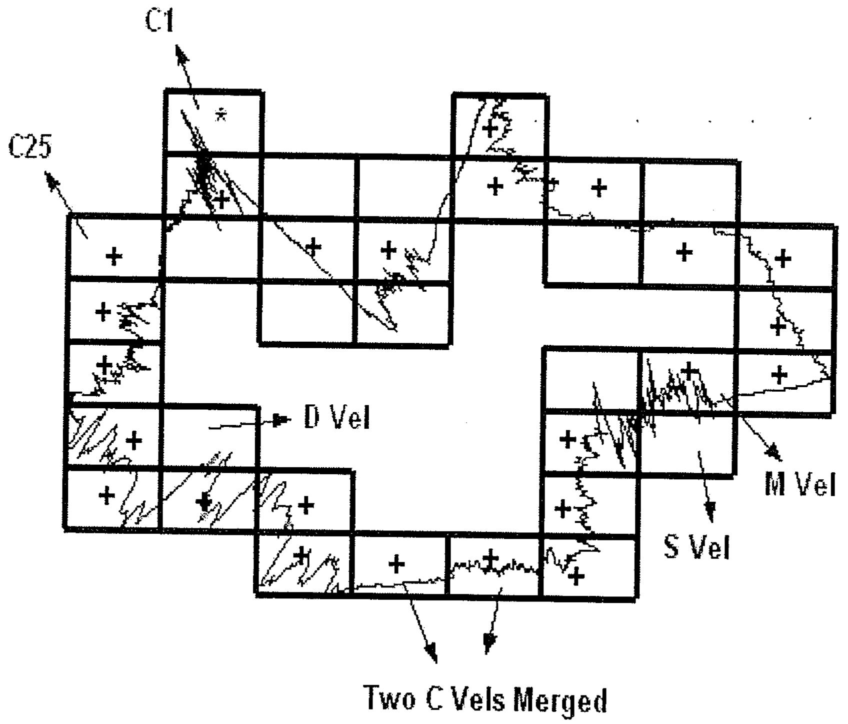


Fig. 8.10 IFS analysis per each C vel (shown by '+') for Boundary-1.

Table 8.5: Linear IFS calculation for a given SFD for 3 of the C vels in Fig. 8.4. Each vel had 4 control points for calculation of IFS.

Vel No.	**LINEAR IFS**					
	a	**b**	**c**	**d**	**e**	**f**
C24	0.1215	0	1.7595	-0.1897	0	13.2421
	0.6623	0	1.1258	-0.3130	0.5798	18.9737
	0.2162	0	1.6061	-0.4153	3.7399	19.2124
C16	0.2417	0	0.0039	0.9665	0	0.2189
	0.2600	0	-0.1853	0.7574	4.3685	5.2991
	0.4983	0	0.0698	-0.0626	9.0685	8.1234
C12	0.2419	0	0.1055	0.2399	0	9.0823
	0.3042	0	-0.2248	0.1824	5.9673	17.3966
	0.4539	0	0.1349	0.1223	13.4708	9.8744

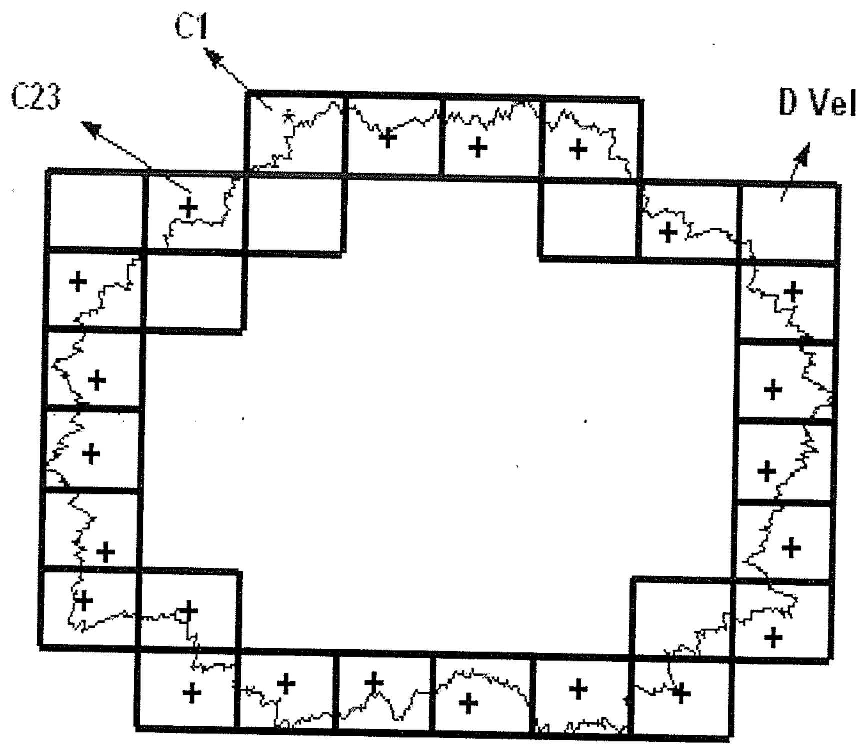


Fig. 8.11 IFS analysis per each C vel (shown by '+') for Boundary-2.

Table 8.6: Linear IFS calculation for a given SFD for 3 of the C vels in Fig. 8.5. Each vel had 4 control points for calculation of IFS.

Vel No.	LINEAR IFS					
	a	**b**	**c**	**d**	**e**	**f**
C4	0.6393	0	-0.0426	0.0059	0	13.1825
	0.1243	0	-0.2924	0.0689	6.1540	14.4121
	0.2364	0	-0.0126	0.3876	7.3503	7.5391
C7	0.2417	0	0.0039	0.9665	0	0.2189
	0.3574	0	0.0227	0.2332	4.9119	5.6548
	0.2397	0	0.8045	0.1325	9.2684	-2.4594
C14	0.2857	0	-0.1361	0.6392	0	2.1941
	0.2381	0	-0.1586	0.6778	2.0000	2.3127
	0.4762	0	0.0390	0.3501	3.6667	2.6115

Table 8.7: Result after analysis step on Boundary-1 for IFS-based approach.

Number of Boundary Points (Bytes)		
	Edge Points with Linear IFS Parameters	
Original	**Total Vels x [IFS +end pts(x,y)]**	**Total**
14759x2 = 29518	24 x [(6x3) + 2x2] + 1 x 2 x [(6x3) + 2x2]	572

Table 8.8: Result after analysis step on Boundary-2 for IFS-based approach.

Number of Boundary Points (Bytes)		
	Edge Points with Linear IFS Parameters	
Original	**Total Vels x [IFS +end pts(x,y)]**	**Total**
6000x2 = 12000	18 x [(6x3) + 2x2] + 1 x 2 x [(6x3) + 2x2]	440

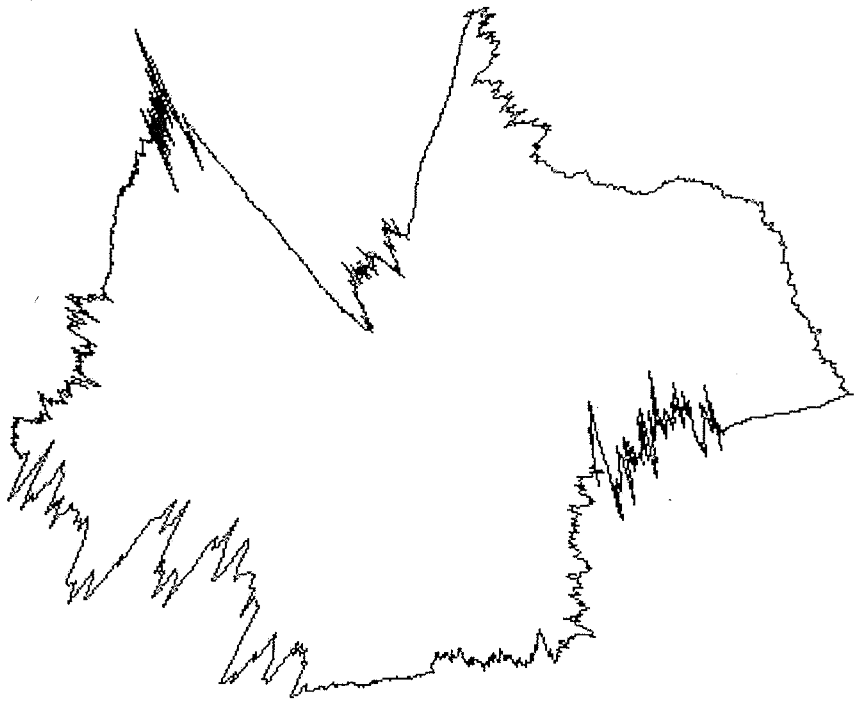
8.4.2 Renyi Dimension

A Rényi dimension spectrum was obtained for the reconstructed fractal boundaries. Figures 8.12 and 8.15 show the original boundaries for the two test images and their respective Rényi spectrum. The reconstructed boundaries and their Rényi spectrum are shown Figs. 8.13 and 8.16, respectively. Figures 8.14 and 8.17 show the comparison between the Rényi spectrum of the original boundary and the reconstructed boundary for Boundary-1 and Boundary-2, respectively. It can be seen from the comparison that the Rényi spectra of the original multifractal boundary and the reconstructed boundary are almost identical. However, it can be seen from Fig. 8.16 that the reconstructed Rényi spectra for Boundary-2 differs from the original for the negative values of q and its Dq values have gone above *2* for $q < -4$. Notice that a single fractal boundary is represented by a line at $Dq = 1.2$. This result shows that the reconstructed fractal boundaries are similar in complexities to the original boundary since the deviation in the spectra only occurs in negative values of q. and they look very similar to each other when observed by the human eye.

8.4.3 Compression Ratio

Tables 8.7 and 8.8 show that multifractal boundaries can be represented very compactly using this technique. The original multifractal boundary of Fig. 7.1a had 14759 points while the one of Fig. 7.1b had 6000 points with each point having xy co-ordinates. After the analysis, one set of IFS points and end points for each vel are obtained. If we consider one byte representation for each IFS value and point coordinates then the compression ratio achieved for Boundary-1 image is about **52:1** and for Boundary-2 image is about **27.27:1**. It is important to note that the number of bytes needed to code the same boundary having double the number of points will remain the same (572 for Boundary-1 and 440 for Boundary-2), thus, increasing the compression by two folds.

(a)



(b)

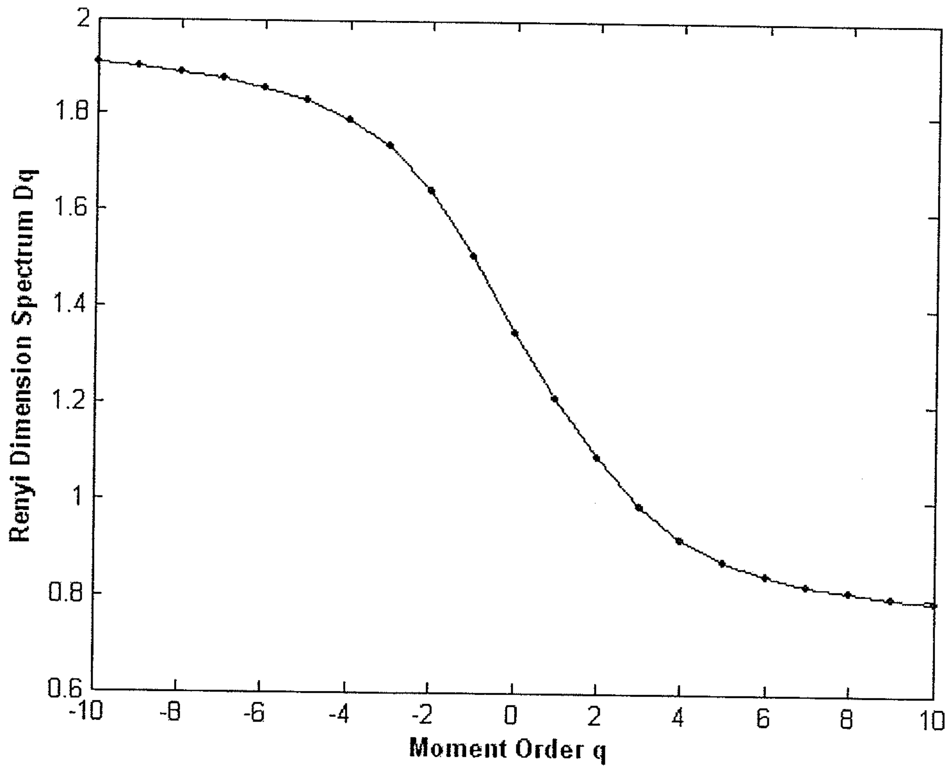
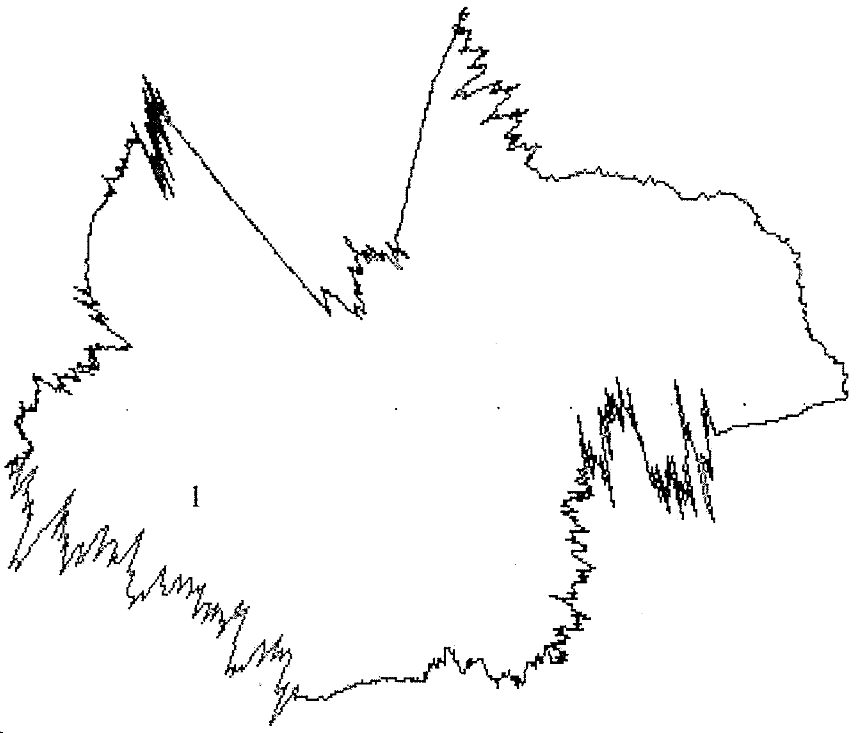
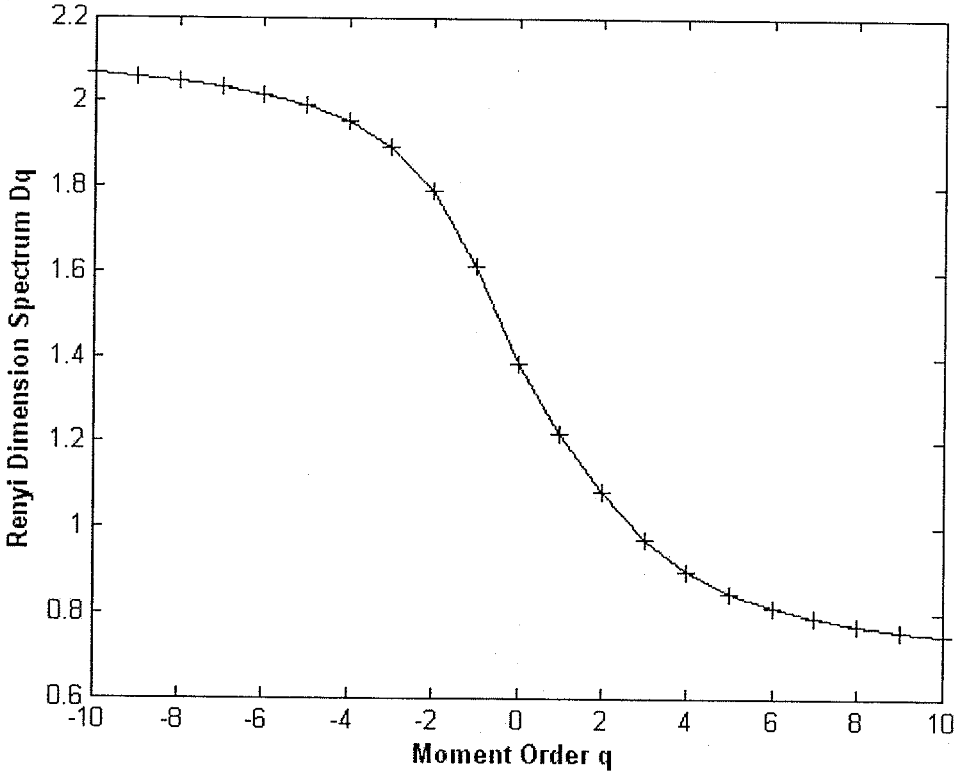


Fig. 8.12 (a) Original boundary (Boundary-1) and, (b) its Rényi dimension spectrum.



