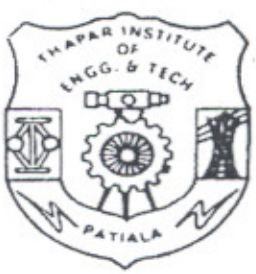
A Cluster Analysis based on Self-Organizing Maps in Fractal Image Compression

A thesis
submitted in partial fulfillment of the requirements
for the award of degree
of

Master of Engineering
in
Software Engineering



Under the Supervision of
Mr. Maninder Singh
Assistant Professor
Computer Science & Engineering Department
Thapar Institute of Engineering & Technology, Patiala.

Submitted By
Lucky Jindal
8023107

Computer Science & Engineering Department
Thapar Institute of Engineering & Technology
(Deemed University)
Patiala-147004
(India)

May 2004

Candidate's Declaration

I hereby certify that the thesis entitled "**A Cluster analysis based on Self-Organizing Maps in Fractal Image Compression**" submitted in the partial fulfillment of the requirement for the award of degree of **Master of Engineering** in **Software Engineering** at Thapar Institute of Engineering & Technology (Deemed University), Patiala, is an authentic work carried out under the supervision and guidance of Mr. Maninder Singh.

The results contained in this thesis have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Lucky Jindal.

Lucky Jindal
Roll No. 8023107

Mr. Maninder Singh
Supervisor
Assistant Professor
Computer Science & Engineering Department
TIET, PATIALA - 147004

This is to certify that the above statement made by the candidate is true and correct to the best of my knowledge and belief.

Countersigned by

Seema Bawa

Ms. Seema Bawa
Assistant Professor & Head
Computer Science & Engineering Department
TIET
PATIALA – 147004

Dr. D. S. Bawa
Dean
Academic Affairs
TIET
PATIALA – 147004

Acknowledgment

This thesis could not have been completed without the support of a number of important people. First and foremost my thanks go to my research guide, **Mr. Maninder Singh**, Assistant Professor, Computer Science & Engineering Department, for always having an open mindset for any problem throughout my research work. His advice over the last six months has been invaluable. He was very helpful in providing the resources necessary for this research work.

I am also thankful to **Ms. Seema Bawa**, Head, Computer Science & Engineering Department, for the motivation and inspiration that triggered me for the research work.

I would also like to thank all the staff members and my colleagues who were always there at the need of the hour and provided with all the help and facilities, which I required, for the completion of my research.

I am indebted to **Ms. Kirandeep Kaur** for her highly valuable comments. She has been very helpful in helping me to decide upon this subject and providing me with ideas on where to find information helpful to the project. Her ideas pervade my thesis.

Thanks go to my parents, for their love and guidance over the last 25 years. They have given me a great head start on life and I am indebted to my comrade who always encouraged me in my high times.

Finally, my thanks go to God, my Creator and Sustainer.

Lucky Jindal
01/8/2K4.

Lucky Jindal

M.E. (Software Engg.)

8023107

Abstract

Fractal image compression is a novel and attractive technique for still image compression. The most significant advantages are high reconstruction quality at low coding rates, rapid decoding, and resolution independence. One of the major limitations is its high encoding time complexity. Many techniques have therefore been developed and are being developed to overcome this limitation.

Different classification and clustering algorithms exist in the literature to classify the domain block pool accordingly so that the search space is reduced for each range block, which can reduce the encoding time complexity to a large extent.

An algorithm based on Kohonen's Self-organizing maps is proposed in this thesis for the cluster analysis of image data. This algorithm, having its roots in neural networks is different from the classical clustering algorithms and based on neighborhood relationships.

Kohonen's SOM algorithm is useful as it rapidly generates high quality clustering and gives good results when using small number of training vectors with an arbitrary initial codebook. This scheme may be combined with other classical algorithms to overcome the limitations of Kohonen's algorithm.

This thesis explains how Self-Organizing Maps clusters the image data and how this concept is used for reducing the encoding time complexity of fractal image compression.

Implementation has been carried out using Matlab6.0 from MathWorks Inc with SOM Toolbox. Experimental results obtained are presented and demonstrate the effectiveness of the proposed algorithm.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
BMU	Best Matching Unit
DCT	Discrete Cosine Transform
DB	Domain Block
ED	Euclidean Distance
FIC	Fractal Image Coding
FID	Fractal Image Decoding
IFS	Iterated Function System
JPEG	Joint Photographic Experts Group
MSE	Mean Square Error
PIFS	Partitioned Iterated Function System
PSNR	Peak Signal to Noise Ratio
RB	Range Block
RMSE	Root Mean Square Error
SOM	Self Organizing Maps
SOFM	Self Organizing Feature Maps
VQ	Vector Quantization

CHAPTER 1

Introduction

1.1 BACKGROUND AND MOTIVATIONS

Digital image processing [1,3], the manipulation of images by computer, is a relatively recent development in terms of humans' ancient fascination with visual stimuli. Public awareness of digital image processing has been greatly increased by video games and digital video special effects used in the entertainment industry. The usage of digital image processing in commercial, industrial, and medical applications and in scientific research continues to grow. One can expect digital image processing to play an important role in the future.

Files that are on the Internet are actually stored on computers all over the world. To get the files to your computer they must be transmitted across a network (Web) of phone lines and other cables. The larger the file size, the longer it takes to transfer the data. For this reason web sites try to use files that are small in size. These smaller files take up less space (storage) on the storage devices where they are stored, and take less time to be transported to your computer. Data compression techniques exploit inherent redundancy and irrelevancy by transforming a data file into a smaller file from which the original image file can later be reconstructed, exactly or approximately. Some data compression algorithms are lossless, while others are not. These coding schemes include Huffman Coding and Run Length Encoding. A lossless algorithm eliminates only redundant information, so that one can recover the image exactly upon decompression of the file. A lossy compression algorithm eliminates irrelevant information as well, and thus permits

only an approximate reconstruction of the original, rather than can exact duplicate. As one might expect, lossy compression algorithms achieve higher compression ratios [2].

Digital representations of images usually require a very large number of bits. In many applications, it is important to consider techniques for representing an image, or the information contained in the image, with fewer bits. Since digital images, by their nature, are quite data intensive, reducing their size can produce solutions. The objective in image compression is to efficiently produce the smallest graphics files without compromising image quality and the motivation if image compression is to reduce the bit-rate for storage and transmission bandwidth. Many image compression techniques have been applied in several areas of image and video processing such as TV transmission, video conferencing and portable video telecommunication.

By eliminating redundant or unnecessary information, lossy image compression is the activity that addresses this aim. These algorithms include Fourier Transform, Cosine Transform, JPEG, and Fractal Image Compression. Image quality is traded off for higher compression ratios. Some are schemes in which the image data is encoded for storage and decoded for display. These coding schemes include GIF. No data is lost in these schemes.

Standard methods of image compression come in several varieties. Currently, the most popular method relies on eliminating high-frequency components of the signal by storing only the low-frequency Fourier coefficients. This method uses a discrete cosine transform (DCT), and is the basis of the so-called JPEG standard. This standard, defined by the Joint Photographic Experts Group, describes ways of taking bit-mapped data for color or gray-scale continuous-tone images and storing it in a smaller number of bytes. Another method, called vector quantization, breaks up images into a small number of canonical pieces and storing only a reference to which piece goes where [4].

DCT-based JPEG compression is quite effective at low or moderate compression ratios, up to ratios of 20 or 25 to 1.Beyond this point, the image becomes very blocky, as the

compression increases and the image quality becomes too poor for practical use JPEG obtains high compression ratios by cutting off the high frequency components of the image. This can also introduce very visible artifacts, in particular for sharp edges in the image. This is known as Gibb's phenomenon [1].

Another drawback of JPEG compression is its resolution dependence. In order to zoom in on a portion of an image and to enlarge it, it is necessary to replicate pixels. The enlarged image will exhibit a certain level of blockiness, which soon becomes unacceptable as the expansion factor increases. Because of this problem, it is sometimes necessary to store the same image at different resolutions, thus wasting storage space. Although JPEG is now a well-established standard for lossy image compression, it has its limits.

A more recent approach comparable to DCT based compression is fractal image compression technique, which overcomes many of JPEG problems.

Decompression speed, resolution independence and high compression ratios distinguish fractal image compression from JPEG/DCT. Many application developers may find fractal image compression preferable for multimedia applications where quick access to high quality images is essential. Microsoft, for example, currently uses fractal image compression in its Encarta multimedia encyclopedia.

Thus, fractal image compression is a lossy method or class of methods that allows images to be stored on computers in much less memory than standard ways of storing images. Fractal Image compression is proposed by Michael Barnsley and Alan Sloan in 1988, based on their work at the Georgia Institute of Technology. Since then it has been the subject of much research, and several variations on the original algorithm have been proposed (though few have been implemented)[4].

Despite all of its advantages, fractal image coding is not considered a primary compression scheme due to a long encoding time as opposed to a short decoding time. Moreover, during image compression the encoding time depends on the compression

ratio we want to achieve, while during decompression the decoding time is not affected by this value. Even highly optimized fractal image coding algorithms are unable to run in real-time on computers today for any reasonably sized image. Many researches are going in this context to reduce the encoding time to make the fractal image compression practically applicable to the areas where it delivers performance as compared to other techniques.

It is of interest to consider the techniques to reduce the computational complexity of encoding process without comprising the image quality. This research work investigates self-organizing maps, a clustering technique as a step to achieve this goal.

There are a number of problems to be solved in fractal image compression to make the process viable and more efficient. They are [4]:

i) Complexity reduction: How can the process of finding a good fractal code be speeded up?
ii) Coding efficiency: How can the parameters of a fractal code be coded in the most efficient way? How to select the parameters in a rate distortion optimal fashion?
iii) Partitioning Methods: Does the flexibility in partitioning scheme lead to improved results in the rate-distortion sense?
iv) Class of transforms: Is there any gain in using wider class of transforms than just block based operators?
v) Decoding complexity: How can the decoding process be speeded up?

In this thesis efforts are made towards solving the problem i). Here speed up of the fractal image compression is addressed with the use of a classification technique.

1.2 PURPOSE OF THESIS

This research work is particularly targeted towards fractal image compression with an idea to minimize the computational requirements to achieve good and desirable reproduction image quality. In this direction some new efficient and effective algorithms are developed.

The first part of this thesis focuses on fractals and the image compression using fractal structures and on the study of different such techniques that can help in speeding up the encoding process. There are many methods and combination of these can be used for fractal image coding. We propose a clustering technique using self-organizing maps to implement fractal image coding. Clustering is an efficient scheme for the implementation of fractal image coding because it gives better compression at the little cost of image quality.

The second part of the thesis deals with the design and development of a clustering technique using self-organizing maps, which has its root in neural networks. This leads to the improvement of encoding time complexity of fractal image compression with a slight degradation in image fidelity.

1.3 ORGANIZATION OF THESIS

This thesis is organized as follows.

Chapter **1** explains some background information, motivations and purposes of this research work. It gives the thesis outline. Chapter **2** first presents an overview of fractals and fractal image compression. Next, it presents the advantages and applications of fractal image compression, and the problems that are encountered in implementing the

fractal image compression. Various methods that can be used for making fractal image fast are also discussed in this section.

The role of classification and clustering schemes in the speeding up of fractal image compression process is presented and how classification improves the performance of fractal image compression. Different classification schemes and clustering algorithms are also mentioned. Then we propose of using one of the most efficient clustering algorithms, self-organizing maps, whose roots are in neural networks in the Chapter **4**. Necessary notations are introduced in this chapter. Advantages and limitations over the classical clustering algorithms are also discussed.

Implementation details and the experimental results that support the validity of the proposed cluster analysis technique done with self-organizing maps in fractal image compression are presented in Chapter **5**. To conclude this thesis we summarize our obtained results in terms of theoretical aspects as well as practical experiences. We balance the advantages and disadvantages of our proposed approach to the existing methods. Finally, the concluding remarks and the contributions of this thesis to the area of fractal image compression are presented in Chapter **6**. Suggestions for future research work are also introduced. Appendix **A** contains reproductions of the set of test images used in the research.

CHAPTER 2

Fractal Image Compression: A Review

2.1 INTRODUCTION

This chapter explains the fractal image compression and reviews the work done by various researchers in the area of fractal image compression. In this chapter, we present the fractal image compression as it is usually presented without any reference to other conventional methods. Nevertheless, fractal methods do have a relation with more conventional compression schemes and will become more and more apparent during the development of this chapter.

2.1.1 Brief History

The term 'fractal' was first used by an IBM mathematician, Benoit Mandelbrot in 1977 to designate objects that are self-similar at different scales. Such objects have details at every scale. A well known example is Mandlebrot set, which is described by a very simple equation yet exhibits an infinite variety of detail. This can be viewed as an extreme form of compression: the equation itself can be described with a few bits of information.

Mandelbrot actually consider fractals for compression, but they could be used for modeling real objects such as clouds, trees or mountains. The images generated by fractal modeling were very realistic and these techniques are now commonly used in many applications using computer-generated images. In 1981 J.J Hutchison gave a revised version of B. Mandelbrot's book 'The Fractal Geometry of Nature' [4,5].

Michael Barnsley, a leading researcher and his co-workers at Georgia institute of technology were the first to recognize the potential interest of fractal methods for image compression. Barnsley developed the theory of iterated function systems first introduced by J.J. Hutchinson in 1981. The first article suggesting fractal image compression was in BYTE magazine in 1988.After the publication, fractal compression became a very fashionable subject. The interest in this area was aroused by the fantastic compression ratios claimed by Barnsley founded up to 10000 to 1. Together with Alan Sloan, Barnsley founded Iterated systems, inc., in 1987, devoted to applications of Iterated function systems, especially fractal image compression. Rather than storing an image pixel by pixel, the goal of fractal image compression is to find a lossy compression algorithm that takes advantage of the self-similarities in an image. Barnsley applied his knowledge of fractals and mathematics to image compression, creating an optimal forms of image compression comparable to JPEG. In 1988,Barnsley his first book 'Fractals everywhere'.

