Range Image Compression

by

Arif Obaid, BASc.

A thesis submitted to the
School of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

MASTER OF APPLIED SCIENCE

Ottawa-Carleton Institute of Electrical Engineering

Department of Electrical Engineering
Faculty of Engineering
University of Ottawa

November 1994

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

ISBN 0-612-15657-5

Canada

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

To my mother and father.....

Acknowledgments

I would like express my sincere appreciation to my thesis supervisor, Dr. Jean-François Rivest, for his patience and guidance during my thesis work. I remain indebted to him for his support and encouragement during the course of this dissertation. Special thanks to Mr. Guy Godin of the National Research Council for his suggestions.

I would also like to thank all the support staff members at the Electrical Engineering department of the University of Ottawa, especially Ms. Lucette Lepage, Ms. Michèle Roy, and Ms. Amanda Lauzon.

My special thanks to Steven Caron for providing many academic and non-academic discussions and to Shahid Chaudry for providing moral support. I would also like to thank Grant Henderson of the Multimedia Communications Laboratory for providing technical help and Amit Jain for helping me during the writing stage of this thesis.

I am thankful to the Telecommunication Research Institute of Ontario (TRIO) for its financial support.

ABSTRACT

Range Images, which are a representation of the surface of a 3-D object, are gaining popularity in many applications including CAD/CAM, multimedia and virtual reality. There is, thus, a need for compression of these 3-D images. Current standards for still image compression, such as JPEG, are not appropriate for such images because they have been designed specifically for intensity images. This has led us to develop a new compression method for range images. It first scans the image so that the pixels are arranged into a sequence. It then approximates this sequence by straight line segments within a user-specified maximum tolerance level. The extremities of the straight-line segments are non-redundant points (NRPs). Huffman coding, with a fixed Huffman tree, is used to encode the distance between the NRPs and their altitudes. A plane-filling scanning technique, known as Peano scanning, is used to improve performance. The algorithms performance is assessed on range images acquired from the Institute for Information Technology of the National Research Council of Canada. The proposed method performs better than JPEG for any given maximum tolerance level. The adaptive mode of the algorithm is also presented along with its performance assessment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Image Compression

The demand for handling digital images is growing rapidly. This is due to the recent technological advances in image acquisition equipment which has led to the use of digitized images in such diverse applications as magnetic resonance imaging (MRI), computed tomography (CT), computer aided design / computer aided manufacturing (CAD/CAM), multimedia, broadcasting, astronomy, remote sensing, facsimile transmission, law enforcement, and advertising [1]. The reason for this enormous interest in digital images is because of the ease with which visual information can be manipulated using digital rather than analog images. A digital signal has many advantages over its analog counterpart in terms of the processing flexibility, robustness to channel errors, and high signal to noise ratio (SNR).

Despite its advantages, there is one major problem with digital images: the large channel bandwidth required for transmission and memory required for storage necessitates the use of compression techniques, irrespective of the advent of broadband networks and advances in storage technology. For example, a 16-bit digital image, 256 by 256 pixels, requires more than 1 Mbits for a complete representation.

Given the need for image compression, the question arises as to how images can be compressed at all. Digital images consist of data that is often redundant or irrelevant and can, therefore, be removed. Redundancy, which is a function of the statistical properties of images, is classified into the following three types [2]:

- spatial, which is due to the correlation between neighboring pixels in an image
- spectral, which is due to the correlation between color planes or spectral bands
- temporal, which is due to the correlation between neighboring frames in a sequence of images.

Irrelevancy, on the other hand, is a function of an observer viewing an image. Limitations of the human visual system (HVS), such as its inability to perceive high spatial frequencies, can also be exploited in many applications [3].

Image compression techniques ideally try to remove both redundant and irrelevant information and then efficiently encode what remains. In practice, however, it is often necessary to throw away some nonredundant and relevant information as well, in order to achieve the necessary degree of compression. These compression techniques can generally be classified as either lossless or lossy [4]. Lossless compression techniques (also known as bit-preserving or reversible compression techniques) preserve all the information in the image and allow exact reconstruction of the original image on a pixel-by-pixel basis. Lossy techniques (also known as irreversible compression techniques), on the other hand, do not preserve all the information, and therefore, introduce some degradation in the reconstructed image. Obviously, lossless compression is ideally desired since no information is compromised; however, since lossy techniques give higher compression than lossless techniques do, they have become much more attractive for applications which can tolerate some form of degradation in the image.

Recently, the Joint Photographic Experts Group (JPEG) [5], of the International Standards Organization (ISO) and the International Consultative Committee on Telephony and

Telegraphy (CCITT), has proposed a standard for still image compression. The standard uses state-of-the-art technology and has found widespread use in image compression applications.

1.2 Problem Definition

A great deal of research has been done on image compression using a variety of techniques; each technique exploiting the spatial, spectral, and temporal redundancies found in an image [6,7]. All of the techniques researched so far have been designed for *intensity* images, which are images which store the relative irradiance, or brightness, of each object in a scene. In this thesis, we have designed a lossy compression algorithm for *range images*.

A range image is a collection of *distance measurements* from a known reference coordinate system to surface points on *objects* in a *scene* [8]. Each pixel represents a distance measurement instead of an intensity value. Range images are known by many names, depending on the context, including depth images, 3-D images, digital terrain map (DTM), topographic map, contour map, surface profiles, etc. A typical range image taken from a range image database [9] is shown in fig. 1.1.

Range images have many advantages over intensity images. When one wants to infer a three-dimensional structure from intensity images, one experiences ambiguity problems that are not found in range images, such as [10]:

- rotation and translation invariance: pixel relationships do not change with rotation or translation of range image unlike that of intensity images.

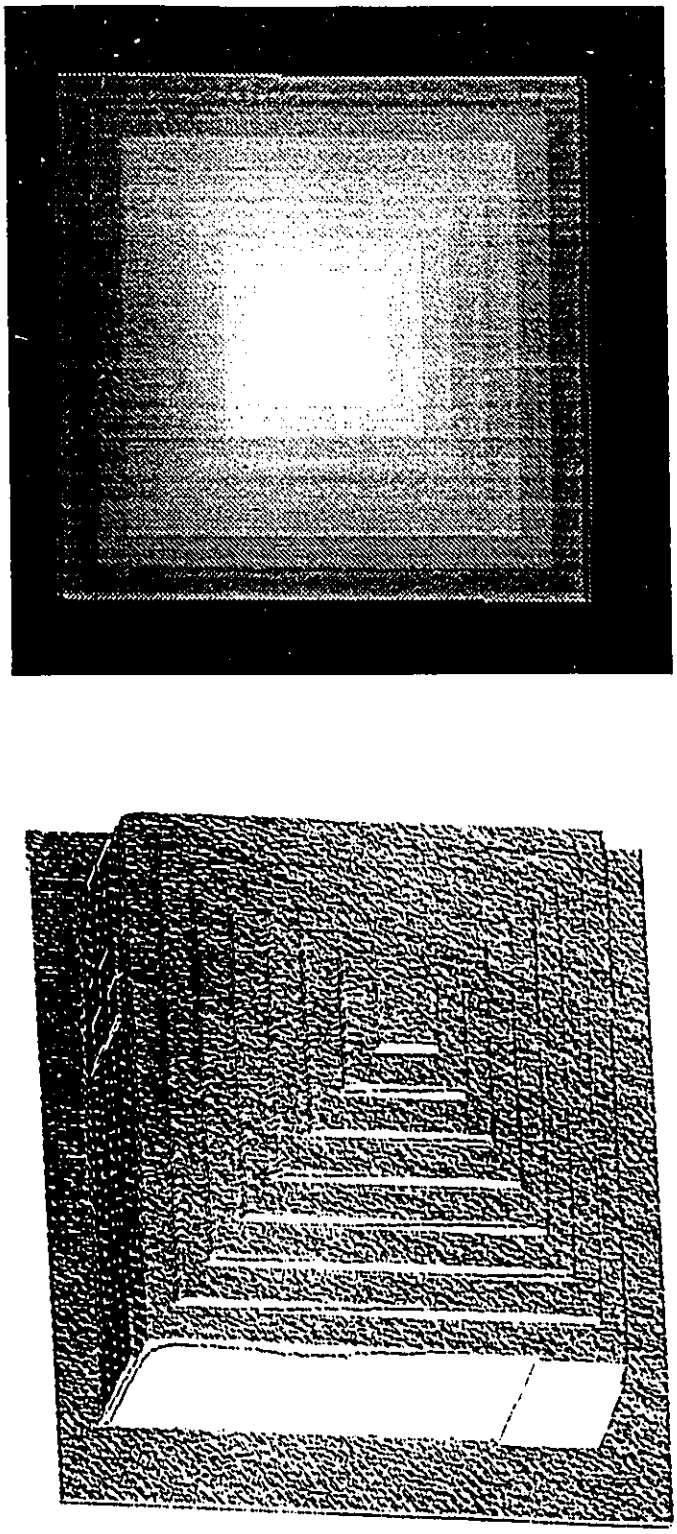


Figure 1.1 2-D and 3-D representation of a range image

- insensitivity to background illumination, surface color, and object texture: range images require no modifications when the illumination source, its spectral characteristics, or its orientation relative to the scanner change.
- scaling invariance: a change in distance from the scanner does not require a scaling factor for range images. Measurement to the surface of the object are absolute.

Another important advantage of range images over intensity images is the lower noise level experienced by range images. This is due to the smooth surface variations that range images possess.

Intensity images are generally meant to be seen by a human observer. Most of the lossy techniques take advantage of the HVS properties. In contrast, range images are not directly visualized. They are generally heavily processed prior to visualization; they are rotated, rendered, shaded, presented using wire-frames, etc. In fact, if they are processed for some industrial application, then the computer would be the only observer. Consequently, the degradation criteria that are appropriate for intensity images are not appropriate for range images.