(a)



(b)

Fig. 8.13 (a) Reconstructed boundary (Boundary-1) using IFS-based approach and, (b) its Rényi dimension spectrum.

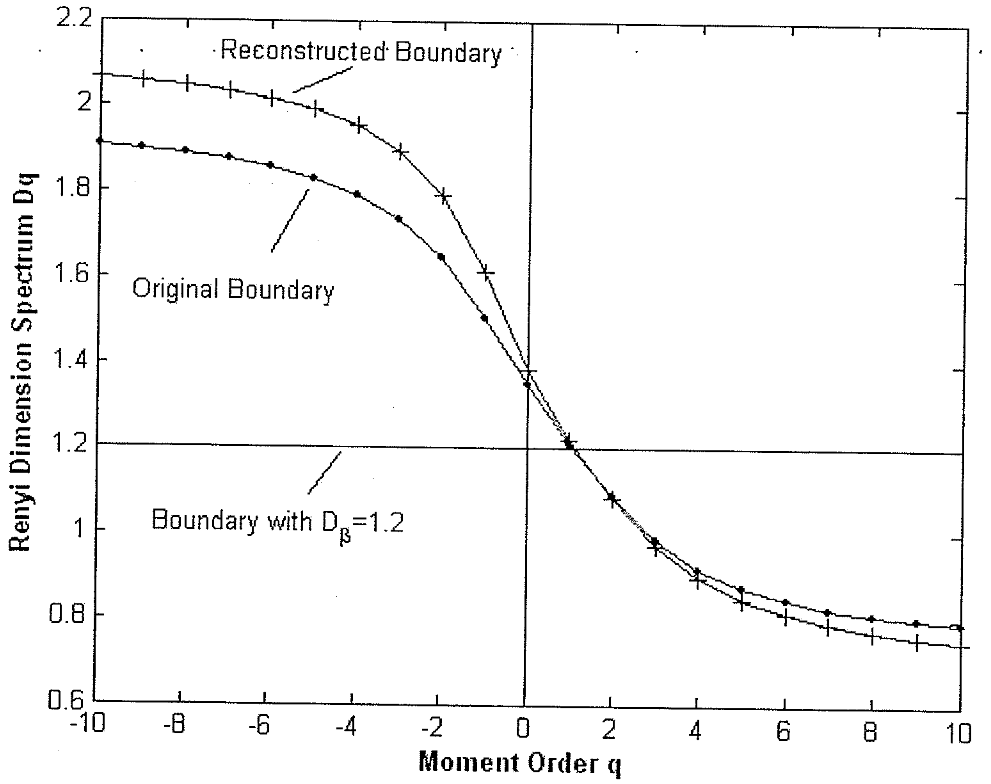
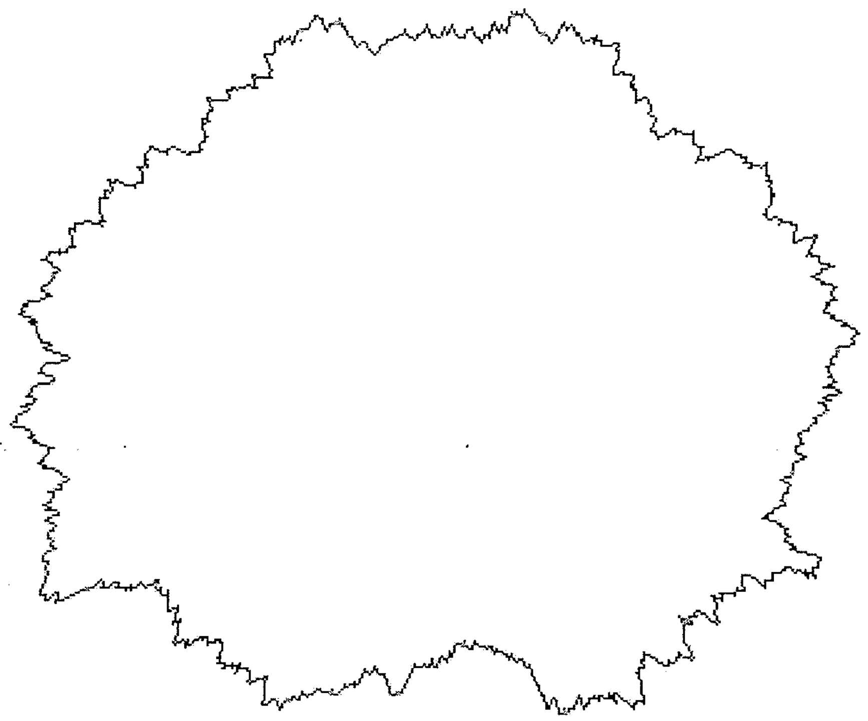


Fig. 8.14 Comparison of Rényi dimension spectrums of original boundary and reconstructed boundary for Boundary-1 using IFS approach.

(a)



(b)

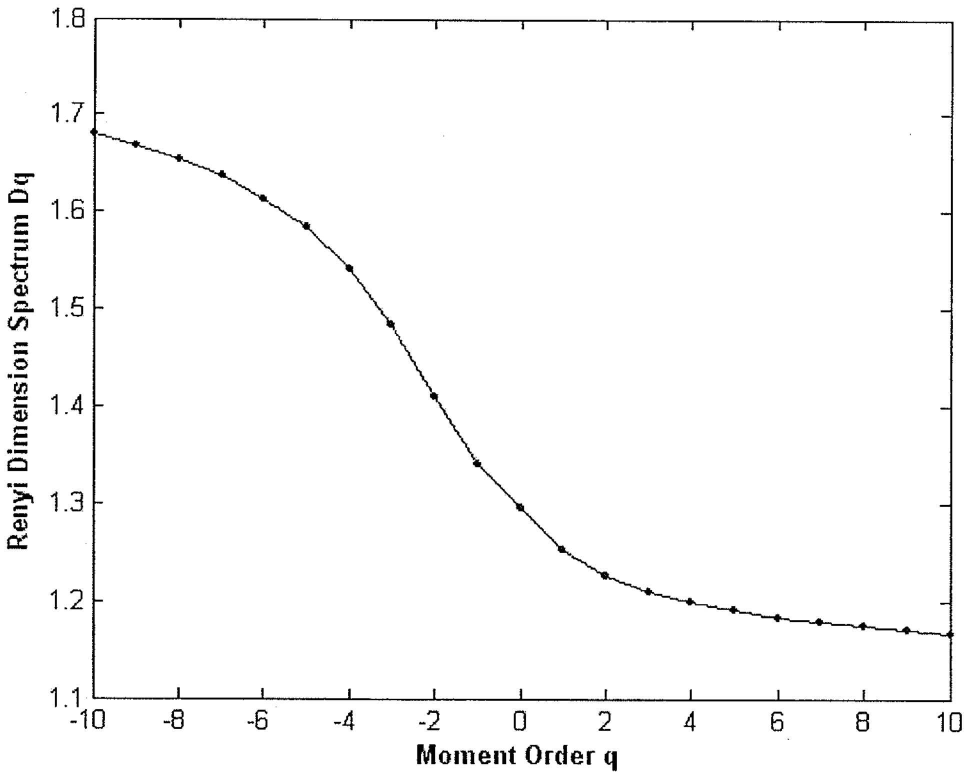
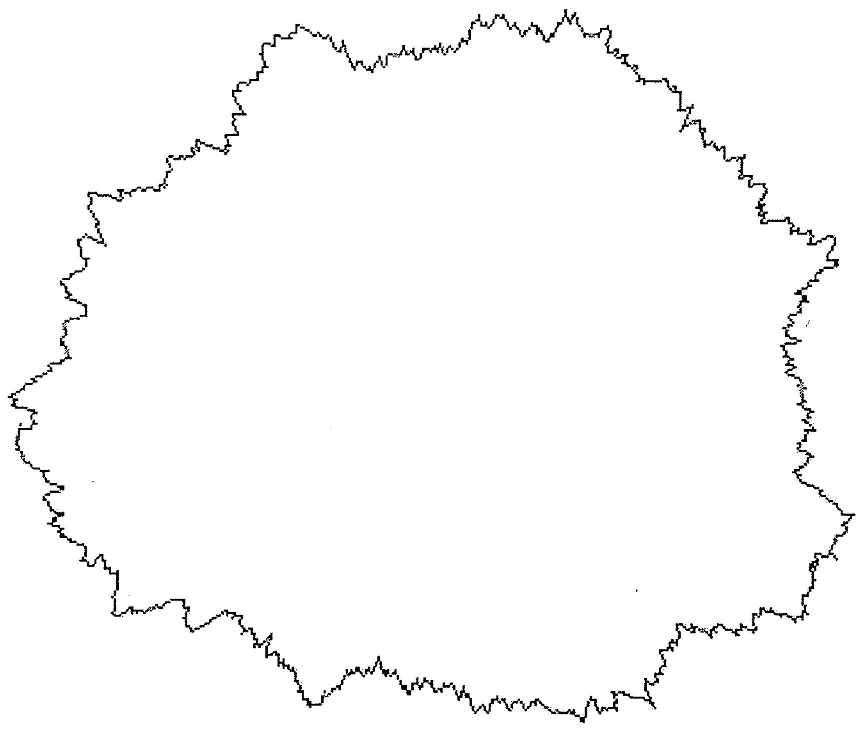


Fig. 8.15 (a) Original boundary (Boundary-2) and, (b) its Rényi dimension spectrum.

(a)



(b)

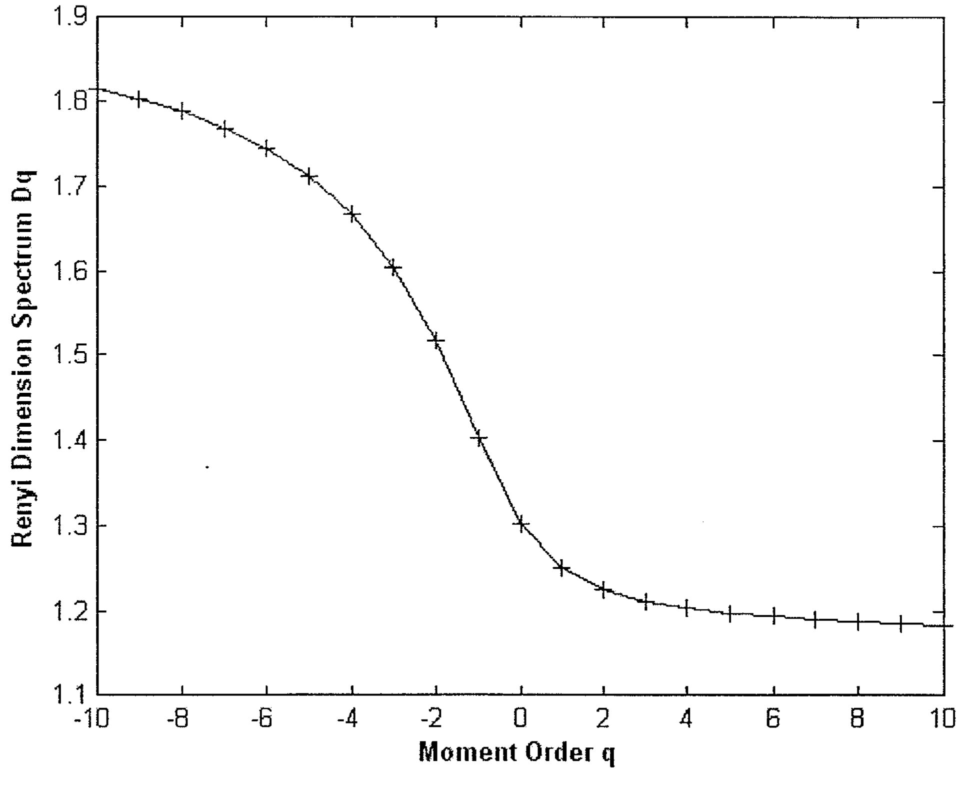


Fig. 8.16 (a) Reconstructed boundary (Boundary-2) using IFS-based approach and, (b) its Rényi dimension spectrum.

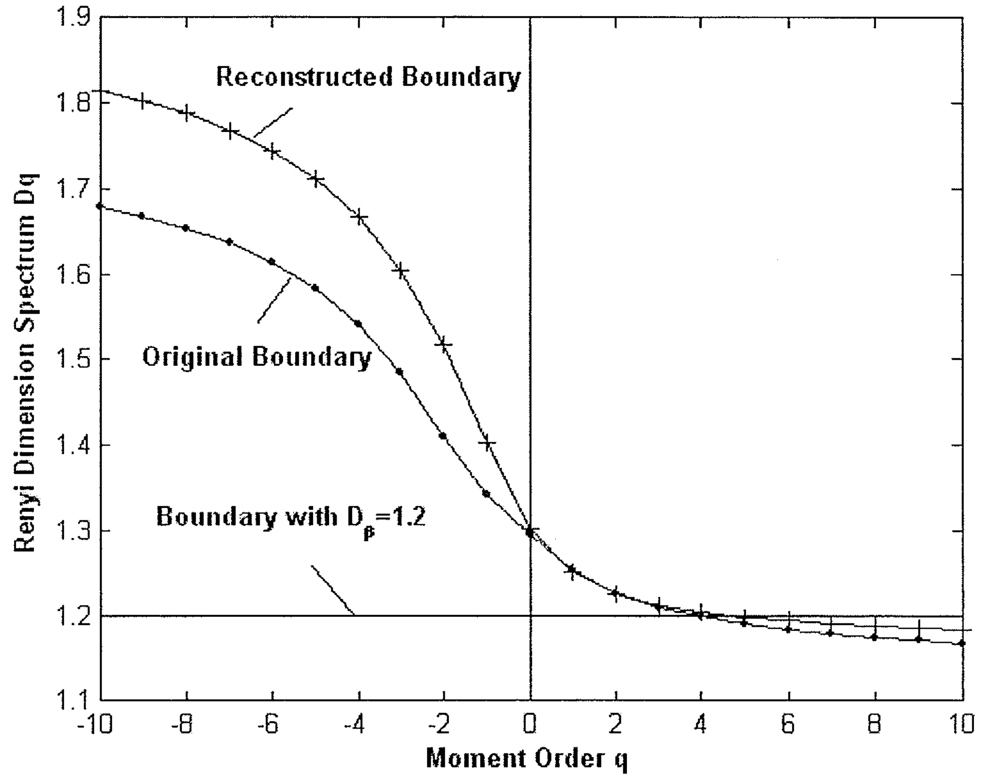


Fig. 8.17 Comparison of Rényi dimension spectrums of original boundary and reconstructed boundary for Boundary-2 using IFS approach.

8.5 Discussion

The significance and usefulness of the results obtained in the earlier sections will be discussed in his section.

8.5.1 Quality Measure Significance

In this thesis the goodness of reconstruction is measured using Rényi dimension spectrum. The main advantage of using fractal measures such as Rényi dimension spectrum is that we measure the function complexity instead of its energy. A motivation for using such complexity-based measures is that the human vision is more sensitive to complexities present in an image instead of variation in energies at individual pixels [Attn54]. Therefore, we are interested in the approximation of the boundary that is similar in complexity to the original. Notice that energy based measures such as signal to noise ratios may also be used as they can distinguish between different objects.

The Rényi dimension spectrum of reconstructed boundaries from both approaches show that its very similar to the one from original boundaries for positive values of q. It can also be observed that the reconstructed boundaries look very similar to the original boundaries and hence it can be easily implied that Rényi dimension spectrum corresponds to the HVS.

