A
Dissertation
On

Fractal Image Compression

Submitted in Partial fulfillment of the requirements
for the award of the degree of
MASTER OF ENGINEERING
in
(Computer Technology & Application)

Submitted By:
TARIQUE ANWAR
College Roll No: 14/CTA/08
University Roll No. 8412

Under the Guidance of:
Mr. Vinod Kumar
Dept. of Computer Engineering
Delhi College of Engineering, Delhi

DEPARTMENT OF COMPUTER ENGINEERING
DELHI COLLEGE OF ENGINEERING
DELHI UNIVERSITY
2010

CERTIFICATE

DELHI COLLEGE OF ENGINEERING
DELHI UNIVERSITY
2009-2010

This is to certify that the work contained in this dissertation entitled " Fractal Image Compression" , submitted by Tarique Anwar**,** University Roll No-8412 in the requirement for the partial fulfillment for the major project in Master of Engineering in Computer Technology & Application, Delhi College of Engineering is an account of his work carried out under my guidance and supervision in the academic year 2009-2010.

Mr.Vinod Kumar
Assistant Professor
Dept. Of Computer Engineering
Delhi College of Engineering, Delhi

ACKNOWLEDGEMENT

It is a great pleasure to have the opportunity to extend my heartiest felt gratitude to everybody who helped me throughout the course of this project.

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my learned supervisor **Mr. Vinod Kumar** for his invaluable guidance, encouragement and patient reviews. With his continuous inspiration only, it becomes possible to complete this dissertation.

I would also like to take this opportunity to present my sincere regards to all the faculty members of the Department for their support and encouragement.

I am grateful to my parents for their moral support all the time, they have been always around to cheer me up, in the odd times of this work. I am also thankful to my classmates for their unconditional support and motivation during this work. Last but not the least, special thanks to the people who are active in the field of Fractal Image Compression.

TARIQUE ANWAR
Master in Engineering
(Computer Technology & Application)
College Roll No. 14/CTA/08
University Roll No. 8412
Department of Computer Engineering
Delhi College of Engineering, Bawana
Road, Delhi-110042

ABSTRACT

In context of medical imaging or satellite images, the change in conditions is to be observed at different times. There is a fact that the images captured of the same scene or the same object at different times may not have the same orientation. For the sake of automating the observation of the progress or the change in the condition of the object, there needs to be a technique to align the input image to the reference image. The technique used for the above process is known as image registration. The work under this project is divided in two phases. The first phase is about the image registration of the medical images. It is implemented using MATLAB. Second phase of the project deals with the compression of the images. The method of image compression chosen is Fractal image compression. Since the data in medical images are very crucial, It is not appreciable to loose data in compression-decompression process. As in fractal image compression, the decoding process involves the iteration on the image. If we increase the number of iterations, more accurate picture we can get back. The above mentioned fact is the reason to choose fractal image compression for the purpose of image compression. We realize the importance of compression methods in our daily life when we store files in a limited storage space or when we have to send a file on a slower network. In this thesis, the methods of compression of images is dealt with. Again, there are different approaches for image compression among which jpeg is a well known one. We judge the ability of a compression method by the compression ratio it provides. It is noted that for the images having fractal properties in terms of self similarity or the images having similar regions, the image compression method known as *Fractal Image Compression* can give better compression ratio.

The fractal image compression algorithms have a common approach which involves the partitioning of the image into smaller non overlapping square subsections range blocks of

predefined size. Then, a search codebook (domain pool) is created from the image taking all the square blocks (domain blocks) of size double of the range blocks and ultimately for each each range block, the most appropriate domain block is selected from the domain pool. It is noted that that what transformations are required to be performed on the range block to match with the domain block.

Table of Contents Page no.

List of Figures……...Page

CHAPTER 1

INTRODUCTION TO FRACTALS

1.1 Overview

A **fractal** is "a rough or fragmented geometric shape that can be split into parts, each of which is (at least approximately) a reduced-size copy of the whole, a property called self-similarity. The term *fractal* was coined by Benoit Mandelbrot in 1975 and was derived from the Latin *fractus* meaning "broken" or "fractured." A mathematical fractal is based on an equation that undergoes iteration, a form of feedback based on recursion.

A fractal often has the following features:

- It has a fine structure at arbitrarily small scales.
- It is too irregular to be easily described in traditional Euclidean geometric language.
- It is self-similar (at least approximately or stochastically).
- It has a Hausdorff dimension which is greater than its topological dimension
- It has a simple and recursive definition.

Because they appear similar at all levels of magnification, fractals are often considered to be infinitely complex (in informal terms). Natural objects that are approximated by fractals to a degree include clouds, mountain ranges, lightning bolts, coastlines, snow flakes, various vegetables (cauliflower and broccoli), and animal coloration patterns. However, not all self-similar objects are fractals—for example, the real line (a straight Euclidean line) is formally self-similar but fails to have other fractal characteristics; for instance, it is regular enough to be described in Euclidean terms.

1.2 History

The mathematics behind fractals began to take shape in the 17th century when mathematician and philosopher Gottfried Leibniz considered recursive self-similarity. Waclaw Sierpinski constructed his triangle in 1915 and, one year later, his carpet. Originally these geometric fractals were described as curves rather than the 2D shapes that they are known as in their modern constructions. The idea of self-similar curves was taken further by Paul Pierre Levy, who, in his 1938 paper *Plane or Space Curves and Surfaces Consisting of Parts Similar to the Whole* described a new fractal curve, the Levy C curve. Georg Cantor also gave examples of subsets of the real line with unusual properties—these Cantor sets are also now recognized as fractals.

Iterated functions in the complex plane were investigated in the late 19th and early 20th centuries by Henri Poincaré, Felix Klein, Pierre Fatou and Gaston Julia. Without the aid of modern computer graphics, however, they lacked the means to visualize the beauty of many of the objects that they had discovered.

In the 1960s, Benoît Mandelbrot started investigating self-similarity in papers such as *How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension*, which built on earlier work by Lewis Fry Richardson. Finally, in 1975 Mandelbrot coined the word "fractal" to denote an object whose Hausdorff–Besicovitch dimension is greater than its topological dimension. He illustrated this mathematical definition with striking computer-constructed visualizations. These images captured the popular imagination; many of them were based on recursion, leading to the popular meaning of the term "fractal".

1.3 Examples

A class of examples is given by the Cantor sets, Sierpinski triangle and carpet, Menger sponge, dragon curve, space-filling curve, and Koch curve. Additional examples of fractals include the Lyapunov fractal and the limit sets of Kleinian groups. Fractals can be deterministic (all the above) or stochastic (that is, non-deterministic).

1.4 In nature

Approximate fractals are easily found in nature. These objects display self-similar structure over an extended, but finite, scale range. Examples include clouds, snow flakes, crystals, mountain ranges, lightning, river networks, cauliflower or broccoli, and systems of blood vessels and pulmonary vessels. Coastlines may be loosely considered fractal in nature.

Trees and ferns are fractal in nature and can be modeled on a computer by using a recursive algorithm. This recursive nature is obvious in these examples—a branch from a tree or a frond from a fern is a miniature replica of the whole: not identical, but similar in nature. The connection between fractals and leaves are currently being used to determine how much carbon is contained in trees.

1.5 In creative works

Fractal patterns have been found in the paintings of American artist Jackson Pollock. While Pollock's paintings appear to be composed of chaotic dripping and splattering, computer analysis has found fractal patterns in his work.

Decalcomania, a technique used by artists such as Max Ernst, can produce fractal-like patterns. It involves pressing paint between two surfaces and pulling them apart.

Fractals are also prevalent in African art and architecture. Circular houses appear in circles of circles, rectangular houses in rectangles of rectangles, and so on. Such scaling patterns can also be found in African textiles, sculpture, and even cornrow hairstyles.

In a 1996 interview David Foster Wallace admitted that the structure of his novel Infinite Jest was inspired by fractals, specifically the Sierpinski triangle.

The song "Hilarious Movie of the 90's" from Pause (album) by the artist Four Tet employs the use of fractals.

CHAPTER 2

IMAGE COMPRESSION

2.1 An Introduction

How does image compression work? It is applied to obtain an image representation while reducing the amount of memory needed as much as possible to encode the image. Image compression is possible because images, in general, are highly non-random, which means that there is repetitive information. Image data, like other meaningful data, are usually structured, and this structure means that the data over different parts of an image are interrelated. For example, consider an image in matrix format, if we take an arbitrary pixel, its gray-level or color will likely be of similar value to that of the neighboring pixels, since they are more likely than not to belong to the same object. If the gray levels or colors are not similar, some more complex relationship may apply: for instance, the pixel might be on the boundary of two objects, or it may be part of a texture pattern. In any case, there are usually some redundant or less informative data because of the image's structure. So image compression is basically a way to capture the image structure and generalize this image structure in coherent and usable form. Compression methods try to eliminate repetitiveness, thus producing a more compact code that preserves the essential and accurate information contained in the original image. Because images require large amounts of data, storing and transmitting this data places a significant load on the computer systems and data transmission facilities used. Compression of data reduces the cost of image storage by increasing the effectiveness of storage resources and increases the effective speed of transmission without broad banding.

2.2 General Concepts

Image compression maps an original image raster into a bit stream suitable for communication over or storage in a digital medium so that the number of bits required to represent the coded image is smaller than that required for the original image. Ideally people would like the coded image to require as few bits as possible so as to minimize the storage space or communication time. They may also require in some applications that the original image be perfectly recoverable from the coded form. If the original image is an analog picture, it is improbable that the digitized information will be exactly the same as the analog original, no matter how many bits are used. However, digital to analog exists with great accuracy and therefore it is possible to convert analog to digital, but the instances of perfect conversion are not common. But people may face a series of issues. For example, the efficiency of compression algorithms, including data compressing rate, the resulting distortion, and its implementation complexity, is a particularly important considerations in hardware implementation and applications.

The basic goal of image compression is the conversion of an original sampled continuous or high bit-rate image into a compressed image with a binary coding having a specific amount of bits per pixel (bpp) so that the decompressed image has the best possible fidelity. The image decompression, or restoration, is the inverse procedure of compression: the compressed images are converted back to the original one, or the best approximation of the original images.

Before using a particular compression technique, it is important to know which category the image compression technique belongs to. There are two broad categories of image compression techniques. The first category consists of methods, which completely preserve the original data. When the compressed image is converted back into its uncompressed form, it is identical with the original image. This kind of technique is called "lossless" compression. For this kind of compression to be effective, there must be some redundancy in the original data. The second category of compression technique consists of methods that only approximate the original data. This category of compression is called "lossy" compression. In general, the less accuracy needed of the resulting image, the greater the compression rate.

Given these two categories of techniques, it is obviously important to know, for any given application, how much degradation in image quality can be tolerated. It is possible to apply objective measures of quality. For example, the original image can be compared with the result of compression followed by decompression. The differences can be expressed as a kind of signal to noise ratio, which can be used as a performance measure of the compression technique.

2.3 Image Compression Methods[8]

The basic idea of image compression is to remove redundancy from the image. This is usually done through mapping the image to a set of coefficients. The resulting set is then quantized to a number of possible values that are encoded by an appropriate coding method.

Differential pulse code modulation (DPCM) is a predictive coding scheme. The basic DPCM system consists of two main components: the predictor and the quantizer. The predictor uses the correlation between pixels to derive an estimate $g^*(i, j)$ for a given pixel value g(i, j) in terms of its neighboring pixels. The predicted value is removed from the actual value at the transmitter. The quantizer discretizes inputs at a specified interval, the prediction residual $h(i, j) = g(i, j) - g^*(i, j)$ to one of a finite number of values. The value is encoded and transmitted to the receiver. After decoding the transmitted codes, the receiver reconstructs the pixel value by adding the predicted value to the quantized prediction error. Therefore, the success of a DPCM compression scheme depends on effective prediction of the current pixel value and efficient quantization of the prediction residual. In the algorithm, the image is divided into non-overlapping 8x8 neighborhoods starting from the upper left corner of the image. Each neighborhood is composed of 4 levels. The upper left corner pixel is a level 1 pixel, which is used to predict values of the remaining pixels through either linear or bilinear interpolation. The advantage of this method is "time-saving", but the disadvantage is that it is not stable. If the adjacent pixels have similar digital numbers, the result is good; if the values of pixels are much different, the result is not good.

The other type of image compression method is based on the discrete cosine transform (DCT). The JPEG algorithm proposed by the Joint Photographic Experts Group (JPEG) contains four modes of operations: sequential encoding; progressive encoding; lossless encoding; and hierarchical encoding, where the sequential and progressive encoding methods are based on the DCT while the lossless mode is based on a predictive method. The hierarchical mode encodes the image at multiple spatial resolutions using either the DCT-based compression or the lossless mode. The encoding methods of JPEG contain three sequential steps: forward discrete cosine transform (FDCT), quantization, and Huffman coding. The processing scheme is applied to a stream of 8 by 8 pixel blocks with 8 bits per pixel. Decompression is achieved by following the processing steps in the opposite direction: Huffman decoding, dequantization, and inverse discrete cosine transform (IDCT). The disadvantages of these methods include: decompression of large digital images is time- consuming; also some geometric degradation effect may occur with higher compression rates. Figure 2.1 shows the JPEG compression scheme.

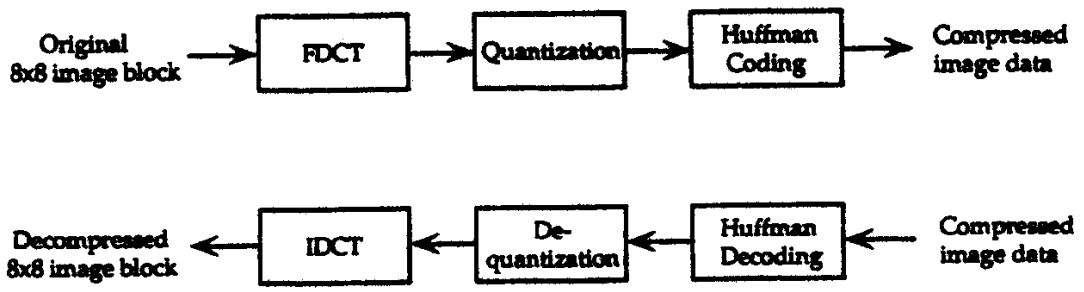


Figure 2.1 Baseline JPEG Compression Scheme (Source: Lammi and Sarjakoski 1996)

CHAPTER 3

THE FRACTAL MODEL

"Fractal" is a new geometry proposed by French Mathematician Mandelbrot in the middle of the 1970s, which is really a revolution in topological space theory and provides the possibility of describing and simulating objects and phenomena precisely in the natural world by using a series of new concepts and mathematical models.