The research community in still image data compression reluctantly uses the mean-squared-error (MSE) to asses image degradations. More elaborate testing involves human viewers. In many range image applications, however, such a criterion is not appropriate. For example, in CAD/CAM, robot navigation, assembly, etc., the critical issue is *maximum absolute error*; any other criterion is not acceptable. Any compression algorithm designed for range image compression, therefore, must be amenable to this requirement.

Range images are acquired using a *range-imaging sensor*, which is any combination of hardware and software capable of producing a range image of a real-world scene under appropriate operating conditions [8]. Range imaging sensors are unique devices in that the

image data points explicitly represent the scene surface geometry in the sampled form. There are four basic types of range sensors [8,11]:

- Point sensor: it measures the distance to a single visible surface point from a single viewpoint along a single ray. The scene objects have to be physically moved in two directions for a range image to be formed.
- Line sensor: it measures the distance to visible surface points that lie in a single 3-D plane that contains the single viewpoint or viewing direction. The scene objects have to be moved orthogonal to the sensing plane for a range image to be formed.
- Field of View sensor: It measures the distance to many visible surface points that lie within a given field of view relative to a single viewpoint or viewing direction. The range image is formed directly using this type of sensor; no scene motion is required.
- Multiple view sensor: it locates surface points relative to more than one viewpoint or viewing direction because all surface points of interest are not visible. It also does not require any scene motion to form range images.

The choice of a range-imaging sensor depends on viewing constraints, scanning mechanisms, object movement possibilities and application.

1.3 Investigated Approach

In this thesis, the problem of 3-D image compression is addressed. Image compression techniques that are available today do not work properly on range images because they are not designed to do so. JPEG is also inappropriate for these images because of the same reason. Therefore, we have developed an algorithm more suitable for these images. It is one of the first techniques specifically designed to compress range images.

Our compression algorithm is a simple, fast, reliable, robust, and cost effective algorithm that may be implemented in real-time. It operates on image data points sequentially and is, thus, one-dimensional. The image data points are approximated by straight line segments. Redundant points are between the extremities of a segment and are, thus, removed. The algorithm finds the end points of the segments based upon a maximum absolute error criterion requiring that the results of the approximation always fall within a user-specified error range. The error range can be adaptive and is not necessarily symmetrical.

Non-redundant point (NRP) altitudes and the distance between them, or run-lengths, are encoded using Huffman codes. Arithmetic coding and Lempel-Ziv-Welch (LZW) coding are also used in this analysis.

The image data points can be scanned in a variety of ways. Unidirectional scanning, such as raster scanning, and multi-directional scanning, such as Peano scanning, are used to develop the algorithm. Furthermore, NRPs can also be found using a two-directional NRP finder which removes redundancy in two orthogonal directions.

The algorithm performance on various range images is outlined and compared with JPEG, the standard for still image compression.

1.4 Main Contributions

The main contributions of this thesis are summarized below:

- Introduce a novel image compression algorithm that is specifically designed for range images.
- Propose a range image coder-decoder (codec) which uses a fixed Huffman tree formed by using various 3-D images as training images.

- Present an in-depth comparison of the proposed range image compression method with JPEG, the still image compression standard.

1.5 Thesis Organization

The thesis is organized as follows. In chapter 2, a review of image compression techniques is presented. First, we detail information theory concepts, like entropy, rate distortion and performance measures. This is followed by a brief review of lossless and lossy image coding schemes. We review four lossless coding techniques: Huffman coding, arithmetic coding, run-length coding, and Lempel-Ziv-Welsh (LZW) coding. Five lossy techniques are then presented, namely, predictive coding, transform coding, vector quantization, subband coding, and fractal block coding. Finally, we describe JPEG, the standard for still image compression.

In chapter 3, our algorithm for range image compression is outlined. First, we describe various scanning techniques , including raster and Peano scanning. Then, the redundancy finder, which is the heart of the algorithm, is described in detail. Three different techniques to reduce the data set are explained. They include Unidirectional, Two-directional, and Multi-directional schemes. We then present the encoding of the non-redundant information. Huffman coding is used with a fixed Huffman tree; justification is provided and the creation of the tree is described. Arithmetic coding and LZW coding are also presented, as an alternative to Huffman coding. Decoding and interpolation of the decoded data are presented next. Finally, we present the procedure to reconstruct the image.

In chapter 4, we present a performance evaluation of the proposed method together with a comparative study of the algorithm with JPEG. First, the performance of the three different redundancy finding methods are analyzed. Then, a comparison between our algorithm and JPEG is presented using various simple to complex range images.

In chapter 5, we outline the adaptive mode of our algorithm. The user specifies a tolerance image instead of a fixed tolerance value. Performance analysis of the adaptive algorithm is given together with some of its properties.

In this thesis, all the simulations are done on range images acquired at the Institute for Information Technology of the National Research Council of Canada, Ottawa, Canada.

Chapter 2

Image Compression Review

In this chapter, we review some of the basic image compression techniques in use today. We begin by reviewing some information theory concepts, such as entropy and rate distortion, followed by a review of lossless and lossy image coding concepts. These include Huffman, Arithmetic, Run length, and Lempel-Ziv-Welsh for lossless and Predictive, Transform, Vector Quantization, and Sub-band coding for lossy compression techniques. Fractal Block coding, which is a new and emerging technique, is also described. Finally, a brief review of JPEG, which is a standard for image compression, is presented.

2.1 Information Theory

An information generating process can be modeled as source which generates symbols from a finite alphabet. For instance, a k-bit image-generating source produces symbols (pixel intensities) from a finite alphabet consisting of 2^k unique symbols. An estimate of the information content and the minimum bound on the number of bits required to code the symbols are fundamental concepts in information theory. These basic concepts are outlined below.

2.1.1 Entropy

An image source, S, generating images of size $N \times M$ pixels, where each pixel is quantized to G gray levels, can generate $G^{N \times M}$ different image patterns. If the probability of a specific image pattern is given by $p(\underline{x})$ where

$$\underline{x} = \{x_{ij}\} \quad ; \begin{Bmatrix} i = 1, \ldots, N \\ j = 1, \ldots, M \end{Bmatrix} \tag{2.1}$$

where x_{ij} is the $(i,j)th$ element of $\underline{x}$, then the average amount of information of the source, H(S), defined as its entropy [12], is given by

$$H(S) = -\frac{1}{N \times M} \sum_{all\ \underline{x}} p(\underline{x}) \log_2 p(\underline{x}) \quad \text{bits / pixel} \tag{2.2}$$

The source entropy is bounded by

$$0 \leq H(S) \leq \log_2 G \tag{2.3}$$

and the source redundancy is given by

$$\text{redundancy} = \log_2 G - H(S) \tag{2.4}$$

If pixels in the image are all statistically independent of each other, then we can express H(S) as:

$$H(S) = -\frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{all\ x_{ij}} p_{ij}(x_{ij}) \log_2 p_{ij}(x_{ij}) \tag{2.5}$$

where p_{ij} is the probability that the pixel X_{ij} of the image S has a value equal to x_{ij}. However, since the statistical information, or source probability, of an image, $p(\underline{x})$, in practice, is not easy to measure or model, the true entropy of the image is very difficult to obtain. Hence, a simpler measure, the first order entropy $H^1(S)$, which is defined on a pixel-by-pixel basis, is often used. It is defined as:

$$H^1(S) = -\sum_{g=1}^{N} P_g \log_2 P_g \tag{2.6}$$

where P_g is the probability of the occurrence of the gray level g. If the pixels of the image are identically and independently distributed (i.i.d), then the entropy of the image is equal to the first order entropy. For 6-bit typical image data, the first order entropy has been estimated at 1.9 bits/pixel [13]. The first order entropy is often referred to as the *memoryless entropy* as it provides a lower bound on the bit rate required for lossless reproduction without exploiting the correlation between pixels.

According to Shannon's *noiseless source coding theorem* [14,15], any source can be losslessly encoded with a code whose average number of bits per symbol is arbitrarily close to, but not less than, the source entropy, $H(S)$. This minimum limit provides a useful criterion for evaluating the performance of various compression techniques.

2.1.2 Rate Distortion Theory

Complementing Shannon's theoretical bound for lossless compression, a theoretical bound for lossy compression for a specified distortion can also be estimated. Rate distortion theory provides a means to determine this limit [16]. The level of degradation is usually controlled by the user by adjusting a set of parameters, such as quantization interval. For a given degradation, or distortion, D, the information content of the degraded source image

is estimated from a rate distortion function, $R(D)$. The rate distortion function specifies the minimum rate at which information about the source output can be achieved, in order to be able to reproduce it with an average distortion that does not exceed D [16].