8.5.2 Reconstruction Artifacts

Some segments of the reconstructed boundary can exhibit slightly different fractality than the original ones. This artifact is visible in the bottom left segment (1) of the reconstructed Boundary-1 for both approaches. This is due to the inaccuracy in D_β

calculation technique caused by relatively spread out points resulting in fewer points within each vel. Similar to other fractal dimension calculation techniques, the accuracy of static vel covering technique depends upon the number of points within a vel. As discussed in Ch.7, if the total number of points in a vel are less than 512 then the calculated value of D_β will be inaccurate. However, this accuracy can significantly improve with more number of points and solves the reconstruction problem. Since both approaches use the static vel covering technique for D_β calculation therefore they are both affected by this limitation.

8.5.3 Performance Comparison of the Two Approaches

In the previous sections, a detailed discussion about the two approaches proposed in this thesis were presented and it was observed that both approaches show promising results. Wavelet based approach shows higher compression ratios but since the fractal curve generation on the boundary segment is done using the MPD technique, its reconstruction quality can vary. MPD is computationally fast but the accuracy of the fractal dimension on the segments of the boundary that it creates can deviate highly from the original one. On the other hand fractal curve created using IFS are very accurate in their fractal dimensions.

For wavelet based approach, the analysis process took about 2 minutes for the Boundary-1 image, while for IFS based approach the analysis process took about 3 minutes. The analysis for Boundary-2 image took less then 1 minute for both approaches.

For all computations, a 2.4 GHz Pentium 4 computer was used, and both approaches were implemented in Matlab 6.5.

Based on the results presented above, this thesis has demonstrated that the object boundaries that are (i) crisp (one pixel wide), (ii) rough (non smooth), (iii) self-similar (fractal), and (iv) non-occluded (not hidden by other object or boundary) can be compactly modelled using the two approaches.

8.6 Summary

In this chapter the results of the two proposed approaches were presented and their merits were discussed. The significance and usefulness of the results were also presented. In the next chapter the conclusion for this thesis and recommendations for future work will be presented.

CHAPTER IX

CONCLUSIONS AND RECOMMENDATIONS

9.1 Conclusions

This thesis was concerned with compact modelling of object boundaries that had fractal or multifractal characteristics. Two approaches, wavelet based and IFS based, were presented for the purpose of modelling such boundaries. In order to test these approaches, two different types of simulated multifractal boundaries were used as data sets. Both of the approaches showed that these boundaries can be efficiently compressed using compact representations.

For both approaches, the first step was to measure the local values of fractal dimension D_β on the given boundary. Therefore, this thesis presented two innovative techniques (static vel covering and moving vel) in order to accurately determine D_βs. However, the accuracy of these techniques depends upon the number of points and the size of the vel. Since some of the parameters such as the vel size have to be determined heuristically therefore, these vel covering techniques may not be very robust.

The wavelet based approach requires smoothing of the multifractal boundary for detection of the second set of points. These are the control points supporting the shape of the object. The actual number of the control points is dependent upon the type of smoothing technique used. This in turn effects the compression ratio.

The MPD algorithm has been employed in the synthesis step of the wavelet based approach. Although this algorithm is computationally fast, the accuracy of the resultant fractal boundary segment and its fractal dimension remains doubtful, specially when the number of points are less then 1000.

In the second analysis step of IFS based approach, the values of IFS are determined in each vel for the given D_β. This is a computationally intensive process, and the time taken to determine each IFS within the vel depends upon the number of control points chosen in the vel. If the boundary structure in the vel is complex i.e. having *wiggles* or snake like shape then more control points are needed.

The wavelet based approach shows higher compression ratios as compared to IFS based approach. But on the other hand the local fractal dimensions D_βs on the reconstructed fractal boundary in IFS based approach are more accurate when compared with wavelet based approach.

Over all both approach have shown promising results in terms of compression and reconstruction quality. The compression ratio achieved for wavelet-based approach was 285:1 while for IFS-based approach it was 175:1. The complexity-based measures (the Rényi fractal dimension spectrum) shows that a boundary modelled on the basis of self similarity resembles the original boundary in terms of complexity.

9.2 Contributions

This thesis has made the following contributions:

- A new approach that uses existing wavelet based techniques to compactly model multifractal object boundaries.
- A new approach that uses existing IFS based techniques to compactly model multifractal object boundaries.
- Two innovate techniques (static vel and moving vel) to determine local fractal dimensions on closed object boundaries.
- The selection of appropriate vel size for determining local fractal dimensions for a given boundary.
- Modification of MPD algorithm to fit on contours and create closed object boundaries.
- Segmented IFS based reconstruction of a closed multifractal object boundary.
- A software system that efficiently implements the wavelet based approach and IFS based approach.

9.3 Recommendations for Future Work

Based on research conducted in this thesis, recommendations are as follows:

- The Rényi spectrum can also be used to determine the optimum size of the vels for accurate calculation of D_β. This can be achieved by minimizing the least squared error between the Rényi spectra of the original boundary and the reconstructed one. The error minimization is done to a certain threshold value while the size of the vels is reduced accordingly. This technique can be further improved by calculating D_β for each vel and plotting them with Rényi spectrum of the original boundary. The size of the vel is then reduced by minimizing the least squared error between the vel and the calculated by the Rényi spectrum. This technique would lead to a covering with variable vel sizes.

- Measuring other potentially more accurate fractal dimensions in the vel instead of D_β. e.g. *regularization dimension* [RoVé98].
- Application of segment wise different smoothing techniques on the multifractal boundary in order to achieve maximum smoothness.
- Design and implementation of the two approaches in hardware chips such as FPGAs.

REFERENCES

[AsBr86] H. Asada and M. Brady, "The curvature primal sketch," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 1, pp. 2-14, January 1986.

[Attn54] F. Attneave, "Some informational aspects of visual perception," *Psychological Review*, vol. 61, pp. 183-193, 1954.

[BaDe85] M. F. Barnsley and S. Demko, "Iterated function systems and the global construction of fractals," *Proc. Roy. Soc.*, London, vol. A399, pp. 243-275, 1985.

[Barn00] M. F. Barnsley, *Fractal Everywhere.* San Diego, CA: Morgan Kaufmann, 2000 (2nd ed.), 531 pp.

[Blum67] H. Blum, "A transformation for extracting new descriptors of shape," in Whaten-Dunn (ed.), *Proc. Symp. Models for the Perception of Speech and Visual Forms*. pp. 362-380, 1967, 470 pp.

[Brad83] M. Brady, "Criteria for representations of shape," in J. Beck, B. Hope, and A. Rosenfeld (eds.), *Human and Machine Vision*, San Diego, CA: Academic Press, pp. 39-84, 1983, pp. 583.

[BrAs84] M. Brady and H. Asada, "Smoothed local symmetries and their implementation," *Intl. J. Robotics Research*, vol. 3, pp. 36-61, 1984.

[Brow00] T. J. Brown, "Combined Evidence Thresholding: A new wavelet regression technique for detail preserving image de-noising," *Proceedings IMVIP 2000*, pp. 83-92, 2000.

[Cast96] K. Castleman, *Digital Image Processing.* Upper Saddle River, NJ: Prentice Hall, 1996, 667 pp.

[ChVC96] J. M. Chen, J. A. Ventura, and H. C. Wu, "Segmentation of planar curves into circular arcs and line segments," *Image and Vision Computing*, vol. 14, no. 1, pp. 71-83, February 1996.

[CoCe01] L. F. Costa and R. M. Cesar, Jr., *Shape Analysis and Classification: Theory and Practice.* Boca Raton, FL: CRC Press, 2001, 680 pp.

[CoDF92] A. Cohen, I. Daubechies, and J. C. Feauveau, "Biorthogonal basis of compactly supported wavelets," *Comm. Pure Appl. Math.*, vol. 45, pp. 485-560, 1992.

[CoTh91] T. Cover and J. Thomas, *Elements of Information Theory.* New York, NY: John Wiley & Sons, 1991, 542 pp.

[CQGa00a] F. A. Cheikh, A. Quddus, and M. Gabbouj, "Contour-based object recognition using wavelet-transform," in *Signal Processing X, Theories and Applications, EUSIPCO 2000*, Tampere, Finland, pp. 2141-2144, September 2000.

[CQGa00b] F. A. Cheikh, A. Quddus, and M. Gabbouj, "Multi-level shape recognition based on wavelet-transform," 4*th IEEE Southwest Symposium on Image Analysis & Interpretation*, Austin, TX, USA, pp. 8-12, April 2000.

[Daub92] I. Daubechies, *Ten Lectures on Wavelets.* SIAM, 1992, 357 pp.

[DaDr99] L. Dalla and V. Drakopoulos, "On the parameter identification problem in the plane and the polar fractal interpolation functions," *J. Approximation Theory*, vol. 101, pp. 289-302, 1999.

[DoJo94a] D. L. Donoho and I. M. Johnstone, "Ideal Spatial Adaptation via Wavelet Shrink-age," *Biometrika*, vol. 81, pp. 425-455, 1994.

[DoJo94b] D. L. Donoho and I. M. Johnstone, "Adapting to unknown smoothness by wavelet shrinkage," *J. Amer. Statist. Assoc.*, vol. 90, pp. 1200-1224, 1995.

[DoJo98] D. L. Donoho and I. M. Johnstone, "Minimax estimation via wavelet shrinkage," *Ann. Statist.* vol. 26, pp. 879-921, 1998.

[DoKP95] D. L. Donoho, K. P. Kerkyacharian and D. Picard, "Wavelet shrinkage: Asymptopia?," *J. Royal Statist. Soc.*, vol. B57, pp.301-369, 1995.

[Dono95] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Trans. Information Theory*, vol.41, pp. 613-627, 1995.

[DuHa72] R. O. Duda and P. E. Hart, "A generalized Hough transformation for detecting lines in pictures," *Comm. ACM*, Vol. 15, pp 1-15, 1972.

[FeKr94] C. Fermuller and W. Kropatsch, "A syntactic approach to scale-space based corner detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-16, pp. 748-751, 1994.

[FeSS80] B. Ferrari, V. Shankar, and F. Sklansky, "Minimal rectangular partition of digitized blobs," *in Proc. 5th Intl. Conf. Pattern Recognition*, pp. 1040-1043, 1980.

[GaBr95] H. Gao and A. G. Bruce, "Waveshrink with semisoft shrinkage," *Research Report*, no. 39, StatSci Division of MathSoft Inc., 1995.

[Gosh85] A. Goshtasby, "Description and discrimination of planar shapes using shape matrices," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-7, pp. 738-743, 1985.

[GoWo92] R. C. Gonzalez and R. E. Woods, *Digital Image Processing.* Reading, MA: Addison-Wesley, 1992, 716 pp.

[GoWo02] R. C. Gonzalez and R. E. Woods, *Digital Image Processing.* Upper Saddle River, NJ: Prentice Hall, 2002, 793 pp.

[Gran72] G. H. Granlund, "Fourier preprocessing for hand print character recognition," *IEEE Trans. Computers*, vol. 21, pp. 195-201, 1972.

[GrMo84] A. Grossmann and J. Morlet, "Decomposition of hardy functions into square integrable wavelets of constant shape," *SIAM J. Math.*, vol. 15, pp. 723-736, 1984.

[HeNe87] P. Heinomen and Y. Neuvo, "FIR-Median hybrid filters," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. ASSP-35, no. 6, pp. 832-838, June 1987.

[Hu62] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. Information Theory*, vol. 8, pp. 179-187, 1962.

[KaCh81] R. Kashyap and R. Chellappa, "Stochastic models for closed boundary analysis: Representation and reconstruction," *IEEE Trans. Information Theory*, vol. 27, pp. 627-637, 1981.

[KaSa89] B. Kartikeyan and A. Sarkar, "Shape description by time series," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-11, pp. 977-984, 1989.

[KhGa98] L. Khriji and M. Gabbouj, "Median rational hybrid filters for image restoration," *IEE Electronic Letters*, vol. 34, no. 10, pp. 977-979, May 1998.

[Kins94a] W. Kinsner, "Fractal dimension: morphological, entropy spectrum, and variance classes," *Technical Report, DEL 94-4*, Department of Electrical and Computer Engineering, University of Manitoba, May 1994, 146 pp.

[Kins94b] W. Kinsner, "Batch and real-time computation of a fractal dimension based on variance of a time series," *Technical Report, DEL 94-6*, Department of Electrical and Computer Engineering, University of Manitoba, May 1994, 22 pp.

[Kins95] W. Kinsner, "Self-similarity: fractals, chaos, scaling and their application," *Technical Report, DEL 95-2*, Department of Electrical and Computer Engineering, University of Manitoba, Jan. 1995, 113 pp.

[Kins96] W. Kinsner, *Fractal and Chaos Engineering.* Winnipeg, MB: Department of Electrical and Computer Engineering; University of Manitoba, 1996, 700 pp.

[KiPK87] H. Kim, K. Park, and M. Kim, "Shape decomposition by collinearity," *Pattern Recognition Letters*, vol. 6, pp. 335-340, 1987.

[Koen84] J. Koenderink, "The structure of images," *Biological Cybernetics*, vol. 50, pp. 363-370, 1984.

[LaKi96] A. Langi and W. Kinsner, "Compression of aerial ortho images based on image denoising," 1996 NASA/Industry Data Compression Workshop, (Snowbird, UT, April 4, 1996), pp. 81-90, 1996.

[LeSC92] J. S. Lee, Y. N. Sun, and C. H. Chen, "Wavelet transform for corner detection," *Proc. IEEE Intl. Conference on Systems Engineering*, Kobe, Japan., pp. 596-599, September 1992.

[LeSC95] J. S. Lee, Y. N. Sun, and C. H. Chen, "Multiscale corner detection by wavelet transform," *IEEE Trans. Image Processing*, vol. 4, no. 1, pp. 100-104, January 1995.

[LiSr90] H. C. Liu and M. D. Srinath, "Partial shape classification using contour matching in distance transformation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-12, no. 11, pp. 1072-1079, November 1990.

[LSCT93] J. S. Lee, Y. N. Sun, C. H. Chen, and C. T. Tsai, "Wavelet based corner detection," *Pattern Recognition*, vol. 26, no. 6, pp. 853-865, 1993.

[MaHi80] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Royal Soc. London*, vol. 207, pp. 187-217, 1980.

[MaHw92] S. G. Mallat and W. L. Hwang, "Singularity detection and processing with wavelets," *IEEE Trans. Information Theory*, vol. 38, no. 2, pp. 617-645, March 1992.

[Mall89] S. G. Mallat, "Multiresolution approximation and wavelet orthonormal bases of L2," *Trans. Amer. Math. Soc.*, vol. 3-15, pp. 69-87, September 1989.

[Mand82] B. B. Mandelbrot, *The Fractal Geometry of Nature.* NewYork, NY: W. H. Freeman, 1982, 480 pp.

[Mall98] S. G. Mallat, *A Wavelet Tour of Signal Processing.* San Diego, CA: Academic Press, 1998, 577 pp.

[MaNi78] D. Marr and H. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes," *Proceedings of the Royal Society of London*, vol. B200, pp. 269-294, 1978.

[Marr82] D. Marr, *Vision: A computational investigation into the human representation and processing of visual information.* San Francisco, CA: W.H. Freeman, 1982, 397 pp.

[MaZh92] S. G. Mallat and S. Zhong, "Characterization of signals from multiscale edges," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-14, no. 7, pp. 710-732, July 1992.

[MMOP96] M. Misiti, Y. Misiti, G. Oppenheim, and J.-M. Poggi, *Wavelet Toolbox User's Guide*. The MathWorks, Inc., 24, Prime Park Way, Natick, Mass. 01760-1500, 1996.

[MPEG00] *MPEG-7 Multimedia Description Schemes*, WD (V3.0), ISO/IEC JTC1/ SC29/ WG11MPEG2000/ N3411 Geneva, CH, June 2000.

MuSB95] F. Murtagh, J. L. Starck, and A. Bijaoui, "Image restoration with noise suppression using a multiresolution support," *Astronomy and Astrophysics,* Supplement series 112, pp. 179 - 189, 1995.

[Neva82] R. Nevatia, "Shape analysis and recognition," in *Machine Perception*, Upper Saddle River, NJ: Prentice Hall, 1982. (Chapter 5, pp. 61-89), 209 pp.

[RiDu92] O. Rioul and P. Duhamel, "Fast algorithms for wavelet transforms," in *IEEE Trans. Information Theory*, vol. 38(2), pp. 569-586, Mar 1992.

[Pavl78] T. Pavlidis, "A review of algorithms for shape analysis," *Computer Graphics and Image Processing,* vol. 7, pp. 243-258, 1978.

[PeFu77] E. Persoon and K. Fu, "Shape discrimination using Fourier descriptors," *IEEE Trans. Systems, Man and Cybernetics*, vol. 7, pp. 170-179, 1977.