Geometry is a branch of mathematics that deals with the shape of objects. But people often describe geometry as uninteresting or monotonous. One of the reasons is that the traditional or classical geometries based on the Euclidean system cannot accurately describe natural objects, such as clouds, mountains, coastlines, etc., because "clouds are not a sphere, mountains are not cones and coastlines are not circles" (Mandelbrot, 1983). The common feature of the shapes of most objects in the natural world is that they are neither regular, nor smooth. All the traditional geometries deal with regular and smooth geometrical shapes. In fact, the classical geometries, which are thought as "strict and accurate", are just inaccurate descriptions of the objects in the natural world; for example, the surface of the earth is always treated as an absolutely smooth spherical surface, or, ellipsoid in an "ideal situation". The value of Euclidean Geometry is dependent upon its use. In many instances it describes reality with great clarity. However, along with the progress of both science and technology, and mankind's knowledge about the world, people have found that Euclidean Geometry is not always useful. A geometry that accurately reflected niche realities was necessary. Therefore, when Fractal Geometry was introduced, it drew many scientists' attention. This geometry found real world applications as soon as it

appeared.

12

3.1 Development of the Theory

During a period of history, some people thought that a curve should be smooth along its entire path, or, at least smooth in recognized "segments", i.e. with few exceptions, every point has its tangent. Later, Italian Mathematician Peano offered a curve which can fill an entire square but is undifferentiable everywhere. Subsequently, people found there are many such curves, now known as "Peano Curve" (Figure 3.1). As it is shown, in classical geometry, a point's dimension is zero; a line's dimension is one; while a plane's dimension is two, and a solid object's dimension is three, suggesting length, width, and height are three linearly independent directions. All dimensions in conventional geometry are integers. In other words, in classical geometry, all curves have a topological dimension D_T=1. Obviously the topological dimension D_T cannot reflect the features of fractal curves. So in fractal geometry, a new concept: *fractal dimension*, denoted as *D*, is used as the parameter to describe the feature of the curves:

$$D = \log N / \log (1/r)$$

Where *D* is the fractal dimension; *N* is the number of segments with same length composing the curve and *1/r* is the ratio of the distance between the two ends of the curve and the length of the segment. The *D* of fractal curves is greater than 1, smaller than or equal to 2 (Figure 3.2). This is the feature of fractal curves, which is different from the general curves. Also, based on the same principle, there are fractal surfaces with $2 < D$

≤ 3 and fractal solid objects with $3 < D \leq 4$. So it can be shown that a fractal dimension is a measurement of the complexity of curves or surfaces: the more complex the curves or the surfaces, the greater the dimension of objects based on its feature.

Central to the concept of fractals is the notion of self-similarity. Self-similarity means that for any curve or surface a portion of the curve or surface can be considered as a reduced image of the whole. In other words, the curves or surfaces are self similar to the object and makes up copies of itself in a reduced scale . In the natural world, however, strict self-similarity seldom occurs. Therefore, the concept of statistical self-similarity is often applied.

3.2 Development in Applications

Since fractal theory was introduced, it has been applied to many fields of study, ranging from physics to music. In computer graphics, fractal theory is applied in many areas; generating complex curves or surfaces or simulating objects in the natural world is an illustration. In recent years, people find that not only is it useful in describing geometrical shapes, but also in describing many natural phenomena: "Fractal Everywhere" (Barnsley 1988). The applications of the fractal are briefly described below.

As mentioned above, the fractal's importance originates because of its accurate description of certain realities, including the measures of some geographical phenomena, such as the length of a coastline. In 1967 Mandelbrot proposed the new concept used to capture and analyze the new information that traditional science could not provide. He asked, "How long is the British coastline?" and advanced the way to measure it by using the fractal concept. After the new geometry appeared, it has been attracting more and more attention by scientists. In other areas, other scientists made effort to use fractal in computer graphics.

Batty (1985) showed a number of examples of simulated landscape, mountainscape and other graphics generated by using the property of "self-similarity" of fractals and pointed out the potential of development of applying fractal in this area.

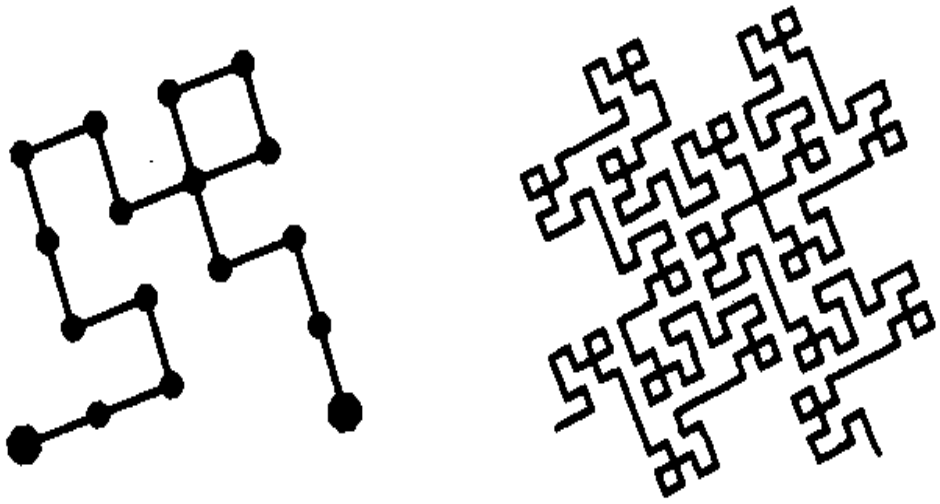
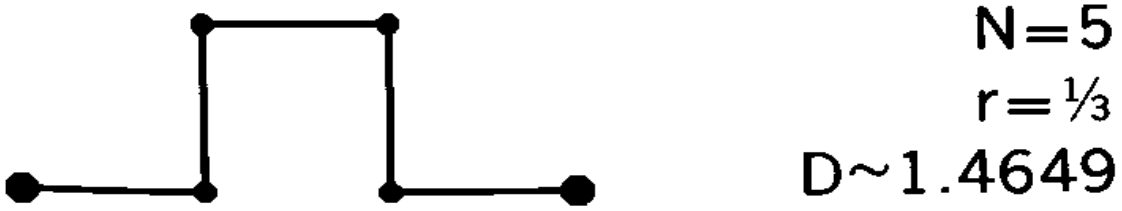


Figure 3.1 Peano Curve
(Source: Mandelbrot 1983)



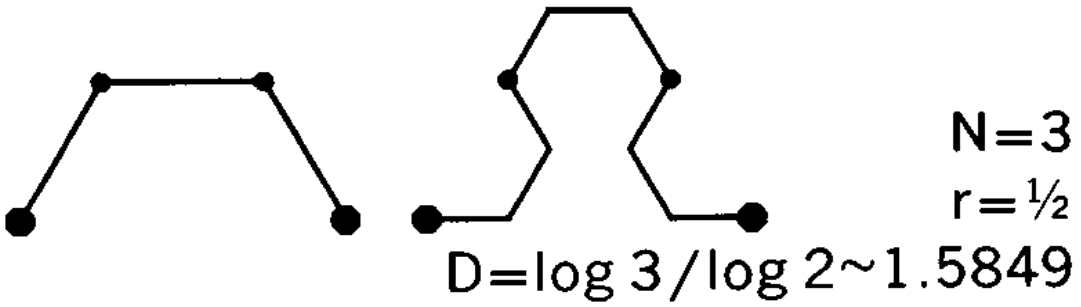


Figure 3.2 Fractal Dimensions
(Source: Mandelbrot 1983)
15

3.3 Considerations

Fractal, the new science and technique, has been developed greatly since it was first advanced. But it is still new and also immature in both theoretical and practical areas. So it can be expected that more development would be obtained in the future.

- Image encoding and compressing is a very significant branch of image processing.

  The research is to organize the images and make them more compact by using the characteristics of images as information source and information carrier in order to extract useful information and store the images effectively, and when needed, restore them without losing useful information. So far there have been encoding and compressing methods available, such as raster encoding, run-length encoding etc. But, the problem is that the methods are based on *the format* of the image, rather than *the individual features* of the images, so it is difficult to reflect the characteristics of the image accurately. This may interfere with the extraction of useful information and the effect of encoding and compressing. The other problem is that the compression rates of current methods for compressing are usually low. Generally speaking, the best result is to reduce the image size to 25 percent of the original image size. So in most cases,

it may not solve the existing problem of storage.

As mentioned above, the fractal has received enormous development since it arose in the 1970s. Then again, the fractal has only about twenty years of history. It is a developing theory and technique; in other words, it needs to be developed in both theory and application.

16

It seems that Mandelbrot himself was involved in a paradox: he was trying to make a new model to replace the old one but what he advanced still deals with some other ideal situation which rarely, or even does not exist in the real world — pure fractal and absolute self-similarity. This may be the limitation, which is from the mathematician's points of view: to seek something perfect, ideal, and strict to meet the requirement of mathematics. But the real world is not exactly as the mathematicians imagine, it is chaotic rather than ordered. Mandelbrot made some modifications to fit his theory into the physical world. Undoubtedly the rise of fractal geometry is a breaking point in this scientific period, but it is not perfect. Issues demand attention and techniques that more clearly define niche reality in terms of fidelity and utility need to be developed. At this point, the cooperation between mathematicians and scientists in practical areas is urgently needed. The former can provide the latter with new theories or new developments in existing theories. The latter can provide the former with experimental data and the demands of practical use. With cooperation, more development of both fractal theory and application can be expected. Using the concepts of

fractals in the practical application of image compression is a needed area of research.

CHAPTER 4

FRACTAL IMAGE COMPRESSION

Image compression is a very significant and necessary branch of image processing. The subsequent goal of image compression is to organize the images so they are more compact by using the characteristics of images as an information source and an information carrier. By organizing the images as effectively and compactly as achievable, useful information can be accurately extracted and the images stored effectively. When needed, these images can without difficulty be restored and they will remain accurate and will not lose functional or constructive information.

To this point, there have been some less effective encoding and compressing methods available. But, the former methods generate in their own fashion a difficulty that interferes with efficiency of use and storage. The associated difficulty arising from dated methods of compression and analysis is that the compression rates of existing methods for compressing images are usually lower than tolerable.

4.1 Attractor and Self-similarity

Fractals have provided a different description of objects, which is completely different from traditional concepts. Fractal theory concentrates on the "self-similarity" of objects, so it can theoretically be used to describe the inside features of different images rather than something on the surface. Therefore the route of compression by means of the fractal is different from traditional compression methods. An intuitive example would be a photocopying machine (Fisher 1992). Imagine a copying machine that reduces the image to be copied by one half and reproduces it three times on the copy tray (Figure 4.1). When we feed the output of this machine back as input, we will find that all the copies seem to be converging to the same final image after several iterations of this process on several input images (Figure 4.2).This final image is called the *attractor* for this copying machine. Because the copying machine reduces the input image, any initial image will be reduced to a point as we repeatedly run the machine.

Thus no matter what initial image is placed on the copying machine, it will not affect the final attractor. In fact, only the position and the orientation of the copies determine what the final image will look like.

From the copies, it can be seen that each copy is formed of three reduced copies of itself. A common feature of these reduced copies and all attractors formed this way is that in the position of each of the images of the original square there is a transformed copy of the whole image. So each image is formed from transformed (and reduced copies) of itself and hence it has the same detail at every scale. This is "*self-similarity*", the unique and important feature of fractal.

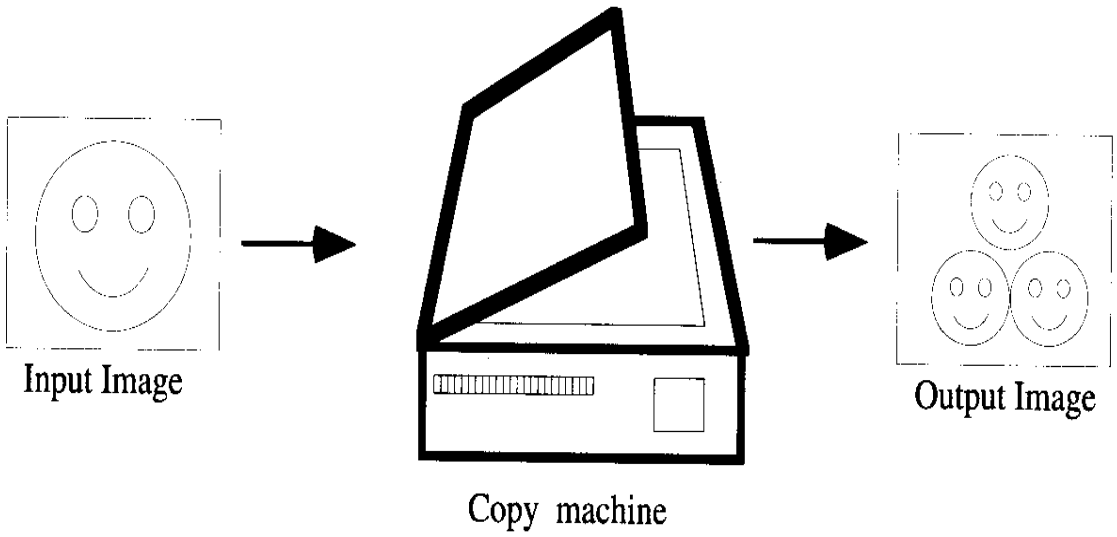


Figure 4.1 Copy Machine
(Fisher 1992)

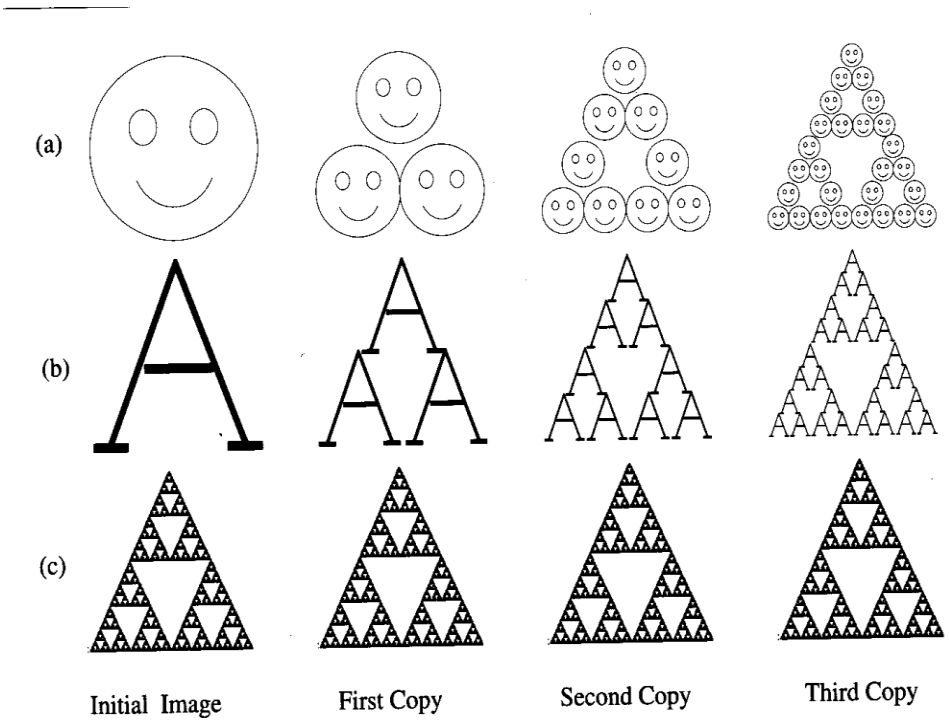


Figure 4.2 Different Initial Images with Similar Final Images
(Fisher 1992)

4.2 Contractive Transformations

Since the way the input image is transformed determines the final result of running the copy machine in a feedback loop, the transformation will be described. Different transformations will lead to different attractors (final images), provided that the transformations must be contractive, that is, given a transformation W, any two points P_1, P_2, in the input image must be closer in the copy. In other words, the distance between the two points:

$$d(W(P_1), W(P_2)) < s\, d(P_1, P_2) \qquad (4\text{-}1)$$

for $s < 1$.