The average mutual information, $I_{NxM}(S,\hat{S})$, between the source and the receiver, defined as:

$$I_{NxM} = \sum_{all\ x,\hat{x}} p(x)\, p(\hat{x}/x)\ \log_2 \frac{p(\hat{x}/x)}{\sum_{all\ x} p(x)\, p(\hat{x}/x)} \tag{2.7}$$

is a measure of the statistical dependence between the source image S and the decoded image $\hat{S}$. Using the axioms of conditional probability, equation (2.7) can be simplified to give

$$I_{NxM} = \frac{1}{N \times M} \sum_{all\ x,\hat{x}} p(x,\hat{x})\ \log_2 \frac{p(\hat{x}/x)}{p(\hat{x})} \tag{2.8}$$

It can be shown that the mutual information between S and $\hat{S}$ is the difference in the entropy of the source image S and the entropy of the same source image S given the reconstructed image $\hat{S}$, i.e.

$$I_{NxM} = H(S) - H(S/\hat{S}) \tag{2.9}$$

For lossless encoding, the average mutual information is given by:

$$I_{NxM} = H(S) - H(S/\hat{S}) = H(S) \tag{2.10}$$

since $H(S/\hat{S}) = 0$ in this case. Furthermore, the mutual information is zero when no transmission occurs between the transmitter and the receiver. Thus, I_{NxM} is bounded by:

$$0 \le I_{NxM} \le H(S) \tag{2.11}$$

The rate distortion function, $R(D)$, can then be defined as the minimum average mutual information between $S$and $\hat{S}$ at a constraint of a fixed average distortion D, as the image size, NxM, tends to infinity [17], i.e.

$$R(D) = \lim_{NxM \to \infty} \left\{ \inf_{all\ p(\hat{x}/x)} I_{NxM}(S,\hat{S}) \right\} \tag{2.12}$$

It should be noted that the distortion D and the mutual information I_{NxM} depend on the type of image compression technique used. Furthermore, there is a minimum value of I_{NxM} that is needed to ensure that the average distortion does not exceed the specified upper limit, D. This minimum value is equal to $R(D)$ [18]. Also, the bit rate required to achieve exact reconstruction (D=0) is the source entropy $H(X)$,

$$R(0) = H(X) \tag{2.13}$$

2.1.3 Distortion Measures

A distortion, or fidelity, measure is a cost function, $d(x,\hat{x})$, used to evaluate compression techniques. The performance of these source coding schemes can be evaluated simply by determining the average distortion introduced by the coding system, $E[d(S,\hat{S})]$, which is,

$$E[d(S,\hat{S})] = \sum_{all\ x,\hat{x}} d(x,\hat{x})\ p(x,\hat{x}) \tag{2.14}$$

where $\hat{x}$ is the output of the input x.

A widely used measures of reconstructed image fidelity, for an NxM size image is the

average mean-squared error (MSE) [6], defined as:

$$e_{ms}^2 = \frac{1}{NxM}\sum_{i=1}^{N}\sum_{j=1}^{M} E\left[(x_{ij} - \hat{x}_{ij})^2\right] \tag{2.15}$$

In practice, the average MSE is often estimated by the average sample mean error, given as:

$$e_{ms}^2 \approx \frac{1}{NxM}\sum_{i=1}^{N}\sum_{j=1}^{M} (x_{ij} - \hat{x}_{ij})^2 \tag{2.16}$$

The SNR corresponding to the above error is usually calculated as:

$$SNR = 10\ \log_{10}\left(\frac{Peak\ Signal\ Value^2}{Mean\ Square\ Error}\right) \tag{2.17}$$

Other distortion measures include Normalized Mean Square Error (NMSE), and its corresponding Signal to Noise Ratio, Mean Absolute Error (MAE), and Maximum Absolute Error (MaxAE), which are defined as follows:

$$NMSE = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(x_{ij} - \hat{x}_{ij})^2}{\sum_{i=1}^{N}\sum_{j=1}^{M}(x_{ij})^2} \tag{2.18}$$

$$SNR = 10\ \log_{10}\left(\frac{1}{NMSE}\right) \tag{2.19}$$

$$MAE = \frac{1}{N\ x\ M}\sum_{i=1}^{N}\sum_{j=1}^{M}(|x_{ij} - \hat{x}_{ij}|) \tag{2.20}$$

$$MaxAE = \max_{i,j}\ (|x_{ij} - \hat{x}_{ij}|) \tag{2.21}$$

where $(|x_{ij} - \hat{x}_{ij}|)$ is the absolute difference between the original image pixel x_{ij} and the reconstructed image pixel $\hat{x}_{ij}$.

The fidelity measures described above may not always be desirable, especially when dealing with images which have to be evaluated visually [19-21]. Therefore, several visual distortion measures have been recently suggested, such as weighted MSE of the *contrast*, instead of the intensity [22,23]. Furthermore, because of the poor correlation between the distortion measures outlined in equations (2.18-2.21) and subjective ratings, considerable effort is being made in the research community to define newer measures to increase this correlation [24,25].

2.2 Lossless Compression

A number of image coding applications require complete reconstruction of compression images. For instance, in medical imaging [26], complete reconstruction of X-rays is necessary to ensure accurate diagnosis. Lossless compression techniques, must, therefore, be reversible processes. i.e. the original image must be recovered accurately. Some of the more common lossless coding strategies used for continuous tone images, such as Huffman, Arithmetic, Run-length, and Lempel-Ziv-Welsh (LZW) coding are briefly surveyed in this section.

2.2.1 Huffman Coding

Huffman coding is a minimum redundancy, prefix code technique that comes arbitrarily close to the minimum bound for compression [27], as defined by Shannon's lossless coding theorem. The Huffman procedure results in a variable length code which assigns

lesser number of bits to symbols with higher probabilities. The coding technique is easy to implement (e.g. the UNIX "compact" program [28]) and is efficiently decodable. However, it has two major drawbacks:

- each symbol is represented by an integral number of bits, implying that the entropy limit cannot be attained unless all the symbol probabilities are negative powers of two.
- the source statistics must either be known a priori or two passes must be made through the data.

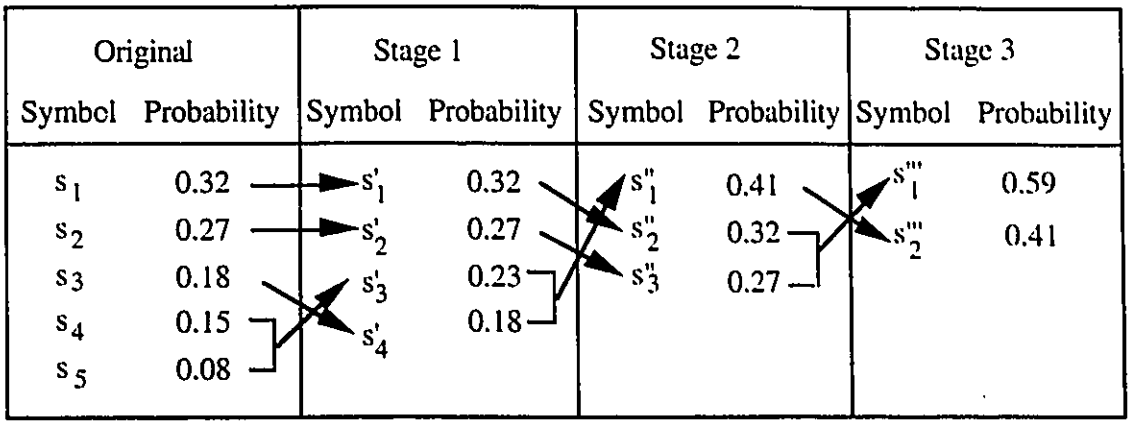
These drawbacks, though, are not very significant when large data sets are encoded.
The efficiency of the Huffman coding scheme is calculated as follows:

$$Efficiency = \frac{H}{R} \text{x } 100\% \tag{2.22}$$

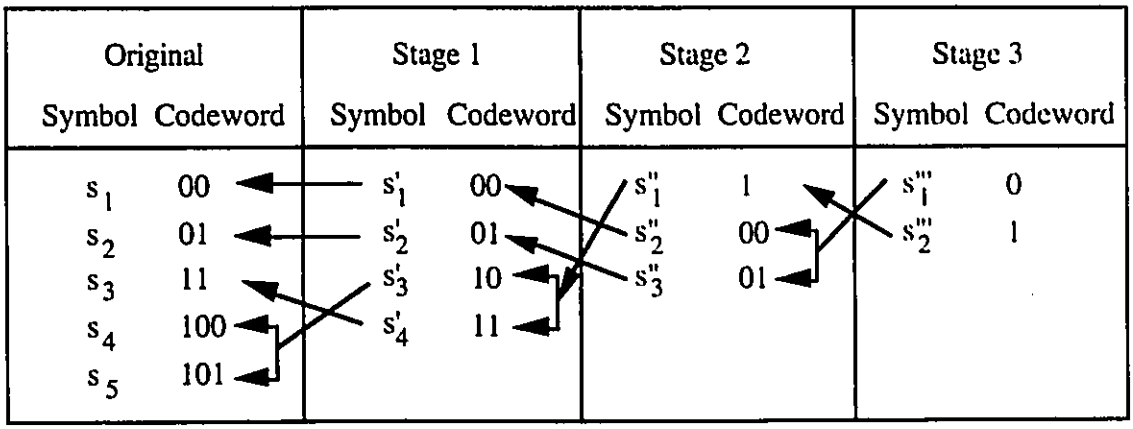
where H is the source entropy and R is the average bit rate after encoding the information source. An efficiency of 100% is obtained when the probabilities of occurrence of all source symbols are negative powers of two [29].
In order to generate Huffman codes for a given source, a Huffman table is made. The table is produced by first ranking all source symbols according to their probability of occurrence. Then, new composite pseudo-symbols are generated by successively combining the lower two symbols and re-ranking the set. This combination process is repeated until a final single composite pseudo-symbol remains. Fig. 2.1(a) describes this source reduction process. Fig. 2.1(b) shows how the Huffman codes are constructed. Each time two pseudo-symbols are combined, one of them is assigned a binary '0' and the other a binary '1'. Codewords are then formed by traversing the table from the single composite pseudo-

symbol left at the end of the table-generating process to each individual source symbol in the set [30].

Decoding is done by simply looking-up the received codes in the Huffman table. This implies that the same Huffman table that was used to produce the codes at the transmitter must be present at the receiver in order to get error-free decoding.



(a)



(b)

Figure 2.1 Huffman code generation process: (a) describes the source reduction process and (b) shows the process of construction of the code

A fixed Huffman tree is generally used to avoid the costly task of transmitting the Huffman table to the receiver. However, if the data is not stationary, then either the table must also be transmitted or an *adaptive* Huffman technique [31,32] must be used.

2.2.2 Arithmetic Coding

Arithmetic coding is an optimal variable length coding technique that encodes data strings by creating a code string which represents a fractional value on the number line between 0 and 1 [33,34]. As the message becomes longer, the interval needed to represent it becomes smaller and the number of bits required to represent the interval increases. The more likely symbols reduce the interval size less than the less likely ones and hence add fewer bits to the message.