[PeJS91] H. Peitgen, H. Jurgens, and D. Saupe, *Fractals for The Classroom: Part 1: Introduction to Fractals and Chaos*. Springer-Verlag, 1991, 450 pp.

[Pent84] A. Pentland, "Fractal-based description of natural scenes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-6, pp. 661-674, 1984.

[QCGa99] A. Quddus, F. A. Cheikh, and M. Gabbouj, "Wavelet-based multi-level object retrieval in contour images," *in Proc. of the Intl. Workshop on Very Low Bit Rate Video Coding (VLBV'99)*, Kyoto, Japan, October 1999.

[QCGa00] A. Quddus, F. A. Cheikh, and M. Gabbouj, "Content-based object retrieval using maximum curvature points in contour images," *Proc. SPIE/EI'2000 Symposium on Storage and Retrieval for Media Databases,* San Jose, Ca., USA, January 2000.

[QuGa00] A. Quddus and M. Gabbouj, "Wavelet based corner detection using singular value decomposition," *in Proc. IEEE Intl. Conference on Acoustic Speech and Signal Processing, ICASSP 2000,* Istanbul, Turkey, pp. 2456-2459, June 2000.

[QuGa02] A. Quddus and M. Gabbouj, "Wavelet based corner detection technique using optimal scale," *Pattern Recognition Letters*, vol. 23, no. 1-3, pp. 215- 220, 2002.

[RaCh92] A. Rattarangsi and R. T. Chin, "Scale-based detection of corners on planar curves," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 4, pp. 430-449, April 1992.

[RaRa97] B. K. Ray and K. S. Ray, "Scale-space analysis and corner detection on digital curves using a discrete scale-space kernel," *Pattern recognition*, vol. 30, no. 9, pp. 1463-1474, 1997.

[Reis96] L. M. Reissel, "Wavelet multiresolution representation of curves and surfaces," *Graphical Models and Image Processing*, vol. 58, no. 3, pp. 198-217, May 1996.

[RiHe74] C. W. Richards and H. Hemami, "Identification of three-dimensional objects using fourier descriptors of the boundary curve," *IEEE Trans. Systems, Man and Cybernetics*, vol. 4, pp. 371-378, 1974.

[RoAd90] D. F. Rogers and J. A. Adams, *Mathematical Elements for Computer Graphics*. NewYork, NY: McGraw-Hill, 1990, 512 pp.

[RoBa79] J. O'Rourke and N. Badler, "Decomposition of three-dimensional objects into spheres," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-1, pp. 295-305, 1979.

[RoVé98] F. Roueff and J. Véhel, "A regularization approach to fractional dimension estimation," *Proceedings of Fractals 98*, pp. 231-237, Malta, October 1998.

[Russ98] J. Russ, *The Image Processing Handbook.* Ann Arbor, MI: CRC Press, 1998, 771 pp.

[SiKi02] S. Siddiqui and W. Kinsner, "Multifractal modelling of arbitrary object boundaries in image segmentation," in *Proc. IEEE Cnd. Conf. Elect. And Comp. Engineering, CCECE 2002 (Winnipeg, Canada, May. 2002)*, vol.2, pp. 950-956, 2002.

[Shen92] M. J. Shensa, "Discrete wavelet transforms: Wedding the a`trous and Mallat algorithms," *IEEE Trans. Signal Processing,* vol. 40, pp. 2464-2482, 1992.

[Smit70] J. Mott-Smith, "Medial axis transformations," in Lipkin and Rosenfeld (eds.), *Picture Processing and Psychopictorics*, San Diego, CA: Academic Press, pp. 267-283, 1970, 526 pp.

[StMB94] J. L. Starck, F. Murtagh, and A. Bijaoui, "Image restoration with noise suppression using the wavelet transform," *Astronomy and Astrophysics,* vol. 288, pp. 343-348, 1994.

[StMB98] J. L. Starck, F. Murtagh, and A. Bijaoui, *Image Processing and Data Analysis: The Multiscale Approach.* NewYork, NY: Cambridge University Press, 1998, 297 pp.

[TaSu89] A. Taza and C. Suen, "Discrimination of planar shapes using shape matrices," *IEEE Trans. Systems, Man and Cybernetics*, vol. 19, pp. 1281-1289, 1989.

[Teag80] M. R. Teague, "Image analysis via the general theory of moments," *J. Optical Society of America*, vol. 70, pp. 920-930, 1980.

[TiBo97a] Q. M. Tieng and W. W. Boles, "Wavelet-based affine invariant representation: A tool for recognizing planar objects in 3-D space," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-19, no. 8, pp. 846-857, August 1997.

[TiBo97b] Q. M. Tieng and W. W. Boles, "Recognition of 2D object contours using the wavelet transform zero-crossing representation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-19, no. 8, pp. 910-916, August 1997.

[TuAB98] M. J. Turner, P. R. Andrews, and J. Blackledge, *Fractal Geometry in Digital Imaging*. San Diego, CA: Academic Press, 1998, 328 pp.

[Vida94] B. Vidakovic, "Non-linear wavelet shrinkage with Bayes rules and Bayes factors," *Discussion Paper 94-24 ISDS*, Duke University, USA, 1994.

[WaWi80] T. Wallace and P. Wintz, "An efficient three-dimensional aircraft recognition algorithm using normalized Fourier descriptors," *Computer Graphics and Image Processing*, vol. 13, pp. 99-126, 1980.

[WoWa75] G. Wahba and S. Wold, "A completely automatic French curve," *Comm. Stat.* vol. 4, pp. 1-17, 1975.

[XuHe94] W. Xu and L. Healy, "Wavelet transform domain filters: A spatially selective noise filtration technique," *IEEE Trans. Image Processing,* vol. 3, no. 6, pp. 747-758, November 1994.

[YoWB74] I. Young, J. Walker, and J. Bowie, "An analysis technique for biological shape," *Computer Graphics and Image Processing*, vol. 25, pp. 357-370, 1974.

[ZaRo72] C. Zahn and R. Roskies, "Fourier descriptors for plane closed curves," *Computer Graphics and Image Processing*, vol. 21, pp. 269-281, 1972.

[Zusn70] L. Zusne, *Visual Perception of Form*. NewYork, NY: Academic Press, 1970, 547 pp.

APPENDIX A

MATLAB CODE

This appendix provides the main MATLAB code used in this thesis.

A.1 Smooth Curve (smooth_curve.m)

```
%-----------------------------------------------------------------------------------------------------------
% Creates closed smooth curve on the given points and stores the curve
% points in new storage array.
% xa_t  x- cordinate array
% ya_t  y- cordinate
% array H     Holder exponent array
% Returns  -None
% Author(s): Sharjeel Siddiqui, 01-Jan-2001 Copyright (c) Sharjeel Siddiqui.
%-----------------------------------------------------------------------------------------------------------

function smooth_curve(xa_t,ya_t,H);

close all;
clear all;

sizxa=size(xa_t);
n=sizxa(1);

iterm=07;

x1=[1:.5:20];

sizex=size(x1);
```

```
y2=zeros(sizex(2),1);


figure(1);
xi=0;
yi=0;


for i=1:sizex(2)
   x=x1(i);
   [y,dy]=neville(xa_t,ya_t,x,n);
   y2(i)=y;
   xi=x;
   yi=y;
end
plot(x1,y2);
hold on;
plot(xa_t,ya_t,'o');
%axis([0,25,min(y2)+2,max(y2)+2]);
clear y;


figure(2);


xi=0;
yi=0;
c=polyfit(xa_t,ya_t,1);


for i=1:sizex(2)
  y=c(1)*x1(i)+c(2);
  if i>1
     plot([xi,x1(i)],[yi,y]);
     hold on;
  end
```

```
   xi=x1(i);
   yi=y;
 end

 plot(xa_t,ya_t,'o');
 axis([0,25,0,15]);
 Title('Curve reconstruction by Linear regression');
 figure(3);

 xi=0;
 yi=0;
 c=polyfit(log(xa_t),log(ya_t),1);

 for i=1:sizex(2)
   y=c(1)*x1(i)+c(2);
   if i>1
      plot([xi,x1(i)],[yi,y]);
      hold on;
   end
   xi=x1(i);
   yi=y;
 end

plot(xa_t,ya_t,'o');
axis([0,25,0,15]);
Title('Curve reconstruction by fitting a nonlinear function(log) on the points with least square error');

figure(5)

s=1:length(xa_t);
sp=1:length(xa_t)/100:length(xa_t);
```

```
xp=spline(s,xa_t,sp);
yp=spline(s,ya_t,sp);

plot(xp,yp);hold on;
plot(xa_t,ya_t,'o');
Title('Curve reconstruction using Cubic Spline interpolation between every two points');

figure(6);

lag=Lagran(xa_t,ya_t,x1);

plot(x1,lag);
hold on
plot(xa_t,ya_t,'o');
Title('(n-1) degree ploynomial construction using Lagrangian formula');


figure(7);

s=1:length(xa_t);
sp=1:length(xa_t)/100:length(xa_t);

xp=Lagran(s,xa_t,sp);
yp=Lagran(s,ya_t,sp);

plot(xp,yp);
hold on
plot(xa_t,ya_t,'o');
axis([0,max(xp)+2,min(yp),max(yp)+2]);
Title('A zoomed window of(n-1) degree ploynomial construction using Lagrangian formula');
```

```
figure(8)
lag=interp1(xa_t,ya_t,x1,'spline');
plot(x1,lag);
hold on
plot(xa_t,ya_t,'o');
Title('Curve Reconstructioin using spline reconstruction by interpolation');

figure(9)
s=1:length(xa_t);
sp=1:length(xa_t)/100:length(xa_t);
xp=interp1(s,xa_t,sp,'spline');
yp=interp1(s,ya_t,sp,'spline');
sizxpp=length(xp);
plot(xp,yp,'k');hold on;
plot([xp(sizxpp),2],[yp(sizxpp),10],'k');hold on;
```

A.2 Mid Point Displacement (mid_point_displacement.m)

```
%-----------------------------------------------------------------------------------------------
% Creates a fractal curve using Midpoint displacement algorithm on the
% given points and stores the new curve points in new storage array.
% xa  x- cordinate array control point
% ya  y- cordinate array control point
% array H     Holder exponent array
% iterm = number of iterations
% Returns - None
% Author(s): Sharjeel Siddiqui, 01-Jan-2001 Copyright (c) Sharjeel Siddiqui.
%-----------------------------------------------------------------------------------------------

function mid_point_displacement(xa,ya,xp,yp,H,iterm)

num=0;  %% total number of parametric points in a particular range

angle=0; %% angle of the support line betwwen two control points

startxp=1;
lenxp=length(xp);
endxp=lenxp;

cuv=5;
sizya=size(ya);
sizyp=size(yp);
sizxa=size(xa);

r_polar=zeros(sizya);
theta_polar=zeros(sizxa);
```

```
for i=1:sizya(1,2)
   r_polar(1,i)=sqrt(xa(1,i)^2+ya(1,i)^2);
   theta_polar(1,i)=atan(ya(1,i)/xa(1,i));
end

no_of_points_after_mpd =129; % pre determined
yalmt=1;
yaulmt=1;

sizz=sizya(1,2)-1; % Number of control points in Y co-ordinate..

yamfull=zeros(65,sizz);
xmfull=zeros(65,sizz);

final_result_xaya=zeros(2,no_of_points_after_mpd *sizz);

%% should be sizz only for closed curves
%otherwsie sizz-1
for z=1:(sizz)

   sign_of_diff_ya=1;
   distance=sqrt( (xa(1,z+1)-xa(1,z))^2 + (ya(1,z+1)-ya(1,z))^2 );
   COS_theta=abs(( xa(1,z+1)-xa(1,z) )) / distance;
   SIN_theta=abs(( ya(1,z+1)- ya(1,z) )) /distance;
   %sign_of_diff_ya=sign( abs(ya(1,z+1))- abs(ya(1,z)) );

  if( (ya(1,z+1) > ya(1,z) && xa(1,z+1) >= xa(1,z)) || ( (ya(1,z+1) < ya(1,z)) &&( xa(1,z+1) <= xa(1,z)) ) )
     sign_of_diff_ya=sign_of_diff_ya;
  elseif( (ya(1,z+1) > ya(1,z) && xa(1,z+1) <= xa(1,z)) || ( (ya(1,z+1) < ya(1,z)) && ( xa(1,z+1) >= xa(1,z))
))
```

```
    sign_of_diff_ya=-sign_of_diff_ya;
  end


  R_theta=[COS_theta, sign_of_diff_ya*SIN_theta; -sign_of_diff_ya*SIN_theta,COS_theta];
  R_theta_minus=[COS_theta, -sign_of_diff_ya*SIN_theta;sign_of_diff_ya*SIN_theta,COS_theta];


  C=(R_theta*[ xa(1,z+1)-xa(1,z) ; ya(1,z+1)-ya(1,z) ] )+ [xa(1,z);ya(1,z)]


  xa(1,z+1)=C(1,1);


  ya(1,z+1)=C(2,1);


  H1=H(1,z);
  maxya=0;
  yam=zeros(2,1);


  %% putting the y co-ordinate pair in to an array 'yam' for mid point
  %% displacemnet algorithm.


  yam(1,1)=ya(1,z);
  yam(2,1)=ya(1,z+1);  %% Column vector
  %newya=zeros(3,1);


  flag=0;


% insert here


  %xdist=abs(abs(xa(z+1,1))-abs(xa(z,1)));
  %ydist=abs(abs(ya(z+1,1))-abs(ya(z,1)));
  num=0;
```

```
 for i=1:iterm


   sizeya=size(yam);

   newya=zeros(sizeya(1,1)*2-1,1);

   k=1;

   for j=1:sizeya(1,1)-1

     newya(k)=yam(j,1);


     if flag==1


       %g1=absya;   % taking the point on the max curvature

       flag=0;

     else


       %% Here comes MPD method.

                  g1=(yam(j,1)+yam(j+1,1))/2 + randn*sqrt((1-2^(2*H1-2))/(2^((i+1)*2*H1)));%randn/
(sqrt(2^(i+1)));

     end

     newya(k+1)=g1;

     newya(k+2)=yam(j+1);

     k=k+2;

   end

   clear yam;

   yam=newya;

   %size(yam)


   clear newya;


 end


 sizeyam=size(yam);
```

```
% calculating number of points needed for X-axis after getting new Y
% axis values by MPD.

inter=abs(xa(1,z+1)-xa(1,z))/(sizeyam(1,1)-1);

%% Generating correspondin X-axis values for the new Y axis value
%% returned by Mid point displacement algorithm.

if (xa(1,z)<xa(1,z+1))

   xm=[xa(1,z):inter:xa(1,z+1)];
else
   xm=[xa(1,z):-inter:xa(1,z+1)];
end

%slope=(ya(z+1)-ya(z))/(xa(z+1)-xa(z));
slope=0 ;
%%  checking the direction of the slope and then rotating the points
%%  accordingly

if sign(slope)==1
   angle=atan(slope);
elseif sign(slope)==-1
   angle=atan(slope);
else
   angle=0;
end

local_slope=0;
local_angle=0;
```

```
  angle=(pi/2)-angle;
  new_angle=angle;

  for k=1:sizeyam(1,1)

%    if ry > 1
%       local_slope= (yam(ry,1)-ya(z))/(xm(1,ry)-xa(z));
%       local_angle=atan(local_slope);
%       new_angle=local_angle+angle;
%    end

    yamfull(k,z)= yam(k,1);
    xmfull(k,z) =xm(1,k);

    C=( R_theta_minus*[xm(1,k)-xa(1,z);yam(k,1)-ya(1,z)]) + [xa(1,z); ya(1,z)];
    xm_prime(k,1)=C(1,1);
    ym_prime(k,1)=C(2,1);
    %new_angle=0;
 end

 figure(10)
 axis off

 for i=1:sizeyam(1,1)-1

   if i==1
     xm1(1,i)=xm(1,1);
     yam1(i,1)=yam(1,1);
   end
   if (i <=sizeyam(1,1)-2)
```

```
        xm1(1,i+1)=yam(i+1,1)*cos(xm(1,i+1));
        yam1(i+1,1)=yam(i+1,1)*sin(xm(1,i+1));
elseif i==(sizeyam(1,1)-1)
       xm1(1,i+1)=xm(1,i+1);
       yam1(i+1,1)=yam(i+1,1);
      end

      hold on;

      plot([xm_prime(i,1),xm_prime(i+1,1)],[ym_prime(i,1),ym_prime(i+1,1)]);
      hold on;

  end

        final_result_xaya(1, (z-1)*no_of_points_after_mpd+1 : (z-1) * no_of_points_after_mpd+1 +
(no_of_points_after_mpd-1) )=xm_prime(:,1)'; % final_result is a row vector while xm_prime is column
        final_result_xaya(2, (z-1)*no_of_points_after_mpd+1 : (z-1) * no_of_points_after_mpd+1 +
(no_of_points_after_mpd-1) )=ym_prime(:,1)'; % final_result is a row vector while ym_prime is column

 hold on;

  clear yam1;
  clear xm1;
  clear yam;
  clear xm;

  xa=x_orig;
  ya=y_orig;
end
save fdatadenoising1-y yamfull;
save fdatadenoising1-x xmfull;
save final_result_xaya1 final_result_xaya;
```