In 1989, A. Jacquin, a doctorate student of Barnsley realized a first automated block based fractal encoding system which can compress digital monochrome images although the algorithm was not sophisticated. Now all the contemporary fractal-coding programs are based on the Jacquin's paper.

In 1992, Y. Fisher, a visiting professor at University of California, San Diego implemented the practical schemes of fractal image coding based on Jacquin's approach. He made many improvements on and alternatives to Barnsley's algorithms.

After this, came along a long list of variations, generalizations and improvements. The first public scheme on fractal image compression was due to E.W. Jacobs and R.D Boss of the Naval Ocean Systems Center in San Diego who used regular partitioning and classification of curve segments in order to compress measured fractal curves. From then on, this lossy compression scheme is the interest of research for many researchers.

2.2 BASIC PRINCIPLES

Fractal Image Compression approach involves the use of fractals to represent the structure of images. Deterministic fractals have the intrinsic property of having extremely high visual complexity, which requires less number of bits for representing given information. They can be described and generated by simple recursive deterministic algorithms or transformations.

Thus, the fundamental principle of fractal image coding is to represent the image by the parameters of the transform under which the image is approximately invariant. This transformation is constructed so that it is contractive.

2.3 WHAT ARE FRACTALS?

Fractals are self-similar geometrical objects. Several parts of a fractal look similar as the entire image but this self- similarity can appear at different scales. Then it is possible to copy the fractal several times on itself. Each part appears to be a reduced copy of the whole image. Fractals are independent of scale.

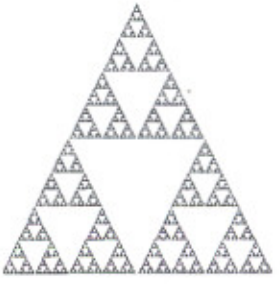


Figure 2.1: Sierpinsky Triangle

A nice example is the Sierpinsky Triangle.

There is no exact definition for the term fractal, but it has certain properties that we can describe:

- ***It is self similar***, something pretty obvious if we look at the Sierpinski Triangle: the triangle contains itself over and over again.
- ***It has detail in every scale***, it's not quite as obvious as self similarity but if we imagine to iterate the copy process infinitely many times, we could zoom in the triangle at an arbitrary level and still encounter the triangle and never reach a level where we could see the picture of Lena again.
- ***There exists a simple algorithmic description***, As we have already seen, the Sierpinski Triangle can be generated with three very simple transformations.

Mandelbrot and Julia sets are most common examples of fractals. These sets are the results of repeatedly iterating a simple mathematical function at different locations (different initial values) until it either converges on a single value, or is found to be unbounded.

2.4 FRACTAL IMAGE COMPRESSION

2.4.1 Description of method

Fractal Image Compression is a simple scheme that allows the encoding of images as 'fractals' and we can generate a complex looking fractal from a small amount of information.

Imagine a special type of photocopying machine that reduces the image to be copied by half and reproduces it three times on the copy (see Figure 2.2 below).

Lets us now consider what happens when we feed the output of this machine back as input. Figure shows several iterations of this process on several input images. We can

observe that all the copies seem to converge to the same final image. We call this image the attractor for copying machine. Since the copying machine reduces the input image, any initial image placed on the copying machine will be reduced to a point as we repeatedly run the machine; in fact, it is only the position and the orientation of the copies that determines what the final image looks like [8].

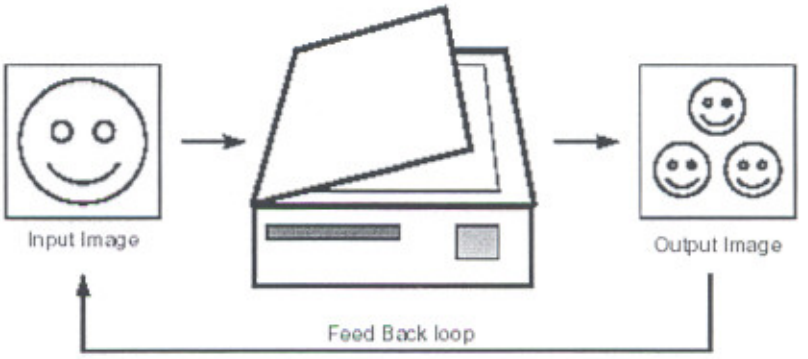


Figure 2.2: Copy Machine that makes 3 reduced copies of the input image

The way the input image is transformed determines the final result when running the copy machine in a feedback loop. However we must constrain these transformations, with the limitation that the transformations must be contractive, that is, a given transformation applied to any two points in the input image must bring them closer in the copy. If points in the copy were spread out the final image would have to be of infinite size. Except for this condition the transformation can have any form.

In practice, choosing transformations of the form

$$w_i \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix}$$

Is sufficient to generate a rich and interesting set of attractors. Such transformations are called affine transformations of the plane. Each can skew, stretch, rotate, scale and translate an input image [6].

A common feature of these transformations that run in a loop back mode is that for a given initial image each image is formed from a transformed (and reduced) copies of itself, and of itself, and hence it must have detail at every scale. That is, the images are fractals. This method of generating fractals is due to John Hutchinson.

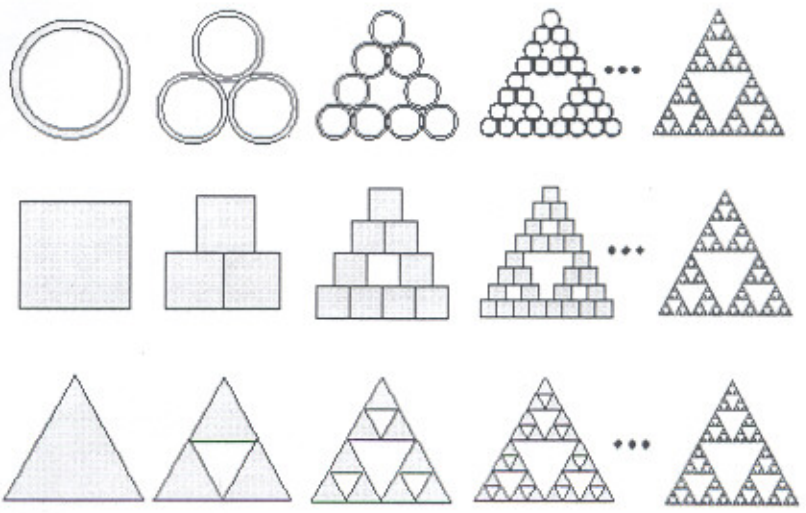


Figure 2.3: Sierpinksi triangles for different initial images

Figure 2.3 (c) Sierpinski Triangle shows several iterations of this process on several input image. What we observe, and what is in fact true, is that all the copies seem to be converging to the same original image. Because the copy process reduces the input image, the copies of any initial image will be reduced to a point as we repeatedly feed the output back as input; there will be more and more copies, but each copy gets smaller and smaller. So the initial image doesn't affect the output image. Infact, it is only the position and orientation of the copies that determines what the final image will look like.

Barnsley suggested that storing the images as collections of transformations could lead to the image compression.

Each transformation $\mathbf{w}_i$ is defined by 6 numbers, which do not require much memory to store on a computer (4 transformations x 6 numbers transformations x 32 bits /number = 768 bits). So if we wish to store a picture of a fern, then we can do it by storing the

numbers that define the affine transformations and simply generate the fern whenever we want to see it. The trick is to find those numbers.

2.4.2 Self -similarity and Self-affinity in images

Fractal image compression [9] is based on the idea that any real-world world image contains affine redundancy, or self-similarity, and that this redundancy can be exploited to compress images.

The property of objects whereby magnified subsets appear similar or identical to the whole and to each other is known as self-similarity. These self –similar portions or called fractal shapes are independent of scale and possess no characteristic size. The non-uniform scaling, where shapes are (statistically) invariant above transformations that scale different co-ordinates by different amounts, is known as self-affinity.

Barnsley used ferns, clouds, and other natural phenomena with obvious repeating motifs as examples. Computer generated fractal compression relies on far more abstract affine redundancies. Small portions of an image are used as the basis of relationships which are difficult, if not impossible, to spot with the human eye. Figure 4 shows some examples of affine redundancy. The white-bordered section is similar to the black-bordered sections, but is exactly twice as large. The entire image is not self-similar, but parts of the image are self-similar with properly transformed parts of itself. It is this restricted redundancy that fractal image compression schemes attempt to eliminate [4,7].

2.4.3 Why it is Fractal Image Compression?

Standard Image compression methods can be evaluated using their compression ratio; the ratio of the memory required to store an image as a collection of pixels and the memory required to store a representation of the image in compressed e.g. the Fern could be generated from 768 bits of data but required 65,536 bits to store as a collection of pixels,

giving a compression ratio of 65,536/768=85.3 to 1. The Compression ratio for the fractal scheme is hard to measure, since the image can be decoded at any scale [4,7].

Figure 2.4: Examples of affine redundancy

2.4.4 Statistical Justification

The most intriguing aspect of fractal compression is its reliance on the presence of self-similarity in natural scenes. To circumvent this highly restrictive assumption we will attempt, in this section, to find a statistical justification of fractal image compression. Fractal compression is often seen as a type of vector quantization. It has, however, a few differences:

- The codebook (all the domain blocks) is contained in the image itself and does not require separate storage.
- The contrast and brightness of the codebook vectors are adjustable so one codebook vector can match a wider class of range blocks.

The reason why domain blocks form a good codebook is related to the invariance properties described in the previous chapter. For a codebook to be efficient, the vectors used to construct it must have the same statistical distribution as the vectors to be coded. We saw that the statistics of an image are invariant under translation, 90-degree rotations, mirror symmetry and scaling. These operations are exactly those involved when matching domain and range blocks. The brightness and contrast adjustments simply perform the appropriate normalization needed when an image is rescaled. Hence, a transformed domain block:

$$s_i G(T_i^{-1}\mathbf{r}) + o_i \text{ for } \mathbf{r} \in R_i$$

extracted from random function G and the corresponding range block,

$$G(\mathbf{r}) \text{ for } \mathbf{r} \in R_i ,$$

are identically distributed. Having two identically distributed random vectors does not imply that these two random vectors will actually have a high probability of taking a similar value at the same time. This matching probability is directly related to how uniform the probability distribution is, as we shall now see. (In the following analysis, we neglect the fact that we have to reject non-contractive mappings. Taking this into account does not change the essence of the development.) Since some error is acceptable when matching blocks, we can partition the set of all possible image blocks in small subsets whose size is related to the matching tolerance. We number those regions from 1 to N and consider that domain and range blocks can be modeled by two discrete random variables V_d and V_r. Both V_d and V_r have a probability p_i of taking value i ε {1,....., N}. The probability **m** of having a match is then the probability that V_d and V_r take the same value at the same time [4]:

$$m = \sum_{i=1}^{N} p_i^2$$

2.5 FRACTAL IMAGE ENCODING

In encoding process, the self-similarities and affine redundancies existing in the image are searched for by partitioning the whole image. These redundancies are represented in the mathematical form or transformations and sent over the channel.

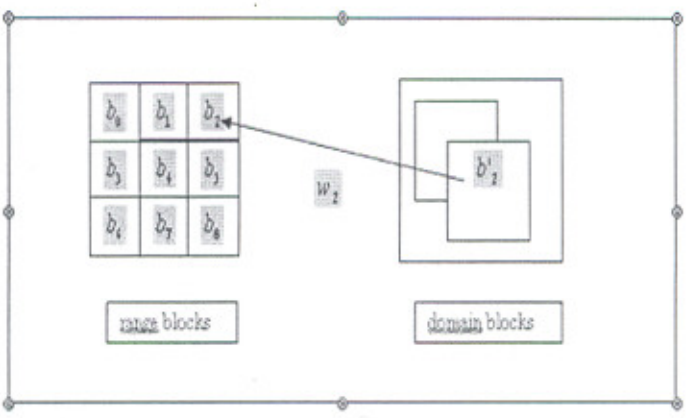


Figure 2.5: Illustration of encoding process

The major steps in encoding of an image in fractal image compression consists of [7]:

(i) Partitioning the given image into non-overlapping squares R_i (range blocks) of size B × B and larger over-lapping square blocks D_i (domain blocks) of size 2B × 2B.This set of blocks form a domain block pool.

(ii) Search for an appropriate domain block in the domain block pool is done for each range block. But, before that, following steps need to be done [8]:

- Domain block pool can be generated by sliding 2B × 2B window within the original image by skipping δ pixels from left to right, top to bottom (Figure 2.5). The window is first located with its bottom left corner at (0,0). It then moves from one position to nest by steps of δ , horizontally to the right or vertically upward in such a way that it remains entirely inside the image support at all times. The steps can be typically chosen to be B or B/2.

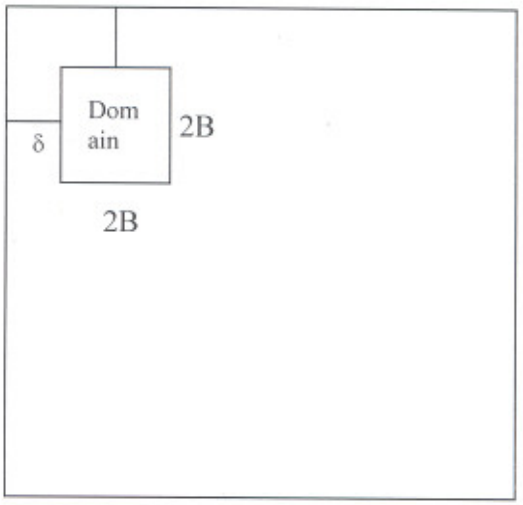


Figure 2.6: The domain block pool

- Shrink every domain block to the size that of range block by averaging the 2x2 neighboring pixel values (Spatial Contraction).