For example, let the source symbol be {a, e, i, o, u} with probability of occurrence 0.1, 0.4, 0.15, 0.3 and 0.05 respectively. If the message to be transmitted is "iou", then it is encoded as shown in fig. 2.2. First, the interval [0,1) is divided into five regions corresponding to the symbol probabilities. Then, the first symbol, "i", reduces the range to [0.5, 0.65). This interval is then further subdivided into five regions using the same symbol probability table. The second symbol, "o", narrows the range to [0.5975, 0.6425). Finally, the "u" symbol reduces the range to [0.64025, 0.6425). Any number in this range can be sent as encoding "iou", e.g. 0.641.

There is, however, a problem of knowing when to stop decoding because the number 0.641 represents not simply "iou" but "iou...". To resolve this ambiguity, a special EOM (end of message) symbol is used; because it occurs only once, it has very low probability and gets assigned a very small fraction of the number line. The decoder works similarly to the encoder in that given the number 0.641, it decodes the symbol "i" first since the number lies within the range [0.5,0.65) assigned for it. It then proceeds sequentially. It should be

noted that in order for the decoder to work properly, it must have access to the symbol probability table. This table, therefore, must be transmitted with the encoded data if it is not fixed.

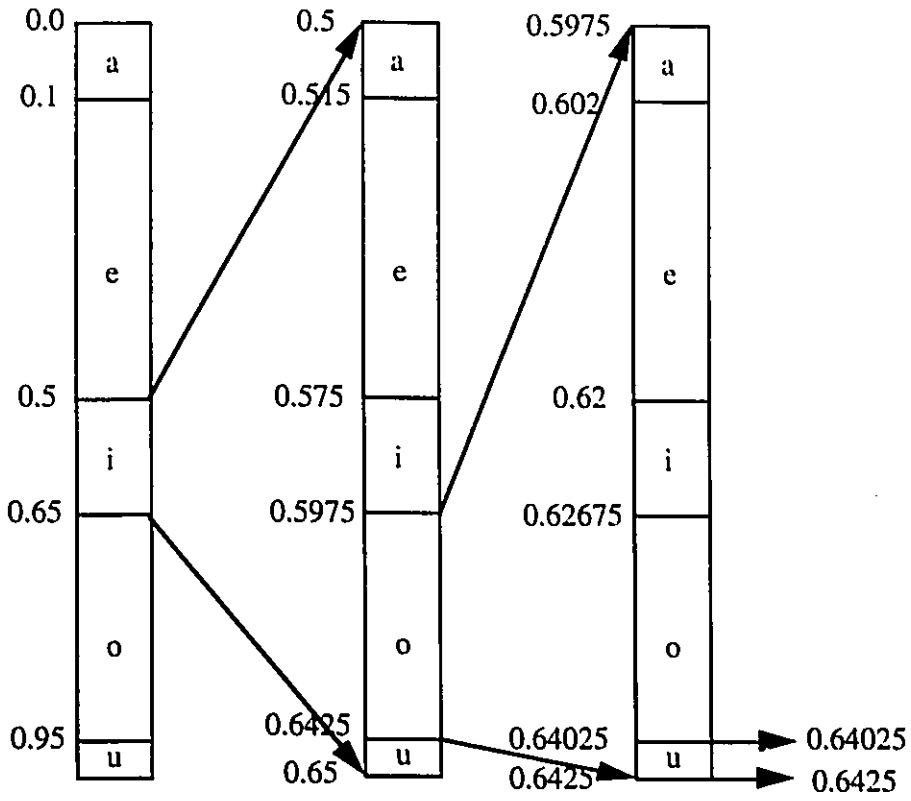


Figure 2.2 Arithmetic coding of the message "iou"

This coding technique may surpass Huffman coding in performance because it achieves fractional bit allocations for source symbols. Huffman coding, on the other hand, is optimal only when the source symbols probabilities are negative powers of two. Furthermore, arithmetic coding is generally faster than Huffman coding and requires only one pass through the data when used adaptively [35]. For high resolution images, such as medical images, though, Huffman and Arithmetic coding have similar performance [72].

Huffman coding should be preferred, however, if the decoder cannot wait for all of the data to be encoded at once before it is transmitted. One example where this is a necessity is in progressive image transmission.

2.2.3 Run-length Coding

Historically, Run length encoding was investigated as a lossless compression technique that could be easily implemented in hardware. It exploits the redundancy found in sources having identical consecutive symbols, such as text, and binary images [37]. For these reasons, it has found extensive usage in facsimile transmission [38].

A *run* is defined as a sequence of consecutive pixels having identical values along a specified direction. Significant bit rate reductions can be achieved when long *runs* are encountered in images. In facsimile applications, each bit plane of the binary image is encoded separately such that runs represent *lengths* of "white" (1) or "black" (2) pixels. These run lengths are generally encoded by some other lossless compression method, such as Huffman or Arithmetic coding.

Run length encoding is not very efficient for complex binary images and non-binary images. Even though compression can be achieved in non-binary images by splitting them into a set of bit planes which are individually run-length coded, it is not very efficient unless the image is inherently simple. However, if the technique is used in conjunction with some other method, it can be highly advantageous. For example, in transform coding [39], the quantized transform coefficients result in long *runs* of zero valued coefficients; run-length coding can be used for efficiently compressing these quantized coefficients.

Run-length coding can also be use in area coding, which can be thought of as a two dimensional coding technique. In area coding [40], an area of connected, continuous

group of pixels of identical value are coded by specifying only the area and the intensity of its pixels, thus significantly reducing redundancy.

2.2.4 Lempel-Ziv-Welsh (LZW) coding

Lempel-Ziv-Welsh (LZW) coding is a lossless compression technique that takes substrings of input characters and maps them into fixed length output codes [41]. The LZW algorithm distinguishes itself from the Lempel-Ziv (LZ) method [42,43] by requiring that for every string in the codebook, its prefix also be in the codebook. 12-bit codecs, which allow 2^{12} entries in the codebook, are generally used in this scheme even though 16-bit codecs have also been implemented. The coding scheme selects the symbol substrings in such a way that all codes in the codebook have almost equal probability of occurrence. Consequently, strings of frequently occurring symbols will contain more symbols in the codebook than strings having infrequent symbols. The method, therefore, exploits character frequency redundancy. Compression is achieved whenever frequently occurring substrings appear in the input sequence. The codebook is transmitted with the codewords so that the decoder can efficiently decipher them.

An example of this process, given by Welsh [41], is illustrated in table 2.1. Given the input data, the symbols are read from left to right. The first character 'a' is assigned a code of 1, and the extended string 'ab' is placed in the codebook and assigned code 4. Similarly 'b' is assigned a code of 2 and its extension ba placed in the table under code 5. Then, the extension of the third symbol, 'ab', is assigned code 4 from the codebook. The process is continued until all symbols have been coded. The codebook thus produced is shown in table 2.2.

Input symbols: a b a b c b a b a b a a a a a a a a
Output codes : 1 2 4 3 5 8 11 12

Table 2.1 Example of LZW coding taken from Welsh [41].

This technique is generally a one-pass procedure which requires no prior information about the input data statistics and has a time of execution proportional to the length of the message to be compressed. It is generally used to compress text, but is certainly not limited to this application.

Input string	code value
a	1
b	2
c	3
ab	4
ba	5
abc	6
cb	7
bab	8
baba	9
aa	10
aaa	11
aaaa	12

Table 2.2 LZW Codebook generation process for input data in table 2.1.

2.3 Lossy Compression

Lossless compression techniques do not, in general, achieve high compression ratios. In most situations, some degradation in quality may be tolerable in order to gain higher compression performance. Some of the more common lossy compression techniques, such as Predictive, Transform, Vector Quantization, and Subband coding are described in this section. A review of an emerging lossy compression technique, known as Fractal Block codir.g is also presented.

2.3.1 Predictive Coding

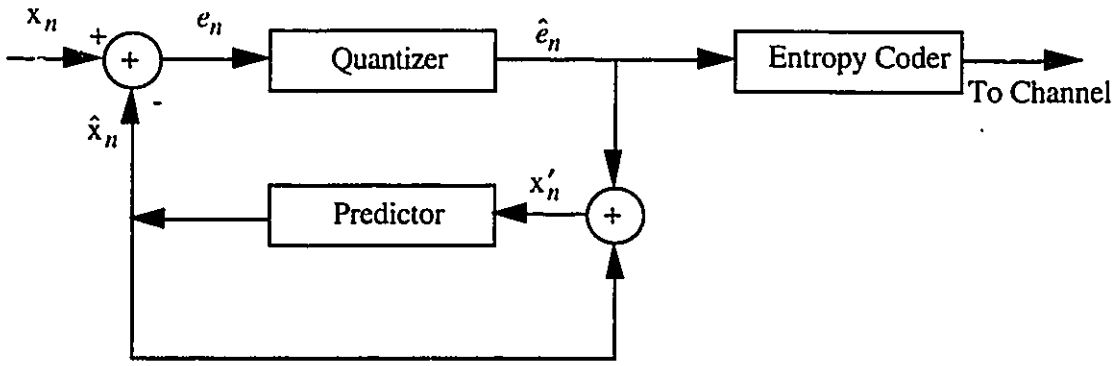
In general, images possess significant amounts of redundant information that can be eliminated to achieve higher compression ratios. This redundant information is evident as high spatial correlation exists between neighboring pixels in an image. Predictive coding techniques try to exploit this spatial redundancy in an efficient manner. Historically, it has been used with success in 1-D speech processing applications [44,45], but has recently gained popularity in image processing applications. Due to the extensive computational complexity in implementing predictive schemes, only the simplest forms of prediction have received much attention in image coding [46]. *Differential pulse code modulation (DPCM)* is the basic compression scheme used in predictive coding techniques.