A.3 Calculate Rényi Dimension Spectrum (Calc_Reyni.m)

```
%-----------------------------------------------------------------------------------------------------------
% Calc_Reyni.m
% loads data from 2 files and calculate reyni spectrum for each one and display
% Returns  -None
% Author(s): Sharjeel Siddiqui, 01-Jan-2004 Copyright (c) Sharjeel Siddiqui.
%-----------------------------------------------------------------------------------------------------------


load final_result_xaya;
figure(100)
for(q=1:21),[D(q,:), R] = Renyi3(final_result_xaya(1,:),final_result_xaya(2,:), 10, q-11),end

plot(log(R),-D)

% based on plot, see where there are straight lines, calculate between 1 to 5 or 6,
t=1:5; % 5 or 5 if you want
% Hint: fine scales are from t=1 to 5
figure(101)

for(k2=1:21),polyD1(k2,:) = polyfit(log(R(1,t)),-D(k2,t),1);end

plot(-10:10,polyD1(:,1),'.-'),grid
hold on;



clear all;
load final_result_xaya1;


figure(102)
```

```
for(q=1:21),[D(q,:), R] = Renyi3(final_result_xaya(1,:),final_result_xaya(2,:), 10, q-11),end

plot(log(R),-D)

% based on plot, see where there are straight lines, calculate between 1 to 5 or 6,
t=1:5; % 5 or 5 if you want
% Hint: fine scales are from t=1 to 5
figure(101)
for(k2=1:21)
   polyD1(k2,:) = polyfit(log(R(1,t)),-D(k2,t),1);
end

plot(-10:10,polyD1(:,1),'+-'),grid

%-----------------------------------------------------------------------------------------------------------
% Reyni3.m
% calculate reyni spectrum
% LAtitude: Latitude vector
% Longitude: Longitude vector
% L: number of levels
% Q: the vector of weights
% Returns  -None
% Author(s): Sharjeel Siddiqui, 01-Jan-2004 Copyright (c) Sharjeel Siddiqui.
% Adopted form Aram Faghfouri's code with his permission
%-----------------------------------------------------------------------------------------------------------

function [D, R] = Renyi3(Longitude,Latitude, L, q)

Lon1 = Longitude - min(Longitude);
Lat1 = Latitude - min(Latitude);
clear Longitude Latitude
```

```
N = length(Lon1); %length of fractal
%NQ = length(Q);

rx = max(Lon1)*1.00001;
ry = max(Lat1)*1.00001;
r = rx/2^max(L); % The level of map segmentation. 4^level is the number of produced cells

R(1,L) = r;


for(level=L:-1:1)
  r = rx/2^level; % The level of map segmentation. 4^level is the number of produced cells
  R(1,level) = r;
  P = zeros(1,2^(2*level));
  N1 =2^level;
  for(i=1:N)
    % P(floor(Lon1(i)/r)+1,floor(Lat1(i)/r)+1) = P(floor(Lon1(i)/r)+1,floor(Lat1(i)/r)+1) + 1;
    P(N1*floor(Lon1(i)/r)+floor(Lat1(i)/r)+1) = P(N1*floor(Lon1(i)/r)+floor(Lat1(i)/r)+1) + 1;
%     P2(floor(Lon1(i)/r)+1,floor(Lat1(i)/r)+1) = P(floor(Lon1(i)/r)+1,floor(Lat1(i)/r)+1) + 1;
%     P(floor(Lon1(i)/r)+1,floor(Lat1(i)/r)+1) = P(floor(Lon1(i)/r)+1,floor(Lat1(i)/r)+1) + 1;
  end

P = P/N;
% N1 = 2^level;
k = 0;
s = find(P>0);
p1 = P(s);

  if(q~=1)
    D(level) = log(sum(p1.^q))/(1-q);
    else
        D(level) = -sum(p1.*log(p1));
    end
end
```

A.4 Vel Covering (rwindow.m)

```
%----------------------------------------------------------------------------------------------------------------
% rwindow.m
% calculates vel covering ( static ve covering) based fractal dimension
% Calculates Reyni Dimesnion in each vel
% calls  chain code and thinning functions
% Pts - The storage array contacinging all the points of the boundary
% Returns  - boxesw storage array that contains various results for static vel covering
% Author(s): Sharjeel Siddiqui, 10-Feb-2001 Copyright (c) Sharjeel Siddiqui.
%----------------------------------------------------------------------------------------------------------------
function boxesw=rwindow(pts,fig)
iter=2;  % number of iteartions in each window
  nryn=41;% used for Reyni calculation
  winsiz=8;  % minimum window size
  z=1;
  winthresh=2;  %% minimum number of points in a window
entropyk=zeros(iter,1);
  hosk=zeros(iter,1);
  intseck=zeros(iter,1);
  Corrk=zeros(iter,1);
  rsize=zeros(iter,1);
maxm=max(pts);
  minm=min(pts);
maxmw=max(pts);
  minmw=min(pts);
spt=size(pts);  %% pts is a   N x 2 matrix
Fdimpts=zeros(size(pts)); % this array is used for passing points of the fractal curve whcih lie in a window
% to the FdimCalcu function for F dim calculation.
maxxdif=(maxm(1)-minm(1))/winsiz;
  maxydif=(maxm(2)-minm(2))/winsiz;
```

```
maxm(1)=ceil(minm(1)+maxxdif);
  maxm(2)=ceil(minm(2)+maxydif);


  %minm=minm/8;


  if ( (mod(maxm(1),2)~=0) & (mod(maxm(2),2)~=0))


    if mod(maxm(1),2)~=0
      maxm=maxm+1;
    end


    if mod(maxm(2),2)~=0
      maxm(2)=maxm(2)+1;
    end
 end



 % BASIC WINDOW ORIGINATION
 % finding the total number of points in each window and omitting the window  if its less then a threshold


 x1=[minmw(1):maxxdif:maxmw(1)];
 y1=[maxmw(2):-maxydif:minmw(2)];


 [xm1,ym1]=meshgrid(x1,y1);


 sxm=size(xm1);
 sym=size(ym1);


 a=sym(1)+1;
 b=sxm(2)*2;
```

```
  mesh2=zeros([a,b]);


  nboxesw=(b/2-1)*(a-1);

  nboxeshor=b/2-1;

  nboxesver=a-1;


 boxesw=zeros(nboxesw,13); % No 13 is used for amount info required for each boz like coordinates  chain
codes etc etc fractla diomesnion


 for i=1:a

   k=1;

    o=2;


   if i==a

      for j=1:b

        mesh2(i,j)=0;

      end

   else


     for j=1:(b/2)


        mesh2(i,k)=xm1(i,j);

        mesh2(i,o)=ym1(i,j);

        o=o+2;

        k=k+2;


     end

  end


end
```

```
sm2=size(mesh2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Basic Windows have been created as a mesh above, now we begin
% Processing basic window or vel, which is shown as bold lines in the
% figure with mesh at the end of this programme. One of the ways that we
% find Fractal dimension inside a vel is spectrum dimension method. To do that we
% calculate fractal dimension therefore we need sub windows/vels in the
% basic vel. Sub vels are made and processd later in the programme

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 y=1;

 for u=1:(sm2(1))

  for v=1:2:(sm2(2)-2)

    %curve_break_flag=0; % used to flag the end of a continuous curve in a box or a vel.
    no_of_curves_in_vel=0;
    end_point_storage_vector =[]; % empty matrix or vector
    last_point=0;
    start_point_on_original_curve=0;
    end_point_on_original_curve=0;

    for i=1:spt(1)

       %pts(i,1)
       %pts(i,2)

       if u<=sm2(1)-1
    if v<=sm2(2)-2

  % boxesw =[x1,y1,x2,y1,x3,y3,x4,y4] has  all 4 coordinates for
  % a rectangular window boxesw[:1] contains number of points
  % in the box

      boxesw(y,2)=mesh2(u,v);
      boxesw(y,3)=mesh2(u,v+1);
      boxesw(y,4)=mesh2(u,v+2);
      boxesw(y,5)=mesh2(u,v+1);
      boxesw(y,6)=mesh2(u+1,v);
      boxesw(y,7)=mesh2(u+1,v+1);
      boxesw(y,8)=mesh2(u+1,v+2);
      boxesw(y,9)=mesh2(u+1,v+1);

      if pts(i,1)>=mesh2(u,v)
       if pts(i,1)<=mesh2(u,v+2)
         if pts(i,2)<=mesh2(u,v+1)
           if pts(i,2)>=mesh2(u+1,v+1)
             Fdimpts(i,1)=pts(i,1);
             Fdimpts(i,2)=pts(i,2);
             % sending the fractal window with
             % points for calculation to the Fdim
             % fucntion.

             % number of points ina box or window.
             boxesw(y,1)=boxesw(y,1)+1;

             % basically we check for a contninuous
             % curves in the present vel
             if (boxesw(y,1)==1 || last_point==0)  % if its the start of curve
```

```
                    start_point_on_original_curve=i;
                    last_point=i;
                  else
                   % if  (curve_break_flag==0)  % if the curve was not broken anytime in the vel keep
on checking
                      if (i==last_point+1)
                        last_point=i;
                      else
                        curve_break_flag=1;   % curve is not continuous anymore in the vel
                        no_of_curves_in_vel = no_of_curves_in_vel+1;
                        temp_vector=[start_point_on_original_curve,end_point_on_original_curve];
                       end_point_storage_vector=[end_point_storage_vector(:,:),temp_vector(:,:)]; %
end_point_storage_vector inreases in size as new end points are added
                        last_point=0; % start again this is the end of the present curve
                         i=i-1; % decrease the value of loop by 1 so that next time we are the start of
the new curve
                      end
                    %end
                  end
                  end_point_on_original_curve=i;


                end
              end
            end
          end


        maxm(1)=mesh2(u,v+2);
        maxm(2)=mesh2(u,v+1);
        minm(1)=mesh2(u,v);
        minm(2)=mesh2(u+1,v+1);


      end
```

```
        end


    end  % end i=1:spt(1)


    % extreme points of the current vel


    %% for the last curve in the vel we have to do the following so
    %% that end_point_storage_vector has all the data for all the
    %% curves in the vel at the end of the loop
    temp_vector=[start_point_on_original_curve,end_point_on_original_curve];


    if (temp_vector(1,1)~=0 && temp_vector(1,2)~=0 && no_of_curves_in_vel==0 ) % if there was only
one curve in the vel the we have to update no_of_curves_in_vel
      no_of_curves_in_vel=1;
    end


                          end_point_storage_vector=[end_point_storage_vector(:,:),temp_vector(:,:)];     %
end_point_storage_vector inreases in size as new end points are added


    size_end_point_storage_vector=size(end_point_storage_vector)
    if (size_end_point_storage_vector(1,2) > 2 )
      wytwuyt=1;
    end
    % At this point we have found all the curves passing
    % through our present vel i.e boxes(z,1), now lets play with
    % curves and find related IFS for each curve in a vel.


    if (boxesw(y,1) > 10 && y==25)
      IFS = calc_IFS_for_vel ( no_of_curves_in_vel, end_point_storage_vector, pts,boxesw);
    end


    % CALCULATE FRACTAL DIMENSION D_B IN EACH VEL
    %Fdimension=Fdimcalcu(pts,Fdimpts,mesh2(u,v),mesh2(u,v+2),mesh2(u,v+1),mesh2(u+1,v+1));
    end_point_storage_vector=[];
    temp_vector=[];

    % SUB-WINDOW ORIGINATION for Box counting and Reyni

    y=y+1;  %% going to next vel

    bsize=(maxm(1)-minm(1))/2;  %% calculating the extremes of the mesh and find the half of it for first
grid scale
    ysize=(maxm(2)-minm(2))/2;

  if ((y-1) <= nboxesw)  % doing y-1 as we had increased above for next vel

    if boxesw((y-1),1)>=winthresh

      for w=1:iter

        rsize(w,1)=bsize;  % keep track of all r sizes for calculation of fractal dimension

        x2=[minm(1):bsize:maxm(1)];
        y2=[maxm(2):-ysize:minm(2)];

        [xm,ym]=meshgrid(x2,y2);

        sxm=size(xm);
        sym=size(ym);

        a=sym(1)+1;
        b=sxm(2)*2;
```

```
mesh1=zeros([a,b]);

nboxes=(b/2-1)*(a-1);

boxes=zeros(nboxes,1);

for i=1:a
  k=1;
  o=2;

  if i==a
    for j=1:b
      mesh1(i,j)=0;
    end
  else

    for j=1:(b/2)

      mesh1(i,k)=xm(i,j);
      mesh1(i,o)=ym(i,j);
      o=o+2;
      k=k+2;

    end
  end

end

sm1=size(mesh1);
```

```
z=1;


figure(fig);

hold on;


for j=1:(sm1(1))


  for k=1:2:(sm1(1,2)-2)


    if ( (j==1) || (j==sm1(1,1) ) )


    plot([mesh1(j,k),mesh1(j,k+2)],[mesh1(j,k+1),mesh1(j,k+1)],'linewidth',2.5);  %% horizontal
lines

      hold on;


    else

      %plot([mesh1(j,k),mesh1(j,k+2)],[mesh1(j,k+1),mesh1(j,k+1)],'linewidth',1);

      %hold on;

    end


    for i=1:spt(1)


      %pts(i,1)

      %pts(i,2)

      if j<=sm1(1)-1

        if k<=sm1(2)-2

          if pts(i,1)>=mesh1(j,k)

            if pts(i,1)<=mesh1(j,k+2)

              if pts(i,2)<=mesh1(j,k+1)

                if pts(i,2)>=mesh1(j+1,k+1)
```

```
                    boxes(z,1)=boxes(z,1)+1;

                 end
               end
             end
           end
         end
       end

      end % for loop

      z=z+1;

    end % for loop
  end % for loop

  cboxes=0;
  intsec=0;
  Hk=0;
  h=0;
  Ck=0;
  Re=0;
  ryn=-10;

  for i=1:nboxes
    if boxes(i)~=0
      cboxes=cboxes+1;
      intsec=intsec+boxes(i);
    end
  end
```

```
for i=1:nboxes
  if boxes(i)~=0
    Hk=Hk+((boxes(i)/intsec)*log(boxes(i)/intsec));
    Ck=Ck+(boxes(i)/intsec)^2;
    for rk=1:nryn
      gq=(boxes(i)/intsec)^(ryn);
      %Reny(rk,w)=Reny(rk,w) + gq;
      ryn=ryn+0.5;
    end
    gq=0;
  end
end


k=2;
m=1;
for j=1:(sm1(1,2)/2)


  for p=1:(sm1(1,1)-2)
      if ( (j==1) | (j==sm1(1,2)/2) )


            plot([mesh1(m,k-1),mesh1(m,k-1)],[mesh1(m,k),mesh1(m+1,k)],'linewidth',2.5); %%
mesh1= x1y1 x2y2 x3y3  x4y4 ..... verticle lines
        hold on;


        %            if (j==sm1(1,2)/2)
% plot([mesh1(m,k-1),mesh1(m,k-3)],[mesh1(m,k),mesh1(m,k)],'linewidth',1.5);
%% mesh1= x1y1 x2y2 x3y3  x4y4 ..... verticle lines
        %             hold on;
        %            end
    else
```

```
%plot([mesh1(m,k-1),mesh1(m,k-1)],[mesh1(m,k),mesh1(m+1,k)],'linewidth',1);   % vertical lines
                hold on;
             end
             %plot([mesh1(1,j),mesh1((sm1(1)-1),j)],[mesh1(1,j+1),mesh1(sm1(1),j+1)]);
             hold on;
             m=m+1

          end

          %% subtrack 1 from m =m-1 as it got increased by one at the end of
          %% the above loop

          if (j==1)
           start_x=mesh1(m-1,k-1); % at this point we have the full verticle bar of the  left side of the box
             start_y=mesh1(m,k);
          end
          if (j==sm1(1,2)/2)
          stop_x=mesh1(m-1,k-1); % at this point we have the full verticle bar of the right side of the box
            stop_y=mesh1(m,k);

           plot([stop_x,start_x],[stop_y,start_y],'linewidth',2.5); %% mesh1= x1y1 x2y2 x3y3  x4y4 .....
verticle lines
           hold on;
        end

        m=1;
        k=k+2;
      end

      bsize=bsize/2;
      ysize=ysize/2
```

```
    entropyk(w)=-Hk;
    hosk(w)=cboxes;  %  array number of vels per iteration
    intseck(w)=intsec;
    Corrk(w)=-Ck;
    Hk=0;
    cboxes=0;
    intsec=0;



  boxsize=zeros(iter-1,1);
  boxnum=zeros(iter-1,1);
  for i=1:iter-1
     boxsize(i,1)=-rsize(i,1)/rsize(i+1,1);

  end

 for i=2:iter
    boxnum(i-1,1)=hosk(i,1)/hosk(i-1,1);

 end

%figure(fig+1);
%plot(log(boxsize),log(boxnum));
%plot(-log(rsize),log(hosk));
%hold on;
%plot(-log(rsize),log(hosk),'o');



if iter >2
```

```
          [sleq,struc]= polyfit(-log(rsize(1:iter-1)),log(hosk(1:iter-1)),1);
         else
          [sleq,struc]= polyfit(-log(rsize(1:iter)),log(hosk(1:iter)),1);
         end


         boxesw(y-1,10)=sleq(1,1);
        %siz(1,1)=rsize(1,1)/rsize(2,1);


      end  % if
    end % if
  end
end


%save boxesw boxesw;
%save nboxeshor nboxeshor;
%save nboxesver nboxesver;
%save winthresh winthresh;
boxesw=thining(boxesw,nboxeshor,nboxesver,winthresh,fig);
boxesw=chaincode(boxesw,nboxeshor,nboxesver,winthresh,fig);


figure(fig+2);
k=1;
maxbox=max(boxesw);
while k<=maxbox(1,11);


for i=1:nboxeshor*nboxesver
if boxesw(i,11)==k
      plot(k,boxesw(i,10),'ro');
      hold on;
      k=k+1;
      break;
```

```
        end
    end


end
```