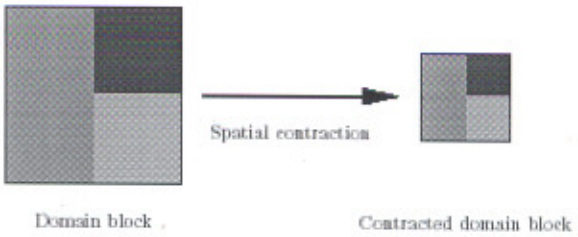


Figure 2.7: Spatial contraction of Domain block

- Application of 8 isometries (rotations and reflections) on every shrunken domain block. This forms the domain block pool.

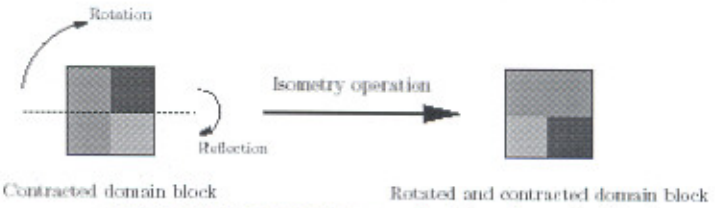


Figure 2.8: An isometry applied to a domain block.

(iii) Find an affine transformation that adjusts the intensity values in the selected domain block to those in the range block. Here, an affine transform is applied on the pixel intensities.

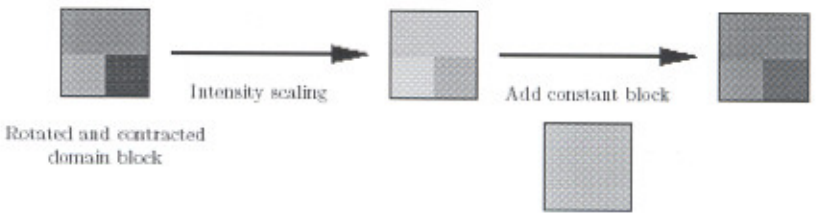


Figure 2.9: An affine transform applied to a domain block

The mapping for (i, j) th range block, $t_{i,j}$ consists of a scaling factor $\mathbf{s_{ij}}$, offset $o_{i,j}$, isometric transformation $\mathbf{T_f}$, $0 \leq f \leq 8$, and Spatial Contraction **S**. An isometric transformation maps a square block to one of the eight isometries obtained from the compositions of reflections and rotations as shown in Table 2.1. The result of applying this mapping to a domain block **D** is an approximation to (i, j) th range block **R**. For isometric transformation T_f : R-->R and spatial contraction function s: D→R as follows (Figure2.9 and 2.10).

$$\mathbf{R} \simeq \mathbf{s_{i,j}} * \mathbf{T_f}\ (\mathbf{S}\ (\mathbf{D})) + \mathbf{o_{i,j}} \qquad \text{Eq (2.1)}$$

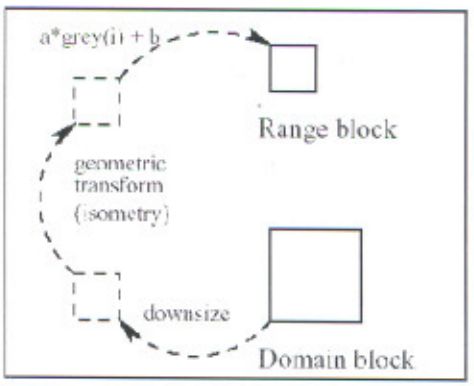


Figure 2.10: Mapping a DB to a RB

Thus, if the image is partitioned into range blocks, then the system can be represented by a set of contractive transformations.

NOTE: If the image is $M \times M$, then there are $\left(\frac{M-2B}{\delta}+1\right)*\left(\frac{M-2B}{\delta}+1\right)$ domain blocks.

We try to find the matrix of the form.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix}$$

The resulting transformation called affine transformation is a combination of a geometrical and massic transformation.

$\mathbf{w_i = G_i \text{ o } M_i}$ where $\mathbf{G_i}$ geometric part, $\mathbf{M_i}$ massic part

GEOMETRIC PART G_i:

The geometric (a, b, c, d, e, f) part of the transformation corresponds to mapping in position and size from the domain cell $\mathbf{D_i}$ to the range cell $\mathbf{R_i}$. In the implementation, we will consider

Sizeof (D_i) = 2 * sizeof (R_i)

So, the pixel values of the range block are the average values of four pixels in the domain block. The geometric part scaled down the domain block in size from 2B×2B to B×B (downsizing) .See Figure 2.11.

- Moves domain block.
- Changes the size of domain block.

MASSIC PART M_i:

The massic (s, o) part of the transformation consists in a modification of the pixel values inside the block. It allows changing the gray level information in order to get good approximation of the block R_i, thus affects the gray levels of the domain blocks with a contrast scaling **a** and the luminance shift **b**. See Figure 2.11.

- Adjusts the intensity and orientation of pixels.

The coefficients a_i, b_i, c_i, d_i, e_i, f_i, s_i and o_i are all transformation parameters and they constitute the fractal code. a_i, b_i, c_i, d_i correspond to one of the eight isometrics, s_i represents a contrast scaling, o_i represents a brightness shift [4,7,8].

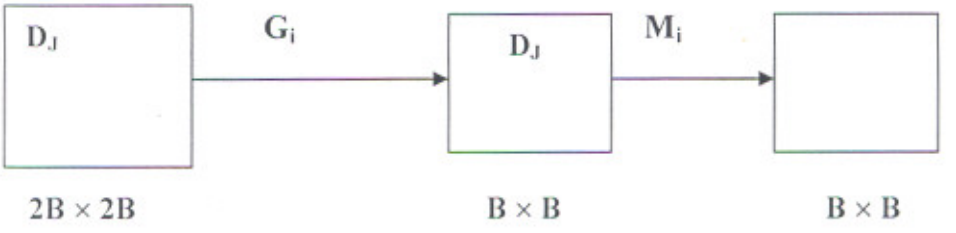


Figure 2.11: Affine transformation G_i o M_i

Table 2.1: Eight Isometric Transformations

f	Function
1	Identity
2	Rotation through +90°
3	Rotation through +180°
4	Rotation through +270°
5	Reflection about mid-vertical axis
6	Reflection and Rotation—90°
7	Reflection and Rotation—180°
8	Reflection and Rotation—270°

Fractal image encoding process consists of determining, for all the range blocks, the mapping parameters in above equation 2.1 such that the distortion between each range

block and its approximation is minimum. In practice, peak signal to noise ratio is used to measure the distance between two images.

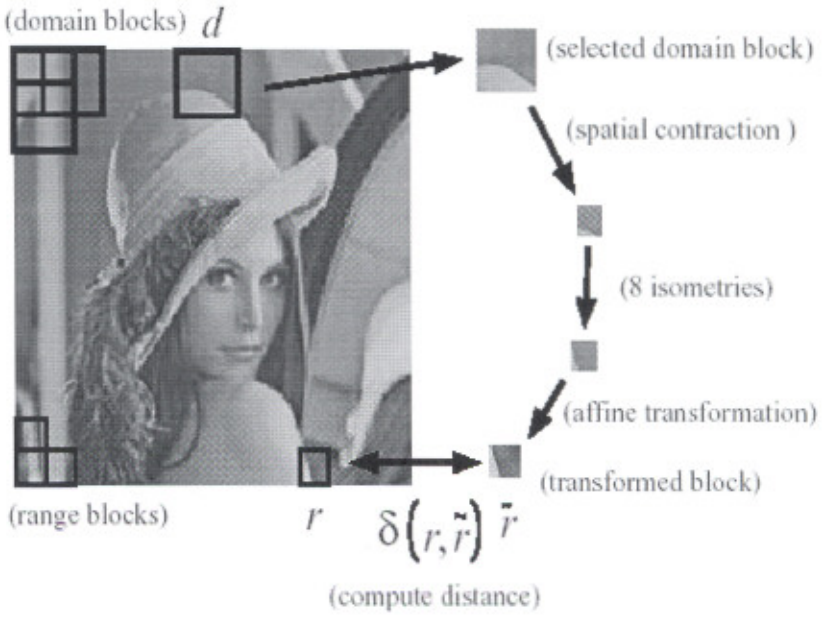


Figure 2.12: Fractal Image Encoding Process

Once encoding is complete, a list containing the selected domain block and transform parameters for each range block represents the image. The image is decoded by iteratively transforming an arbitrary initial image using the transform consisting of the union of the transforms for each range block.

2.5.1 Distance Measure

Fractal Image Coding requires the definition of similarity measure between the image block RB and the transformed image block DB. For this, Root Mean Square (RMS) metric is preferred because it can be easily computed and fits well with visual perception for relatively small block sizes.

The distance between two images is defined as the sum over the partition of all block distortions.

If we have two squares containing n pixel intensities, **a** (from DB) and **b** (from RB). We can seek s and o to minimize the following quantity. (MSE calculation)

$$R = \sum_{i=1}^{n} (s \cdot a_i + o - b_i)^2.$$

Eq (2.2)

Where **n** is the total number of pixels in the range block.

The fractal image coding is to find the minimum approximate error for a domain block (DB) through affine transformation to match a range block (RB) by minimizing the **rms** error between them (Square root of the above quantity R) [4].

In order to decrease the computing time, we optimize the luminance transformation parameters and the MSE calculation. The MSE between a range block RB and the transformed codebook block DB of size B x B is given by equation. This error is optimal when the luminance transformation parameters are optimal. To minimize the equation, the partial derivatives according to the contrast parameter **a** and the brightness **b** must be set to zero.

The output of the compression process is the set of matches. For every range block, the location of the corresponding domain block and the transformation applied to it are recorded in Fractal Image File (FIF) format, which consists of a header and the packed list of affine coefficients for the chosen maps [6].

2.5.2 Difficulties in Implementation of Encoding process:

The whole process of searching the right DB for one RB is what makes fractal encoders very tricky to implement. The basic algorithm for fractal image compression, proposed by Barnsley, is time consuming, due to the requirement of search for the best fitting domain block for each range block. The size of the range blocks is an important parameter in the encoding scheme. An encoding with small range blocks results in a high

signal to noise ratio but a low compression ratio. With a higher size of range blocks, the compression ratio is increased but the details in the range are lost. To capture fine details while preserving a high compression ratio, the quad-tree scheme is introduced.

Many efforts have been undertaken to speed up the encoding process. Most of the techniques attempt to accelerate the searching either using classification of domains and ranges or by using fewer domains.

Several techniques proposed by various researchers the reduction of encoding complexity are described in the section 2.10 briefly. This research work proposes one such technique with experiments and evaluations in the next chapters.

2.6 FRACTAL IMAGE DECODING

Decompressing the information in the fractal-compressed file is less complex than the compression process. Given the fractal code of an image (FIF file or compressed image), the image reconstruction is done by iterating the transformation on any arbitrary image.

For each range R, the domain D_i that maps to it, is shrunk by two in each dimension by averaging non-overlapping groups of 2 × 2 pixels. The shrunken domain pixel values are then multiplied by s_i, added to o_i, and placed in the location in the range determined by the orientation information. Thus, each domain block of the initial image is transformed to its corresponding range block.

Actual value of the pixel range intensity is given by

$z = s_i * d_{ij} + o_i$,

Where d_{ij} is the domain pixel value at (i, j).

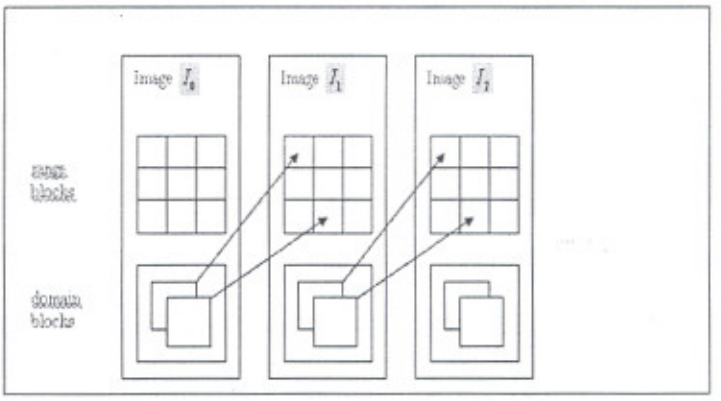


Figure 2.13: Illustration of decoding process

This constitutes one iteration of decoding procedure. The decoding step is iterated until the fixed point is approximated, i.e. until further iteration doesn't change the image or until the change is below some small threshold value.

As the decoding proceeds, the image converges to a stable image. Usually 4 iterations are adequate to reconstruct the decoded image.

To measure the quality of the produced image, we use peak to signal to noise ratio (PSNR)[9]. For 8 bit grey scale images the PSNR measured in dB is defined as

$$PSNR = 10Log\frac{255^2}{rms-error} = 10Log\frac{255^2}{\frac{1}{\#pixels}\sum(p'_{ij}-p_{ij})^2}$$

Where $\mathbf{p}_{ij}$ and $\mathbf{p'}_{ij}$ denote the pixel intensities in the original and the reconstructed images respectively.

Decoding at larger size:

The transformations that encode an image are resolution independent. This property has been cited in the popular technical press as one of the main advantages of fractal image

compression. The underlying model uses, functions of infinite resolution, and the fixed point reached here is such a function. Therefore, during decoding we can select any resolution.

Post-Processing:

Since the ranges are encoded independently, there is no guarantee that the pixel values will be smooth at the block boundaries. The eye is sensitive to such discontinuities, even when they are small. It is possible to minimize these blocking artifacts by post processing the image. In this implementation, only pixel values at the boundary of the range blocks are modified using a weighted average of their values.

2.6.1 Efficient storage

To increase the compression ratio, the following bit allocation schemes are used [6].

Quad tree: One bit is used at each quadtree level to denote a further recursion or ensuing transformation information. At the maximum depth, however, no such bit is used, since the decoder knows that no further division is possible.

Scaling and offset: Five bits are used to store the scaling and seven for the offset.

Domains: The domains are indexed and referenced by this index. However, when the scaling value is zero, the domain is irrelevant, and so no domain or orientation information is stored in this case.