In the case of a plane, if the points have coordinates $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, then

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (4\text{-}2)$$

This condition is natural and obvious, because if the transformation is not contractive, points in the copy will be spread out. It follows that the final image will be of infinite size. Except for this condition, the transformations can have any form. In practice, the transformation can be affine, which will be sufficient to yield an interesting set of attractors.Contractive transformations have the pleasing property such that when they are repeatedly applied, they converge to a point. This point remains fixed when further iterations are applied.

4.3 Iterated Function Systems (IFS)

An *iterated function system* (IFS) consists of a collection of *contractive transformations* $\{w_i: R^2 \longrightarrow R^2 \mid i = 1, 2, \ldots, n\}$ which map the plane R^2 into itself. This collection of transformations defines a map:

$$W(\cdot) = \bigcup_{i=1}^{n} w_i(\cdot) \qquad (4\text{-}3)$$

where $W(\cdot)$ denotes the map, which consists of a set of transformations $w_i(\cdot)$; $(\cdot)$ is a group of points in the plane R^2. The map W is not applied to the plane, but as an alternative, it is applied to sets, collections of points in the plane. The special copy machine mentioned above, which is running in a feedback loop, is a good metaphor for IFS: Given an input set S, we can compute $w_i(S)$ for each i (corresponding to making a reduced copy of the input image S), and take the union of these sets (corresponding to assembling the reduced copies). Then W is a map on the space of subsets of the plane, in other words, a map on the space of

images, not on the whole plane R^2 (Fisher 1992).

If a contractive map W on a space of image exists, then there is a special image, called the *attractor*, denoted as x_w, with the properties:

1. If the copy machine is applied to the attractor, the output is equal to the input. The image is fixed, and the attractor x_w is called the *fixed point* of W. Then we have:

$$W(x_w) = x_w = w_1(x_w) \cup w_2(x_w) \cup \ldots \cup w_n(x_w) \quad (4\text{-}4)$$

2. Given an input image S_0, we can run the copy machine once to get $S_1 = W(S_0)$; twice to get $S_2 = W(S_1) = W(W(S_0)) \equiv W^{o2}$, and so on. The superscript "o" indicates that we are using iterations, not exponents; for that reason, W^{o2} is the output of the second iteration. The attractor, which is the result of running the copy machine in a feedback loop, is the limit set

$$x_w \equiv S_\infty = \lim_{n\to\infty} W^{on}(S_0) \quad (4\text{-}5)$$

which is not dependent on the choice of S_0.

3. x_w is unique. If we find any set S and an image transformation (map) W satisfying $W(S) = S$, then S is the attractor of W; that is, $S = x_w$. It means that only one set will satisfy the fixed-point equation in 1 above.

4.4 Partitioned Iteration Function System

Theoretically, each image has a unique fixed point. But in practice, it is impossible to find a unique fixed point for a whole image. Thus, the image should be partitioned as different parts, and the fixed points for the corresponding parts should be obtained through different transformations (or maps). The above IFS will be different from different parts of the image, and therefore it is called a partitioned iteration function system (PIFS). The copy machine metaphor can be extended to describe the process. Besides the copy machine's features described above, which include:

- The number of copies of the original pasted together to form the output;
- A setting of position and scaling, stretching, skewing, and rotation factors for each copy.

The extended one has more:

- contrast and brightness adjustment for each copy;
- mask that selects, for each copy, a part of the original to be copied.

The new features are applied to allow the transformation of gray-scale images. In particular, the final listed feature allows partitions of an image into pieces which are each transformed separately.

What happens when we copy an original image using this machine? A portion of the original, which is denoted as D_i, is copied to a part of the produced copy, denoted as R_i. The D_i and R_i are called *Domains* and *Ranges* respectively. The transformation between D_i and R_i is denoted as W_i. As before, the copy machine runs in a feedback loop: its own output is fed back as its new input repeatedly. The extended copy machine can be called a partitioned copy machine. The mathematical analogue of a partitioned copy machine is a *partitioned*

iterated function system.

Using affine transformation on the gray scale images, the gray level adds another dimension, so the transformation w_i will take the format like :

$$w_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix} \quad (4\text{-}6)$$

where *wi* is the transformation; *x* and *y* are the coordinate of a point in the image; *ai*, *bi*, *ci*, *di*, *ei*, and *fi* are the coefficients of the transformation, s_i controls the contrast and o_i controls the brightness of the transformation.

4.5 General Procedure

The image compression procedure is in fact an image encoding process while the reconstructing or restoring procedure corresponds to image decoding. Let (*M*, *d*) denote a metric space of digital images, where *d* is the distance measure (which is invariant to translation and rotation) in the given space. Let *U* be an original image that we want to encode or compress. We need to find a contractive image transformation *T*, defined from the space (*M*, *d*) to itself, for which *U* is an approximate fixed point. So, there exists a number $s < 1$ such that for any *u*, *v* belonging to (*M*, *d*), we have

$$d(T(u),T(v)) \leq s\,d(u,v) \quad (4\text{-}7)$$

and

$$d(U, T(U)) \approx 0 \quad (4\text{-}8)$$

The scalar *s* is called the contractivity of the transformation *T*. So if a suitable *T* is chosen, it can be expected that the original image will be reconstructed as close approximation through a series of *T* transformations from an initial image, which could be much simpler than the original image itself. In other words, after a series of *T*

transformations, the initial image will converge to the original one.

Reconstruction can be accomplished by partitioning the original image in a suitable manner that will obtain the suitable transformation *T* to meet the requirement of compression rate and computational issues. So the task of compressing an image includes three important parts:

1) Partition the image and find transformations for each partitioned part;
2) Encoding (compressing) the image; and
3) Decoding (decompressing) the image.

4.6 Quadtree Method

Under quadtree method, the partitioning of image is discussed followed by the encoding and decoding algorithms.

4.6.1 Quadtree Partition

Quadtree is an image structure, which appeared at the end of the 1970s, and was developed and applied in the 1980s and the 1990s. In the first case, Klinger and Dyer (1976) advanced the concept of evenly decomposing in order to build a representation of complex image data designed for computer searching, and defined the corresponding method of decomposition. The basic idea is to decompose images by region, rather than by rows or columns, with the intention that the structural information of images can be better reflected. One of the advantages of decomposing the image is: if an image is too big to be loaded into the available computer memory , this loaded image can be divided into sub-

images to process without breaking the structural information of the image.

If we decompose an image into quadrants continuously, the result will be a quadtree. To get a quadtree of an image, the procedure of evenly decomposing the image is: Suppose we have a 2^n x 2^n binary image in which points with value "1" represent the "feature" points as black ones and points with value "0" represent background points as white ones. First, the whole image is served as the root node. If the node does not consist of all "1" value or all "0" value points, it is called a gray node and needs further decomposition. As the first step, divide the whole image into four 2^{n-1} x 2^{n-1} sub-images, then decide if further decomposition is needed. The sub-images are not only son nodes of the whole image but also the root nodes of its own. If one node consists of all "1" value points or all "0" value points, then stop decomposing the node; if the node has both "1" value points and "0" value points, then decompose it again until all sub-images consist of points with the same values. (Figure 4.3).

In the very beginning, the quadtree took the format of the chain-structure and hierarchical style. Each node has six fields in which only one is with the value of the node and the other five are all chain fields pointing to the father node and the four son nodes of the current node. Also, since there are a lot of gray nodes as intermediate nodes in a quadtree, only ¾ nodes in total are black or white. In fact, only those points are meaningful, more specifically, only black nodes are requisite for processing. Obviously, this kind of hierarchical quadtree occupies huge amounts of storage. Albeit, the structure is clear and simple: one fourth of the storage space is used to accumulate the intermediate nodes; and in each node, there are five out of six fields being used to store the pointers, and only one field to store the value of the node. The redundancy is very large; subsequently, the efficiency is

very low. At this stage, the quadtree could not be put into practice mainly because of these disadvantages.

Gargantini (1982) advanced a new structure of quadtrees, called the linear quadtree (Figure 4.4). The differences of Gargantini's linear quadtree and the previous quadtrees are:

1) It only stores black nodes;
2) It codes each node;
3) The codes of nodes imply the path from root to nodes; and
4) Each region on an image can be represented as an ordered series of nodes.

The main advantages of linear quadtrees include:

1) The storage and processing time depend only on the amount of black nodes;
2) They remove the chain structure and a large amount of redundancy.

Based on Gargantini's work, Lauzon et al. (1985) introduced one coding method for linear quadtrees, called "Two Dimensional Run-Encoding", simply "2DRE". It takes advantage of the "Morton series" so that the codes of the quadtree structures become more compact.

At this stage, the quadtree structure became practical, applicable, and operational. Then the quadtree structure began to be used in image processing and image encoding areas.

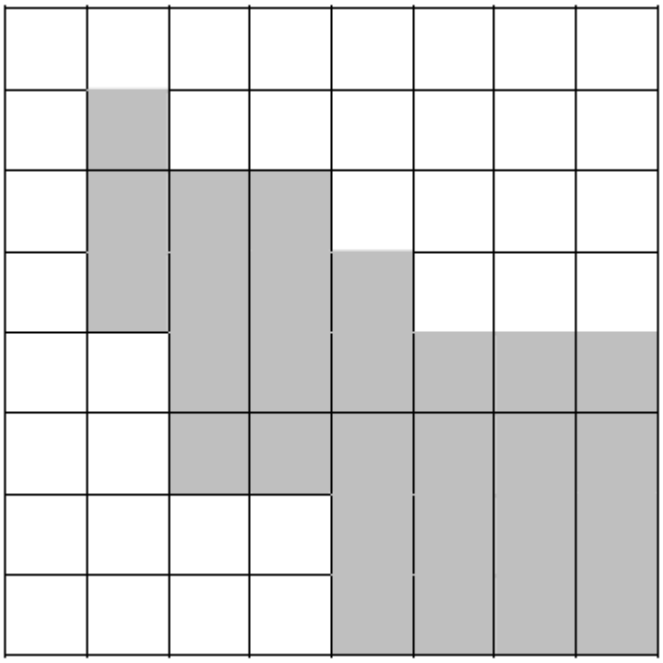
Based on the basic concepts and theoretical foundation that were discussed above, the process of image compression starts with image partitioning. As the first step, the image is partitioned by some collection of ranges R_i. Then for each R_i, a domain D_i, which has a low rms error, is found from some collection of image pieces. The ways of partitioning the image into domains and ranges are both quadtree method. The sets R_i and D_i, determine s_i and

o_i as well as a_i, b_i, c_i, d_i, e_i, and f_i . Then a transformation $W = \mathrm{U}w_i$, which encodes the original image, is obtained.

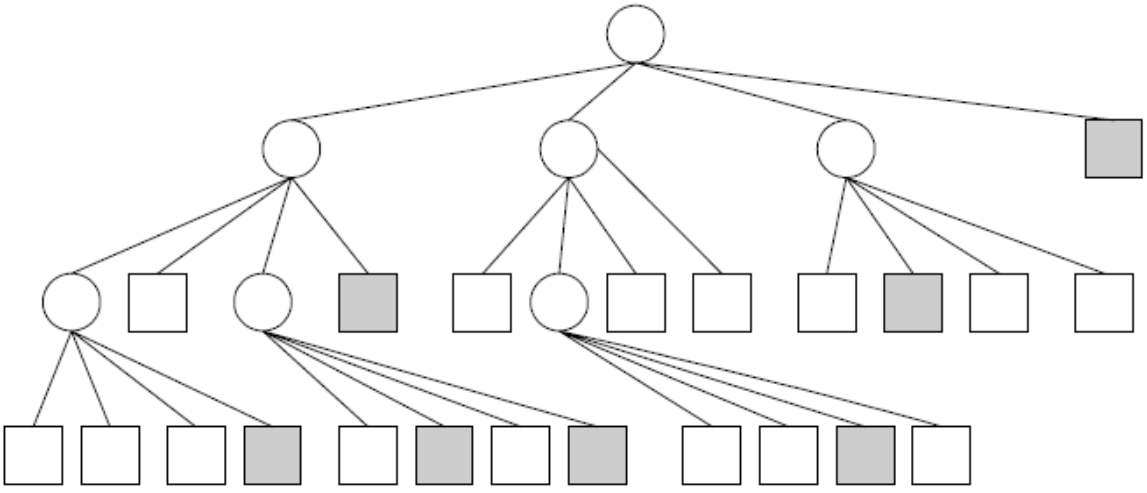
A quadtree partition is a representation of an image as a binary space partitioning tree in which each node, corresponding to a square portion of the image, contains four sub-nodes, corresponding to the four quadrants of the square. The root of the tree is the initial image (Figure 4.5).

First, the image is divided into domains with different sizes using the quadtree method and the domain pool is built, including sizes and positions of the domains. At that time, ranges are selected. Based on the minimum tree depth, various initial numbers of the quadtree partitions are prepared. The squares at the nodes are compared with domains in the domain pool (or domain library) *D*, which are twice the range size. The pixels in the domain are averaged in groups of four so that the domain is reduced to the size of the range, and the affine transformation of the pixel values is found that minimizes the rms difference between the transformed domain pixel values and the range pixel values. All the potential domains are compared with a range. If the resulting optimal rms value is above a pre-selected threshold and if the depth of the quadtree is less than a pre- selected maximum depth, then the range square is subdivided into four quadrants, and the process is repeated. If the rms value is below the threshold, the optimal domain and the affine transformation on the pixel values are stored. Thus one map w_i is found. The collection of all such maps $W = \mathrm{U}w_i$ constitutes the encoding.