A.5 Chain codes (chaincode.m)

```
%--------------------------------------------------------------------------------------------------
% chaincode.m
% calculates chain codes on a thinned ( vel covering)
% boxesw - The storage array contacinging all the points of the boundary
% nboxeshor - horxontal points
% nboxesver - Vertilca points
% winthresh - number of points acting as threshold
% Returns  - boxesw storage array that contains various results for static vel covering
% Author(s): Sharjeel Siddiqui, 10-Feb-2001 Copyright (c) Sharjeel Siddiqui.
%--------------------------------------------------------------------------------------------------
function boxesw=chaincode(boxesw,nboxeshor,nboxesver,winthresh,fig)


sizboxesw=size(boxesw);
chaindir=zeros(sizboxesw(1,1),1);
k=1;
prevpos=0;
nextpos=0; %  first window with some boundary inside
status=0;


for i=1:sizboxesw(1,1)
   if ((boxesw(i,1) > winthresh) & (boxesw(i,12)==0)) % starting from the first box which has more points
then the threshold.. as our first box in chaincode
      nextpos=i;
      boxesw(i,11)=1;
      status=1;
      break;
   end
end
   if nextpos~=0
      figure(fig);
```

```
                plot((boxesw(nextpos,2)+boxesw(nextpos,4))/2,(boxesw(nextpos,3)+boxesw(nextpos,7))/
2,'r+','linewidth',1.5);
        hold on;
    end
    while status

        dir=chainfind(boxesw,nextpos,sizboxesw,nboxeshor,nboxesver,winthresh,prevpos);

        prevpos=nextpos;
        nextpos=dir(1,2);

        if boxesw(nextpos,11)==0
            boxesw(nextpos,11)=k;
            k=k+1;
                plot((boxesw(nextpos,2)+boxesw(nextpos,4))/2,(boxesw(nextpos,3)+boxesw(nextpos,7))/
2,'ro','linewidth',1.5);
            hold on ;
        else
            status=0;
        break;
    end

  end

 j=0;
```

A.6 Thining Algorithm (Thinning.m)

```
%---------------------------------------------------------------------------------------------------
% thinning.m
% Applies thinning algorith on vels (vel covering)
% boxesw - The storage array contacinging all the points of the boundary
% nboxeshor - horxontal points
% nboxesver - Vertilca points
% winthresh - number of points acting as threshold
% Returns  - boxesw storage array that contains various results for static vel covering
% Author(s): Sharjeel Siddiqui, 10-Feb-2001 Copyright (c) Sharjeel Siddiqui.
%---------------------------------------------------------------------------------------------------

function boxesw=thining(boxesw,nboxeshor,nboxesver,winthresh,fig);

  sizboxesw=size(boxesw);
  NP=0;
  SP=0;

  P2=0;
  P3=0;
  P4=0;
  P5=0;
  P6=0;
  P7=0;
  P8=0;
  P9=0;

  hold on;
  figure(fig);
  hold on ;
```

```
 stopf=0;


while stopf==0

    stopf=1;

 for o=1:2


  for j=1:sizboxesw(1,1)

    j

   if ( ( boxesw(j,12)==0) & (boxesw(j,1)>= winthresh))


     %% we always look forward


     if  j<nboxeshor

           NP=0;

           SP=0;

           P9=0;

           P2=0;

           P3=0;

           if ((boxesw(j+1,1)>= winthresh) & (boxesw(j+1,12)==0))

             NP=NP+1;

             SP=SP+1;

             P4=1;


          else

             if  ( (boxesw(j+nboxeshor+1,1) >= winthresh ) & (boxesw(j+nboxeshor+1,12)==0))

                  SP=SP+1;

             end


            p4=0;


          end
          if ((boxesw(j+nboxeshor+1,1)>= winthresh ) & (boxesw(j+nboxeshor+1,12)==0))
            NP=NP+1
            P5=1;



         else
            if  ( (boxesw(j+nboxeshor,1) >= winthresh ) & (boxesw(j+nboxeshor,12)==0))
                 SP=SP+1;
            end
            P5=0;

         end

         if (( boxesw(j+nboxeshor,1)>= winthresh ) & (boxesw(j+nboxeshor,12)==0))  %% only counting
NP
              NP=NP+1;
              P6=1;

         else

              P6=0;
         end



         if j~=1

              if (( boxesw(j+nboxeshor,1)>= winthresh ) & (boxesw(j+nboxeshor,12)==0))  %% only
counting NP
              NP=NP+1;
              P6=1;
```

```
      else
        if ( ( boxesw(j+(nboxeshor-1),1) >= winthresh) & (boxesw(j+(nboxeshor-1),12)==0))
          SP=SP+1;
        end
        P6=0;
      end


      if (( boxesw(j+(nboxeshor-1),1)>= winthresh ) & (boxesw(j,12)==0))
        NP=NP+1;
        P7=1;

      else
        if ( ( boxesw(j-1,1) >= winthresh) & (boxesw(j-1,12)==0))
          SP=SP+1;
        end
        P7=0;
      end


      if (( boxesw(j-1,1)>= winthresh ) & (boxesw(j-1,12)==0))
        NP=NP+1;
        P8=1;

      else

        P8=0;

      end

    end
```

```
elseif ((mod(j,nboxeshor)==0) & (j~= nboxeshor*nboxesver))

    NP=0;
    SP=0;
    P3=0;
    P4=0;
    P5=0;

   if j==nboxeshor

     P2=0;

     if ( ( boxesw(j-1,1)>= winthresh ) & (boxesw(j-1,12)==0))
          NP=NP+1;
          P8=1;
    else
        P8=0;

    end

  else

    if (( boxesw(j-nboxeshor,1)>= winthresh ) & (boxesw(j-nboxeshor,12)==0))
         NP=NP+1;
         P2=1;
    else
        P2=0;

    end
```

```
          if (( boxesw(j-1,1)>= winthresh ) & (boxesw(j-1,12)==0))
               NP=NP+1;
               P8=1;

         else
             if ( (boxesw(j-(nboxeshor+1),1) > winthresh ) & (boxesw(j-(nboxeshor+1),12)==0))
                 SP=SP+1;
             end

           P8=0;

         end


         if (( boxesw(j-(nboxeshor+1),1)>= winthresh ) & (boxesw(j-(nboxeshor+1),12)==0))
              NP=NP+1;
              P9=1;

         else

           if ( (boxesw(j-nboxeshor,1) >= winthresh ) & (boxesw(j-nboxeshor,12)==0))
                SP=SP+1;
              end


           P9=0;

        end


       end
```

```
        if (( boxesw(j+nboxeshor,1)>= winthresh ) & (boxesw(j+nboxeshor,12)==0))   % boundary condition
common

                NP=NP+1;

                P6=1;

                SP=SP+1;


        else

            if  ( (boxesw(j+(nboxeshor-1),1) >= winthresh ) & (boxesw(j+(nboxeshor-1),12)==0))
                    SP=SP+1;

            end


            P6=0;


        end



        if (( boxesw(j+(nboxeshor-1),1)>= winthresh ) & (boxesw(j+(nboxeshor-1),12)==0))   % boundary
condition common

                NP=NP+1;

                P7=1;



        else

            if  ( ( boxesw(j-1,1) >= winthresh ) & (boxesw(j-1,12)==0))

                    SP=SP+1;

            end


            P7=0;


        end
```

```
elseif (mod(j-1,nboxeshor)==0)    % going in to left column boundary

    NP=0;

    SP=0;

    P9=0;

    P8=0;

    P7=0;

   if j==((nboxeshor*nboxesver) -(nboxeshor-1))

     P6=0;

    if ( ( boxesw(j+1,1)>= winthresh ) & (boxesw(j+1,12)==0))
          NP=NP+1;
          P4=1;
    else
        P4=0;

    end

   else

    if (( boxesw(j+nboxeshor,1)>= winthresh ) & (boxesw(j+nboxeshor,12)==0))
          NP=NP+1;
          P6=1;
    else
        P6=0;

   end
```

```
        if (( boxesw(j+1,1)>= winthresh ) & (boxesw(j+1,12)==0))
            NP=NP+1;
            P4=1;


        else
            if  ( (boxesw(j+nboxeshor+1,1) > winthresh ) & (boxesw(j+nboxeshor+1,12)==0))
                SP=SP+1;
            end


         P4=0;


        end


       if (( boxesw(j+(nboxeshor+1),1)>= winthresh ) & (boxesw(j+(nboxeshor+1),12)==0))
           NP=NP+1;
           P5=1;


       else


         if  ( (boxesw(j+nboxeshor,1) >= winthresh ) & (boxesw(j+nboxeshor,12)==0))
             SP=SP+1;
           end
           P5=0;


       end
     end


    if (( boxesw(j-nboxeshor,1)>= winthresh ) & (boxesw(j-nboxeshor,12)==0))   % boundary condition
common

          NP=NP+1;
          P2=1;
```

```
            SP=SP+1;

    else
        if ( (boxesw(j-(nboxeshor-1),1) >= winthresh ) & (boxesw(j-(nboxeshor-1),12)==0))
              SP=SP+1;
        end

        P2=0;

    end

     if (( boxesw(j-(nboxeshor-1),1)>= winthresh ) & (boxesw(j-(nboxeshor-1),12)==0))   % boundary
condition common
            NP=NP+1;
            P3=1;

    else
        if ( ( boxesw(j+1,1) >= winthresh ) & (boxesw(j+1,12)==0))
              SP=SP+1;
        end

        P3=0;

    end

  elseif (j >((nboxeshor*nboxesver) -nboxeshor) )

       NP=0;
```

```
SP=0;
P7=0;
P6=0;
P5=0;

if ((boxesw(j-1,1)>= winthresh) & (boxesw(j-1,12)==0))
  NP=NP+1;
  SP=SP+1;
  P8=1;

else
  if ( (boxesw(j-(nboxeshor+1),1) >= winthresh ) & (boxesw(j-(nboxeshor+1),12)==0))
      SP=SP+1;
  end

  P8=0;

end

if ((boxesw(j-(nboxeshor+1),1)>= winthresh ) & (boxesw(j-(nboxeshor+1),12)==0))
  NP=NP+1
  P9=1;


else
  if ( (boxesw(j-nboxeshor,1) >= winthresh ) & (boxesw(j-nboxeshor,12)==0))
      SP=SP+1;
  end
  P9=0;

end
```

```
        if (( boxesw(j-nboxeshor,1)>= winthresh ) & (boxesw(j-nboxeshor,12)==0))  %% only counting NP
                NP=NP+1;
                P2=1;


         else
                P2=0;
         end
if j~=nboxeshor*nboxesver

            if (( boxesw(j-nboxeshor,1)>= winthresh ) & (boxesw(j-nboxeshor,12)==0))  %% only counting
NP
               NP=NP+1;
               P2=1;


             else
               if ( ( boxesw(j-(nboxeshor-1),1) >= winthresh) & (boxesw(j-(nboxeshor-1),12)==0))
                  SP=SP+1;
               end
               P2=0;
             end



            if (( boxesw(j-(nboxeshor-1),1)>= winthresh ) & (boxesw(j-(nboxeshor-1),12)==0))
               NP=NP+1;
               P3=1;


            else
              if ( ( boxesw(j+1,1) >= winthresh) & (boxesw(j+1,12)==0))
                 SP=SP+1;
              end
            P3=0;

           end



           if (( boxesw(j+1,1)>= winthresh ) & (boxesw(j+1,12)==0))

             NP=NP+1;

             P4=1;

       else

             P4=0;


           end


         end


       else

           NP=0;

           SP=0;


         if (( boxesw(j-nboxeshor,1)>= winthresh ) & (boxesw(j-nboxeshor,12)==0))  %% only counting
NP

            NP=NP+1;

            P2=1;


          else

            if ( ( boxesw(j-(nboxeshor-1),1) >= winthresh) & (boxesw(j-(nboxeshor-1),12)==0))

              SP=SP+1;

            end

            P2=0;

          end
```

```
            if (( boxesw(j-(nboxeshor-1),1)>= winthresh ) & (boxesw(j-(nboxeshor-1),12)==0))  %% only
counting NP

               NP=NP+1;

               P3=1;


            else

               if ( ( boxesw(j+1,1) >= winthresh) & (boxesw(j+1,12)==0))

                  SP=SP+1;

               end

               P3=0;

            end



            if (( boxesw(j+1,1)>= winthresh ) & (boxesw(j+1,12)==0))  %% only counting NP

               NP=NP+1;

               P4=1;


            else

               if ( ( boxesw(j+(nboxeshor+1),1) >= winthresh) & (boxesw(j+(nboxeshor+1),12)==0))

                  SP=SP+1;

               end

               P4=0;

            end



            if (( boxesw(j+(nboxeshor+1),1)>= winthresh ) & (boxesw(j+(nboxeshor+1),12)==0))  %%
only counting NP

               NP=NP+1;

               P5=1;


            else

               if ( ( boxesw(j+nboxeshor,1) >= winthresh) & (boxesw(j+nboxeshor,12)==0))
            SP=SP+1;
          end
          P5=0;
        end


          if (( boxesw(j+nboxeshor,1)>= winthresh ) & (boxesw(j+nboxeshor,12)==0))  %% only
counting NP
          NP=NP+1;
          P6=1;


        else
          if ( ( boxesw(j+(nboxeshor-1),1) >= winthresh) & (boxesw(j+(nboxeshor-1),12)==0))
            SP=SP+1;
          end
          P6=0;
        end


          if (( boxesw(j+(nboxeshor-1),1)>= winthresh ) & (boxesw(j+(nboxeshor-1),12)==0))  %%
only counting NP
          NP=NP+1;
          P7=1;


        else
          if ( ( boxesw(j-1,1) >= winthresh) & (boxesw(j-1,12)==0))
            SP=SP+1;
          end
          P7=0;
        end


        if (( boxesw(j-1,1)>= winthresh ) & (boxesw(j-1,12)==0))  %% only counting NP
          NP=NP+1;
          P8=1;
```

```
            else
              if ( ( boxesw(j-(nboxeshor+1),1) >= winthresh) & (boxesw(j-(nboxeshor+1),12)==0))
                SP=SP+1;
              end
              P8=0;
            end

             if (( boxesw(j-(nboxeshor+1),1)>= winthresh ) & (boxesw(j-(nboxeshor+1),12)==0))  %%
only counting NP
              NP=NP+1;
              P9=1;

            else
              if ( ( boxesw(j-nboxeshor,1) >= winthresh) & (boxesw(j,12)==0))
                SP=SP+1;
              end
              P9=0;
            end

        end

        if ((o==1) & (NP>=2) & (NP<=6) & (SP==1) & (P2*P4*P6==0) & (P4*P6*P8==0))
          boxesw(j,13)=1;
          stopf=0;
          plot((boxesw(j,2)+boxesw(j,4))/2,(boxesw(j,3)+boxesw(j,7))/2,'rx','linewidth',1.5);
          hold on ;
        end
        if ((o==2) & (NP>=2) & (NP<=6) & (SP==1) & (P2*P4*P8==0) & (P2*P6*P8==0))
          boxesw(j,13)=1;
          stopf=0;
          plot((boxesw(j,2)+boxesw(j,4))/2,(boxesw(j,3)+boxesw(j,7))/2,'rx','linewidth',1.5);
          hold on ;
        end


      end  % if


    end  % for
      for j=1:sizboxesw(1,1)
        if boxesw(j,13)==1
          boxesw(j,12)=1;
        end
      end
  end % for o
end % while
```