Orientation: Three bits are used to store the orientation of the domain-range mapping.
The above algorithm is implemented and tested for various gray level images. The results obtained are better compared with those for exhaustive search method. The results are shown in Table 2.2 for exhaustive search method. The confidence gained by the above-completed works has motivated the researcher to suggest new methods. It is proposed to

improve the above methods by incorporating new classification or cluster analysis schemes.

Table 2.2: Encoding Time and Quality of decoded Lena image of size 256x 256 for Exhaustive Search Method

Compression Ratio	Exhaustive search		
	Enc.Time (hrs:min)	RMS Error	PSNR (dB)
4.5	3:57	7.212	30.97
8.6	2:26	7.999	30.07
10.4	2:13	8.692	29.438
21.1	1:24	12.609	26.117
42.7	1:00	16.941	23.552

A Simple Illustrative Example.

Consider a 256 × 256 image in which each pixel can be one of the 256 levels of grey. Let it segmented into non-overlapping range blocks of size 8 × 8 and overlapping domain blocks of size 16 × 16.This would result in 1024 range blocks. Let D be collection of domain blocks with 241* 241=58,081 squares. For each R_i, we have to search through all of D to find a $D_i \varepsilon D$ which minimized the eq.2:2. That we have to find the part that looks almost like R_i. This domain is called cover of the range R_i. There are 8 ways to map one square onto another, so that this means comparing 8*58,081=464,648 squares with each of the 1024 range squares. Also a square in D has 4 times as many pixels as in R_i, we must either sub sample or average the 2 × 2 sub-squares corresponding to each pixel of R_i, The results are good .The original image required 65536 bytes of storage whereas the transformations required only 3968 bytes giving the compression of 16.5:1.

2.7 ADVANTAGES OF FRACTAL IMAGE COMPRESSION

Fractal image compression insures competitive compression ratio and image fidelity, so it is good for multimedia applications where quick access to high quality images is essential [4,7,8,9].

- **High Compression Ratios**. Fractal image compression can offer higher compression ratios than JPEG coding, while maintaining near equivalent PSNR values [7].

- **No Codebook.** Fractal image coding used a virtual codebook while encoding the image, and does not require the storage or transmission of any part of a codebook or synchronization around a common codebook. Fractal image coding defines an image as iterated function systems, with only the parameters of this function needing to be stored or transmitted.

- **Fast Decompression.** Fractal image decoding is linear with respect to the number of pixels in an image. These iterations will cause the initial image to converge to the fixed-point solution of the function.

- **Resolution Independence.** The image may be decoded at a different resolution than that of the original image, with higher resolution being obtained through the fractal nature of the image itself.

- **High Quality Decoded Image.** Fractal image coding has been classified as a second generation coding scheme because fractal image compression takes into account the Human Visual System since fractal curves look very similar to natural images.

- **Combination with Other Techniques.** Fractal image coding can easily be adapted to work in combination with other coding techniques.

2.8 WEAKNESS OF FRACTAL IMAGE COMPRESSION

- Fractal image compression suffers from long encoding time due to the requirement of search for the best fitting domain block for each range block.
- The image has to be partitioned into blocks. Because errors are correlated within a block but uncorrelated across neighboring blocks, a very distracting artifact results. The image appears to be made up of "tiles".
- The success of this scheme seems to rely exclusively on some "mystic" property of natural images to exhibit some self-similarity among parts of itself.
- At low compression ratios, conventional transform methods generally perform better.
- Not suitable for Medical images [4,5,7,8,9].

2.9 APPLICATIONS

- Since fractal compression is an asymmetrical process, taking much longer to compress an image than to decompress it. This characteristic limits the usefulness of fractally compressed data to applications where image data is constantly decompressed but never recompressed. Fractal compression is therefore highly suited for use in image databases and CD-ROM applications and browsing archives where encoding is done only once, while decoding is done often.
- Fractal performs very well for highly sampled, natural images.
- Fractals are good for transmitting images over Internet. Resolution independence property is desired, as target browsers have different resolutions.

The content and resolution of the source bitmap can greatly affect fractal compression. Images with a high fractal content (e.g., faces, landscapes, and intricate textures) result in much higher compression ratios than images with a low fractal content (e.g., charts, diagrams, text, and flat textures). High-resolution images may be compressed to achieve higher compression ratios and will still retain a high image quality. To retain a high

quality for lower resolution images, the resulting compression ratio will be much lower. Images with a greater bit depth (such as 24-bit true color images) will also compress more efficiently than images with fewer bits per pixel (such as 8-bit gray-scale images).

Netscape 4.02 has a plug-in of Fractal Image Format (FIF). Microsoft Encarta had all its images compressed in this format [4,7,8,9,10].

2.10 COMPLEXITY REDUCTION METHODS IN FRACTAL IMAGE COMPRESSION

Due to the unacceptably high complexity of encoding of the fractal image compression scheme a lot of work has been directed at reducing the encoding complexity. The methods suggested may be classified as [4]:

a) Domain Pool Restriction
b) Block Classification
c) Adaptive Clustering
d) 1-D Functional Methods
e) Feature Vectors and
f) Transform Domain Block Matching
g) Different Partitioning Schemes
h) Other methods for encoding complexity reduction

a) Domain Pool Restriction

The first step in the reduction of computational complexity of the encoding process is to restrict the domain pool. Not all possible domain blocks are searched. Fisher suggested restriction of the domain pool to blocks twice the size of the range blocks with adjacent domain blocks overlapping one another by the size of the range blocks. Saupe suggested

restriction of domain blocks to those with high variances. Barthel et al restricted the domain pool to the nearest neighbors and search for the appropriate domain block was carried out in a spiral path around the range block. Classification and clustering schemes also fit to some extent in the framework of domain pool reduction.

b) Block Classification

Classification can be done on the basis of metric or non-metric features. Jacquin introduced block classification scheme similar to the one proposed by Ramamoorthi and Gersho for Vector Quantization of images. In this method the domain and range blocks are classified as shade edge and mixed blocks. Edge blocks are further classified as simple and mixed edge blocks. For a range block from any one of the above categories only domain blocks from the same category are searched. Fisher et al proposed a more elaborate classification scheme based on variance of the blocks with 72 classes and Jacobs developed an archetype classification scheme. In this scheme the archetype form for a particular codebook block is given by the block that can best cover all others having the same archetype best in the least squares sense. Starting from an arbitrary classification using codebook blocks from a library made from many images the classification scheme is iterated till the best archetype form for each block is determined. The iteration is stopped when there is no more change in the selection of archetype blocks in further iterations. These archetype blocks derived from training images are used for classification of the range and domain blocks of images [6,Ch 4,14].

c) Adaptive Clustering

In the methods described earlier the classification scheme is decided upon before it is applied to any image, thus relies on the heuristically chosen criteria. With more analytical and adaptive methods, better control of the involved tradeoffs (encoding time, number of classes, image quality) may be possible. The set of classes is designed adaptively depending on target images.

Lepsoy and Oien proposed an adaptive codebook-clustering algorithm in which classification is image dependent. Boss and Jacobs [6,Ch 4] considered an archetype classification based on a set of training images. A first algorithm based on the Kohonen's Self-Organizing Map (SOM) for codebook clustering was presented by Bogdan [11]. Domain blocks are clustered, yielding a classification with a notion of distance, not used in traditional schemes. Designing the clusters on an initial training set rather than determining them adaptively for each image may avoid the computational cost of clustering during encoding.

d) 1-D Functional Methods

Bedford et al proposed a method in which the range and domain blocks are compared by projecting each of them on a common fixed unit vector. A particular range block is compared only with domain blocks whose result of comparison with the unit vector is close to that of the result of comparison of the range block. This method has further been extended to include more such vectors.

e) Feature Vectors

Saupe first proposed this method. In this method a small set of d real-valued keys are assigned to each range and domain block, which make up the d-dimensional feature vector. These keys are constructed such that searching in the domain pool is restricted to a small neighborhood around the d-dimensional key corresponding to a particular range block. Thus the sequential search of the domain block is substituted by a nearest neighbor search.

Kominek used a much simpler scheme for arriving at the feature vector but used the r-trees algorithm for searching. In this algorithm the d-dimensional space of the feature vectors is divided into d-dimensional rectangles. For a range block with its feature vector located in one rectangle only domain blocks with feature vectors in the same rectangle are searched.

Frigaard et al used two dimensional feature vectors the grey scale standard deviation (a continuous feature) and the number of dominant grey levels (a discrete feature) in each block. Bani-Eqbal used 4-dimensional feature vectors.

f) Transform Domain Block Matching

Wohlberg et al proposed a scheme where block matching is done in the DCT domain. Here they make use of the well-known energy compaction property of the DCT to reduce the number of dimensions of the feature vector space where a nearest neighbor search is performed. Also the fact that the DCT of the transformed blocks for the usual transformations employed can be obtained from the DCT of the original block with just multiplication by ± 1 is put to use.

g) Different Partitioning Schemes

When the whole image is partitioning into range blocks they can be partitioned into different ways based on squares and rectangles such as Fixed-Size partitioning, Quadtree partitioning, Triangular partitioning, HV partitioning etc. The performances of these ways depend on the content of the image. So we can select any of the existing or hybrid methods [6,13,14].

h) Other methods for speeding up the encoding process

Using parallel architecture can speed up the searching process as well as encoding process. Some form of pipelining may be incorporated. For example, a master processor could determine ranges and deal with input/output, while each of several sub-processors compares a range to some subset of the domain pool. Each sub processor would find the best domain in its subset and return it to the master processor, which would find the optimum domain and send the next range for matching [8].

2.11 CLASSIFICATION AND CLUSTERING SCHEMES IN FRACTAL IMAGE COMPRESSION

One way to reduce the domain pool search is the block classification or codebook clustering. Before encoding, all the domain blocks are classified. During encoding a potential range is classified and only the domains with same or near classification are searched through.

This research work is focused on reducing the computational complexity of fractal image compression by using an appropriate classification and clustering scheme [6].

2.11.1 Classification and its significance

The domain range comparison step of the encoding is very computationally intensive. A classification scheme is used in order to minimize the number of domains to be compared with a range block. Before encoding, all the domain blocks are classified. During encoding a potential range is classified and only the domains with same or near classification are searched through. This significantly reduces the number of domain range comparisons (See Figure 2.14). By 'near' classification, we mean that the squares that would have been classified differently if their pixel values were slightly different. The idea of using a classification scheme was developed independently. Many classification schemes are possible [7].

The classes and classification schemes were introduced by Ramamoorthi and Gersho [15] for the vector quantization of the images. The domain blocks were classified based on their perceptual features. The classes were: Shade class with no significant gradient, Midrange class with moderate gradient but no definite edges and a Mixed class of blocks with no distinct orientation. A matching domain for each range was only sought within the same class.

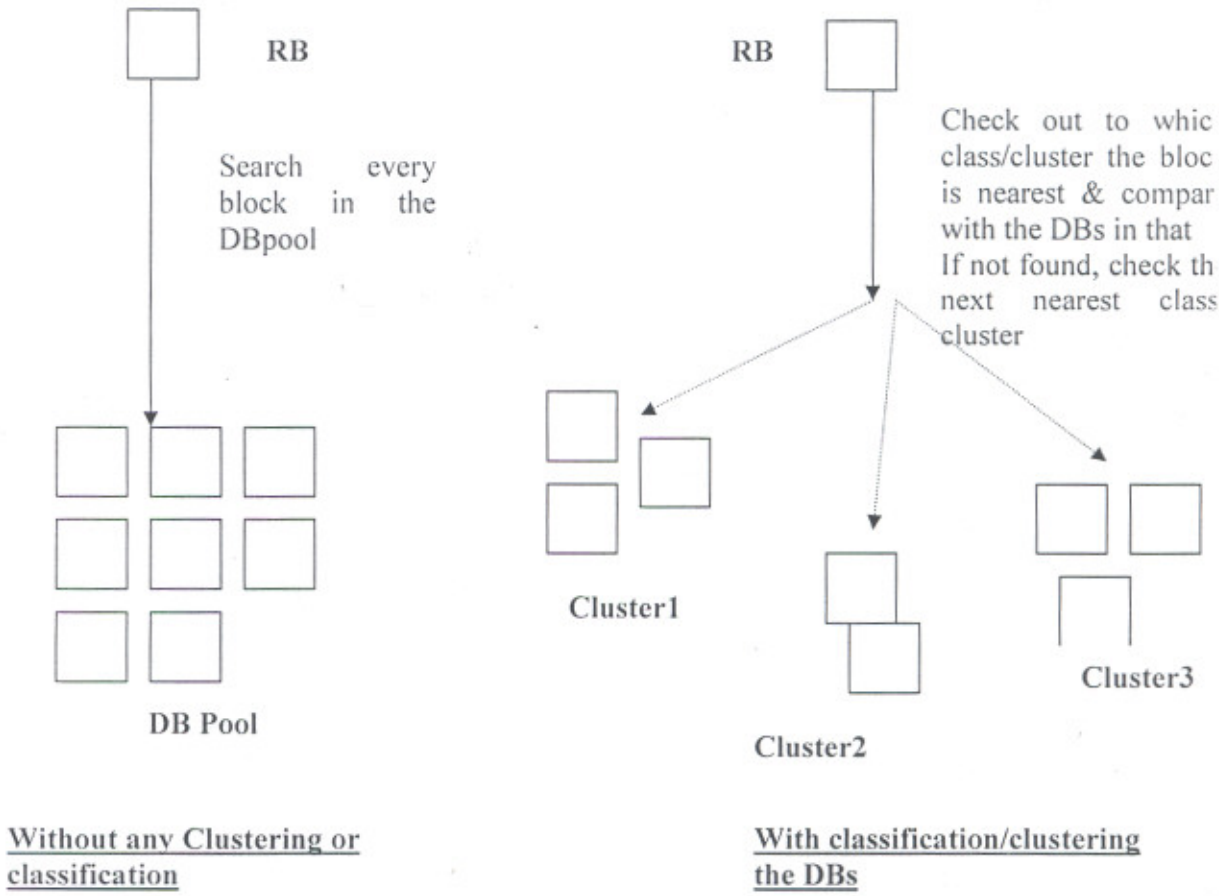


Figure 2.14: Search Process- With and Without clustering

Classification based on Non-metric features:

Jacquin used a classification scheme [16-19] in his original approach to reduce the number of comparisons. His classification scheme was based on that given by Ramamoorthi and Gersho [15]. Blocks were: shade blocks, edge blocks and midrange blocks. A shade block is smooth with no significant gradient. An edge block presents a strong change in intensity. A midrange block has moderate gradient but no definite edge.