29



(a)



(b)

Figure 4.3 A Black-White Image and Its Quadtree

a. The Image

b. The Quadtree

30

	003						
	021	030	031				
	023	032	033	122			
		210	211	300	301	310	311
		212	213	302	303	312	313
				320	321	330	331

Figure 4.4 Coding of a Linear Quadtree

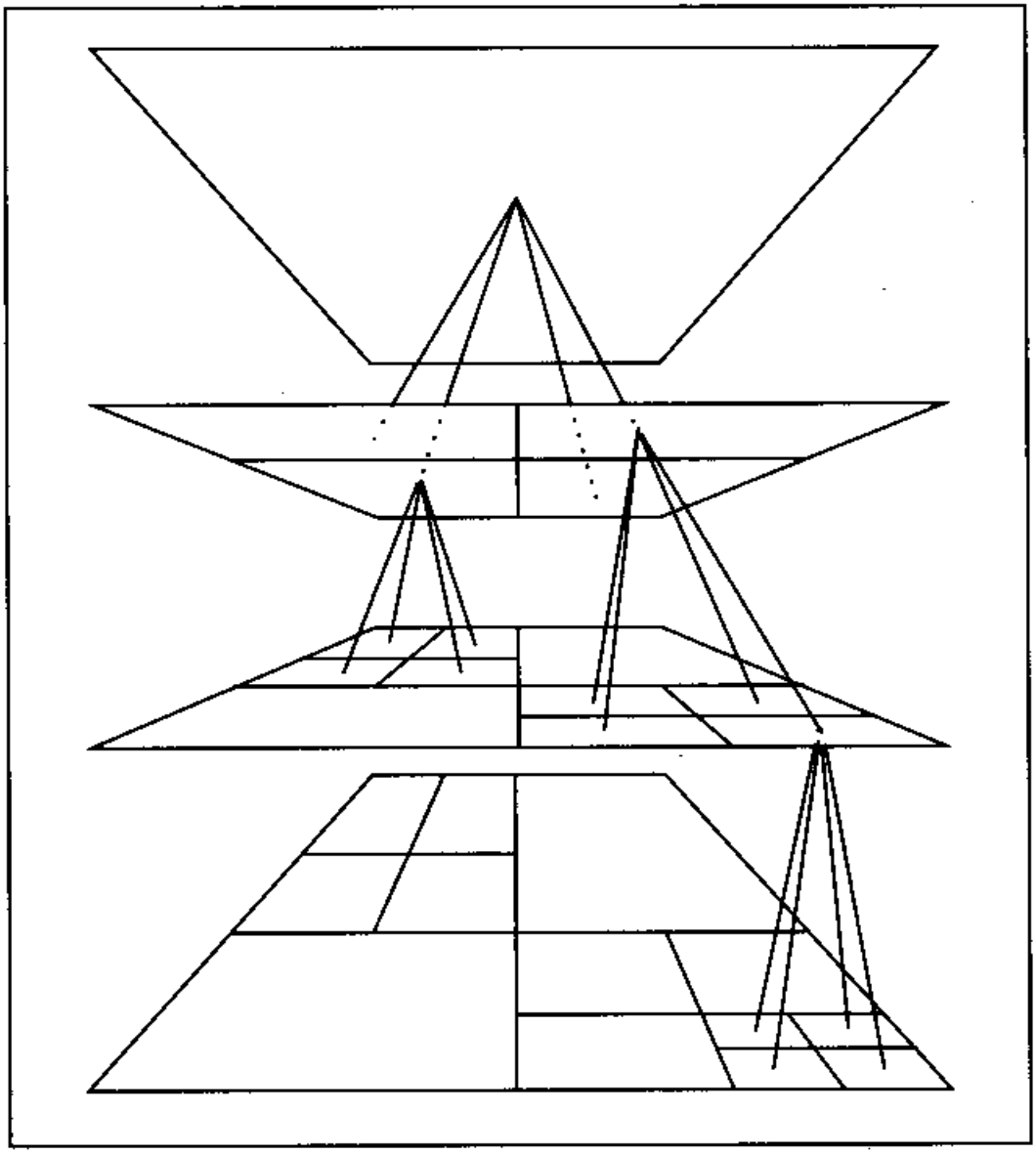


Figure 4.5 Quadtree Partition

(Fisher, Y., et al, 1992)

4.6.2 Encoding (Compressing) Algorithm

The basis for the encoding procedure is like this: an image is partitioned into parts that can be approximated by other parts after some scaling operations (Figure 4.6). The result of the procedure is a set of transformations, which, when iterated from *any* initial image, possess a fixed point approximating the original image. In the restored image, fractal characteristics can be seen: 'zooming' into the restored image, finer and finer details will appear. Accordingly, self-similarity (a part of the image can be approximated by other part of the same image), and recursive subdivision, which are the important characteristics of fractals, are put into use in the encoding procedure. This is why the procedure is named fractal.

The steps of the encoding (compressing) procedure are as follows:

1) Determine the parameters for compressing:
    i) Image name, image size, minimum partition exponent, maximum partition exponent (both of which determine the size of domains and ranges), tolerance for fidelity, e.g. xxx.img, 256x256, 4 (corresponding to 16x16 blocks), 6 (corresponding to 4x4 blocks), 0 (corresponding to tolerance as zero).
2) Read the image to be compressed.
3) Process domains
    a. Scale the image by calculating the average values of each four-pixel

group, then save the calculated values into an array domain

b. Divide the image (in 'domain') into overlapping domains (16x16 or 8x8)

c. Divide each domain block into 4 quadrants and calculate the variance of each quadrant.

33

d. Classify the domains into 24 classes (Figure 4 .7) according to the order of the variances of the quadrants of the domain blocks. Record the position, the size and the class of the domain blocks in the corresponding class chain.

e. After processing the 16x16 domains, the procedure is repeated until you reach the smallest domains (4 x 4) as specified by the maximum partition exponent.

4) Record the parameters into the output file including image size, maximum partition, minimum partition exponents, and maximum and minimum values of the image.

5) Process ranges

a. Partition the original image into ranges according to the quadtree method. If the minimum partition is not reached: go ahead to continue the partition until the minimum partition is reached (e.g. 16x16 range).

b. Classify the range according to the same rule as did for domains, i.e. 24 classes based on the order of the variances of the quadrants of the block.

c. Search the domain class chains to find the domain which matches the

current range best by calculating the error between the domain and range. Record the position and the rotating factor of the domain with the smallest error.

d. Check a) if the smallest error is less or equal to the tolerance and

   b) if the maximum partition is reached.

   i. If both are 'Yes', go to (5);

   ii. If a) is 'Yes' and b) is 'No', go to (5);

   iii. If a) is 'No' and b) is 'Yes', go to (5);

   iv. If both are 'No', put a bit '1' as a flag into the output file, then

   v. Divide the current range into 4 sub-ranges according to the quadtree method, then go to (2).

e. Write A, B, the rotating factor, the position of the domain with smallest error, into the output compressed file.

6) Calculate the compression rate: the number of bytes of the original image divided by the number of bytes in the output compressed file. The parameters for the encoding algorithm include:

- The rms tolerance threshold e_c
- The maximum depth of the quadtree partition;
- The minimum depth of the quadtree partition;
- The type of domain pool
- The maximum allowable scaling factor s_{max}
- The number of classes compared with a range.
- The maximum and minimum values of the image

An encoding of an image consists of the following data:

- The final quadtree partition of the image
- The scaling and offset values s_i and o_i for each range
- For each range, a domain that is mapped to it
- The symmetry operation (orientation) used to map the domain pixels onto the range pixels.

Figures 4.8 and 4.9 show an example of the partition and transformation.

4.6.3 Decoding (Decompressing) Algorithm

Decoding an image consists of iterating *W* from any initial image. The quadtree partition is used to determine all the ranges in the image. For each range R_i, the domain D_i, which maps to it is shrunk by two in each dimension by averaging non-overlapping groups of 2 x 2 pixels. The shrunken domain pixel values are then multiplied by si, added to oi, and placed in the location in the range determined by the orientation information. This constitutes a decoding iteration. The step is iterated until the fixed point is approximated, that is, until further iteration does not change the image or until the change is below some small threshold value.

The steps of the decoding (decompressing or restoring) procedure are as follows:

1) Determine the parameters of the restoring procedure:
   a) Hypothetical image (the "initial image") name,
   b) Compressed file name and
   c) Number of iteration.
2) Read the input file (the compressed file).
3) Read the hypothetical image into an array 'image', which will be used as the domains. The hypothetical image can have any values (e.g. all 0's, or all 1's, etc.).

4) Get the parameters of the compressed image from the input file:

   a) Image size,

   b) Maximum partition,

   c) Minimum partition exponents, and

   d) Maximum and minimum values of the original image.

5) Prepare a blank image ‘image1’ for the restored image.

36

6) Read the transformations from the input file:

   a) Set ‘level’=0.

   b) Checking level is less/more than minimum partition.

      i) If ‘Yes’, divide the blank image block into 4 sub-blocks, with level +1, put sub-blocks into a queue, then go to (3);

      ii) If ‘No’, i.e. level is equal to minimum partition, go to (4).

   c) Check the queue. If it is not empty:

      i) Obtain the first one and remove it from the queue,

      ii) Then go to (2); if it is empty, go to (7).

   d) Checking if ‘level’ is less than maximum partition and the flag bit ‘1’ in the input file.

      i) If both are ‘Yes’, divide the current block into 4 sub-blocks

      ii) With level +1, put them into the queue, then go to (3);

      iii) If a) or b) or both are ‘No’, i.e. either the maximum partition is iv)

      Reached or good transformation is encountered, then go to (4).

      v) Get the transformation parameters from the input file: A, B,

      rotating factor and position of the domain, then put them

      and the corresponding positions of the restored image into a

transformation chain.

vi) Go to (3).

vii) Finish the procedure starting from step 6).

7) Do the transformations:

a) Set a counter=0;

b) check if counter is greater than the predetermined iteration number.

If 'Yes', go to (8);

If 'No', go to (3).

37

c) Check the transformation chain. If it is

empty, counter+1, then go

to b); if it is not empty, go to d).

d) Get the parameters in the transformation chain, calculating the pixel values for the restored image by scaling the hypothetical image and using Y=AX+B, if necessary, rotating it (A, B and rotating factor are all in the transformation chain, Y represents the restored value, and X the 'initial value' in hypothetical image), then put the pixel values into 'image1'.

e) Go to c).

f) Copy 'image1' to 'image'. The later will be the 'hypothetical image' or the domain of the next iteration.

g) Counter+1, then go to (2).

h) Finish the iteration.

8) Post-process the restored image by using maximum and minimum values of the original image.

9) Write ‘image’ into output file.

Figure 4.10 is the initial image of which all pixels are ‘white’, i.e. gray level is 255; 4.11 and 4.12 show the result of decompressing procedure at the first and the tenth iteration respectively. It can be seen that after ten iterations, the restored image became very similar to the original one.

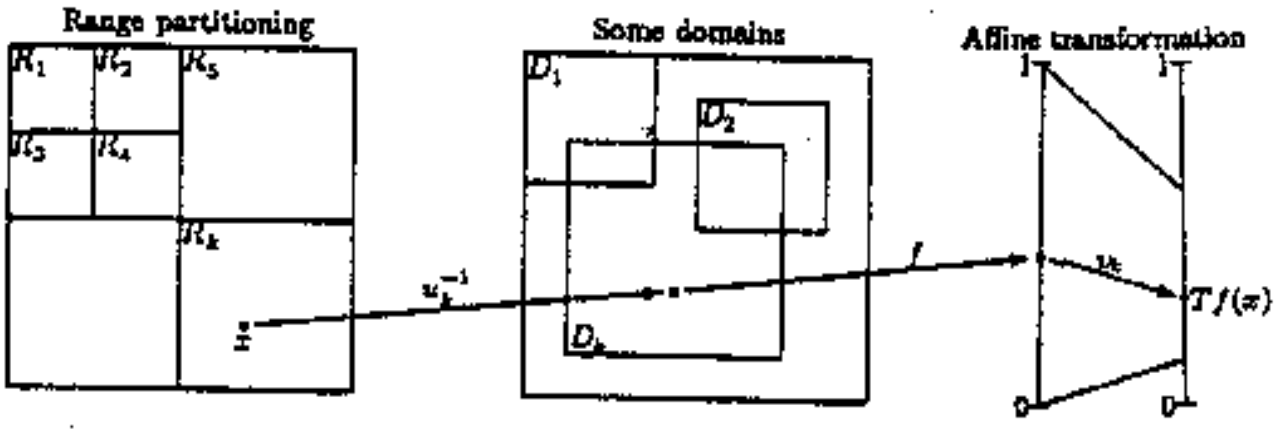


Figure 4.6 Partition with Self-Similarity(Source: Saupe et al 1994)