A.7 Various functions for Wavelet based Approach

```
%-----------------------------------------------------------------------------------------------------------
% findcor.m
% Finds the high curvature and corner points on a given binary image
% G - Source boundary image
% P  - returns corner points
% Author(s): Sharjeel Siddiqui, 10-Feb-2002 Copyright (c) Sharjeel Siddiqui.
%-----------------------------------------------------------------------------------------------------------


function p=findcor(G)
[x,m]=bmpread(G);
[row,col]=size(x)

nx=1-thr10(x,1);


b=boundaty_track(nx);
phi=calc_orient(b);
fb=fixb(b,phi);
nphi=myorient(fb);
w=dwt(nphi,4);
c=cor_with_WTMM(w,0.4,0.15,0.65);
mynmark(fb,c(:,1),row,col);
p1=fb(c(:,1),:);
p=[p1 c(:,2)];
figure(10);
plot(nphi);
XLABEL('Path Length [Pixels]');
YLabel('Phi(t) [radians]');
Title(' Orientation-space representation');
```

```
return


% Mark the HCPS and Corners
function mynmark(b,c,row,col)


o=zeros(row,col);


r=size(b,1);


for i=1:r
o(b(i,1),b(i,2))=1;
end


mark2(o,b(c,:));


return
```

```
%-----------------------------------------------------------------------------------------------------------
% boundary_track.m
% Tracks the boundary
% x - bitmap points
% new  - returns corner points
% Author(s): Sharjeel Siddiqui, 10-Feb-2002 Copyright (c) Sharjeel Siddiqui.
%-----------------------------------------------------------------------------------------------------------

function new=boundaty_track(x)

[r,c]=size(x);
count=0;

for i=1:r
  fnd=find(x(i,:));
  if ~isempty(fnd)
    start=[i fnd(1)];
    break;
  end
end
start


if x(start(1)+1,start(2)-1)==1; %if x(start(1)+1,start(2)-1)==1;
  cur=[start(1)+1 start(2)-1];
else
  cur=[start(1)+1 start(2)];
end


last=start;
```

```
new(1:2,:)=[start;cur];

n=2;



while mean(abs(cur-start))~=0



 blk=x(cur(1)-1:cur(1)+1,cur(2)-1:cur(2)+1);

 if cur-last==[1 -1]

   if blk(1,2)==1

     n=n+1;

     new(n,1)=cur(1)-1;

     new(n,2)=cur(2);


  elseif blk(1,1)==1

    n=n+1;

    new(n,1)=cur(1)-1;

    new(n,2)=cur(2)-1;


  elseif blk(2,1)==1

   n=n+1;

   new(n,1)=cur(1);

   new(n,2)=cur(2)-1;


  elseif blk(3,1)==1

   n=n+1;

   new(n,1)=cur(1)+1;

   new(n,2)=cur(2)-1;


 elseif blk(3,2)==1

  n=n+1;
```

```
    new(n,1)=cur(1)+1;
    new(n,2)=cur(2);

   elseif blk(3,3)==1
    n=n+1;
    new(n,1)=cur(1)+1;
    new(n,2)=cur(2)+1;

   elseif blk(2,3)==1
    n=n+1;
    new(n,1)=cur(1);
    new(n,2)=cur(2)+1;
   else
   end

%****last=blk(1,2)********
 elseif cur-last==[1 0]
  if blk(1,1)==1
   n=n+1;
   new(n,1)=cur(1)-1;
   new(n,2)=cur(2)-1;

  elseif blk(2,1)==1
   n=n+1;
   new(n,1)=cur(1);
   new(n,2)=cur(2)-1;

  elseif blk(3,1)==1
   n=n+1;
   new(n,1)=cur(1)+1;
   new(n,2)=cur(2)-1;
```

```
    elseif blk(3,2)==1
     n=n+1;
     new(n,1)=cur(1)+1;
     new(n,2)=cur(2);

    elseif blk(3,3)==1
     n=n+1;
     new(n,1)=cur(1)+1;
     new(n,2)=cur(2)+1;

    elseif blk(2,3)==1
     n=n+1;
     new(n,1)=cur(1);
     new(n,2)=cur(2)+1;

    elseif blk(1,3)==1
     n=n+1;
     new(n,1)=cur(1)-1;
     new(n,2)=cur(2)+1;
    else
    end

%*****last=blk(1,1)*******
  elseif cur-last==[1 1]
   if blk(2,1)==1
    n=n+1;
    new(n,1)=cur(1);
    new(n,2)=cur(2)-1;

   elseif blk(3,1)==1
  n=n+1;
  new(n,1)=cur(1)+1;
  new(n,2)=cur(2)-1;

 elseif blk(3,2)==1
  n=n+1;
  new(n,1)=cur(1)+1;
  new(n,2)=cur(2);

 elseif blk(3,3)==1
  n=n+1;
  new(n,1)=cur(1)+1;
  new(n,2)=cur(2)+1;

 elseif blk(2,3)==1
  n=n+1;
  new(n,1)=cur(1);
  new(n,2)=cur(2)+1;

 elseif blk(1,3)==1
  n=n+1;
  new(n,1)=cur(1)-1;
  new(n,2)=cur(2)+1;

 elseif blk(1,2)==1
  n=n+1;
  new(n,1)=cur(1)-1;
  new(n,2)=cur(2);
 else
 end
```

```
%****last=blk(2,1)***********
 elseif cur-last==[0 1]
  if blk(3,1)==1
   n=n+1;
   new(n,1)=cur(1)+1;
   new(n,2)=cur(2)-1;

  elseif blk(3,2)==1
   n=n+1;
   new(n,1)=cur(1)+1;
   new(n,2)=cur(2);

  elseif blk(3,3)==1
   n=n+1;
   new(n,1)=cur(1)+1;
   new(n,2)=cur(2)+1;

  elseif blk(2,3)==1
   n=n+1;
   new(n,1)=cur(1);
   new(n,2)=cur(2)+1;

  elseif blk(1,3)==1
   n=n+1;
   new(n,1)=cur(1)-1;
   new(n,2)=cur(2)+1;

  elseif blk(1,2)==1
   n=n+1;
   new(n,1)=cur(1)-1;
   new(n,2)=cur(2);
   elseif blk(1,1)==1
     n=n+1;
     new(n,1)=cur(1)-1;
     new(n,2)=cur(2)-1;
   else
   end

%***last=blk(3,1)*******
 elseif cur-last==[-1 1]
   if blk(3,2)==1
     n=n+1;
     new(n,1)=cur(1)+1;
     new(n,2)=cur(2);

   elseif blk(3,3)==1
     n=n+1;
     new(n,1)=cur(1)+1;
     new(n,2)=cur(2)+1;

   elseif blk(2,3)==1
     n=n+1;
     new(n,1)=cur(1);
     new(n,2)=cur(2)+1;

   elseif blk(1,3)==1
     n=n+1;
     new(n,1)=cur(1)-1;
     new(n,2)=cur(2)+1;

   elseif blk(1,2)==1
```

```
        n=n+1;
        new(n,1)=cur(1)-1;
        new(n,2)=cur(2);

      elseif blk(1,1)==1
        n=n+1;
        new(n,1)=cur(1)-1;
        new(n,2)=cur(2)-1;

      elseif blk(2,1)==1
        n=n+1;
        new(n,1)=cur(1);
        new(n,2)=cur(2)-1;
      else
      end

%****last=blk(3,2)*****
   elseif cur-last==[-1 0]
     if blk(3,3)==1
       n=n+1;
       new(n,1)=cur(1)+1;
       new(n,2)=cur(2)+1;

     elseif blk(2,3)==1
       n=n+1;
       new(n,1)=cur(1);
       new(n,2)=cur(2)+1;

     elseif blk(1,3)==1
       n=n+1;
       new(n,1)=cur(1)-1;
     new(n,2)=cur(2)+1;

    elseif blk(1,2)==1
     n=n+1;
     new(n,1)=cur(1)-1;
     new(n,2)=cur(2);

    elseif blk(1,1)==1
     n=n+1;
     new(n,1)=cur(1)-1;
     new(n,2)=cur(2)-1;

    elseif blk(2,1)==1
     n=n+1;
     new(n,1)=cur(1);
     new(n,2)=cur(2)-1;

    elseif blk(3,1)==1
     n=n+1;
     new(n,1)=cur(1)+1;
     new(n,2)=cur(2)-1;
    else
    end

%*****Last=blk(3,3)****
  elseif cur-last==[-1 -1]
   if blk(2,3)==1
    n=n+1;
    new(n,1)=cur(1);
    new(n,2)=cur(2)+1;
```

```
elseif blk(1,3)==1
 n=n+1;
 new(n,1)=cur(1)-1;
 new(n,2)=cur(2)+1;

elseif blk(1,2)==1
 n=n+1;
 new(n,1)=cur(1)-1;
 new(n,2)=cur(2);

elseif blk(1,1)==1
 n=n+1;
 new(n,1)=cur(1)-1;
 new(n,2)=cur(2)-1;

elseif blk(2,1)==1
 n=n+1;
 new(n,1)=cur(1);
 new(n,2)=cur(2)-1;

elseif blk(3,1)==1
 n=n+1;
 new(n,1)=cur(1)+1;
 new(n,2)=cur(2)-1;

elseif blk(3,2)==1
 n=n+1;
 new(n,1)=cur(1)+1;
 new(n,2)=cur(2);
else
end
```

```
%****last=blk(2,3)******
 elseif cur-last==[0 -1]
   if blk(1,3)==1
    n=n+1;
    new(n,1)=cur(1)-1;
    new(n,2)=cur(2)+1;

   elseif blk(1,2)==1
    n=n+1;
    new(n,1)=cur(1)-1;
    new(n,2)=cur(2);

   elseif blk(1,1)==1
    n=n+1;
    new(n,1)=cur(1)-1;
    new(n,2)=cur(2)-1;

   elseif blk(2,1)==1
    n=n+1;
    new(n,1)=cur(1);
    new(n,2)=cur(2)-1;

   elseif blk(3,1)==1
    n=n+1;
    new(n,1)=cur(1)+1;
    new(n,2)=cur(2)-1;

  elseif blk(3,2)==1
   n=n+1;
   new(n,1)=cur(1)+1;
```

```
      new(n,2)=cur(2);


    elseif blk(3,3)==1
      n=n+1;
      new(n,1)=cur(1)+1;
      new(n,2)=cur(2)+1;
    else
    end
  else
  end

last=cur;
cur=new(n,:);
%pause

end

cur

return
```

```
%------------------------------------------------------------------------------------------------
% calc_orient.m
% computes orientation profile
% b - Tracked boundary points
% phi  - returns corner points
% Author(s): Sharjeel Siddiqui, 10-Feb-2002 Copyright (c) Sharjeel Siddiqui.
%------------------------------------------------------------------------------------------------

function phi=calc_orient(b)

[sz,c]=size(b);
d=mydiff(b,2);
phi=myatan(d);
np=[fliplr(phi) phi fliplr(phi)];
t=-6:6;
g=(1/(2*pi*2)^(1/2))*exp(-t.^2/4);
phi=wkeep(conv(g,np),sz);

return
```

```
%---------------------------------------------------------------------
% dwt.m
%DWT Single-level discrete 1-D wavelet transform.
%     computes the wavelet decomposition using Spline
%     Wavelets (Defined by Mallat) with Dyadic Scaling.
%
%       x-> Input row vector
%       level-> Max level of decomposition
%
%     See also DWTMODE, DWTPER, IDWT, WAVEDEC, WAVEINFO.

%     Uses DYADDOWN, WKEEP, WFILTERS.
% Author(s): Sharjeel Siddiqui, 10-Feb-2002 Copyright (c) Sharjeel Siddiqui.
%---------------------------------------------------------------------


function [w,ap] = dwt(x,level)


LoF_D=[ 0 0 0.1250 0.3750 0.3750 0.1250 0];
HiF_D=[0 0 0 -2 2 0 0];
lambda=[1.5 1.12 1.03 1.01];

lx = length(x);
w=[];
ap=[];
%Making Symmetry
xd=[fliplr(x) x fliplr(x)];
ld=length(xd);

% Start.
```

```
for n=1:level
 lf=length(LoF_D);


   % Decomposition.
     if n <=4
       d  = wkeep((1/lambda(n))*conv(xd,HiF_D),ld);
     else
       d  = wkeep(conv(xd,HiF_D),ld);
     end


    a  = wkeep(conv(xd,LoF_D),ld);
  xd=a;
  w=[w;d];
  LoF_D=dyadup(LoF_D,2);
  HiF_D=dyadup(HiF_D,2);
 ap=[ap;a];


 end
 %save Last Appx.
 w=[w;a];


%Correcting for Symmetry
w=wkeep(w,[level+1 lx]);


return
```

```
%---------------------------------------------------------------------------------------------------
% cor_with_WTMM.m
% Finds the high curvature and corner points using WTMM
% w - Source boundary image
% t1  - threshold1
% t2  - threshold2
% t3 - threshold3
% Author(s): Sharjeel Siddiqui, 10-Feb-2002 Copyright (c) Sharjeel Siddiqui.
%---------------------------------------------------------------------------------------------------

function c=cor_with_WTMM(w,t1,t2,t3)

[row,col]=size(w);
level=row-1;

% Compute WTMM
for i=1:level
  %wmm(i,:)=mymodm(w(i,:)); % original
  wmm(i,:)=abs( mymodm(w(i,:)) ) % changes by SAS
  figure(i);
  plot(wmm(i,:));
end
awm=abs(wmm);

% Compute Score at each level
for i=1:level
 cmx=max(awm(i,:));
 score(i,:)=awm(i,:)/cmx;
%subplot(level,1,i), plot(score(i,:));
end
%pause
```

```
%Start from last level
%mw=max(awm(level,:));
pos=find(awm(level,:)>t2);
lp=length(pos);
count=0;


for p=1:lp
  cp=pos(p);
  cwm=wmm(level,cp);
  cscr=score(level,cp);
 for l=level-1:-1:1
  cw=wmm(l,:);
   while sign(cw(cp))~=sign(cwm) & cp>1,
    cp=cp-1;
   end
  if (score(l,cp)>t1 & score(l,cp)-cscr>-0.2) | score(l,cp)>=t2
%if  cscr>0.1
    flag=1;
    cwm=wmm(l,cp);
    cscr=score(l,cp);
  else
    flag=0;
    break
  end
 end
if flag==1 & cscr>0.1
  count=count+1;
  c(count,1)=cp;


  c(count,2)=cscr;
 end
end

cc=c(:,1);

Extracting end corners

len=length(cc);
x1=score(1,:);

x1(cc)=0;

for i=1:len

  if cc(i)>=9 & cc(i)<=col-9

    for j=cc(i)-1:-1:cc(i)-9

      err=abs(x1(j)-c(i,2));

    if x1(j)~=0 & err<0.2 & x1(j)>t3

      count=count+1;
c(count,1)=j;
c(count,2)=x1(j);
break;
end
```

```
   end


   for j=cc(i)+1:cc(i)+9
 err=abs(x1(j)-c(i,2));


     if x1(j)~=0 & err<0.2 & x1(j)>t3


       count=count+1;


       c(count,1)=j;


       c(count,2)=x1(j);


       break;
 end


  end



 end


end


[c(:,1),ind]=sort(c(:,1));


c(:,2)=c(ind,2);


%Further check for multiple entries


[row,col]=size(c);
```

```
indx=[];

count=0;

for i=1:row-1

  if c(i,:)==c(i+1,:)

    count=count+1;

    indx(count)=i;

  end


end

if ~isempty(indx)

  c(indx,:)=[];

end

return
```