Fisher et al used a similar scheme with more classes (72 classes) also chosen to distinguish between blocks of different appearance based on the relative averages of quadrants of each block [7,Ch 3].

Classification based on Metric features:

Frigaard et al [20] computed features in a space with a metric, searching blocks with features within some distance threshold of the range block features. The features utilized were the block standard deviation and the number of "dominant grey levels" which is the number of distinct pixel values for which the number of pixels with that value exceed some threshold.

Novak [21] classified blocks according to a set of invariant features based on the moments of the block pixel values in a triangular partition. An alternative set of features may be defined by calculating inner products with a fixed set of vectors. These inner products provide lower bounds on distances between domain and range blocks, allowing many of the domains to be excluded from the actual distance calculation.

Gotting and Ibenthal [22] transformed the standard invariant representation into a set of features, which were also invariant to the block isometries, arranging them in a tree structure to speed up the search. A tree search has also been applied to a pyramid of progressively coarser resolution domains, with the search progressing at each level in the region of the best match in the previous level [23,24].

2.11.2 Fisher's Classification scheme

A square sub-image is divided into upper left, upper right, lower left and lower right quadrants numbered sequentially. On each quadrant, we compute values proportional to the average and variance: if the pixel values in quadrant **i** are $\mathbf{r_1}^i, \mathbf{r_2}^i \ldots\ldots \mathbf{r_n}^i$ for i =1, 2 ,3..n , We compute

$A_i = \Sigma r_j^i$ and $Vi=\Sigma(r_j^i)^2 - Ai^2$

It is always possible to orient the sub image so that A_i are ordered in one of the following three ways:

Major Class 1: $A_1 \geq A_2 \geq A_3 \geq A_4$.

Major Class 2: $A_1 \geq A_2 \geq A_4 \geq A_3$.

Major Class 3: $A_1 \geq A_4 \geq A_2 \geq A_3$.

These correspond to the brightness levels shown in the Figure 2.15.

Figure 2.15: Classification of image based on brightness

Once the rotation of the square has been fixed, each of the three major classes has 24 subclasses consisting of 24 orderings of the V_i. Thus, there are 72 classes in all if the scaling value s_i is negative; the orderings in the classes above are rearranged. Therefore, each domain is classified in two orientations, one orientation for positive s_i and the other for negative s_i. For a range block belonging to one particular class, only domain blocks of the same class are searched.

In this approach no notion about distance between the classes was defined so that one cannot search in the nearest neighboring class in case no match in the selected class is found [7,Ch 3].

2.11.3 Clustering

The original goal of clustering is to find similarity between elements and to group them together based on a threshold of similarity between the elements. There are many clustering algorithms available.

Clustering, a variation of classification speeds up fractal compression by partitioning the domain blocks in **k** clusters based on specific properties found in the block. Only the domain blocks in the same cluster are searched resulting in a speedup of a factor of **k**, not including any overhead required to classify the blocks.

According to a standard text book on pattern recognition [22] "Clustering algorithms are methods to divide a set of **n** observations into **k** groups so that members of the same group are more alike than members of different groups . . . the groups are called clusters". Since the main point about a cluster solution is that members of the same group are more alike than members of different groups, one has to have a means of measuring the likeness of such members.

The idea is to divide the codebook (set of domain blocks which are shrunken to the size of range blocks and under gone with eight different transformations) i.e. the domain pool is divided into set of clusters. Each cluster is having a representative called cluster center. Cluster centers should ideally lie in the areas of feature space with high data density. Codebook therefore captures the internal structure of the data set. Figure 2.16, 2.17 and 2.18 shows the data set and clustered data set [7,Ch 9].

This organization of codebook permits efficient encoding. The general procedure is:

i. **Initialization:** Sub divide the codebook into clusters D1, D2 ...Dn by computing the cluster centers C1, C2,..., Cn and grouping the codebook blocks around the cluster centers. The centers have same size as the blocks.

Figure 2.16: Data set shows the dumps of feature vectors

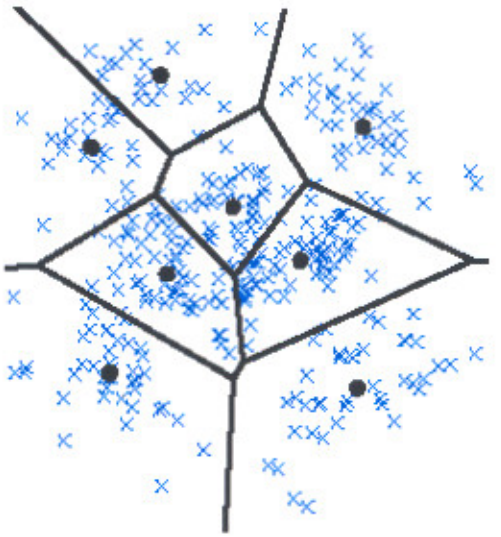


Figure 2.17:Cluster boundaries and cluster centers after a cluster algorithm

ii. Encoding: For each range block, first the cluster with cluster center with minimum distance to range block is searched. Then in that cluster, all the codebook blocks are searched for the one that most resembles the range.

To use such a clustering procedure, Firstly we need a measure or function to find a cluster center, function to find if a range resembles the codebook block and a procedure to find the good cluster centers.

A multitude of cluster algorithms have been developed. Our goal is to use the optimal clustering algorithm. An optimal clustering algorithm reaches the reasonably good solution in a reasonable amount of time. It provides the clusters with minimum variability between the elements within a cluster and maximizing the value between elements within different clusters. The choice of a clustering algorithm depends on the data type, purpose and the application. In the literature a wide variety of algorithms have been proposed for different applications and sizes of data sets [33,34]. Some of them are:

- K-means/Modified K-means clustering algorithm
- Nearest neighborhood algorithm
- LBG-algorithm
- Lloyd's algorithm
- Clustering using wavelet transformation.

Hybrid of many algorithms can be used to take the advantage of strength of various techniques.

2.11.4 Selection of Best Algorithm

The criteria widely accepted for evaluating the partitioning a data set are:

i. The separation of the clusters, and
ii. Their compactness.

In general terms, we want clusters whose members have a high degree of similarity (or in geometrical terms are close to each other) while we want the clusters themselves to be widely spread.

Here the term "optimal" implies parameters that lead to partitions that are as close as possible (in terms of similarity) to the actual partitions of the data set. Then, the

algorithm and the respective set of input parameters resulting in the optimal partitioning of a data set may be selected.

2.12 Adaptive Clustering

Block classification methods usually rely on heuristically chosen criteria. If more analytical or adaptive methods could be found, there would be better control of the involved trade offs (encoding time, number of classes, image quality). Adaptation tends to maximize the attainable image quality while keeping the total complexity of adaptation, classification and subsequent encoding low.

Lepsoy and Oien[25,26] introduced the adaptive codebook-clustering algorithm for classification. After pointing out that the best match for a range block is the domain block whose projection in a given subspace is mostly parallel to it, Lepsoy and Oien proposed a clustering algorithm for classifying blocks [1,Ch 9].

They reduced the search effort by identifying clusters of domain blocks in the domain pool, the cluster centers being located by applying the GLA [28,32] to the domain blocks using a distortion measure based on an invariant representation equivalent to the cosines of the angles between vectors. The optimum domain for each range was located by a comparison of the range with each cluster center, followed by a comparison with all cluster members of the best cluster.

A similar clustering approach, but using the standard invariant representation was evaluated by Hamzaoui [12]. A clustering approach based on the GLA has also been applied to domain blocks in a triangular partition [48].

Boss and Jacobs [1,Ch4] avoided the computational cost of clustering during encoding by designing the clusters on an initial training set rather than determining them adaptively for each image.

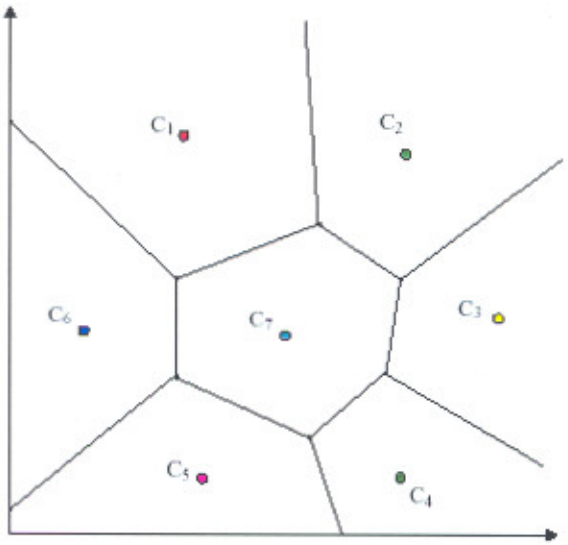


Figure 2.18:Representation of Clusters in 2D space

Different classical clustering algorithms exist in the literature, use of each depending on the type of application and purpose. We, in this thesis, are concerned with how self-organizing maps, a clustering and visualization tool, based on neural networks are used in cluster analysis in fractal image compression and how it helps in performance improvement. The experiments are also done and evaluated to compare it with other techniques.

CHAPTER 3

Self-Organizing Maps

3.1 INTRODUCTION

Self-Organizing Maps or Kohonen's SOMs is one of the best and robust neural network algorithm based on unsupervised (machine) learning in contrast to many other neural network algorithms. A SOM learns to classify the data without any external supervision. Thus SOM is visualization and a clustering tool.

The Self-Organizing Map (SOM) was introduced by Teuvo Kohonen in 1982 so also called Kohonen's map. The SOM originated from the LVQ (Learning Vector Quantization) network the underlying idea of which was also Kohonen's in 1972. A SOM is a simple analog of human brain's way of organizing information in a logical manner. In the brain, neurons tend to cluster in groups. The connections within the group are much greater than the connections with the neurons outside of the group. Kohonen's network tries to mimic this in a simple way. The first application area of the SOM was speech recognition, or perhaps more accurately, speech-to-text transformation [35].

SOMs are unsupervised Artificial Neural Networks which are mathematically characterized by transforming high dimensional data into two dimension representation or map or grid of neurons enabling automatic clustering of the input, while preserving high order topology. A SOM has one layer of source nodes and one layer of output neurons. The weights between input nodes and output layer learn to cluster input patterns.

Because no teacher is available for training, a similarity measure is used to express the closeness between patterns in self-organizing paradigms [41].

The virtue of self-organizing maps (SOMs) is their ability to extract intrinsic topological structure hidden in multidimensional data despite the simplicity of their algorithm. In reality, SOMs are too flexible and fit to any data distribution, no matter which topology they postulate. Thus, using SOM, we are faced with the difficulty of determining the best topology from a number of possibilities.

The SOM algorithm is regarded as a vector quantization (VQ) algorithm with a topological constraint, which gives stability and robustness to the original VQ algorithm. By gradually eliminating the constraint, the SOM algorithm ultimately converges to one of the solutions of VQ. In fact, many applications have used SOMs in this manner.

Thus, SOM can be defined as:
Self-organizing map is a feed-forward neural network approach that uses an unsupervised training algorithm and through a process called self-organization, configures the output units into a topological representation of the original data.

In contrast to the classical clustering methods a self-organizing map provides easy visualization, imposes few assumptions and restrictions, and is able to handle large data sets to detect isolate patterns and structures in the data. Self-organizing maps have thus becoming of increasing interest for exploratory data analysis and data mining and image processing also in finance and economics [38].

3.2 SOM MODEL

The basic idea of SOM is simple yet effective. It realizes a mapping between input space and output space that preserves topology; in other words, if vectors are near from each

other in the input space, their projection in the output space will be close too. This property of SOMs is utilized in this research.

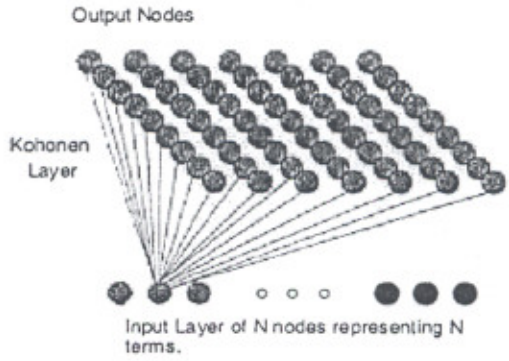


Figure 3.1: Kohonen's SOM with 2-D output space

The way that SOMs go about organizing themselves is by competing for representation of the samples. Neurons are also allowed to change themselves by learning to become more like samples in hopes of winning the next competition. It is this selection and learning process that makes the weights organize themselves into a map representing similarities. See Figure 3.1.

Assume that no classes are known in prior. Pattern is submitted to the input layer to identify clusters. We use pattern similarity as clustering criteria. To define a cluster, we need to establish a basis for assigning patterns to the domain of particular cluster. During training, dissimilar vectors are rejected [35-40].

3.3 THEORETICAL BACKGROUND

The basic Self-Organizing Map (SOM) can be visualized as a sheet-like neural-network array (see Figure 3.1), the cells (or nodes) of which become specifically tuned to various input signal patterns or classes of patterns in an orderly fashion.

Although the neurons of the network can be arranged in a 2-dimensional grid, or a line, or some other geometry (Figure 3.2). Thus, the location of the nodes within the network has some geometrical significance.