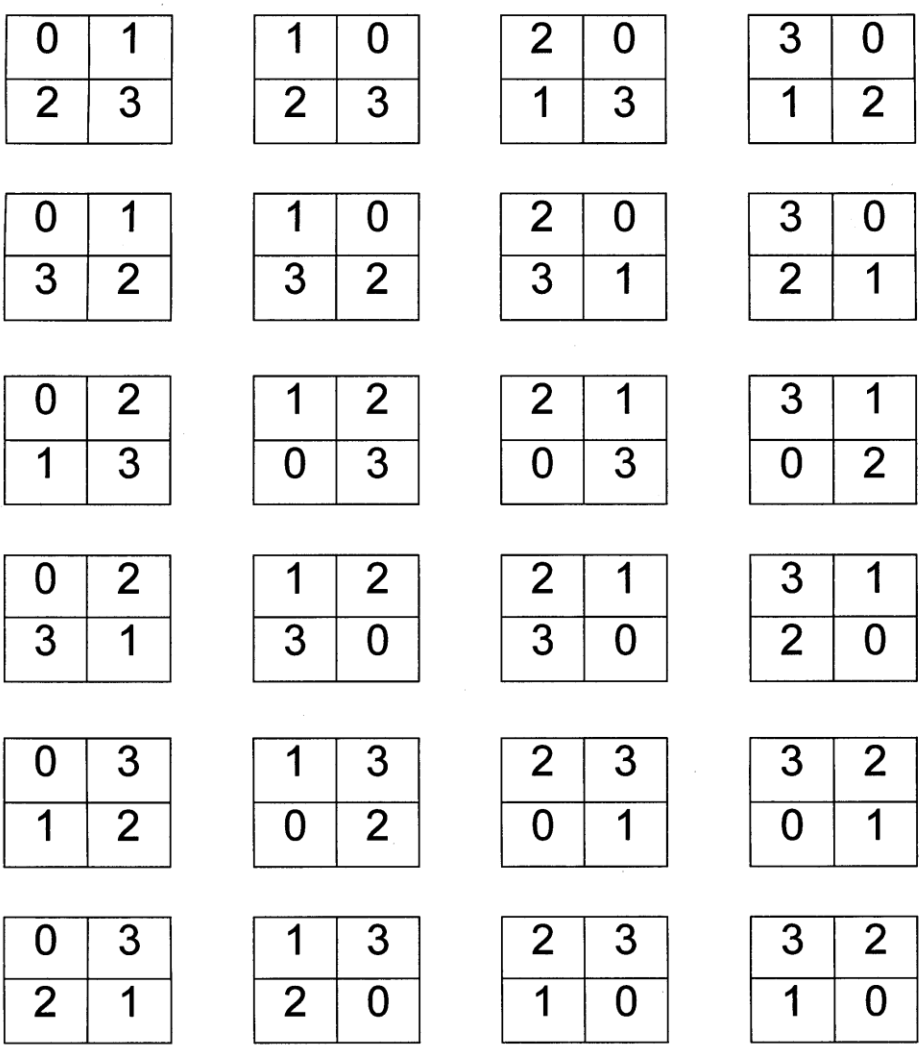


Figure 4.7 Twenty Four Classes
(0, 1, 2, and 3 represent the ordered values of variances of the quadrants. There are twenty four situations for the whole combination)

39

Figure 4.8 Image of San Francisco

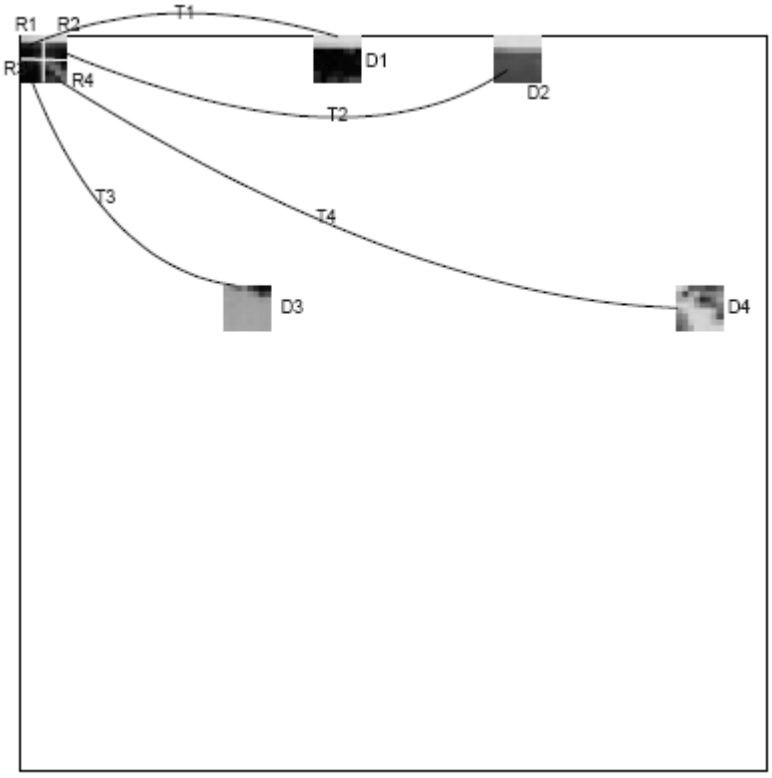


Figure 4.9 Domains, Ranges, and Transformation for the image in Fig.4.8
(R1 through R4: first four ranges in the image; D1 through D4: domains corresponding to R1 through R4; T1 through T4: transformations between Di and Ri, i=1,2,3,4)

40

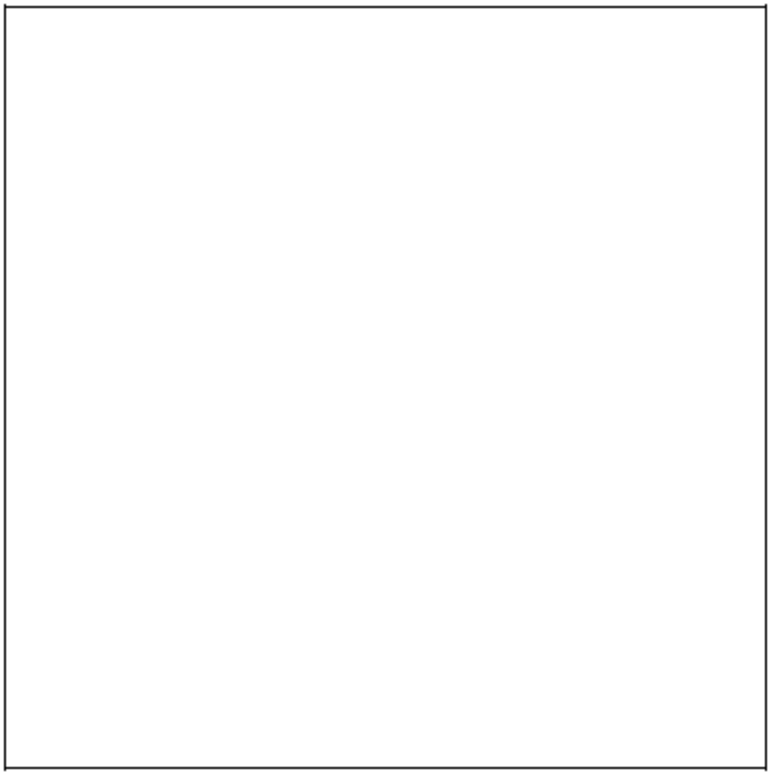


Figure 4.10 Initial image (all pixels' DN is 255)

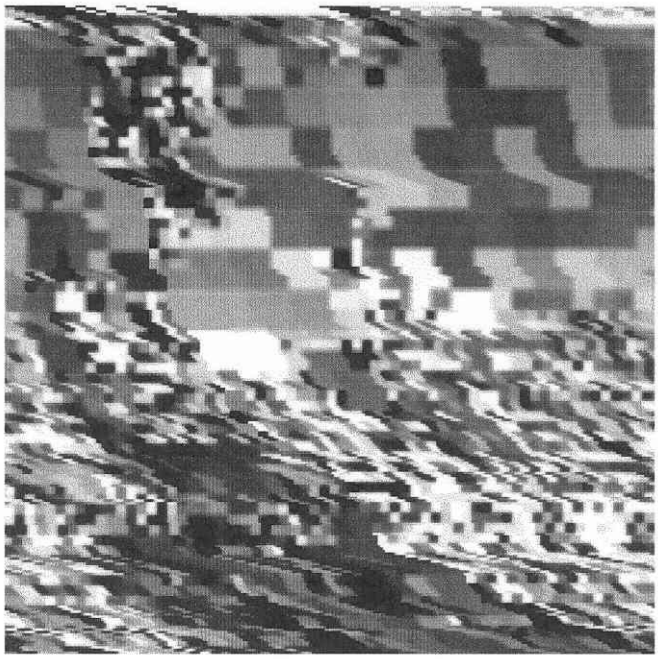


Figure 4.11 First Iteration of Decompression
41

Figure 4.12 Tenth Iteration of Decompression

4.7 New Algorithm

This method reduces the encoding time of fractal image compression by performing less searches by excluding many of domain blocks from the domain pool. The idea is based on the observation that many domains are never used in a typical fractal encoding, and only a fraction of this large domain pool is actually used in the fractal coding. There is a very large set of domain blocks in the domain pool with high entropy, which are not used. Thus, it is possible to reduce the search time by discarding a large fraction of high entropy blocks, which affect only a few ranges. For these ranges a sub-optimal domains with smaller entropy may be found. In this way, the domain pool is constructed from blocks with the lowest entropy instead of all domains. The algorithm for encoding is as below:

Step 1: Initialization (domain pool construction)

Choose parameter E ;
Divide the input image into *N* domains, D_j
For (j =1; j $\leq$ *N*; j ++) **{**
Ent =entropy (D_j);
If (Ent $\leq$ E)
Push D_j onto domain pool stack **}**

Step 2: Choose a tolerance levels C ;

Step 3: Search for best matches between range and domain blocks

For (i =1 ; i $\leq$ num_range ; i ++) **{**
min_error =C ;
For (j =1 ; j $\leq$ num_domain; j ++) {
Compute **s**, **o**;
If (0$\leq$ **s**<1.0)
If (E(R_i , D_j) < min_error) {
min_error = E(R_i , D_j);

```
            best_domain[i] =j ;}
}
```

43

If (min_error = = C)

Set R_i uncovered and partition it into 4 smaller blocks

Else

Save_coefficients(best_domain, s, o);

}

At the end of step 1 the domain pool has *num_domain* domain according to E value.

CHAPTER 5

IMAGE REGISTRATION

5.1 INTRODUCTION

Image registration is the process of overlaying one or more image to a reference image of the same scene taken at different time, from different view point and/or different sensor. Difference between images is introduced due to different imaging condition such that yields highest similarity between the input and the reference images. Image registration geometrically aligns two images the reference image and input image. Image registration is a crucial step in all image analysis tasks in which the final information is gained from the combination of various data sources like in image fusion, change detection, and multichannel image restoration. Typically, registration is required in remote sensing(multispectral classification, environmental monitoring, change detection, image mosaicing, weather forecasting, creating super-resolution images, integrating information into geographic information systems (GIS)), in medicine(combining computer tomography (CT) and NMR data to obtain more complete information about the patient, monitoring tumor growth, treatment verification, comparison of the patient's data with anatomical atlases),in cartography (map updating), and in computer vision (target localization, automatic quality control), to name a few.

In general, its applications can be divided into four main groups according to the manner of the image acquisition:

Different viewpoints (multiview analysis):-Images of the same scene are acquired from different viewpoints. The aim is to gain larger a 2D view or a 3D representation of the scanned scene. Examples of applications: Remote sensing mosaicing of images of the surveyed area. Computer vision—shape recovery (shape from stereo).

Different times (multitemporal analysis):- Images of the same scene are acquired at different times, often on regular basis, and possibly under different conditions. The aim is to find and evaluate changes in the scene which appeared between the consecutive image acquisitions. Examples of applications: Remote sensing—monitoring of global land usage, landscape planning. Computer vision—automatic change, detection for security monitoring, motion tracking. Medical imaging—monitoring of the healing therapy, monitoring of the tumor evolution.

Different sensors (multimodal analysis):-Images of the same scene are acquired by different sensors. The aim is to integrate the information obtained from different source streams to gain more complex and

detailed scene representation. Examples of applications: Remote sensing—fusion of information from sensors with different characteristics like panchromatic images, offering better spatial resolution, color/multispectral images with better spectral resolution, or radar images independent of cloud cover and

solar illumination. Medical imaging—combination of sensors recording the anatomical body structure like magnetic resonance image (MRI), ultrasound or CT with sensors monitoring functional and metabolic body activities like positron emission tomography (PET), single photon emission computed tomography (SPECT) or magnetic resonance spectroscopy (MRS). Results can be applied, for instance, in radiotherapy and nuclear medicine.

Scene to model registration:- Images of a scene and a model of the scene are registered. The model can be a computer representation of the scene, for instance maps or digital elevation models (DEM) in GIS, another scene with similar content (another patient), 'average' specimen, etc. The aim is to localize the acquired image in the scene/model and/or to compare them.

Examples of applications: Remote sensing—registration of aerial or satellite data into maps or other GIS layers.

Computer vision—target template matching with real-time images, automatic quality inspection. Medical imaging—Comparison of the patient's image with digital anatomical atlases, specimen classification.

5.2 Image Registration process

The registration process involves finding a single transformation imposed on the input image by which it can align with the reference image. It can be viewed as different combination of choice for the following four component.

(1) Feature space
(2) Search space
(3) Similarity measure
(4) Search strategy

The **Feature space** extracts the information in the images that will be used for matching. The **Search space** is the class of transformation that is capable of aligning the images. The **Similarity measure** gives an indication of the similarity between two compared image regions. The **Search strategy** decide how to choose the next transformation from the search space, to be tested in the search to spatial transformation.

Registration process mainly consists in determining the unknown transformation parameters required to map the input image to the reference image in order to compare and analyze both in a common reference frame. The task of determining the best spatial transformation for the registration of images can be characterized by four major components (Brown, 1992):

1. Feature space
2. Search space
3. Similarity measure
4. Search strategy

5.3 Feature Space:-

The feature space represents the information in the images that will be used for matching. This may be the image itself (i.e. pixels values), but we can also use distinctive objects (closed boundary regions, edges, contours, line intersections, corners, etc.) manually or automatically detected. These features can be represented by their point representatives (centers of gravity, line endings, distinctive points), which are called control points (CPs) in the literature. Images are usually preprocessed in an attempt to extract intrinsic structures and reduce the effects of sensor noise.

First, we have to decide what kind of features is appropriate for the given task. The features should be distinctive objects, which are frequently spread over the images and which are easily detectable. Usually, the physical interpretability of the features is demanded. The detected feature sets in the reference and sensed images must have enough common elements, even in situations when the images do not cover exactly the same scene or when there are object occlusions or other unexpected changes. The detection methods should have good localization accuracy and should not be sensitive to the assumed image degradation. In an ideal case, the algorithm should be able to detect the same features in all projections of the scene regardless of the particular image deformation.

The number of common elements of the detected sets of features should be sufficiently high, regardless of the change of image geometry, radiometric conditions, presence of additive noise, and of changes in the scanned scene. The 'remarkableness' of the features is implied by their definition.

According to the feature space employed, we can identify three classes of registration algorithms pixel-based, and feature-based, transform-based,.

Pixel-based registration:-

Pixel-based algorithms work directly with the (totality of) pixel values of the images being registered. Preprocessing is often used to suppress the adverse effects of noise and differences in acquisition or to increase or uniformize pixel resolution. The main advantage of this approach is a more global vision of the algorithm, which increases its robustness.

Feature-based registration:-

Feature-based algorithms work on a set of characteristic features extracted from the images. The dimensionality of the features is usually drastically smaller than the dimensionality of the original image data. The extraction process is highly non-linear, mostly using thresholding.