DPCM

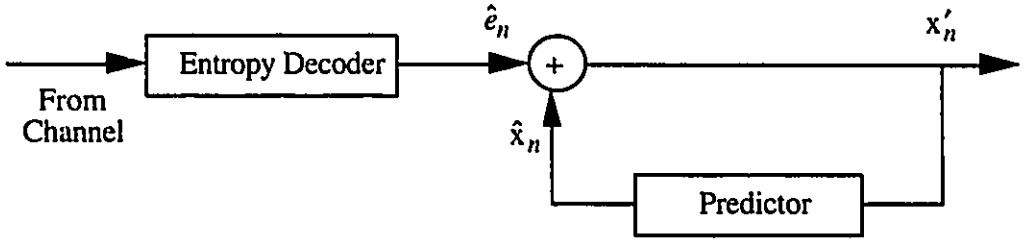
The block diagram of a DPCM system is shown in fig. 2.3. In this method, only the error, or difference, between the current pixel and its predicted value is encoded. DPCM takes advantage of the high spatial correlation in the image to predict the intensity of the current

pixel based upon previous or neighboring pixels. The simplest form of prediction for the current pixel intensity is to use the previous pixel intensity, while a more elaborate prediction rule may be a weighted combination of many neighboring pixels.

As shown in fig. 2.3, the predicted estimate $\hat{x}_n$ is subtracted from the actual pixel value, x_n, and the error, $e_n = x_n - \hat{x}_n$, is quantized, entropy coded and transmitted.



DPCM Transmitter



DPCM Receiver

Figure 2.3 Block diagram of a DPCM system

The quantized error value is in turn used to predict the next data sample x_{n+1}. In general, linear predictions of the form,

$$\hat{x}_n = \sum_{i=1}^{p} w_i x_{n-i} \tag{2.23}$$

are employed [47]. The coefficients $\{w_i\ ;\ i = 1,..p\}$ are called predictor coefficients or tap weights which are computed from the long term statistics of the image data. The predictor can be optimized in terms of the mean-squared error such that the "energy" of the predictor error is minimized. The optimal set of tap coefficients w_i^* can then be obtained by solving the Weiner-Hopt equation [44]:

$$\sum_{i=1}^{p} w_i^* R(j-i) = R(j) \tag{2.24}$$

where $R(j)$ is the autocorrelation function of the source image. The feedback arrangement whereby the output of the DPCM receiver is used for prediction rather than the past input values prevents accumulation of quantization errors at the receiver. Furthermore, it provides prediction gains comparable to the case where the prediction is based on the past input values. Typically, the prediction gain improves with filter order, p, up to order 3 or 4, then saturates [48].

DPCM is designed on the basis that the source image is stationary. However, since many images have statistics that changes significantly, a fixed predictive coder may not yield satisfactory performance. The efficiency and performance of these encoders can be improved by having them adapt to the slowly time-variant statistics of the source image. Two kinds of commonly used adaptive systems include coders with [48]:

- an adaptive predictor with a fixed quantizer, or
- an adaptive quantizer with a fixed predictor.

Both approaches update their coefficients instantaneously, thus keeping track of the changes in the statistics of the input data

DPCM is generally a two-pass process; the first pass generates a differential image and the second one encodes the data using an entropy encoding technique. Predictive coding systems are generally simpler to implement and require lesser amounts of memory than do

other lossy compression techniques. However, compression ratios are generally not very high.

2.3.2 Transform Coding

Transform coding, which is also known as Block Quantization, is a technique that de-correlates the image data by applying a linear transform to the image. The image data is transformed from spatial to frequency domain by the application of this orthogonal transform. The resulting transform coefficients are less correlated than the original data and, therefore, only a finite set of them need to be encoded. Transform coding exploits the fact that, for a typical image, a large amount of energy is concentrated in a small number of transform coefficients. Typically, transform coefficients above a predetermined threshold are discarded, resulting in some loss of detail in the resulting reconstructed image.

Transform coding is generally implemented by partitioning the image into small blocks, typically of 8x8 pixels, which are transformed and encoded separately. Fig. 2.4 outlines a block diagram of a transform coding scheme. Here, an image, $\boldsymbol{I}$, first undergoes an orthogonal transform, T, resulting in the formation of a matrix of transform coefficients, Q,

$$Q = \left(\left(Q_{ij} \right) \right) \tag{2.25}$$

where Q_{ij} are individual transform coefficients. In general, only the lower frequency coefficients are retained since they contain most of the energy of the image. These coefficients are then quantized using a variable step-size quantizer which uses a small step size for the visually important low frequency transform coefficients and a larger step size for the not-so-important higher frequency coefficients. The quantized coefficients, $\hat{Q}_{ij}$,

represented by the matrix $\hat{Q}$ in the figure, are then entropy coded using a variable bit-length encoder specified by a bit allocation map. The inverse transform, T^{-1}, is performed at the receiver to recover the coded image after all the coefficients have been de-quantized.

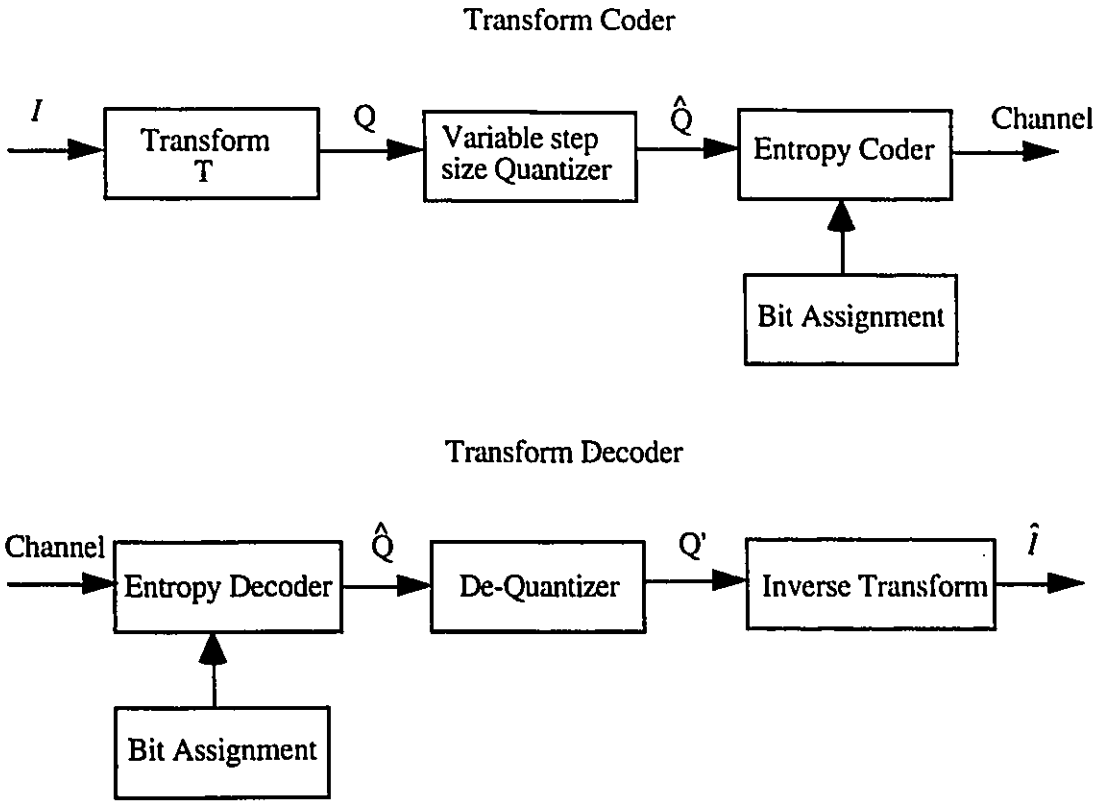


Figure 2.4 Block diagram of a transform coding scheme

Transformation

A number of different applicable transforms are available, and the choice of any particular transform upon its energy compaction properties and its computational efficiency. Some transforms that have been investigated for image coding include:

• **Kahuenen-Loeve Transform (KLT)** [50]: It is an *optimal* transform, i.e. one which completely de-correlates the transform coefficients. However, it is not used in practice because it is computationally intensive and difficult to implement.

• **Discrete Fourier Transform (DFT)** [51]: It has good energy compactness properties and possesses efficient algorithms for implementation (computational complexity is $O(N \log N)$ for an N-point transform). Unfortunately, it degrades the reconstructed image significantly due to Gibbs's phenomena and is thus not used for applications which cannot tolerate much degradation.

• **Discrete Cosine Transform (DCT)** [52]: It has better energy compactness properties than the DFT, equally efficient algorithms for implementation, and approaches the KLT in performance. Due to these properties, it has become the most popular transform. It is used in the CCITT/ISO standards for still image compression (JPEG) and video compression (MPEG).

• **Wavelet Transform** [53]: It represents an image as a superposition of wavelets, which are functions generated from one single function by dilation and transformations. Its major advantage is that it preserves edges better at higher compression ratios and is, thus, better suited towards the human visual system (HVS).

Quantization

The transform coefficients are usually individually quantized using a b_{ij} bit uniform quantizer for the (i,j)th coefficient. The bit assignment are constrained by

$$B_{ij} = \sum_{i=1}^{N} \sum_{j=1}^{M} b_{ij} \tag{2.26}$$

where B_{ij} is the number of bits allocated to the $(i,j)th$ transform coefficient.