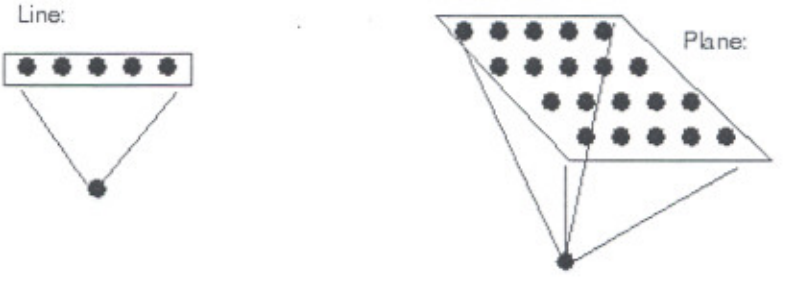


Figure 3.2: Kohonen's SOMs

The output space is supposed to be one or two-dimensional array of nodes (See Figure 3.2). Suppose that an input pattern has N features and is represented by a vector x in an N-dimensional pattern space. i.e. each neuron on the grid has, as many input connections as there are number of attributes to be used for the classification. The network maps the input pattern to an output space. The output space preserves a certain topological orderness.

All neurons or output nodes are physically arranged in a square grid. Each neuron on the grid has, as many input connections as there are number of attributes to be used for the classification. It is thus possible to define k-neighborhoods on the grid, which include all neurons whose distance from one neuron is less or equal to k. Each of the node s assigned a weight vector m_i. Weight vectors has same dimensionality as that of input vectors. Kohonen proposed to allow the output nodes interact laterally, leading to the self-organizing feature map [42].

Note that self-organizing maps are performed in a way similar to the k-means algorithm used in statistics. Although, the latter has been shown to perform differently and less satisfactory than the first.

Although, it is important to evaluate the complexity of the algorithm. The computational complexity of SOM is K^2 where K is the number of map units. Each learning step requires O(K) computations, and to achieve a sufficient statistical accuracy the number of iterations should be at least some multiple of *K*.

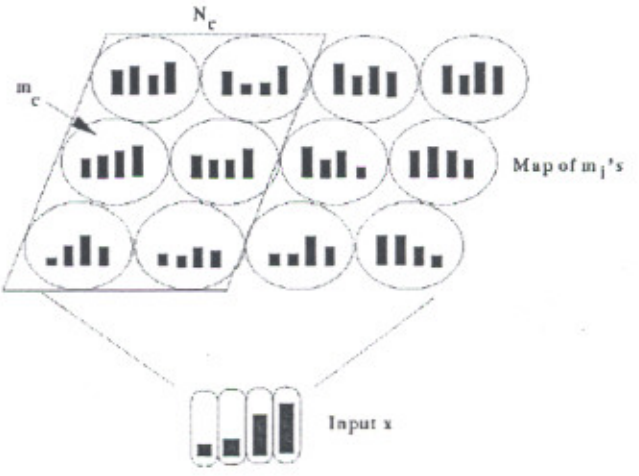


Figure 3.3: Basic architecture of SOM

3.4 MAIN SOM ALGORITHM

With this approach an input vector is presented to the network (typically a multilayer feed-forward network) and the output is compared with the target vector. If they differ, the weights of the network are altered slightly to reduce the error in the output. This is repeated many times and with many sets of vector pairs until the network gives the desired output.

The Main Algorithm goes as:

(i) Assume output nodes are connected in an array (usually 1 or 2 dimensional)

(ii) Assume that the network is fully connected - all nodes in input layer are connected to all nodes in output layer.

(iii) Use the competitive learning algorithm as follows:

SOM formation process has four steps [36]

(i) Initialization of weights

(ii) Competition

(iii) Cooperation

(iv) Adaptation / Learning

Having completed the learning process, the mapping can be processed by calculating for each input distance vector the winning neuron.

During the mapping process a new input vector may quickly be given a location on the map, it is automatically classified or categorised. There will be one single winning neuron: the neuron whose weight vector lies closest to the input vector. (This can be simply determined by calculating the Euclidean distance between input vector an weight vector.)

3.4.1 Learning Algorithm Overview

A SOM does not need a target output to be specified unlike many other types of network. Instead, where the node weights match the input vector, that area of the lattice is selectively optimized to more closely resemble the data for the class the input vector is a member of. From an initial distribution of random weights, and over many iterations, the SOM eventually settles into a map of stable zones. Each zone is effectively a feature classifier, so you can think of the graphical output as a type of feature map of the input space.

(i) Initialization of Weights

Each node's weights must be initialized, prior to training. So the first step in constructing a SOM is to initialize the weight vectors to small-standardized random values. You select a sample vector randomly and search the map of weight vectors to find which weight best represents that sample. Typically these will be set to small-standardized random values. Since each weight vector has a location, it also has neighboring weights that are close to it. The weight that is chosen is rewarded by being able to become more like that randomly selected sample vector. In addition to this reward, the neighbors of that weight are also rewarded by being able to become more like the chosen sample vector.

(ii) Competition

This input pattern is presented to the self-organizing map and each unit or node or neuron determines its activation. Every node is examined to calculate which one's weights are most like the input vector. That node is called 'winner ' or 'Best Matching Unit' (BMU).

a) Calculating the Best Matching Unit

Determine the winning output node or 'winner' i, where w_i is the weight vector connecting inputs to output node i.

Usually, the Euclidean distance between weight vector and input pattern is used to calculate a unit's activation but other distance measures can be chosen.

To determine the best matching unit, one method is to iterate through all the nodes and calculate the Euclidean distance between each node's weight vector and the current input vector. The node with a weight vector closest to the input vector is tagged as the BMU.

Best Matching Unit is:

$i(x) = \arg\min ||x-w_j||, j = 1,2,3,..,l$

Where $\mathbf{x}$ is the current input vector and $\mathbf{w}_j$ is the node's weight vector.

b) **Determine the location where the topological neighborhood of excited neurons is to be centered**

(iii) Cooperation

When a node wins a competition, not only is its weights adjusted, but those of the neighbors are also changed. They are not changed as much though. The farther the neighbor is from the winner, the smaller its weight change.

a) Determining the Best Matching Unit's Local Neighborhood

Within each iteration, after the BMU has been determined, the next step is to calculate which of the other nodes are within the BMU's neighborhood. All these nodes will have their weight vectors altered in the next step.

First the radius of the neighborhood is calculated and then it is determined if each node is within the radial distance or not.

In neighborhood, each vector is surrounded by other 6 (hexagonal) or 8 (rectangular arrangement) vectors.

You can see that the neighborhood shown above (Figure3.4) is centered on the BMU node **i** (node with red mark) and encompasses most of the other nodes.

A Guassian can be used to define the neighborhood kernel.

$$h_{ci}(t) = \exp\left(-\frac{\|r_c - r_i\|^2}{2 \cdot \delta(t)^2}\right).$$

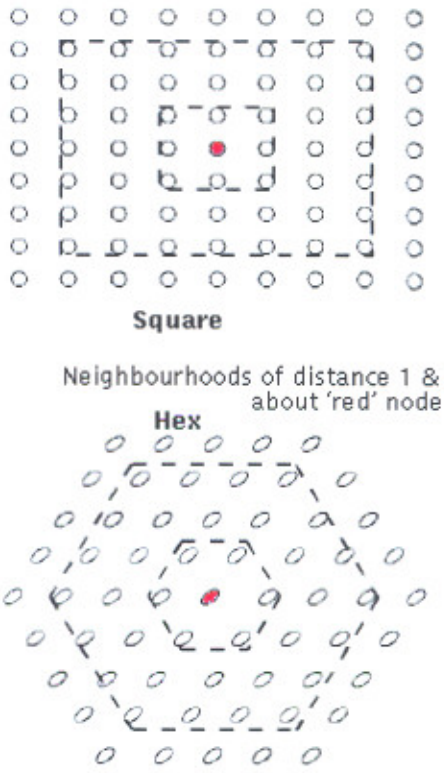


Figure 3.4: BMU's neighborhood-Square and Hexa

Where $\|r_c - r_i\|^2$ denotes the distance between the winner node **c** and input vector **i** and a time-varying parameter δ enable formation of large clusters in the beginning and fine-grained input discrimination towards the end of the learning process. Some other neighborhood functions can also be used instead of guassian function.

A unique feature of the Kohonen's learning algorithm is that the area or size of the neighborhood shrinks over time. This is accomplished by making the radius of the neighborhood shrink over time.

(iv) Learning Process (Adaptation)

The learning process is competitive and unsupervised, meaning that no teacher is needed to define the correct output (or actually the cell into which the input is mapped) for an input. Weights vectors of neurons are adapted to match the input vectors in a training set.

a) Adjusting the Weights

Then weights of all the BMU's neighborhood neurons are adjusted by an amount inversely proportional to the distance. Weight vector of every node within the BMU's neighborhood (including the BMU) is adjusted in relation with the input vector according to the following equation:

$$m_i(t+1) = m_i(t)+\alpha(t).h_{ci}(t).\ [\ x(t) - m_i(t)\ ]$$

Where **t** represents the time-step and **α** is a small variable called the learning rate, which decays with time. This equation is applied to all neurons inside the neighborhood of winning neuron **i**. See Figure 3.4 & 3.5.

Basically, what this equation is saying, is that the new adjusted weight for the node is equal to the old weight ($\mathbf{m_i}$), plus a fraction of the difference (α) between the old weight and the input vector (**x (t)**).

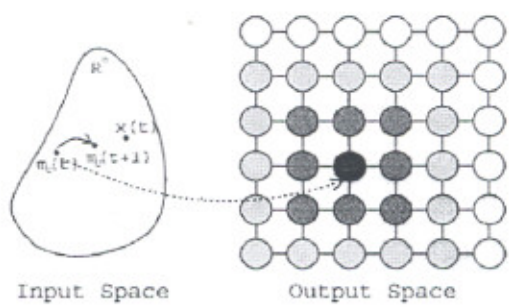


Figure 3.5: Adaptive process

b) Repeat the iterations till predetermined number is reached

Furthermore, as training goes on, the neighborhood gradually shrinks. And at the end of training, the neighborhoods have shrunk to zero size. The training process is terminated

if the RMS errors of all the input are reduced to an acceptable level or prescribed number of iterations are reached. Training a SOM requires no target vector.

The number of training steps must be fixed prior to training the SOM because the rate of convergence in the neighborhood function and the learning rate are calculated accordingly.

Upon repeated presentation of the training data, weights tend to follow the distribution. After the training is over, the map should be topologically ordered. This means that **n** topologically close (using some distance measure e.g. Euclidean) input data vectors map to **n** adjacent map neurons or even to the same single neuron. When input is mapped into regions close to each other in the grid of neurons or output patterns, this can be viewed as a neural network version of k-means clustering [35].

Thus the overall goal of training the net is that nearby outputs correspond to nearby inputs. E.g. if x1 and x2 are two input vectors and t1 and t2 are the locations of the corresponding winning output nodes, then t1 and t2 should be close if x1 and x2 are similar. A network that performs this kind of mapping is called a feature map.

During the training process, the map is built, the neural network organises itself, using a competitive process. The network must be given a large number of input vectors, as much as possible representing the kind of vectors that are expected during the second phase (if any). Otherwise, all input vectors must be administered several times.

Note that after initialization, the SOM training is done in two phases: first rough training and then fine-tuning. Rough training phase with large (initial) neighborhood radius and large (initial) learning rate, and fine-tuning phase with small radius and learning rate. If tighter control over the training parameters is desired, the respective initialization and training functions, e.g. som_batchtrain (in SOM toolbox), can be used directly. There is also a graphical user interface tool for initializing and training SOMs.

Following the description above, the input of the self organizing map is the set of our experimentally obtained images, while the output is a two-dimensional array of nodes, with an image assigned to each of them. These assigned images start with some arbitrary value but, as the algorithm proceeds, will tend to approximate the probability density distribution of the input data

In this way we always have two pieces of information within this scheme, one is provided directly by the code vectors, and the other is the final assignment of each input vector to a given node vector, which results in a classification of the data set.

Kohonen showed that this algorithm has some interesting properties. First, it represents most faithfully those dimensions of the input space along which the variance in the sequence of inputs $\mathbf{X}_i$ is most pronounced. These will often correspond to the most important features of the inputs. Second, it tries to preserve continuity, i.e., it maps similar inputs vectors ($\mathbf{X}_i$) to neighboring locations. Finally, it reflects differences in the sampling density of the input space in a natural way: regions from which inputs have occurred more frequently are mapped onto larger domains and therefore with better resolution.

During self-organizing procedure, the topologically close relationship of the organized information is maintained which is the advantage over using other neural network approaches.

3.4.2 Topology Preservation

Topology means local or neighborhood properties. Topology preservation map attempts to map input patterns to nodes such that "nearness" is preserved; similar patterns mapped to nearby neurons. (Impossible in general, due to geometrical differences between input space and network geometry.

The topology preservation measure describes how well the SOM preserves the topology of the studied data set. Unlike the mapping precision measure, it considers the structure of the map. For a strangely twisted map, the topographic error is big even if the mapping precision error is small.

A simple method for calculating the topographic error:

$\varepsilon_k = 1/N \Sigma u(x_k)$

Where, $u(x_k) = 1$, if first & second BMU of x_k are not next to each other.

$= 0$, otherwise

3.4.3 Distances& Metrics

We need a metric to check the similarity or dissimilarity between the clusters. Standard choice is Euclidean Distance given by $ED=\|x-m_i(x)\|$. We minimize the Euclidean distance. Advantage of using this metric is that there is no need of normalization but other distances may be used.

The city-block metric distance is a simpler measure than Euclidean distance. The expression for the city-block metric distance is given as:

$$d_2(x, y) = \sum_{i=1}^{k} |x_i - y_i|$$

Then, the relative entropy (Kullback Leibler distance) can be employed. The formulation for this distance measure is given as:

$$d_3(x, y) = \sum_{i=1}^{k} y_i \log \frac{y_i}{x_i}.$$

To summarize, the SOM algorithm in few steps is described as:

Step_0: **Convert the images I_i into vectors X_i**

I_i into X_i

Step_1: **Initialize the m_j vectors.**

Initialize the $\mathbf{m}_j$ vectors associated with each output node to random values.