Landmark-based methods use a relatively small and sparse set of landmarks; these are important points which can be (manually or automatically) identified in both images. Extrinsic markers refer to specifically designed artificial features attached to the object (or subject, in medical imaging) before acquisition to serve as landmarks. Unfortunately, extrinsic markers are difficult to deploy. In medical imaging they are not patient-friendly either. If extrinsic markers are not available, we have to content ourselves with features intrinsic to the images. In that case, however, the automatic landmark identification suffers from lack of robustness. The manual landmark identification is often tedious, time-consuming, imprecise, and unreproducible. If the images cannot be characterized using points, it might be more appropriate to use curves such as edges or volume boundaries.

Transform-based registration:-

Transform-based algorithms exploit properties of the Fourier, wavelet, Hadamard, and other transforms, making use of the fact that certain deformations manifest themselves more clearly in the transform domain. These methods are used mainly in connection with linear deformation fields. Nevertheless, there

are examples of methods that estimate locally linear optical flow using Gabor filters and B-spline wavelets. Typical characteristics of the transforms employed are linearity and independence on the actual image contents.

5.4 Search Space:-

The search space is made of all possible transformations. The considered sensor model determines the characteristics of the search space (i.e. which transformations to consider and in which range). If no a priori information is available, the model should be flexible and general enough to handle all possible degradations which might appear.

The input image is transformed by means of the mapping functions. Image values in non-integer coordinates are computed by the appropriate interpolation technique.

Some popular transformations are as follows:-

Transformation of the Cartesian coordinate system:-

It consists of rotation, translation and scaling only

$$u = s\,x\cos(\theta) - s\,y\sin(\theta) + tx$$

$$v = s\,x\sin(\theta) + s\,y\cos(\theta) + ty$$

s is the scaling, θ is the rotational, and (tx, ty) are the translational differences between the images. This model is often called 'shape-preserving mapping' because it preserves angles and curvatures and is unambiguously determined by two CPs. This transformation is useful when registering images as rigid bodies.

Affine transformation:-

Slightly more general but still linear model is an affine transform

$$u = a_0 + a_1 x + a_2 y$$

$$v = b_0 + b_1 x + b_2 y$$

This can map a parallelogram onto a square. This model is defined by three non-collinear CPs, preserves straight lines and straight line parallelism. It can be used for multiview registration assuming the distance of the camera to the scene is large in comparison to the size of the scanned area, the camera is perfect (a pin-hole camera), the scene is flat, and the present geometric distortion has no local factors.

Projective Transformation:-

If the condition on the distance of the camera from the scene is not satisfied the perspective projection model

$$u = (a_0 + a_1 x + a_2 y)/(dx+ey+1)$$

$$v = (b_0 + b_1 x + b2\ y)/(dx+ey+1)$$

should be used. This model exactly describes a deformation of a flat scene photographed by a pin-hole camera the optical axis of which is not perpendicular to the scene. It can map a general quadrangle onto a square while preserving straight lines and is determined by four independent CPs.

5.5 Similarity Measure:-

The similarity measure function, or cost function, gives an indication of the similarity between two compared image regions. The function may either be based on direct pixel intensity comparisons, or on other geometrical features within the regions. Here we focus some of the direct intensity comparisons based similarity measures

Sum of Squared Differences

A simple similarity measure is the sum of the differences between compared pixel intensities within the area, squared. In case of color photographs we may look at each color channel separately and take the average. Equation 2.4 defines the function, known as SSD (Sum of Squared Differences) mathematically.

$$ESSD(h,k) = \sum_{i}^{h}\sum_{j}^{W}(\mathrm{I_i(i+h,j+k) - I_0(i,j))^2}$$

Where Ii denotes the intensity value at image i, and h, k defines the offset point in the input image for which the similarity measure should be calculated. The main weakness of this measure is that intensity shifts between the images may cause large differences even at the most optimal matching position, potentially leading to inaccurate results and misalignment.

Sum of Absolute Differences:-

Method of minimizing intensity difference called as sum of absolute difference(SAD),which exhibit minimum in the case of minimum in the case of perfect matching

$$\mathrm{SAD} = \frac{\sum_i^n |I_r(i) - T(I_S(i)|}{N} \ldots$$

Where I_r(i) is the intensity value at position i of the reference image R and I_s(i) is the intensity value at position i of the input image S and T geometrical transformation.

SAD is suitable for monomodal image registration only when image difference among the data are sufficiently small. In spite of high computational efficiency, these cost function can lead to monomodal data registration when intensity is significantly changed.

Normalized cross correlation:-

Classical area-based methods like cross-correlation (CC) exploit for matching directly image intensities, without any structural analysis. Consequently, they are sensitive to the intensity changes, introduced for instance by noise, varying illumination, and/or by using different sensor types. The normalized cross-correlation can be written as

$$CC = \frac{\sum_{i=0}\sum_{j=0}((R_{ij} - R) * ((I_{ij} - I))}{\sqrt{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(R_{ij} - R)^2 * \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(I_{ij} - I)^2}}$$

Where R_{ij} is the intensity of the reference image pixel and R is the mean intensity I_{ij} is the intensity of the input image and I is the mean intensity.

This measure of similarity is computed for window pairs from the input and reference images and its maximum is searched. The window pairs for which the maximum is achieved are set as the corresponding ones .If the sub pixel accuracy of the registration is demanded, the interpolation of the CC measure values needs to be used. Although the CC based registration can exactly align mutually translated images only, it can also be successfully applied when slight rotation and scaling are present.

Two main drawbacks of the correlation-like method is the flatness of the similarity measure maxima (due to the self-similarity of the images) and high computational complexity. Despite of these limitations, the correlation like registration methods are still often in use, due to its easy hardware implementation, which makes it useful for real-time applications.

Variance

Another similarity measure is to look at the *standard deviation* of the intensity differences of the compared region. This approach is less susceptible to failure due to intensity shifts than squared differences, since the intensity difference should be of approximately the same amount for each compared pixel, resulting in small variance and standard deviation. Equation gives a variance based similarity measure at offset (h, k).

$$E_V(h, k) = \sqrt{Var(I_{diff}(h, k))}$$

Where I_{diff} is the set of intensity differences within the compared region.

It may also be possible to improve results by utilizing spatially varying weights or biases to compensate for mean intensity variations between the images.

Mutual Information Method

Mutual information similarity measured is based on information theory. Mutual information compares the statistical dependence between two images. The similarity measure works from the pixel values and makes few assumptions about the surface properties of the objects or the imaging process. As a result, mutual information is robust with respect to changes in lighting and even imaging modality.

To introduce and explain the mutual information, we start at the basics with the definition of entropy and its interpretation. We then turn to mutual information, its multiple forms of definition and its properties.

5.6 Entropy

The desire for a *measure of information* (commonly termed *entropy*) of a message stems from communication theory. This field concerns the broadcast of a message from a sender to a receiver. The first attempts to arrive at an information measure of a message focused on telegraph and radio communication, sending Morse code or words. However, picture transmission (television) was already considered in the important paper by Hartley In 1928, he defined a measure of information of a message that forms the basis of many present-day measures. He considered a message a string of symbols, with *s* different possibilities for each symbol. If the message consists *n* of symbols, there are s^n different messages possible (assuming there are no syntactic rules). He sought to define an information measure that increases with message length. The measure complies s^n but the amount of information would increase exponentially with the length of the message and that is not realistic. Hartley wanted a measure *H* that increases linearly with *n*, i.e. *H=K n.*, where *K* is a constant depending on the number of symbols *s* . He further assumed that, given messages of length *n1*and n2 from s1 and *s2* numbers of symbols, respectively, if $s1^{n1}=s2^{n2}$, i.e., the number of possible messages is equal, then the amount of information per message is also equal. These two restrictions led him to define the following measure of information

$$H= n \log s= \log s^n \quad (1)$$

Hartley's information measure depends on the number of possible outcomes: the larger the number of possible messages, the larger the amount of information you get from a certain message. If there is only a single message possible, you gain no information (log 1=0) from it, because you already knew you would

receive that message. In this respect, the measure can also be viewed as a measure of *uncertainty*. When there are more different messages you could possibly receive, you are more uncertain which one you will actually receive. And, again, if there is only one, there is no uncertainty.

A drawback of Hartley's measure is that A drawback of Hartley's measure is that it assumes all symbols(and, hence, all messages of a given length) are equally likely to occur. Clearly, this will often not be the case. In the previous paragraph, for example, the letter "e" has occurred 229 times and the letter "q" only twice. Shannon introduced an adapted measure in 1948 , which weights the information per outcome by the probability of that outcome occurring. Given events $e_1 \ldots\ldots, e_n$ occurring with probabilities $p_1 \ldots\ldots$, p_m the *Shannon entropy* is defined as

$$H = \sum_i p_i \log \frac{1}{p_i} = -\sum_i p_i \log p_i \qquad (2)$$

If we apply to Shannon's entropy the assumption that all outcomes are equally likely to occur, we get

$$H = -\sum_i \frac{1}{s^n} \log \frac{1}{s^n} = \sum_i \frac{1}{s^n} \log s^n = \log s^n \qquad (3)$$

which is exactly Hartley's entropy.

Although the second definition of the Shannon entropy in (2) is more commonly used, the first one more clearly explains the meaning. The term signifies that the amount of information gained from an event with probability is inversely related to the probability that the event takes place. The more rare an event, the more meaning is assigned to occurrence of the event. The information per event is weighted by the probability of occurrence. The resulting entropy term is the *average* amount of information to be gained from a certain set of events.

In line with Hartley's entropy, we can also view Shannon's entropy as a measure of uncertainty. The difference is that Shannon's measure depends not only on the number of possible messages, but also on the chances of each of the messages occurring. When all messages are equally likely to occur, the entropy is maximal, because you are completely uncertain which message you will receive. When one of the messages has a much higher chance of being sent than the other messages, the uncertainty decreases. You expect to receive that one message and in most cases you will be right. The amount of information for the individual messages that have a small chance of occurring is high, but, *on average*, the information

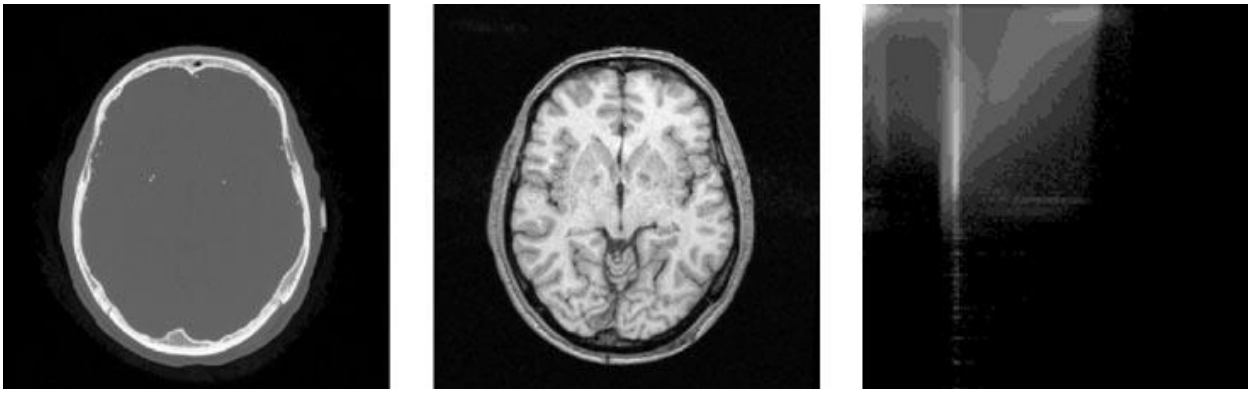
(entropy/uncertainty) is lower. As a hypothetical example, let us assume a 1-yr old child uses the words "mummy," "daddy," "cat," and "uh-oh." If the child uses all words almost as frequently, with a slight preference for "mummy," the respective percentages of times the words are used could be 0.35, 0.2, 0.2, and 0.25. The entropy of the child's language is then -.35 log .35-.2 log .2-.2 log .2- .25log .25=1.96. Sometime later, the vocabulary may have expanded and changed to ("mummy" 0.05), ("daddy" 0.05), ("cat" 0.02), ("train" 0.02), ("car" 0.02), ("cookie" 0.02), ("telly" 0.02), and ("no" 0.8). Now one word is dominant and the entropy of the language has dropped to 1.25. There is less uncertainty about which word the child will utter. Whatever you ask, the answer is almost certainly "no."

The Shannon entropy can also be computed for an image, in which case we do not focus on the probabilities of letters or words occurring, but on the distribution of the gray values of the image. A probability distribution of gray values can be estimated by counting the number of times each gray value occurs in the image and dividing those numbers by the total number of occurrences. An image consisting of almost a single intensity will have a low entropy value; it contains very little information. A high entropy value will be yielded by an image with more or less equal quantities of many different intensities, which is an image containing a lot of information.

In this manner, the Shannon entropy is also a measure of dispersion of a probability distribution. A distribution with a single sharp peak corresponds to a low entropy value, whereas a dispersed distribution yields a high entropy value. Summarizing, entropy has three interpretations: the amount of information an event (message, gray value of a point) gives when it takes place, the uncertainty about the outcome of an event and the dispersion of the probabilities with which the events take place.

Mutual information

The research that eventually led to the introduction of mutual information as a registration measure dates back to the early 1990s. Woods *et al.* first introduced a registration measure for multimodality images based on the assumption that regions of similar tissue (and, hence, similar gray values) in one image would correspond to regions in the other image that also consist of similar gray values (though probably different values to those of the first image). Ideally, the ratio of the gray values for all corresponding points in a certain region in either image varies little. Consequently, the average variance of this ratio for all regions is minimized to achieve registration.



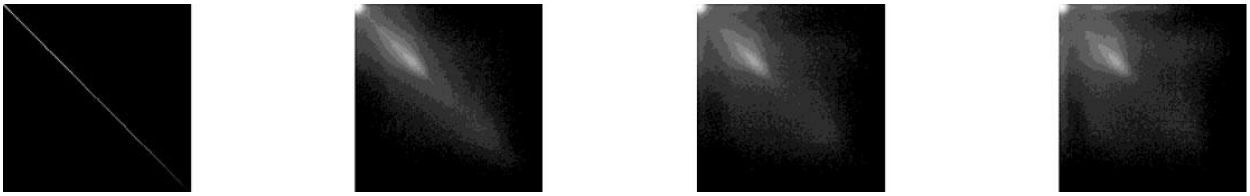
(a) (b) (c)

Fig 5.1. Example of a feature space for (a) a CT image and (b) an MR image. (c) Along the axes of the feature space, the gray values of the two images are plotted: from left to right for CT and from top to bottom for MR. The feature space is constructed by counting the number of times a combination of gray values occurs. For each pair of corresponding points (x; y), with x a point in the CT image and y a point in the MR image, the entry (I (x); I (y)) in the feature space on the right is increased. A distinguishable cluster in the feature space is the stretched vertical cluster, which is the rather homogeneous area of brain in the CT image corresponding to a range of gray values for the MR image.