Bit Allocation

The bit allocation procedure in a transform coding technique must be optimized in order to achieve high compression. This implies that transform coefficients be assigned an unequal number of bits with significant coefficients occupying most of the bits. If the transform coefficients are stationary gaussian random variables with variances, σ_{ij}, then the optimal bit assignment for the (i,j)th coefficient is given by

$$b_{ij} = \delta + 0.5\log_2 \frac{W_{ij}\sigma_{ij}^2}{D^*} \tag{2.27}$$

where δ is a correction factor that reflects the performance of practical quantizers, and W_{ij} is a weighting function allowing noise shaping. D^* is the weighted noise power

$$D^* = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} W_{ij}E_{ij}^2 \tag{2.28}$$

where E_{ij} is the noise introduced in the (i,j)th coefficient due to quantization.

Adaptive Transform Coding

An adaptive transform coding scheme can also be used for non-stationary data sets. In such techniques, the bit allocation map changes to match the image statistics [54]. Transform coding is generally more computationally complex and requires more memory for implementation than Predictive coding. However, it outperforms predictive coding in terms of compression and is therefore more popular.

2.3.3 Vector Quantization

Vector Quantization (VQ) is a powerful compression technique that yields high compression ratios for low bit-rate image coding [55,56]. It is based on coding vectors, or blocks, instead of scalar values within the image. According to rate distortion theory, coding vectors instead of scalars always yields better performance. This is the reason why VQ has gained popularity in recent years.

VQ is basically the quantization $q(\cdot)$, of an N-dimensional vector V into another N-dimensional vector $\hat{V}$, i.e.

$$q(V) = \hat{V} \tag{2.29}$$

It can be viewed as a pattern recognition problem whereby blocks of data are classified into a discrete number of categories, or cells, in a way that optimizes some fidelity criterion, such as MSE.

A generalized block diagram of a VQ scheme is shown in fig. 2.5. This compression technique first partitions the image into N-dimensional vectors and then uses a previously generated codebook with a finite set of reconstruction levels, or reproduction vectors, to represent each image vector. The actual image vectors, V_i, are then assigned the closest matching reproduction vector, $\hat{V}_i$, from the codebook. This matching vector is generally chosen by optimizing a distortion measure, $d(V_i, \hat{V}_i)$. A quantizer that minimizes distortion is known as an *optimum* quantizer [57]. The most commonly used distortion measure is the one that minimizes mean-square error, i.e.

$$d\left(V_i, \hat{V}_i\right) = \frac{1}{N}\sum_{i=1}^{N}\left(V_i - \hat{V}_i\right)^2 \tag{2.30}$$

In some applications, however, a weighted MSE is used, i.e.

$$d(V_i, \hat{V}_i) = \frac{1}{N}\sum_{i=1}^{N} w_i (V_i - \hat{V}_i)^2 \qquad (2.31)$$

where w_i are the tap weights.

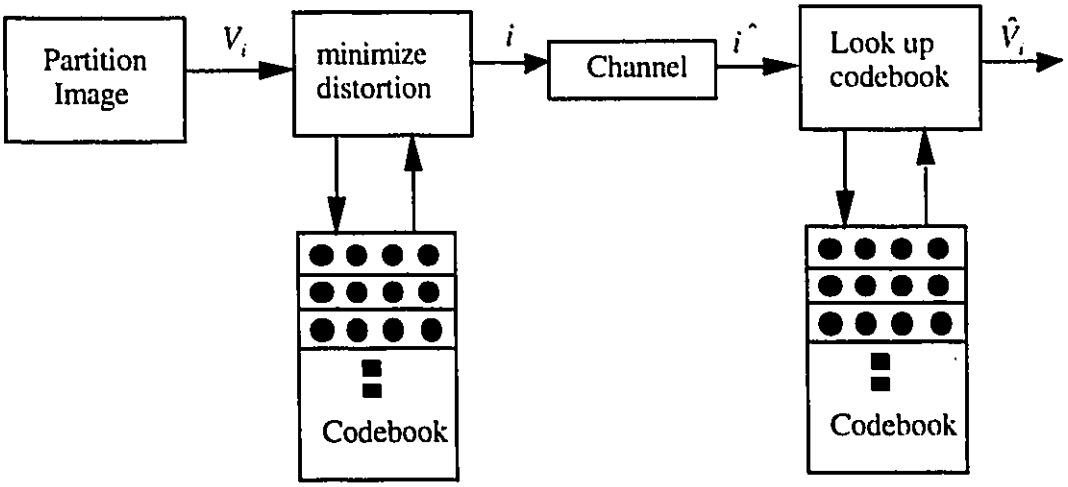


Figure 2.5 Block diagram of a Vector Quantizer

Once a match has been made, only the *index* , i, of the corresponding matched vector from the codebook is encoded. The image is reconstructed at the decoder by a simple table look-up operation, using the index as an address to a table containing the codewords.

In order to partition the image into P regions optimally, the following two conditions must be met:

- The optimal quantizer should employ the nearest neighbor selection rule, that is,

$$q(V_i) = \hat{V}_i \quad iff \quad d(V_i, \hat{V}_i) < d(V_i, \hat{V}_j) \quad for \quad all \quad j \qquad (2.32)$$

- Each codeword should be chosen to minimize the average distortion in each region. In other words, each codeword should be the centroid of its region.

The simplest partitioning scheme is one where neighboring pixels are jointly treated as vectors. More sophisticated schemes have also been proposed in the literature [53].

Codebook Design

The codebook is generated by choosing optimum reconstruction vectors from a training set of images that sufficiently represent the class of images to be coded. An optimal algorithm to generate the codebook, proposed by Linde *et. al.* [57], is the generalized Lloyd or the LBG algorithm. The algorithm starts with an initial codebook C_0, containing N initialized codewords, and a set of M training vectors. It then assigns each training vector to its nearest neighbor codeword. Each codeword is iteratively modified such that it minimizes its distortion relative to the vectors assigned to it. The iterative process is repeated until the difference between two successively generated codewords is sufficiently small.

The codebook can be initialized by selecting N evenly spaced vectors or, alternatively, by using a binary splitting algorithm [57], where the centroid of each partition of the training set is successively split, using the LBG algorithm until N codewords are generated. Once an optimal set of codevectors have been selected, they are arranged in the suitable hierarchical setup that allows efficient searching and matching during coding and decoding. In literature, codebooks are classified as either universal or image adaptive [56]. In universal VQ (UVQ), both the transmitter and the receiver have a copy of the same codebook beforehand, thus reducing the overhead and increasing compression. The codebook, though, must be generated from a large set of training vectors selected from different types of images. In image adaptive VQ (IAVQ) [61], the codebook is transmitted to the receiver as a side information resulting in high overhead. However, this technique

results in a superior image quality because IAVQ codebooks match the image statistics better.

UVQ generally requires a codebook of large size to ensure a good quality reconstructed image. This increases the bit rate for index transmission as well as the coding complexity. It also requires heavy computing load if an exhaustive search is used and heavy memory requirements to store the codebook. Recently, however, some techniques have been proposed which aim at decreasing the size of the codebook significantly. Examples include classified VQ [58,59], predictive VQ [56] and finite state VQ [60].

2.3.4 Subband Coding

In subband coding [62], the frequency components of an image are first split into separate frequency sub-images or subbands, each of which contain a limited range of spatial frequencies. These different subbands are then downsampled because each sub-image has a reduced bandwidth compared to the original image. The subbands are coded using different coders so as to take advantage of the properties of the individual sub-images. Compression is achieved by allocating different number of bits to the subbands based on their visual importance. The image is reconstructed by upsampling the decoded subbands, applying appropriate filters, and adding the subbands together. The partitioning of the frequency components is referred to as the *analysis stage*, and the reconstruction of the image from the coded subbands referred to as the *synthesis stage*, as shown in fig. 2.6.

Generally, separable two dimensional Quadrature Mirror Filters (QMF) are used to partition the frequency components. This is because they avoid aliasing and are separable. Fig. 2.7 shows the shape of a 1-D idealized pair of QMF filters.

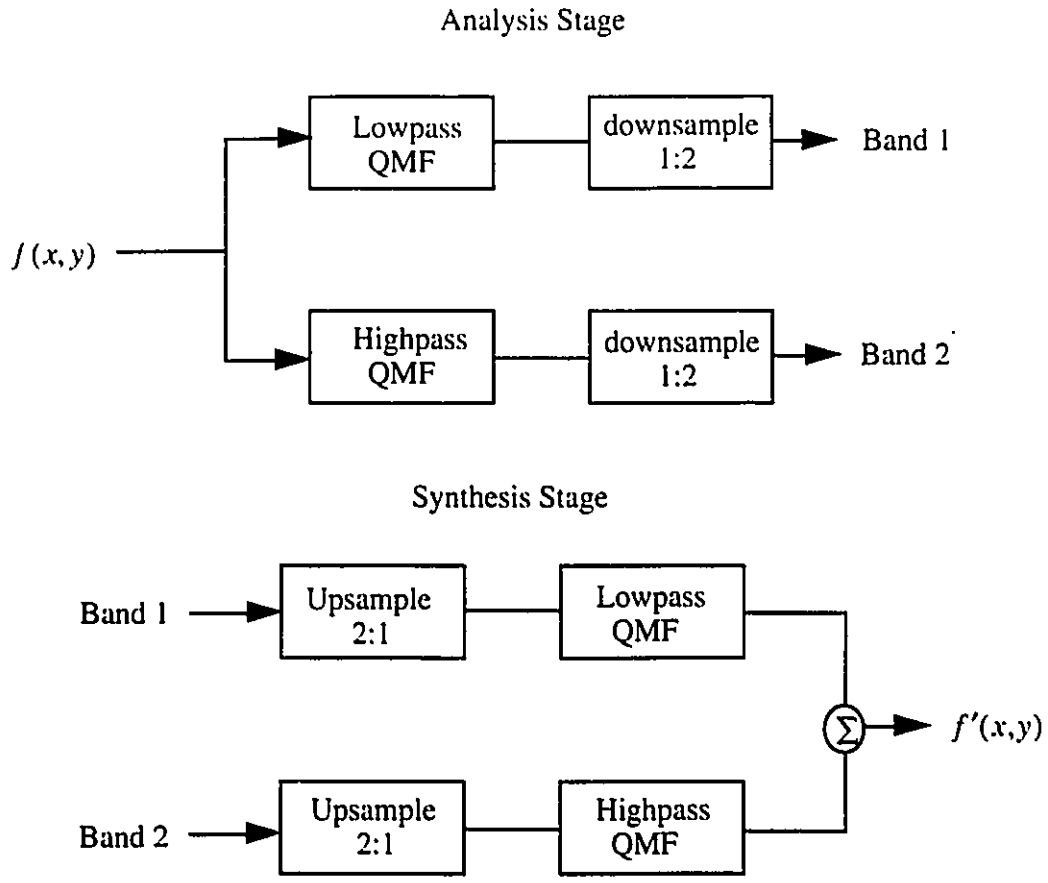


Figure 2.6 Block diagram of a 2-band coding system

Subband coding is an extremely flexible coding technique that adopts well to the channel or to the human visual system characteristics. Moreover, it has the following major advantages [63]:

- it confines quantization errors to each subband, thus reducing visual distortion in the reconstructed image.
- it allows progressive image transmission (PIT). Low frequency components can be transmitted and higher frequency ones added gradually.
- it uses different coding schemes, each suited to the information content of each subband, to encode different subbands.