Step_2: **Present one input vector.**

Step_3:**Compute distance to all nodes at time t.**

Compute the distance $\mathbf{d}_{ij}$ between the input vector $\mathbf{m}_i(t)$ and each output node j using:

$$\mathbf{d}_{ij} = \| \mathbf{X}_i(t) - \mathbf{m}_j(t) \|$$

Where $\mathbf{m}_j(t)$ is the code vector associated to node **j**.

Step_4: **Select output node with minimum distance.**

Select the output node as the node associated with the vector $\mathbf{m}_j(t)$ that minimizes $\mathbf{d}_{ij}$.

Step_5: **Select output node with minimum distance.**

Select the output node as the node associated with the vector $\mathbf{m}_j(t)$ that minimizes $\mathbf{d}_{ij}$.

Step_6: **Repeat**

Go to **step 2** until process converge.

3.5 VISUAL INSPECTION OF SOM

To visually inspect a SOM, U –matrix, a matrix that contains the distances between all neighboring neurons, can be calculated to find groups formed by dense sets of grid neurons. U –matrix thus shows the cluster structure of the map. These distances can be displayed color-coded on the low-dimensional representation of the map, because the distances are scalar values whatever the dimension of the underlying system (Figure 3.5).

It can be efficiently visualized using gray shade. Dark shades represent small distances, while bright shades represent large distances. High values of the U-matrix indicate a cluster border; uniform areas of low values indicate clusters themselves [44].

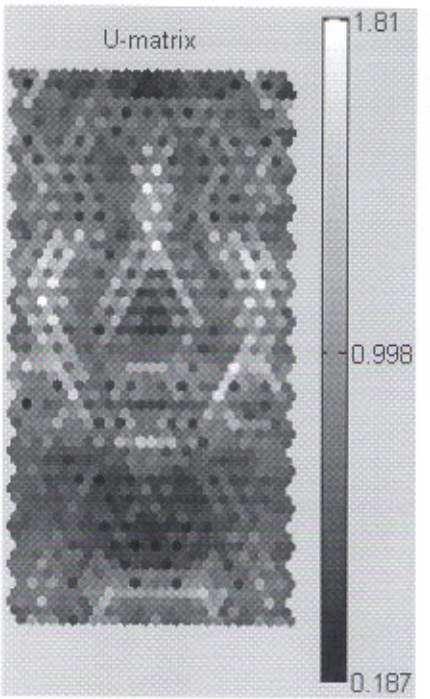


Figure 3.6: Visualization of SOM with U-matrix

CHAPTER 4

Cluster Analysis with Self-Organizing Maps

4.1 PROPOSED CLUSTERING APPROACH

In this thesis, we propose the use of self-organizing maps, unsupervised learning algorithm as a clustering tool in the encoding process in fractal image compression. This thesis discusses how cluster analysis with self-organizing maps is done in fractal image compression and how it helps in improvement of encoding time complexity. The experiments are also done and evaluated to compare it with other techniques.

Let $\{\pm \phi (D_1), \pm \phi (D_2), \ldots\ldots, \pm \phi (D_{Nd})\}$ be the set of projected codebook blocks. We want to partition the set of projected DBs into disjoint sets called clusters with each set having a cluster representatives or cluster centers such that vectors in the same cluster are closer to each other than vectors in different clusters. The quality of clustering can be measured by a criterion function that one tries to optimize. For example, one can choose to construct the cluster centers such that sum of square of Euclidean Distances, ED = $\| x - m_i(x) \|$ is minimized.

Here $m(x_i)$ denotes the cluster center closest to the projected codebook block $\mathbf{x}_i$. A cluster of center **m** is formed by grouping around **m** all projected codebook blocks having **m** as their nearest neighbor. With Kohonen's algorithm, more than one cluster center is considered for each vector $\mathbf{x}_i$. Iterative optimization algorithms are frequently used to find optimal partitions. Unfortunately, they guarantee only local optimization.

After the cluster centers have been designed, the set of projected codebook blocks $\{\pm \phi(D_1), \pm \phi(D_2), \ldots, \pm \phi(D_{Nd})\}$ is clustered by mapping each vector $\pm\phi(D_{Nd})$ to its nearest cluster center. A range block R is encoded in two steps. First, we map its feature vector $\phi(R)$ to its closest cluster center $m(\phi(R))$. Then the range block R is compared only to the codebook blocks whose feature vectors are in the cluster of center $m(\phi(R))$. This corresponds to a 1-class search. We can evidently search in more classes by considering the next nearest cluster centers of $\phi(R)$. This will yield more accurate encoding at the expense of increased time.

4.2 NOTATIONS AND MATHEMATICAL BACKGROUND

Let us assume that a sampled image is partitioned into non-overlapping square blocks of size $N \times N$ called range blocks. This is not a restriction since it will be clear how the principles described carry over to more general partitions. We consider each range block as a vector R in the linear vector space R^n, where $n=N \times N$. The conversion from a square sub-image of side length N to a vector of length $n=N^2$ can be accomplished e.g. by scanning block line by line. The domain pool is a collection of square blocks, which are typically larger than the ranges and taken also from the image, called domain blocks. The domain pool is enlarged by including blocks obtained after applying the eight isometries of the square to the domain blocks. Finally, by pixel averaging, the size of these blocks is reduced to the size of a range block. The resulting blocks are called codebook blocks. In encoding process, for a range block, $R \varepsilon R^n$, a search through the domain blocks $D_1, D_2 \ldots D_n \varepsilon R^n$ is required. Let $E(D_i, R)$ denote the least squares error of an approximation of the range block R by an affine transformation of the codebook block D_i .In terms of formula, this is

$$E(D_i, R) = \min_{a,b \in \mathbb{R}} \|R - (aD_i + bC)\|.$$

Where C is the block of constant intensity C=(1.....1)/ √n. The codebook block D_i which gives the smallest error E (D_i, R) is selected on the condition that the value of the scale factor **a** for the codebook block D_i ensures the convergence of decoding process.

4.3 PROPOSED ALGORITHM FOR ENCODING PROCEDURE

The proposed algorithm consists of following sequence of steps:

Step_1: Partition the image into range blocks using square partition.

Step_2: Partition the image into domain blocks of size two times the range blocks size based on square partitioning.

Step_3: Shrink the DBs to the same size as that of RBs, so that comparisons are possible.

Step_4: Apply all the possible transformations (eight) to the DBs and convert into a vector form to form a codebook (set of all vectors).

Step_5: Classify this codebook into 3 classes based on intensity or brightness, subtracting the mean of each block.

Step_6: Apply the clustering algorithm (SOMs). We can visualize the regions with high density in the data space. These regions are clusters.

Step_7: We can find the cluster representatives (cluster centers) that represent each cluster.

Step_8: For each RB,

- Find out the best nearest cluster.
- In the corresponding DB class, check out the cluster, whose cluster center resembles the RBs the most, i.e. cluster center is a minimum distance from the RB.
- Within the chosen cluster, determine the optimum DB that resembles the RB the most i.e. DB which is at minimum distance from the RB in that cluster.

Step_9: All the mappings are stored in FIF (Fractal Image File) format and sent over the channel.

4.4 ADVANTAGES BY USING SOMS IN FIC:

Probably, one best thing about SOMs is that they are very easy to understand. It's very simple, if they are close together and there is grey connecting them, and then they are similar. If there is a black ravine between them, then they are different.

SOM algorithm is robust, fast and efficient as compared to many classical algorithms. This method efficiently classifies and computes without degrading the image fidelity. One needs not to know the number of clusters in prior. It automatically based on the image data calculates the appropriate number of cluster.

SOM converges faster although it performance is equivalent to that of LBG algorithm. SOM provides high quality clustering in case of high dimensional data. SOMs are flexible and fit into any data distribution.

4.5 LIMITATIONS

One major limitation with SOMs is getting the right data. Unfortunately we need a value for each dimension of each member of samples in order to generate a map. Sometimes this simply is not possible and often it is very difficult to acquire all of this data so this is a limiting feature to the use of SOMs often referred to as missing data.

Another limitation is that every SOM is different and finds different similarities among the sample vectors. SOMs organize sample data so that in the final product, the samples are usually surrounded by similar samples, however similar samples are not always near each other. So sometimes, a lot of maps need to be constructed in order to get one final good map.

Success of the algorithm depends on the successful training of the SOMs. It doesn't give us direct information but merely clusters and classifies the data based on the set of attributes used.

The final major limitation with SOMs is that they are very computationally expensive since as the dimensions of the data increases, dimension reduction visualization techniques become more important, but unfortunately then time to compute them also increases.

To overcome this limitation, we can employ any of the variants of SOM algorithm. A rich variety of versions of the basic SOM algorithm have been developed. Some of the variants aim at improving the computational complexity of the SOM, while some aim at the improvement of topology preservation.

4.6 ALGORITHM FOR DECODING PROCEDURE

The decoding procedure is same as that used in simple fractal decompression process. Decompression process is quicker as compared to other image compression techniques.

The Decoding procedure is as follows:

(i) At the decoder end, any initial or black or white image is taken and square partitioned.
(ii) The FIF file sent contains the information about RB size, location of Domain blocks, transformations applied and scaling and offset information.
(iii) For each RB, the information is read and whole process is reversibly applied.
(iv) With every iteration we move closer to the original image, The final image which can't be changed further is the attractor image which is closest approximate of original image.

4.7 DIFFERENCE BETWEEN CLASSICAL CLUSTERING ALGORITHMS AND SOMS

In the classical clustering algorithms, the main drawback is that number of clusters should be known in prior. SOM is visually efficient, robust and fast as compared to classical clustering methods. SOMs avoid the problem of dead units that arise in LBG algorithm. It automatically determines the number of clusters based on certain statistics. Moreover, classical clustering algorithms are not much suitable for high dimensional data while the SOMs are.

The main drawback with SOM clustering is that there is fixed size in terms of number of units and their arrangement. Secondly, hierarchical relations between the input data are not mirrored.

A sample algorithm for fractal image compression with SOMs used for clustering is as follows:

i) Classify the image data into three classes based on quadrant brightness or variance, subtracting the mean of each class.

ii) For each class, generate a set of clusters using a clustering algorithm.

iii) For each range block within its class, find the cluster with cluster center of minimum distance form the clusters in same class. Then In the cluster with resembling cluster center (centroid), Find the block with minimum distance (resembling block) from range block.

CHAPTER 5

Implementation Details and Experimental Results

5.1 IMPLEMENTATION

A powerful and flexible implementation of SOM is given by SOM toolbox for Matlab computing environment by MathWorks Inc. This Toolbox contains functions for creation, training, different kind of visualizations and analysis of properties of the SOMs and data, e.g. SOM quality, clusters on the map and correlations between variables. The Toolbox is available free of charge under the GNU General Public License from http://www.cis.hut.fi/projects/somtoolbox. It requires no other toolboxes, but the basic functions of Matlab. Other software packages for SOM implementation are also available as SOM for R, SOM_PAK package.

Although the Matlab Neural Network Toolbox does have unsupervised clustering functions, past experience has shown that they do not work very well. SOM Toolbox has myriad visualization functions. U-matrix can be used to visualize the clusters in image data, which shows the distances between all the neighboring neurons.

The experiments were done on the P-3 machine with 550MHz CPU running Windows 98 Operating System. The purpose of the experiments was to compare the encoding time complexity and image quality of this algorithm with simple fractal image compression process. The results described here were obtained with a code, which is not optimal in terms of speed. Our main purpose was to show the benefits of clustering achieved in fractal image compression in terms of encoding time complexity.

SOMs find clusters within the data and organize the clusters into 1 or 2 dimensional grid such that nearby points on the grid correspond to nearby clusters.

5.2 EXPERIMENTAL RESULTS

The experiments have been aimed at finding the encoding time complexity reduction with the loss of image quality that results for clustering using SOMs, compared to the full search case.

Complexity Reduction:

For 16 x 16 bit Lena image, encoded with range block of 2 x 2 and domain block with step size of 4.The codebook contains 128 vectors and 3 clusters.

Average: 5.54 sec
Minimum: 4.50 sec
Maximum: 5.38 sec

64 x 64 bit Lena image is encoded with range block of 4 x 4 and domain block of 8 x 8 with step size of 4.The codebook contains 6728 vectors and 5 clusters.
In 10 attempts, the time consumption was following:

Average: 630.49 sec
Minimum: 620.0 sec
Maximum: 640.98 sec

Loss of Image Quality:

The effect is examined here for three test images: 16 x 16 Lena image, 64 x 64 pixel Lena image and 64 x 64 pixel Peppers image. Table 5.1 summarizes the results. PSNR, a rough

indicator of reconstructed image quality is calculated for every reconstruction. The drop in image quality is visible but not annoying. We judge this loss to be comparable to the loss experienced with the complexity reduction.