A.8 Wavelet based Combined Evidence Thrsholding (wavelet_evidence.m)

```
%--------------------------------------------------------------------------------------------------------
% wavelet_evidence.m
% Applies the wavelet based combined evidnce thrsholding on a signal
% Returns  - None
% Author(s): Sharjeel Siddiqui, 10-Feb-2001 Copyright (c) Sharjeel Siddiqui.
%--------------------------------------------------------------------------------------------------------
clear all;
close all;

load leleccum;

%Set the variables. Type

mother_wavelet='db1';

s = leleccum(1:512);
figure(1)
plot (s);

l_s = length(s);

total_scales=4;
power_corr_result=0;
power_wavelet_coef=0;
sum_of_coeff=0;

E_lower=0.1;
E_upper=0.6;
```

```
T=[25,7,3];

corr_n=zeros(1,total_scales-1)
corr_2n=zeros(1,total_scales-1);

[C,L] = wavedec(s,total_scales,mother_wavelet);

for (i=1:total_scales)
   % put each level of detail coefficents in det_ceof vector

  det_coef=detcoef(C,L,i);

  sum_of_coeff= sum(det_coef.^2);
  power_wavelet_coef = power_wavelet_coef + sum_of_coeff;

  if (i <4)

       det_coef_next =detcoef(C,L,i+1);

       % upsample the next detailed scale so that its equal to the
       % previuos higher one . We need this to do correlation between
       % the 2 scales.

       det_coef_next_upsampled = upsample(det_coef_next,2);

      corr_result= corr2(det_coef,det_coef_next_upsampled);

      corr_n(1,i)=corr_result;

      power_corr_result = power_corr_result + corr_result^2;
```

```
    end

end

corr_2n = corr_n * sqrt (power_wavelet_coef/power_corr_result);

corr_2n_minimum = min ( abs(corr_2n) );

C_new=[];

for (i=1:total_scales-1)
   % put each level of detail coefficents in det_ceof vector

  det_coef=detcoef(C,L,i);
  clear W;
  W=[];
  for (j=1:L(1, (total_scales+1) - (i-1)) ) %% L=[ ca3 cd3 cd2 cd1 Ao]

      curr_coeff=abs(det_coef(1,j));

      if (curr_coeff > T(1,i))
         E1=0;
      elseif (curr_coeff==0)
         E1=1;
      else
         E1= ( exp( curr_coeff ) - exp(1) ) / (1-exp(1));
      end
```

```
        if ( abs( corr_2n(1,i) ) >= curr_coeff )
           E2=0;
        elseif ( abs( corr_2n(1,i) ) == corr_2n_minimum )
           E2=1;
        else
           E2 = (curr_coeff - abs( corr_2n(1,i) ) ) / (curr_coeff - corr_2n_minimum );
        end


        E= sqrt( E1*E2);



        if E >= E_upper
           F=0;
        elseif (E <= E_lower)
           F=1;
        else
           F = (E_upper-E)/ (E_upper-E_lower);


        end


        W(1,j)=F*det_coef(1,j);


    end


   C_new=[W(1,:),C_new]; %%  construct the new cofecient vector for all scales  for reconstruction at the
end



end


%% since dont go till the last scale in the above loop  therefore we add
%% the lastoriginal apporximaton and detail coeffiecnts in the
%% reconstructing vector


det_coef=detcoef(C,L,total_scales);
C_new=[det_coef,C_new];


app_coef = appcoef(C,L,mother_wavelet,total_scales);
C_new=[app_coef,C_new];


%% now reconstruct


A0 = waverec(C_new,L,mother_wavelet);


figure(2)
plot (A0);
```

A.9 Various functions for IFS based Approach

```
%------------------------------------------------------------------------------------------------
% calc_IFS_for_vel.m
% Calculate the Linear IFS with in a vel
% no_of_curves_in_vel - number of vels
% boxesw - The storage array containing all the points of the boundary
% end_point_storage_vector - horxontal points
% Returns  - The calculated value of the IFS
% Author(s): Sharjeel Siddiqui, 12-Jul-2003 Copyright (c) Sharjeel Siddiqui.
%------------------------------------------------------------------------------------------------

function IFS=calc_IFS_for_vel ( no_of_curves_in_vel, end_point_storage_vector, pts,boxesw)

%% Plot all the curves the vel and do the following
%% - Calculate D_Beta for each curve
%% - Calculate IFS (linear) for each curve
%% - Reconstruct the curves from the IFS using Random iteration Algorithm (optional-enable the code)

no_of_points_for_IFS=4; %% Change this number if you want to give more control points per curve for IFS
num_iter=1000;          %% number of iterations needed for IFS reconstruction
%% Our IFS takes two separate vectors

X=zeros(1,no_of_points_for_IFS);
F=zeros(1,no_of_points_for_IFS);

size_pts=size(pts);  %% pts is a  N x 2 matrix

size_of_end_point_storage_vector = size( end_point_storage_vector);

for i=1:(size_of_end_point_storage_vector(1,2) / 2)  %no_of_curves_in_vel
```

```
 start_point_on_original_curve = end_point_storage_vector(1,i*2-1);
 end_point_on_original_curve  = end_point_storage_vector(1,i*2);

if(start_point_on_original_curve ~= end_point_on_original_curve)
 X(1,1)=pts(start_point_on_original_curve,1);
 F(1,1)=pts(start_point_on_original_curve,2);

 X(1,no_of_points_for_IFS)=pts(end_point_on_original_curve,1);
 F(1,no_of_points_for_IFS)=pts(end_point_on_original_curve,2);

 figure;
 hold on;

       minimum_interval  =  floor  (  (end_point_on_original_curve-start_point_on_original_curve)/
no_of_points_for_IFS );
 xi=2;

 %% Plotting original curve and finding control points for IFS
 for j=1: (size_pts(1,1)-1) % we are plotting 2 adjacent points at a time

  if (j >=start_point_on_original_curve && j < end_point_on_original_curve )

    plot( [pts(j,1),pts(j+1,1)],[pts(j,2),pts(j+1,2)]);
    hold on;

    %% Make control points for IFS. For the current curve we take atleast 4 points including end
    %% points to find the IFS .

    %% type of sampling without the end points
```

```
        if ( j >= (start_point_on_original_curve + minimum_interval * xi ) && (xi < no_of_points_for_IFS ) )


           X(1,xi)=pts(j,1);

           F(1,xi)=pts(j,2);


           xi=xi+1;

        end


      end


    end


    d=0.5;

    %% calling our IFS calculation function and pasing given DB


    if (start_point_on_original_curve ~= end_point_on_original_curve)

       V=IFSCalc(d,X,F);


 %reconstructed_curve=Random_ifs_reconstruction(V,num_iter,[boxesw(1,2),boxesw(1,4),boxesw(1,7),bo
 xesw(1,3)],1); % axis=[boxesw(1,2),boxesw(1,4),boxesw(1,3),boxesw(1,5)]

                  Random_ifs(V,num_iter)     %,[boxesw(1,2),boxesw(1,4),boxesw(1,7),boxesw(1,3)],1);  %
 axis=[boxesw(1,2),boxesw(1,4),boxesw(1,3),boxesw(1,5)]

   end


 end  % if (start_point_on_original_curve ~= end_point_on_original_curve


end
```

```
%-----------------------------------------------------------------------------------------------
% IFSCalc.m
% calculation of IFS according to passed control points X,F, dimension d
% and number of iterations num_iter
% Returns  - The calculated value of the IFS
% Author(s): Sharjeel Siddiqui, 12-Jul-2003 Copyright (c) Sharjeel Siddiqui.
%-----------------------------------------------------------------------------------------------


function V=IFSCalc(d,X,F)
%d=1.2;


[X,order]=sort(X);


F=F(order);


b=max(F)+2;
a=min(F)-2;



N=size(X);


for n=2:N(1,2)
  A(1,n-1)=(X(1,n)-X(1,n-1))/( X(1,N(1,2))-X(1,1) );


  D(1,n-1)=( ( X(1,N(1,2)) * X(1,n-1) )- ( X(1,1)* X(1,N(1,2) ) ) )/( X(1,N(1,2))-X(1,1) );


  S_max(1,n-1)=min( [( b-F(1,n) ) / ( b-F(1,N(1,2) ) ),( b-F(1,n-1) )/( b- F(1,1) ),( a-F(1,n) )/( a-F( 1,N(1,2)
) ),( a-F(1,n-1) )/( a-F(1,1) )] );


  S_min(1,n-1)=max( [( a-F(1,n-1) )/ ( b-F(1,1) ),(a-F(1,n))/(b-F(1,N(1,2) ) ),( b-F(1,n-1))/( a-F(1,1) ),(b-
F(1,n))/(a-F(N(1,2) ) ) ] );
```

```
  end



  N=size(A);
  sum=0;
  found_flag=0;

 count_iter=0;
 while (found_flag==0)
   count_iter=count_iter+1;
   alpha=rand(1);
   for n=1:N(1,2)-1
     S(1,n)= alpha*min(S_max(1,n),1) +(1-alpha)*max(S_min(1,n),-1); %
     sum=sum + (  (A(1,n)^(d-1))*S(1,n) );
   end

     S(1,N(1,2))=(1-sum)/(A(1,N(1,2) )^(d-1));
     if  ( S(1,N(1,2)) >= max(S_min(1,N(1,2)),-1) && S(1,N(1,2)) <= min(S_max(1,N(1,2)),1) )
       found_flag=1;
     else
       found_flag=0;

     end
 if (count_iter>5000)
   count_iter
   break;
 end
end

N=size(X);
```

```
for n=2:N(1,2)

   C(1,n-1)= ( (F(1,n)-F(1,n-1) )/( X(1,N(1,2))-X(1,1) ) ) - S(1,n-1)* ( (F(1,N(1,2))-F(1,1))/(( X(1,N(1,2))-
X(1,1) )) );

  E(1,n-1)=( ( X(1,N(1,2)) * F(1,n-1) )- ( X(1,1)* F(1,n) ) )/( X(1,N(1,2))-X(1,1) ) - S(1,n-1)*( ( X(1,N(1,2))
* F(1,1) )- ( X(1,1)* F(1,N(1,2)) ) )/( X(1,N(1,2))-X(1,1) );

end

  V(:,1)=A(1,:)';
  V(:,2)=0;
  V(:,3)=C(1,:)';
  V(:,4)=S(1,:)';
  V(:,5)=D(1,:)';
  V(:,6)=E(1,:)';
  V
```

```
%-----------------------------------------------------------------------------------------------
% Random_ifs.m
% Calculates the attractor of the given IFS using RIA
% Returns  - None
% Author(s): Sharjeel Siddiqui, 12-Jul-2003 Copyright (c) Sharjeel Siddiqui.
%-----------------------------------------------------------------------------------------------


%  system in the matrix M.  K data points are plotted in the image.
%   M represents n 2x2 linear functions of the form f(x) = Ax+b.
%   M is an nx6 matrix containing the data for one function per row.
%   A is the optional axis set [xmin,xmax,ymin,ymax], for fractals
%   with larger transformations.  Defaults to [-0.1,1.1,-0.1,1.1].
%   R (optional, default 0) can be used to reuse the same plot as
%   before, where the flag 1 = reuse same plot, 0 = start new plot.

function Random_ifs(M,K,A,R)

%first do some elementary error checking
%check arguments
error(nargchk(2,4,nargin))
%check matricies
figure(1);
a = size(M);
if a(2) < 6
  error('M must be atleast nx6 matrix.');
elseif (a(2)==6)
  l = size(K);
  if l(1) > 1 | l(2) > 1 | K(1) <= 20
    error('K must be 1x1 vector > 20.')
  end
  reuse = 0;
  if nargin >= 3
    q = size(A);
    if q(1) ~= 1 | q(2) ~= 4
      error('A must be a 1x4 axis specification, [xmin,xmax,ymin,ymax].');
    end
  end
  if nargin == 4
    if R ~= 0 & R ~= 1
      error('R must be either a 1 (reuse plot) or 0 (start new plot).')
    end
    reuse = R;
  end

  % make the prob matrix
  P = [];
  for i = 1 : a(1)
    t = abs(det( [ M(i,1:2); M(i,3:4) ] ));
    if t == 0
      t = .001;
    elseif t > 1
      error(strcat('Matrix M contains a non-contraction function in row:  ',int2str(i)),'.')
    end
    P(1,i) = t;
  end

% fix for to big or small elements because an IFS satisfies the contraction
% theorem, sum(P) < 1
s = sum(P);
if s < 1
  % add difference, moved over all elements
```

```
    P = P + ((1-sum(P))/length(P)) * ones(1,length(P));
  end
  %start graph
  if reuse == 0
    hold off
  end


  x=[0;0];
  p = plot(x(1),x(2),'.','EraseMode','none','MarkerSize',4,'Color','blue');
  if nargin >= 3
    axis(A);
  else
    axis([-0.1,1.1,-0.1,1.1]);
  end
  hold on


  % The iteration - skip first 20 points to not have any wierd data
  for i=1:20
    r = randp( P );
    y = [ M(r,1:2); M(r,3:4) ] * x + [ M(r,5:6)' ];
    x=y;
  end
  hold on;
  axis off
  for i=21:K
    r = randp( P );

    newx=M(r,1)*x + M(r,2)*y +M(r,5);
    newy=M(r,3)*x + M(r,4)*y +M(r,6);
    x=newx;
    y=newy;
    %plot(x,y); hold on
    set(p,'XData',x,'YData',y);
    drawnow;
  end

elseif (a(2)==12)
  figure(2)
   l = size(K);
  if l(1) > 1 | l(2) > 1 | K(1) <= 20
    error('K must be 1x1 vector > 20.')
  end
  reuse = 0;
  if nargin >= 3
    q = size(A);
    if q(1) ~= 1 | q(2) ~= 4
      error('A must be a 1x4 axis specification, [xmin,xmax,ymin,ymax].');
    end
  end
  if nargin == 4
    if R ~= 0 & R ~= 1
      error('R must be either a 1 (reuse plot) or 0 (start new plot).')
    end
    reuse = R;
  end

  % make the prob matrix
  P = [];
  for i = 1 : a(1)
    t = abs(det( [ M(i,1:3); M(i,4:6); M(i,7:9) ] ));
    if t == 0
```

```
      t = .001;
    elseif t > 1
      error(strcat('Matrix M contains a non-contraction function in row:  ',int2str(i)),'.')
    end
    P(1,i) = t;
end

% fix for to big or small elements because an IFS satisfies the contraction
% theorem, sum(P) < 1
s = sum(P);
if s < 1
    % add difference, moved over all elements
    P = P + ((1-sum(P))/length(P)) * ones(1,length(P));
end
%start graph
if reuse == 0
    hold off
end

x=[0;0];
%p = plot(x(1),x(2),'.','EraseMode','none','MarkerSize',4,'Color','red');
if nargin >= 3
%   axis(A);
else
 %  axis([-0.1,1.1,-0.1,1.1]);
end
hold on
x=0;
y=0;
t=0;
% The iteration - skip first 20 points to not have any wierd data
```

```
  for i=1:20
    r = randp( P );
    %y = [ M(r,1:3); M(r,4:6);M(r,7:9) ] * x + [ M(r,10:12)' ];
    newt=M(r,1)*t+ M(r,2)*x + M(r,3)*y +M(r,10);
    newx=M(r,4)*t+ M(r,5)*x + M(r,6)*y +M(r,11);
    newy=M(r,7)*t+ M(r,8)*x + M(r,9)*y +M(r,12);
    t=newt;
    x=newx;
    y=newy;

    x=y;
  end
  hold on;

  for i=21:K
    r = randp( P );

    newt=M(r,1)*t+ M(r,2)*x + M(r,3)*y +M(r,10);
    newx=M(r,4)*t+ M(r,5)*x + M(r,6)*y +M(r,11);
    newy=M(r,7)*t+ M(r,8)*x + M(r,9)*y +M(r,12);
    t=newt;
    x=newx;
    y=newy;

    plot(x,y); hold on
  end
  drawnow;
end
```