Subband coding, therefore, outperforms other coding schemes including transform coding. However, it is not as popular as DCT based coding schemes because of its greater complexity.

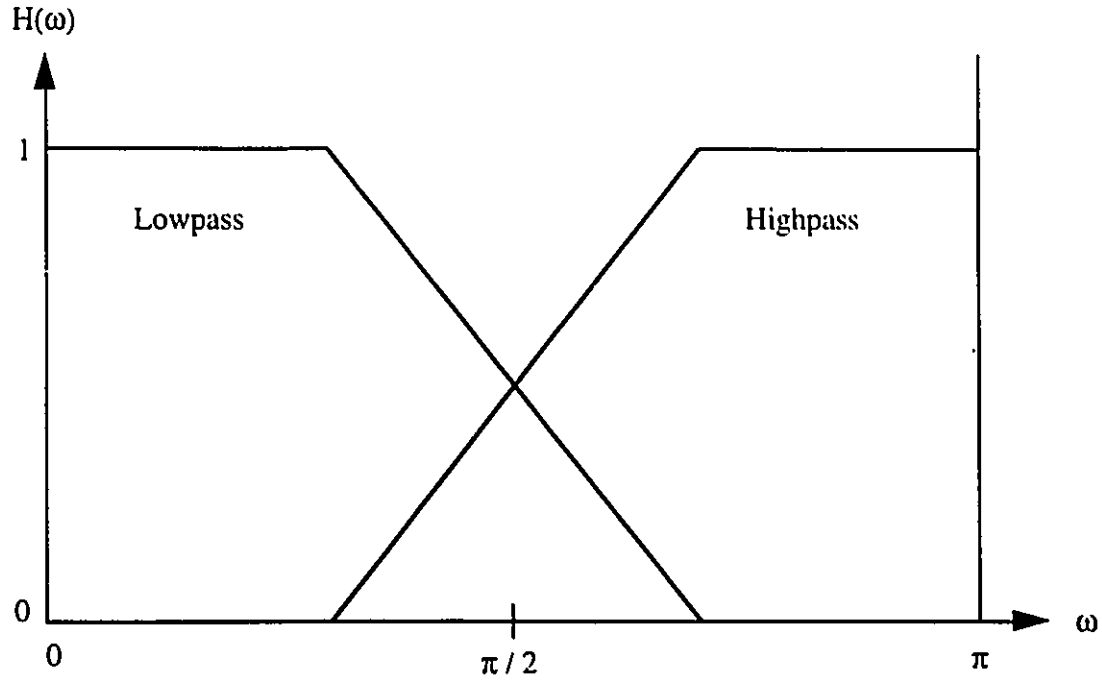


Figure 2.7 One dimensional idealized QMF pair

2.3.5 Fractal Block Coding

One of the new coding schemes available today is fractal block coding (FBC) [64]. It is based on a fractal theory of iterated transformations and has the following two main characteristics:

- it exploits image redundancy through *self-transformability* on a blockwise basis
- it approximates an original image by a *fractal* image

The coding-decoding system is based on the construction of a specific image transformation - a fractal code - which, when iterated on any image produces a sequence of images which converge to a fractal approximation of the original image. The technique, which shares many aspects with Vector Quantization, has performance which is comparable to that of state-of-the-art VQ's [64].

In FBC, an image is modeled as a fractal object; fractal objects being highly redundant in the sense that they are made up of transformed copies of either themselves or part of themselves [65,66]. The fractal image is then partitioned into non-overlapping blocks, which are usually square of size 4x4 or less, but can of any size and shape. An inter-block distortion measure is used, such as the least mean-square distortion, to "fit" blocks using a set of discrete image transformations, such as reflections and rotations.

Fractal Coder

Let an image of size rxr be partitioned into $\{R_i; 0 \le i < N\}$ different sized regions, called range cells. Recall that an image transformation τ, has the following form:

$$\tau = \sum_{i=0}^{N-1} \tau_i \qquad (2.33)$$

where τ_i is the i^{th} transformation. Then, given a range cell R_i (of size BxB), the construction of the transformation τ_i, which maps onto this cell is divided into the following two steps, as shown in fig. 2.8:

- select an image domain block of size DxD, which is contracted to a block of size BxB.
- select the proper *processing* of the domain block. i.e. find the block transformations which minimize distortion.

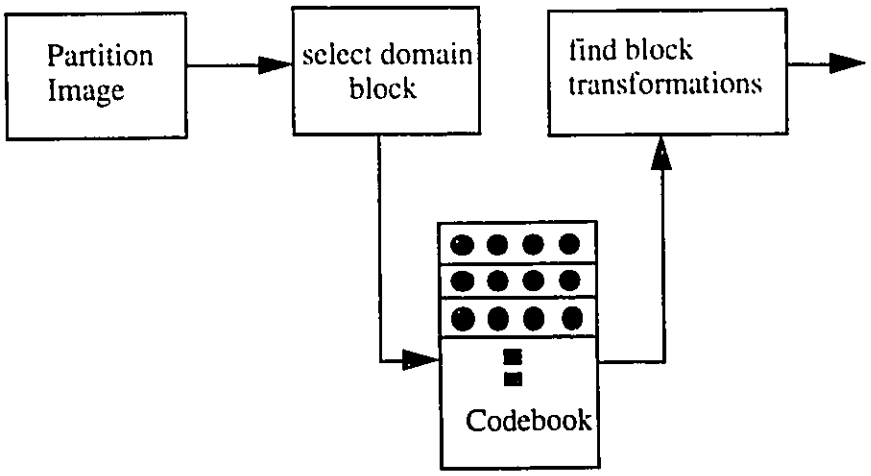


Figure 2.8 Fractal Block encoder

A pool of domain blocks, made up of all image blocks, can be extracted from the original image along with a pool of transformations, made of all discrete block transformations. Such a pool is called a *global pool* or *virtual codebook*. Range blocks can then be encoded by finding a "best fit" from the codebook, i.e. one which minimizes distortion. In order to do so, a directed search is conducted using the following two techniques:

- discarding *a priori* large subsets of the global pool.
- analyzing the range block to encode along with a set of potential domain blocks.

The result of this search is a blockwise discrete image transformation which leaves the original image approximately invariant, and produces a fractal code. Note that the encoder does not rely on the existence of any "universal codebook" but simply on the redundancy present in the original image.

Fractal Decoder

The original image, I_0, is reconstructed using any initial image by simply iterating the fractal code, τ, until convergence to a stable decoded image is observed. The sequences of

images $\left\{I_n = \tau^n I_0\right\}_0^\infty$, where I_n is the reconstructed image formed after the n^{th} iteration, is called the *reconstruction sequence* for the code τ. The mapping of an image is done sequentially such that for each cell index i, the transformation τ_i is applied to the current image block over the domain cell D_i, and mapped onto the range cell R_i. The number of iterations required, given a convergence threshold, to achieve convergence, is approximately the same for different initial images.

It should be noted that domain image blocks are not needed at the decoder, hence the term "virtual codebook". The codebook is used only during the encoding phase.

Fractal block coding is a variable bit-rate encoding scheme; bit rates depend not only on the image complexity but also on the system specifications. One of its major disadvantages is its computing requirements for encoding. This is because a typical global pool is usually much larger than a codebook that, say, vector quantizer would use. However, it shows great promise because of its performance and may emerge as the leading encoding scheme in the near future.

2.4 JPEG : Still Image Compression Standard

In 1990, the Joint Photographic Experts Group (JPEG), of the CCITT and the International Standards Organization (ISO), proposed a standard for still image compression [5]. Its goal was to develop a method for continuous-tone image compression which met the following requirements:

- be near the state-of-the-art with regard to compression ratios and visual image fidelity.
- be applicable to practically any kind of continuous tone digital image, regardless of its size, color space, complexity, or statistical properties

- have tractable computational complexity so that software and hardware implementation is viable.

The JPEG standard specifies four modes of operation, each one providing a unique functionality for a specific class of applications. They are [36]:

- Baseline Sequential : each image component is encoded in a single left-to-right, top-to-bottom (raster) scan.
- Lossless Compression : each image is encoded such that exact recovery of every source image sample value is guaranteed.
- Progressive coding : each image is encoded in multiple stages such that the image progressively builds up in multiple coarse-to-fine stages.
- Hierarchical coding : each image is encoded at multiple resolutions so that lower resolutions can be accessed before retrieving the full resolution version.

For each mode, one or more codecs are specified. The baseline sequential mode uses a DCT based codec, whereas the lossless uses a DPCM based codec. The progressive mode utilizes a modified baseline codec and the hierarchical mode has the freedom to chose between any of the codecs used in the other three modes. We will, therefore, only describe the baseline and lossless modes.