Table 5.1: PSNR ratios with and without SOM clustering

Image	Block Size	Codebook size	Clustering	PSNR (dB)
Lena 16 x 16	2 x 2	128	Full Search	35.0
			3 Clusters	22.763
Lena	4 x 4	6728	Full Search	32.8
			5 Clusters	24.55
Peppers	4 x 4	6728	Full Search	33.67
			4 Clusters	22.9

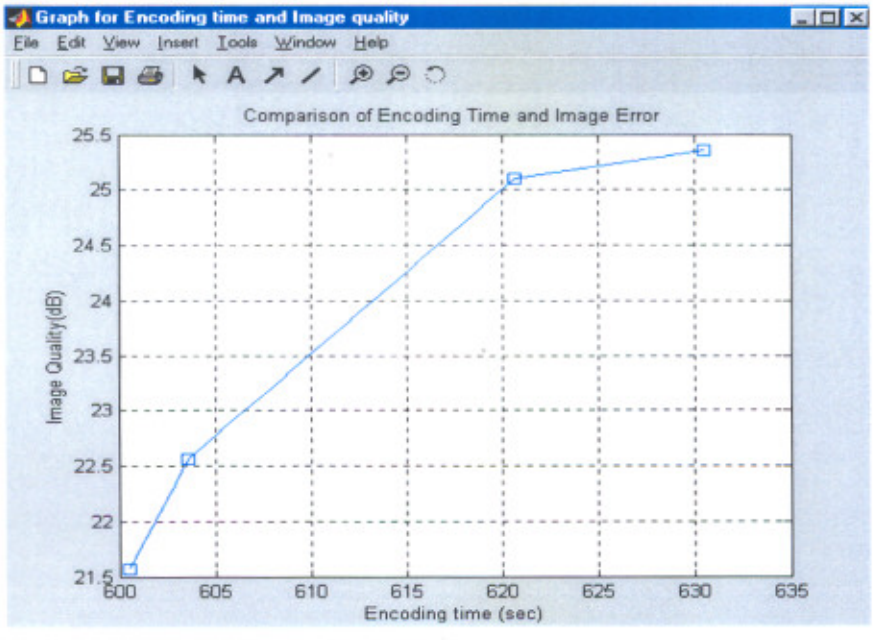


Figure 5.1: Graph for encoding time Vs image quality

While the simulations have been carried out on different images with similar results, the standard Lena image of 64 x 64 is used in the section for illustration purposes.

The above graph (Figure 5.1) shows the experimental results regarding how image quality varies with encoding time. While reducing the encoding time, image quality suffers.

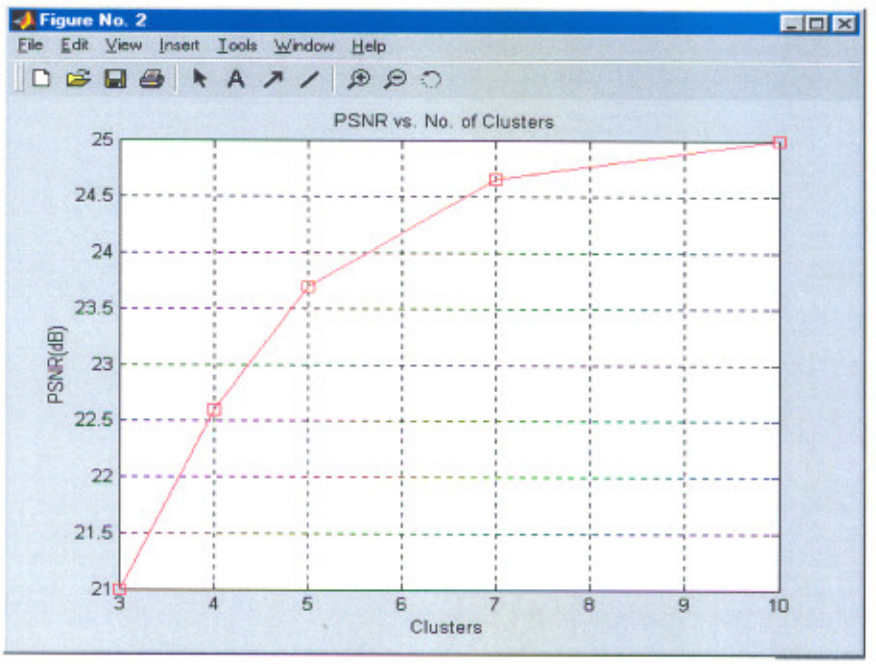


Figure 5.2: Graph for PSNR Vs No of clusters

The above graph (See Figure 5.2) shows the experimental results the relation how the PSNR value is affected with the number of clusters.

With the number of clusters, PSNR value i.e. image quality suffers, and as the image quality suffers with increasing number of clusters in the fractal image encoding process employing SOM clustering, the encoding time reduces. Thus, to speed up the encoding process, we have to make some compromise against image quality .We can adjust, to have reduced acceptable encoding speed with an acceptable image quality.

CHAPTER 6

Conclusion and Future Work

6.1 CONCLUSION

We have introduced a theory and technique for reducing the encoding time complexity in fractal image encoder. The methods uses classification of blocks in the codebook so that each range block needs only be compared to the codebook blocks in its own class. The classes are adaptive and can be tuned to the original image or to a fixed training set. We have used a neural network approach for the cluster analysis of image data. Self-organizing maps which are at present considered one of the best neural network algorithms for its efficiency, robustness and speed has been used in this research for the classification and clustering of the image data.

This technique is advantageous as it rapidly generates high quality clustering and no prior information regarding clustering is required. The image quality is affected, but this loss is weighted against the advantage of a fast encoder. Since SOM is meant for high dimensional data, so its beneficial to use SOMs for real-life color images.

To overcome the advantages of neural network algorithm, this algorithm can be combined with other classical algorithms to gain better results.

Thus our algorithm provides a faster alternative to the other simple fractal image compression schemes classified with classical clustering algorithms for clustering analysis.

6.2 FURTHER RESEARCH

While the research described in this thesis has contributed to improved fractal image compression, many significant questions remain unresolved. The most significant of these are presented here as issues particularly deserving of further research.

An extension of the evaluations described in this thesis to a wider range of fractal coding and clustering schemes is clearly desirable, and evaluations based on a significantly larger ensemble test images are required to confirm the tentative conclusions reached here.

While using neural network techniques for classification, we can use spectral bands for the classification in case of color images, which will give better clustering although research in this thesis is restricted to grayscale images.

We can combine a SOM algorithm with classical clustering algorithms to overcome the limitations of SOMs and to gain the advantage of strength of various classical techniques to derive a more effective hybrid scheme.

Additional reduction in encoding time complexity can be achieved by applying different algorithms as fractal quadtree partitioning schemes or any other efficient partitioning scheme depending on the image content if we know. Efficient searching techniques like nearest neighbor search algorithm or closest codeword search algorithm can be applied for finding the cluster centers and best candidates inside the clusters. Many efficient algorithms are available in the literature.

References

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley Publishing Company, New York, 1992.

[2] K. Sayood, *Introduction to Data Compression*, TMH, 2000.

[3] A.K. Jain, *Fundamentals of Digital Image Processing*, Englewood Cliffs, NJ: Prentice Hall, 1989.

[4] Y. Fisher, *Fractal Image Compression: SIGGRAPH 1992 Course Notes,* Computer Graphics, SIGGRAPH 1992 Proceedings, 26(4).

[5] H.R. Mahadevaswamy, *New Approaches to Image Compression,* PhD Thesis, December 2000.

[6] B. Wohlberg, G. Jager, *A Review of the Fractal Image Coding Literature*, IEEE Transactions on Image Processing, Vol.8, No.12, pp. 1716-1729, Dec. 1999.

[7] Yuval Fisher, *Fractal Image Compression: Theory and Application*. Springer-Verlag, New York, USA, 1995.

[8] A. Schuler, *An Introduction To Fractal Image Encoding.*

[9] Shinhaeng Lee, *Parallel processing architecture for fractal image compression*, PhD Thesis, Tohoku University, Japan.

[10] E.W. Jacobs, Y. Fisher and R.D. Boss, *Image Compression: A Study Of The Iterated Transform Method*, Signal Processing, Vol.29, pp 251-263, 1992.

[11] Farhad Sadaghiani, *Fractal Image Compression,* May 2001.

[12] R. Hamzaoui, *Codebook clustering by self-organizing maps for fractal image compression*, NATO ASI Conf. Fractal Image Encoding and Analysis, Trondheim, July 1995.

[13] E. Reusens, *Partitioning Complexity Issue for Iterated Function systems based on image coding*, Signal Processing, Vol.1, pp 171-174, 1994.

[14] B. Wohlberg, *Fractal Image Compression and the Self-Affinity Assumption: A Stochastic Signal Modeling Perspective*, PhD Thesis, and August 1996.

[15] B Ramamoorthi, A Gersho, *Classified Vector Quantization of images*, IEEE Vol COM-34, No. 11, November 1986.

[16] A.E. Jacquin, *Fractal Image Coding: A Review*, Proceedings of the IEEE, Vol.81, No.10, pp.1451-1465, October 1993.

[17] A.E. Jacquin, *Image Coding Based On A Fractal Theory Of Iterated Contractive Image Transformations*, IEEE Transactions on Image Processing, IP-1 (1), pp.18-30, 1992.

[18] A.E. Jacquin, *A Novel Fractal Block-Coding Technique For Digital Images*, Proceedings IEEE International conference on Acoustic, Speech, and Signal Processing, pp.2225-2228, 1990.

[19] C. Frigaard, J. Gade, T. Hemmingsen, and T. Sand. *Image compression based on a fractal theory.* Internal Report S701, Institute for Electronic Systems, Aalborg University, Aalborg, Denmark, 1994.

[20] M. Novak. *Attractor coding of images.* In Proceedings PCS'93 (International Picture Coding Symposium), page 15.6, Lausanne, Switzerland, March 1993.

[21] D. Gotting, A. Ibenthal, and R.-R. Grigat. *Fractal image coding and magnification using invariant features.* In NATO ASI on Fractal Image Encoding and Analysis, Trondheim, Norway, July 1995.

[22] M. Kawamata, M. Nagahisa, and T. Higuchi. *Multi-resolution tree search for iterated transformation theory-based coding.* In Proceedings ICIP-94 (IEEE International Conference on Image Processing), volume III, pages 137–141, Austin, TX, USA, November 1994.

[23] H. Lin and A. Venetsanopoulos, *A pyramid algorithm for fast fractal image compression.* In Proceedings ICIP-95 (IEEE International Conference on Image Processing),Vol III, pages 596–599, Washington, D.C., USA, October 1995.

[24] S. Lepsoy, *Attractor Image Compression - Fast Algorithms and Comparisons to Related Techniques.* PhD thesis, The Norwegian Institute of Technology, Trondheim, Norway,June 1993.

[25] S. Lepsoy and G. E.Oien. Fast attractor image encoding by adaptive codebook clustering. In Y. Fisher, editor, *Fractal Image Compression: Theory and Application*, chapter 9, pages 177–197. Springer-Verlag, New York, NY, USA, 1995.

[26] F. Davoine, M. Antonini, J.-M. Chassery, and M. Barlaud. *Fractal image compression based on Delaunay triangulation and vector quantization*. IEEE Transactions on Image Processing, 5(2):338–346, February 1996.

[27] Gordon W. Paynter, *Fractal Image Compression*, October 1995.

[28] D. Saupe, M. Ruhl, *Evolutionary Fractal Image Compression*, Proceedings of IEEE International Conference on Image Processing, ICIP'96. Vol. I, pp. 129-132, Switzerland, September 1996.

[29] D. Saupe, R. Hamzaoui, *A Guided Tour of the Fractal Image Compression Literature*, July 1999.

[30] C.J. Wein, I. F Blake, *On the performance of Fractal compression with clustering*, IEEE Transactions on Image Processing, Vol 5 No. 3, March 1996.

[31] J. Han, M. Kamber, *Data Mining – Concepts and Techniques*, 2001.

[32] V. Faber, *Clustering and the Continuous k – means Algorithm*, Los Alamos Science No. 22, 1994.

[33] D Lee, S. Baek, K Sung, *Modified K – means Algorithm for Vector Quantizer Design*, IEEE Signal Processing Letters, Vol 4 No. 1 January 1997.

[34] T. Kohonen, *Self-Organizing Maps*, Springer, Berlin, Heidelberg, 1995.

[35] J. Vesanto and E. Alhoniemi, *Clustering of the Self-Organizing Maps*, IEEE Transaction on Neural Networks, 2003.

[36] C. Amerijekx, J.D Legat, *Image Compression using Self Organizing Maps*, System Analysis and Modeling Simulation, Vol 43,No. 11, November 2003.

[37] J. Vesanto, J. Himberg, E. Alhoniemi and J. Parhankangas, *Self-organizing map in Matlab: the SOM Toolbox*, In Proceedings of the Matlab, DSP Conference 1999, Espoo, Finland, pp. 35-40, 1999.

[38] R.M. Gray, *Quantization and Compression Topic 4: Clustering*, Stanford University 2003.

[39] Ludwig Schwardt, *Clustering*, University of Stellenbosch, March 2003.

[40] A. Flexer, *On the use of Self organizing maps for clustering and visualization.*

[41] Juha Vesanto, Johan Himberg, Esa Alhoniemi and Juha Parhankangas, *Self-Organizing Map in Matlab: the SOM Toolbox*, Laboratory of Computer and Information Science Helsinki University of Finland, February 18, 2000.

[42] T. Kohonen, *Chapter 13 Software Tools for Self-Organizing Maps Guido Deboeck*, Neural Network Research Center.

[43] S. Nakamura, Numerical Analysis and Graphical Visualization with Matlab, Prentice Hall PTR, NJ, 1996.

[44] http://www.cis.hut.fi/projects/somtoolbox -*Self-Organizing Map for Data Mining in MATLAB: the SOM Toolbox.*

Appendix A

ORIGINAL TEST IMAGES

Figure A.1: 64 x 64 Lena.

Figure A.2: 64 x 64 Peppers.

Figure A.3: 64 x 64 Eye.

Figure A.4: 64 x 64 Boat.

Figure A.5: 64 x 64 Bird.

Figure A.6: 64 x 64 Mandrill.

Papers Communicated/Accepted/Published

The following papers have been communicated/accepted/published so far:

1. Lucky Jindal, Maninder Singh, "A Neural Network Approach for the Improvement in Encoding Time in Fractal Image Compression", The 4th IASTED International Conference on VISUALIZATION, IMAGING, AND IMAGE PROCESSING, VIIP, 2004, September 6-8, 2004. (**Communicated**)

2. Lucky Jindal, Maninder Singh, "4th Indian Conference on Computer Vision, Graphics and Image Processing", December 16-18, 2004, Kolkata. **(Communicated)**