Hill *et al* proposed an adaption of Woods' measure. They constructed a *feature space*, which is a two-dimensional (2-D) plot showing the combinations of gray values in each of the two images for all corresponding points. Fig. 5.1 shows an example of such a feature space for a magnetic resonance (MR) and a computed tomography (CT) image. The difference with Woods' method is that instead of defining regions of similar tissue in the images, regions are defined in the feature space. These regions are based on the clustering one finds in the feature space for registered images.

The feature space (or joint histogram) changes as the alignment of the images changes. When the images are correctly registered, corresponding anatomical structures overlap and the joint histogram will show certain clusters for the gray values of those structures. . As the images become misaligned, structures will also start overlapping structures that are not their anatomical counterparts in the other image. Consequently, the intensity of the clusters for corresponding anatomical structures will decrease and new combinations of gray values emerge such as skull and brain or skin and background. This will manifest itself in the joint histogram by a dispersion of the clustering. Fig. 5.2contains several histograms of an

MR image with itself for different rotations of one image with respect to the other. Clearly, the histogram shows increasing dispersion as the misregistration increases.



(a) (b) (c) (d)

Fig. 5. 2 Joint gray value histograms of an MR image with itself. (a) Histogram shows the situation when the images are registered. Because the images are identical, all gray value correspondences lie on the diagonal. (b), (c), and (d) show the resulting histograms when one MR image is rotated with respect to the other by angles of 2_, 5_, and 10_, respectively. The corresponding joint entropy values are (a) 3.82; (b) 6.79; (c) 6.98; and (d) 7.15..

Using this characteristic of the joint histogram of two images, measures of dispersion emerged, to use for image registration. Hill *et al.* proposed the third-order moment of the joint histogram, which measures the skewness of a distribution. Both Collignon *et al* and Studholme *et al* suggested to use entropy as a measure of registration. As we have explained earlier, entropy measures the dispersion of a probability distribution. It is low when a distribution has a few sharply defined, dominant peaks and it is maximal when all outcomes have an equal chance of occurring. A joint histogram of two images can be used to estimate a joint probability distribution of their gray values by dividing each entry in the histogram by the total number of entries. The Shannon entropy for a joint distribution is defined as

$$-\sum_{i,j} p(i,j) log\ p(i,j)$$

By finding the transformation that minimizes their joint entropy, images should be registered.

Once entropy, a measure from information theory, had been introduced for the registration of multimodality medical images, another such measure quickly appeared: mutual information. It was pioneered both by Collignon *et al.* and by Viola and Wells]. Applied to rigid registration of multimodality images, mutual information showed great promise and within a few years it became the most investigated measure for medical image registration.

Definition:-

Two discrete random variables A and B with marginal probability distributions $p_A(a)$ and $p_B(b)$ and joint probability distribution $p_{AB}(a,b)$ are statistically independent if $p_{AB}(a,b) = p_A(a).\ p_B(b)$, while they are maximally dependent if they are related by a one-to-one mapping T: $p_A(a) = p_B(T(a)) = p_{AB}(a,T(a))$. The mutual information $I(A,B)$of A and B measures the degree of dependence of and as the distance between the joint distribution $p_{AB}(a,b)$ and the distribution associated to the case of complete independence $p_A(a).\ p_B(b)$,, by means of the Kullback-Leibler measure i.e.,

$$I(A,B) = \sum_{a,b} p_{AB}\ (a,b) \log \frac{p_{AB}(a,b)}{p_A(a).p_B(b)} \qquad (4)$$

Mutual information is related to the information theoretic notion of entropy by the following equations

$$I\ (A,\ B) = H\ (A) + H\ (B) - H\ (A,B) \qquad (5)$$

$$= H\ (A) - H\ (A/B) \qquad (6)$$

$$= H\ (B) - H\ (B/A) \qquad (7)$$

with $H(A)$ and $H(B)$ being the entropy of A and B respectively, $H(A,B)$ their joint entropy, and $H(A/B)$ and $H(B/A)$ the conditional entropy of A given B and of B given A , respectively. $H(A)$, $H(A/B)$ and $H(A,B)$ are defined *as*

$$H(A) = -\sum_a p_A\ (a) \log p_A\ (a) \qquad (8)$$

$$H(A/B) = -\sum_{a,b} p_{A,B}\ (a,b) \log p_{A/B=b}\ (a) \qquad (9)$$

$$H(A,B) = -\sum_{a,b} p_{A,B}\ (a,b) \log p_{A,B}\ (a,b) \qquad (10)$$

with $p_{A/B=b}(a)$ the conditional probability of A given $B=b$. The entropy $H(A)$is known to be a measure of the amount of uncertainty about the random variable A, while $H(A,B)$ is the amount of uncertainty left in A when knowing B. Hence, from (6), $I(A,\ B)$ is the reduction in the uncertainty of the random variable A

by the knowledge of another random variable B , or, equivalently, the amount of information that B contains about A . If A and B are independent,

$p_{AB}(a,b) = p_A(a).\ p_B(b)$ and , $I(A,B) = 0$ while if A and B are one-to-one related, $I(A,B)=H(A)= H(B)$. It can be shown] that mutual information $I(A,B)$ is nonnegative $(\ I(A,B)>0)$ for any two random variables A and B.

Properties Of Mutual information:-

Independence: $I(A,B) = 0 \Leftrightarrow p_{AB}(a,b) = p_A(a).\ p_B(b)$

Symmetry : $I(A,B) = I(B,A)$

Self information: $I(A,A)=H(A)$

Lower bound: $I(A,B)\geq 0$

Upper bound: $I(A,B) \leq \min(H(A),H(B))$

$\leq (H(A) + H(B))/2$

$\leq \max(H(A), H(B))$

$\leq H(A,B)$

$\leq (H(A) + H(B))$

Data processing: $I(A, B) \geq I(A, T(B))$

5.7 Search Strategy

The search strategy governs how the search space is explored, and has a great impact on the efficiency of the image registration process. Here we review a few common strategies, leaving the genetic algorithm and Hill climbing algorithm to a more thorough account in the next chapter.

Exhaustive Search

A simple *exhaustive search* (or *full search*) where all states of the search space are computed, is guaranteed to find the optimal alignment. However, keeping in mind that similarity measure computation

usually has complexity on the order of the number of pixels in the compared areas, this is a rather computationally intensive and time consuming strategy, which quickly becomes infeasible to apply on everything but very limited search sets and spaces.

Step Search

A faster alternative to the exhaustive search is the step search strategy. Various brands of this strategy exist, the basis however being a method that iteratively converges towards the optimal alignment with logarithmic order complexity. In each iteration, a center point is defined, and four points are selected at an equal step distance from this point towards north, south, west and east. Similarity measures are calculated for the four points and the center point, and the best point forms the center point of the next iteration that starts with half the previous step distance.

This quickly converges towards a minimum point. Unfortunately, the method is susceptible to trap itself around a local optimum and miss the global one, resulting in misalignment. Chen suggests that the method is used to make fast estimations of the optimal position.

Gradient Descent

Another approach is to perform *gradient descent* on an appropriate similarity measure function. Gradient descent works by calculating the gradient at the current point, and iteratively moves in the negative gradient direction (in case minimization of the similarity measure is our objective). Obviously, the method requires a differentiable and rather smooth objective function surface.

A parameter called the *learning rate* affects how great the leap in the negative gradient direction should be. The initial learning rate must be suited to fit the objective function. A high learning rate makes faster convergence possible, but introduces the risk of ”shooting over the target” by causing moves that passes over the minima valley and worsens the similarity measure. Because of this, the learning rate is usually decreased when a step in the negative gradient direction does not result in a better similarity measure. Adaptive approaches do this, and also make a small increase on the learning rate if a step results in a better similarity measure. Gradient descent requires an initial starting point on the function, which is usually randomly selected.

Multi-resolution Search

In *multi-resolution* (or *subpixel*) image registration, the search space is reduced by searching at different resolutions in an hierarchical manner. The search is made on a set of down sampled versions of the

original images, and matches are iterat- ively refined at higher resolutions. The concept of image wavelet2 decomposition has successfully been put to use in multi-resolution registration. A wavelet-based technique called the *iterative refinement algorithm* (IRA) combined with a genetic algorithm-based search strategy have proven to give accurate and fast results.

5.8 Search Space

When solving an image registration problem, we look for a particular solution that will be better than all or almost all other feasible solutions. Depending on the number of parameters n that constitute a solution, an n-dimensional *search space* consisting of the set of all possible solutions is created. If we mark each point in the search space with the corresponding cost for that solution, we get a landscape-like hypersurface. Our aim with the search is to find the lowest valley in this landscape. This is often a rather time consuming process, since the hypersurface rarely behaves in a smooth and predictable way.

CHAPTER 6

IMPLEMENTATION

6.1 Code

genim.m

```
clear all;
close all;
tic
xdis=[1 20];
ydis=[1 10];
rot=[-20 0];
%scale=[.50 .65];
n=10;
numit=30;
nummut =1;
f=@image_fit;
x=xdis(2)-xdis(1);
y=ydis(2)-ydis(1);
t=rot(2)-rot(1);
%s=scale(2)-scale(1);
```

```
%x=1:3:60;
%y=1:3:60;
%t=-60:3:0;
%s=.4:.01:.6;
ipopx= rand(1,n)*x+xdis(1);
ipopy= rand(1,n)*y+ydis(1);
ipopt= rand(1,n)*t+rot(1);
%ipops= rand(1,n)*s+scale(1);
%ipopx= x;
%ipopy= y;
%ipopt= t;
%ipops= s;
for it=1:numit
  for i=1:n
    fpop(i)=-feval(f,ipopx(i),ipopy(i),ipopt(i));
  end


    maxf(it) = max(fpop);
    meanf(it) = mean(fpop);
    % subtract lowest fitness in order to normalize
    m=min(fpop);
    fpop=fpop-m;
    cpop(1) = fpop(1);
    for i=2:n, cpop(i) = cpop(i-1) + fpop(i); end

    % SELECTION
    total_fitness = cpop(n);
    % use roulette selection (-> need pos. fitness!)
    for i=1:n
        p=rand*total_fitness;
        % now find first index
        j=find(cpop-p>0);
        if isempty(j)
            j=n;
        else
            j=j(1);
        end
        parentx(i)=ipopx(j);
        parenty(i)=ipopy(j);
        parentt(i)=ipopt(j);
       %parents(i)=ipops(j);
    end
  %
 % pop, fpop, parent,pause

    % REPRODUCTION
    % parents 2i-1 and 2i make two new children 2i-1 and 2i

    % crossover
    % use arithmetic crossover
    for i=1:2:n
        r=rand;
        ipopx(i)   =     r*parentx(i) + (1-r)*parentx(i+1);
        ipopx(i+1) = (1-r)*parentx(i) +     r*parentx(i+1);
```

```
        ipopy(i)   =     r*parenty(i) + (1-r)*parenty(i+1);
        ipopy(i+1) = (1-r)*parenty(i) +     r*parenty(i+1);

        ipopt(i)   =     r*parentt(i) + (1-r)*parentt(i+1);
        ipopt(i+1) = (1-r)*parentt(i) +     r*parentt(i+1);

        %ipops(i)   =     r*parents(i) + (1-r)*parents(i+1);
        %ipops(i+1) = (1-r)*parents(i) +     r*parents(i+1);
    end

    % mutation
    % use uniform mutation
    for i=1:nummut
        z=rand;
        %ipopx(ceil(z*n)) = x(1) + z*(x(n)-x(1));
        %ipopy(ceil(z*n)) = y(1) + z*(y(n)-y(1));
        %ipopt(ceil(z*n)) = t(1) + z*(t(n)-t(1));
        %ipops(ceil(z*n)) = s(1) + z*(s(n)-s(1));
        ipopx(ceil(z*n)) = xdis(1) + z*x;
        ipopy(ceil(z*n)) = ydis(1) + z*y;
        ipopt(ceil(z*n)) = rot(1) + z*t;
    end
end
ipopx,ipopy,ipopt
for i=1:n, fpop(i) =-feval(f, ipopx(i),ipopy(i),ipopt(i)); end
fpop
ffinal=max(fpop);
j=find(fpop==ffinal);
       if isempty(j)
            j=n;
        else
            j=j(1);
        end
%f,j
xfinal=ipopx(j);
yfinal=ipopy(j);
tfinal=ipopt(j);
%sfinal=ipops(j);
xfinal,yfinal,tfinal,ffinal

%load image
IM1=imread('image.jpg');
IM2=imread('imager.jpg');
IM1=double(IM1);
IM2=double(IM2);
IM3=double(IM2);
%figure(2);
%imshow(IM2);
%z1=imabsdiff(IM1,IM2);
%x=[52.0707;44.7016;-13.5511;.5240]
%sfinal=2;
%IM2=imresize(IM2, sfinal(1), 'bilinear');
J=imrotate(double(IM2),tfinal(1),'bilinear'); %rotated cropped IMAGE2
```

```
[n1 n2]=size(IM1);
[n3 n4]=size(J);
   if xfinal>n3-n1
        xfinal=n3-n1-1;
        %IM1(1:n1, 1:n2)=255;
    end

    if yfinal>n4-n2
       yfinal=n4-n2-1;
        %IM1(1:n1, 1:n2)=255;
    end

    if xfinal<0
        xfinal=0;
        %IM1(1:n1, 1:n2)=255;
    end

    if yfinal<0
        yfinal=0;
        %IM1(1:n1, 1:n2)=255;
    end

position1=1:n1;
position2=1:n2;
xx=round(position1+xfinal);
yy=round(position2+yfinal);

IM2=round(J(xx, yy));
%IM2=imresize(IM2,.5, 'bilinear');
[n11 n22]=size(IM2);
n11, n22
%z2=imabsdiff(IM1,IM2);
c3=imsubtract(IM2,IM1);
%c2
err=abs(sum(sum(c3))/(n11*n22))
%c2=corr2(IM1,IM2)
subplot(1,3,1), imshow(IM1, [ ]), title('Image 1')
subplot(1,3,2), imshow(IM3, [ ]), title('Image 2')
subplot(1,3,3), imshow(IM2, [ ]), title('Registered Image 2')
toc
```

image_fit.m

```
function f=image_fit1n(x,y,t)

%load image
IM1=imread('imager.jpg');
IM2=imread('image.jpg');
IM1=double(IM1);
IM2=double(IM2);
% s=2;
%IM2=imresize(IM2, s, 'bilinear');
J=imrotate(double(IM2), t,'bilinear'); %rotated cropped IMAGE2
J=abs(J)*255/max(max(J));

[n1 n2]=size(IM1);
```

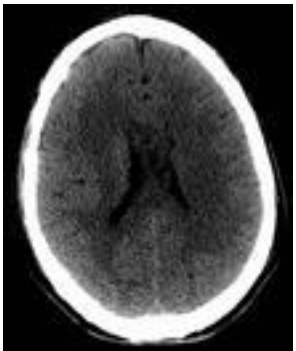
```
[n3 n4]=size(J);
%n1, n2,n3,n4

if ((n3<n1)&(n44<n2))
    x=1:n3;
    y=1:n4;
    IM1=IM1(x,y);
end
if ((n3<n1)&(n4>=n2))
    x=1:n3;
    y=1:n2;
    IM1=IM1(x,y);
end
if ((n3>=n1)&(n4<n2))
    x=1:n1;
    y=1:n4;
    IM1=IM1(x,y);
end
%if n1>n3-x/2
    %f=1000;
    %message=strvcat('The scaling factor is too small.', 'Press Ctrl+C to
stop.',...
        %'Increase x0(4) and restart.');
    %disp('Press Ctrl+C to stop.')
    %Errordlg(message)
    %pause;
%else
[n1 n2]=size(IM1);
[n3 n4]=size(J);
    if x>n3-n1
        x=n3-n1-1;
        IM1(1:n1, 1:n2)=255;
    end

    if y>n4-n2
       y=n4-n2-1;
        IM1(1:n1, 1:n2)=255;
    end

    if x<0
        x=0;
        IM1(1:n1, 1:n2)=255;
    end

    if y<0
        y=0;
        IM1(1:n1, 1:n2)=255;
    end
   if ((x<=n3-n1)&(y<=n4-n2))
    xt=1:n1;
    yt=1:n2;

    xx=round(xt+x);
    yy=round(yt+y);
```

```
    IM2=round(J(xx, yy)); % selecting part of IMAGE2 matching the size of
IMAHE1
   end
   if ((x<=n3-n1)&(y>n4-n2))
    xt=1:n1;
    yt=1:n2;

    xx=round(xt+x);
    yy=round(yt);

    IM2=round(J(xx, yy)); % selecting part of IMAGE2 matching the size of
IMAHE1
   end
   if ((x>n3-n1)&(y<=n4-n2))
    xt=1:n1;
    yt=1:n2;

    xx=round(xt);
    yy=round(yt+y);


    IM2=round(J(xx, yy)); % selecting part of IMAGE2 matching the size of
IMAHE1
   end
   if ((x>n3-n1)&(y>n4-n2))
    xt=1:n1;
    yt=1:n2;

    xx=round(xt);
    yy=round(yt);


    IM2=round(J(xx, yy)); % selecting part of IMAGE2 matching the size of
IMAHE1
   end

    rows=size(IM1,1);
    cols=size(IM2,2);
    N=256;

    h=zeros(N,N);

    for ii=1:rows;    %  col
        for jj=1:cols;   %   rows
            h(IM1(ii,jj)+1,IM2(ii,jj)+1)= h(IM1(ii,jj)+1,IM2(ii,jj)+1)+1;
        end
    end

    [r,c] = size(h);
    b= h./(r*c); % normalized joint histogram
    y_marg=sum(b); %sum of the rows of normalized joint histogram
    x_marg=sum(b');%sum of columns of normalized joint histogran

    Hy=0;
    for i=1:c;    %  col
```

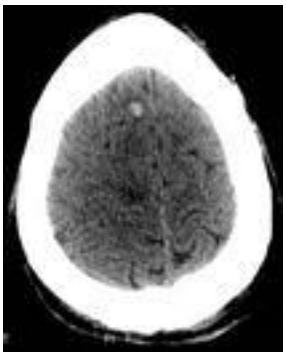
```
        if( y_marg(i)==0 )
            %do nothing
        else
            Hy = Hy + -(y_marg(i)*(log2(y_marg(i)))); %marginal entropy for
image 1
        end
    end

    Hx=0;
    for i=1:r;    %rows
        if( x_marg(i)==0 )
            %do nothing
        else
            Hx = Hx + -(x_marg(i)*(log2(x_marg(i)))); %marginal entropy for
image 2
        end
    end
    h_xy = -sum(sum(b.*(log2(b+(b==0))))); % joint entropy

    f=-(Hx+Hy-h_xy);% Mutual information
    %x
end
```

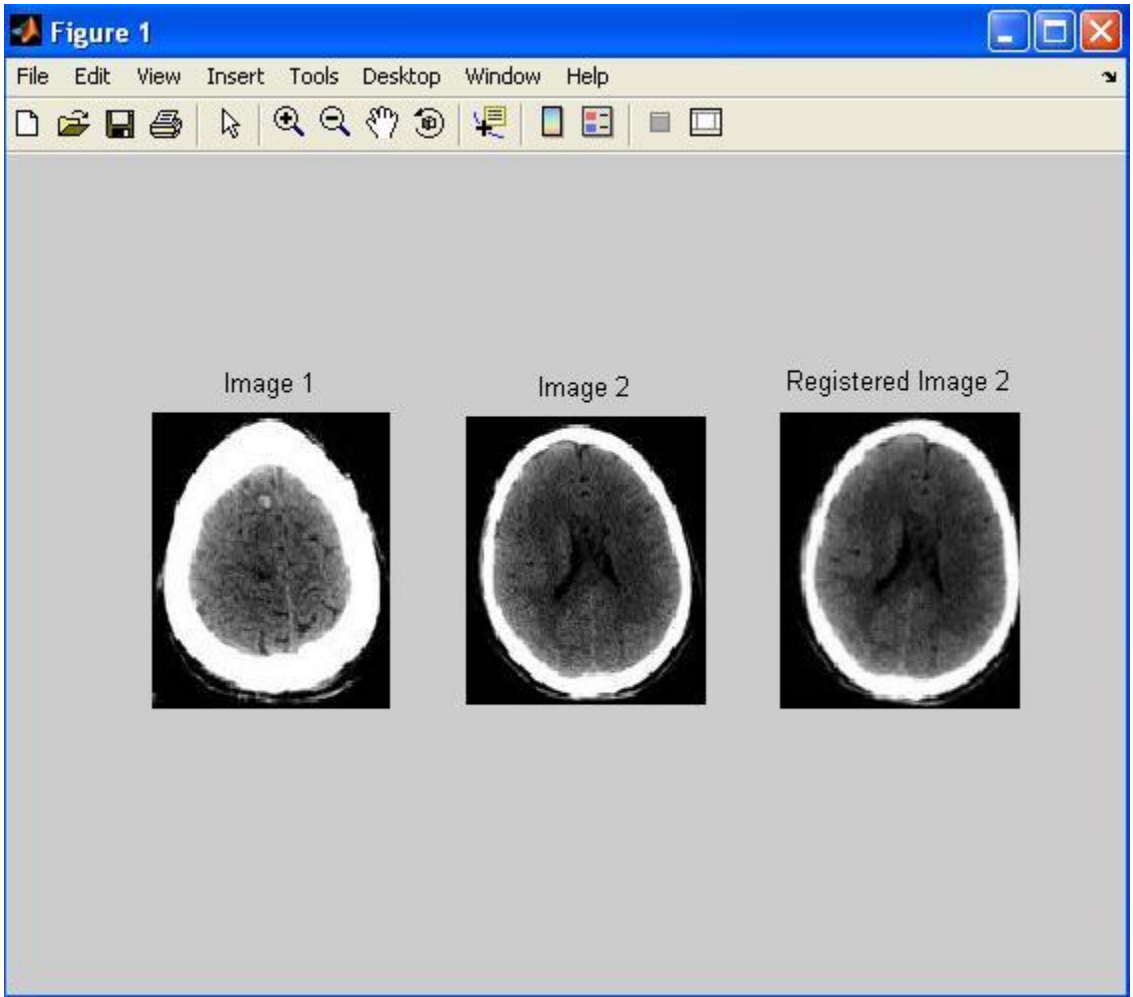
6.2 Output

6.2.1 Refrence Image



6.2.2 Input Image





6.2.3 Registered Image

CHAPTER 7

CONCLUSION AND FUTURE WORK

7. Conclusion and Future Work

Using MATLAB in the windows XP SP2 environment with 512 MB of RAM, the image registration program is executed and gives the registered image. The image is then to be applied for the fractal image compression. For the fractal image compression, all the techniques used for, the common problem is to anyhow reduce the searching time, which in this algorithm is done by reducing the size of the domain pool. The search time referred is the time taken to search the best domain block for each of the corresponding range block. So, for the sake of future work, the compression method can be made faster by reducing the domain pool using some different technique.

8. Refrences

[1] Barbara Zitova´, Jan Flusser**, "**Image registration methods: a survey", Image and Vision Computing 21 (2003) 977–1000.

[2] Geoffrey Egnal, Kostas Daniilidis, **"**Image Registration Using Mutual Information**"** University of Pennsylvania Department of Computer and Information Science Technical, Report No. MS-CIS-00-05.

[3] Frederic Maes, Dirk Vandermeulen, and Paul Suetens, "Medical Image Registration Using Mutual Information", PROCEEDINGS OF THE IEEE, VOL. 91, NO. 10, OCTOBER 2003.

[4] Giuseppe Pascal, Luigi Troiano,"A Niche Based Genetic Algorithm For Image Registration**"**, Proceedings of the ICEIS 2007 - Ninth International conference on Enterprise Information Systems, Volume AIDSS, Funchal, Madeira, Portugal, June 12-16, 2007.

[5] Josien P. W. Pluim, *J. B. Antoine Maintz, and Max A. Viergever. "*Mutual-Information-Based Registration of Medical Images: A Survey**",** IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 22, NO. 8, AUGUST 2003 **.**

[6] Flávio Luiz Seixas, Luiz Satoru Ochi, Aura Conci, Débora C. M. Saade, "Image Registration Using Genetic Algorithms", *GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.* ACM 978-1-60558-130-

9/08/07.

[7] L. G. Brown, “A survey of image registration techniques,” *ACMComput. Surv.*, vol. 24, no. 4, pp. 325–376, Dec. 1992.

[8] Z. D. Liu “Fractal image compression methods: A review” Proceedings of the Third International Conference on Information Technology and Applications (ICITA`05) 1/2005 IEEE

[9] M. Barnsley and L. Hurd. Fractal Image Compression. on Image Processing: Mathematical Methods and Applications. pp. 183-210, Clarendon Press, Oxford, 1997.

[10] Y. Fisher. Fractal Image Compression: Theory and Applications. Springer-Verlag, New York,1994.

[11] C. M. Kung, P. L. Lee, and J. H. Jeng. Fast fractal image compression using visual-based structure similarty index. *The 21th IPPR Conference on Computer Vision, Graphics,and Image Processing,*August 2008.

[12] G. Caso, P. Obrador and C.-C. Kuo. Fast Methods for Fractal Image Encoding. Proc. SPIE Visual Communication and Image Processing 95, Vol. 2501, pp.583-594, 1995.

[13] B. Bani-Eqbal. Enhancing the Speed of Fractal Image Compression. Optical Engineering, Vol. 34, No.6, pp.1705-1710, June 1995.

[14] C.S. Tong and W. Man. Adaptive Approximation Nearest Neighbor Search for Fractal Image Compression. IEEE Trans. on Image Processing, 11(6), pp.605-615, 2002.

[15] E.W. Jacobs, Y. Fisher, and R.D. Boss. Image Compression: A study of the Iterated Transform Method. Signal Process, Vol. 29, pp. 251-263, 1992.

[16] D. Saupe. Lean Domain Pools for Fractal Image Compression. Proceedings IS&T/SPIE 1996 Symposium on Electronic Imaging: Science & Technology Still Image Compression II, Vol. 2669, Jane 1996.

[17] M. Polvere and M. Nappi. Speed-Up in Fractal Image Coding: Comparison of Methods. IEEE Trans. on Image Processing, Vol. 9, No. 6, pp.1002-1009, June 2000.

[18] B. Wohlberg and G. Jager. A review of the Fractal Image Compression Literature. IEEE Trans. on Image Processing, Vol. 8, No. 12, pp. 1716-1729, Dec. 1999.

[19] D. Saupe and R. Hamzaoui. Complexity Reduction Methods for Fractal Image Compression. In I.M.A. Conf. Proc. on Image Processing; Mathematical methods and applications, Sept., J.M. Blackedge (ed.), 1994

[20] C.S. Tong, and M. Pi. Fast Fractal Encoding Based on Adaptive Search. IEEE Trans. on Image Proc.10, No.9, pp.1269-1277, Sept. 2001.

[21] Barnsley, M. F., 1988. *Fractals Everywhere*. New York: Academic Press Inc., London.

[22] Barnsley, M. F., Sloan, A., 1988. A Better Way to Compress Images. *Byte* 88(1):25-32.

[23] Barnsley, M. F., and Hurd, L. P. 1993. *Fractal Image Compression*. AK Peters, Ltd., Wellesley, Massachusetts.

[24] Fisher, Y., 1992. Fractal Image Compression Using Iterated Transforms, in *Image and Text Processing*, edited by Storer, J. A., Kluwer Academic, Boston. pp 34-61.

[25] Fisher, Y., 1997. *Fractal Image Compression: Theory and Application*. Springer- Verlag, New York.

[26] Gomes J. and Velho, L., 1997. *Image Processing for Computer Graphics*. Springer Verlag, New York.

[27] Jacquin, A. E., 1992. Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations. *IEEE Transaction on Image Processing*, 1(1): 18-30.

[28] Jacquin, A. E., 1993. Fractal Image Coding: A Review. *Proceedings of the IEEE*, 81(10): 1451-1465.

[29] Knipe, J. and Li, X., 1998. *On the Reconstruction of Quadtree Data*. *IEEE Transactions on Image Processing* 7(12): 1653-1660.

[30] Mandelbrot, B. B., 1967. How Long is the Coast of Britain? Statistical Self- similarity and Fractional Dimension. *Science* 156: 636-638.

[31] Mandelbrot, B. B., 1983. *The Fractal Geometry of Nature*. Freeman , New York,.

[32] Saupe, D. and Hamzaoui, R., 1994. A Review of the Fractal Image Compression Literature. *Computer Graphics* 28(5): 268-276.

[33] Wohlberg, B. and Jager, G., 1999. A review of the fractal image coding literature .*IEEE Transactions on Image Processing* 8(12): 1716-1729.