2.4.1 Baseline Sequential Mode

Fig. 2.9 and Fig. 2.10 outlines the key processing steps of the baseline sequential JPEG coder and decoder. The image to be encoded is first partitioned into non-overlapping blocks of 8x8 pixels. Each block is then sequentially fed through the DCT based encoder. If each block contains multi-components (such as in color images), then each component is fed into the system either one at a time or is alternated interleaving the 8x8 sample blocks

from each component in turn. In order to decrease the average entropy of the image pixels, the pixel values are shifted from unsigned integers in the range $\left[0,2^{p}-1\right]$ to signed integers with the range $\left[-2^{p-1},2^{p}-1\right]$, where p is the number of bits required to represent each pixel. This is generally referred to as a *level shift* operation. Each 8x8 block is then transformed using the DCT into 64 orthogonal basis functions, or DCT coefficients. The 64 DCT coefficients represent 64 unique two dimensional "spatial frequencies" which comprise the input image "spectrum". The transform uses the following equation:

$$F(u,v) = \frac{C(u)\,C(v)}{4}\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\,\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right] \qquad (2.34)$$

where: $C(t) = \begin{cases} 1/\sqrt{2} & for \quad t=0 \\ 1 & otherwise \end{cases}$,

$f(x,y)$ = the source image data, and

$F(u,v)$ = transform coefficients

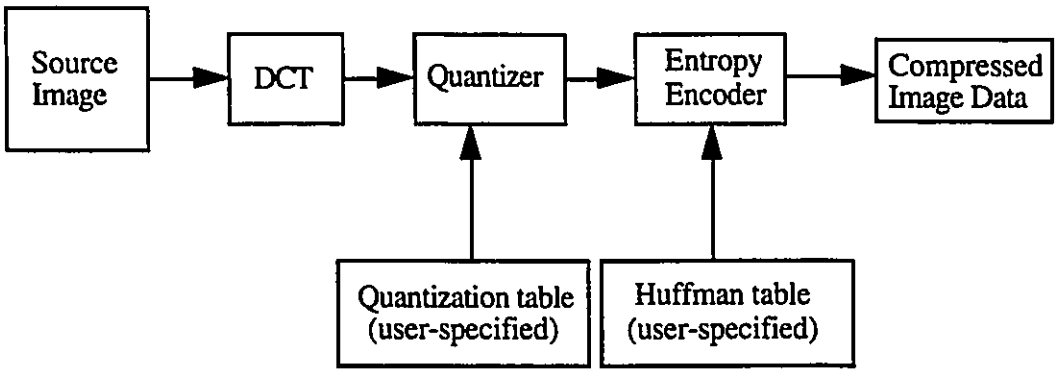


Figure 2.9 Baseline sequential mode encoder

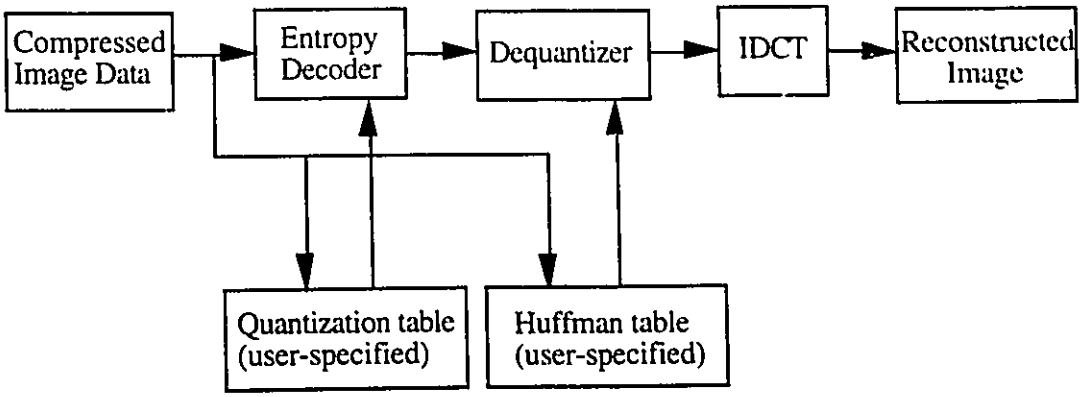


Figure 2.10 Baseline sequential mode decoder

Fig. 2.11 describes a DCT transformed block. The first coefficient of each pixel block, i.e. the C_{00} coefficient, is called the DC coefficient and the remaining 63 coefficients are known as AC coefficients.

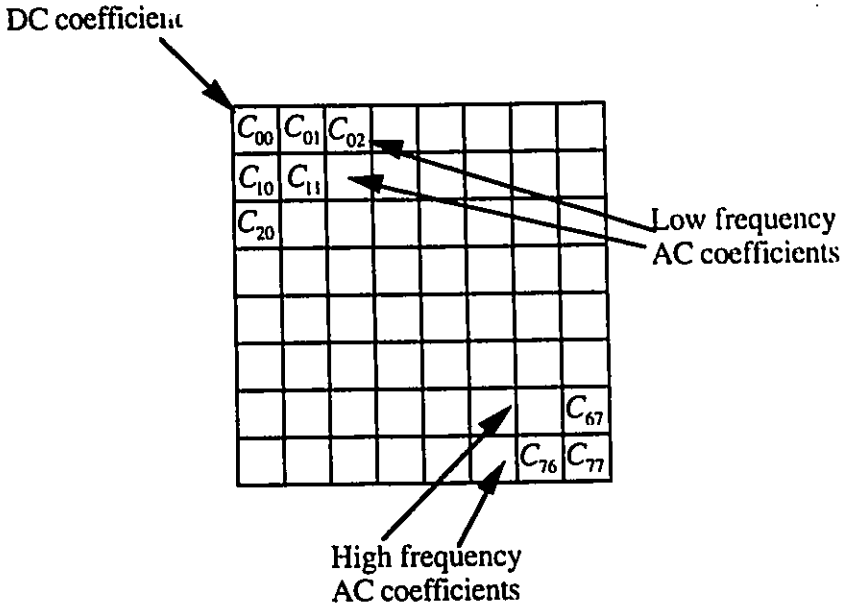


Figure 2.11 A typical 8x8 DCT transformed pixel block

The DCT concentrates most of the image energy in the lower spatial frequencies. Thus, encoding only the lower frequency components can yield high compression.

Furthermore, in a typical 8x8 block for a typical source image, most of the spatial frequencies have zero or near-zero amplitudes and, therefore, need not be encoded. It should be noted that the DCT introduces no loss to the source image blocks; it merely transforms them to a domain where they can be encoded more efficiently.

All of the transform coefficients are then quantized using a midstep quantizer and rounded to the nearest integer, as shown in Fig. 2.12. Quantization is defined as the division of each DCT coefficient by its corresponding quantizer step size, $Q(u,v)$, followed by rounding to the nearest integer:

$$F^{Q}(u,v) = \left\lfloor \frac{F(u,v)}{Q(u,v)} \right\rfloor \tag{2.35}$$

where $\lfloor x \rfloor$ is the integer round operation. The quantized value, $F^{Q}(u,v)$, is, thus, normalized by the quantization step size.

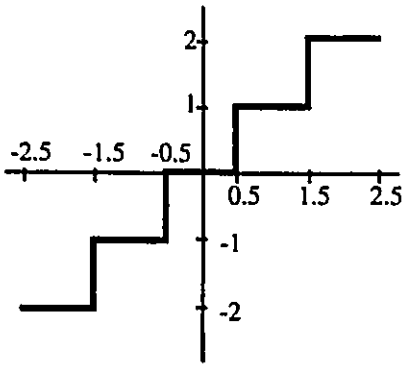


Figure 2.12 A uniform midstep quantizer

Quantization reduces the coefficient magnitude and increases the number of coefficients with zero value. We note that quantization results in a loss of image information which may appear as distortions in the reconstructed image. The quantization step sizes are not fixed

in the JPEG standard; they increase in size with higher frequency AC coefficients. Furthermore, a quantization table is not specified in the standard even though one is suggested. This is to allow for the best possible reconstructed image quality depending on the application. Hence, there is a tradeoff between step size and compression: smaller the step size, the better the image quality and smaller the compression ratio. Quantization is a many-to-one mapping, and is therefore lossy. Compression quality, or information loss, is controlled by the proper choice of a quantization table.

After quantization, the coefficients are fed into the entropy encoder. The DC coefficients are differentially encoded because there is usually a strong correlation between adjacent block DC terms. This process is depicted in Fig. 2.13. The rest of the DCT coefficients are first reordered in a one dimensional sequence using a zig-zag sequence, as shown in Fig. 2.14, and then encoded by using either Huffman or arithmetic coding.

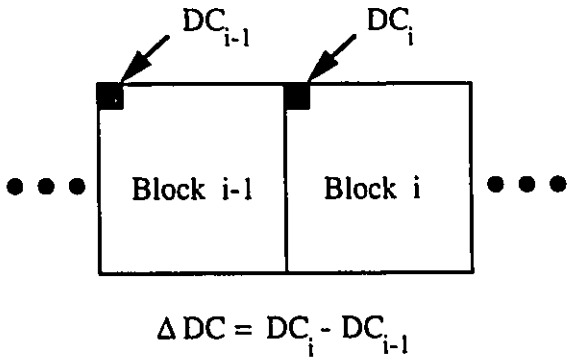


Figure 2.13 Differential DC encoding

JPEG does not specify a Huffman table, giving the user freedom to choose one according to his application. However, it does provide a few sample tables. The baseline compression algorithm is allowed to use a maximum of two DC and two AC Huffman tables. However, JPEG has defined an *extended* sequential algorithm for special applications. It is similar to the baseline algorithm except that a maximum of eight tables are allowed: four each for the DC and AC coefficients.

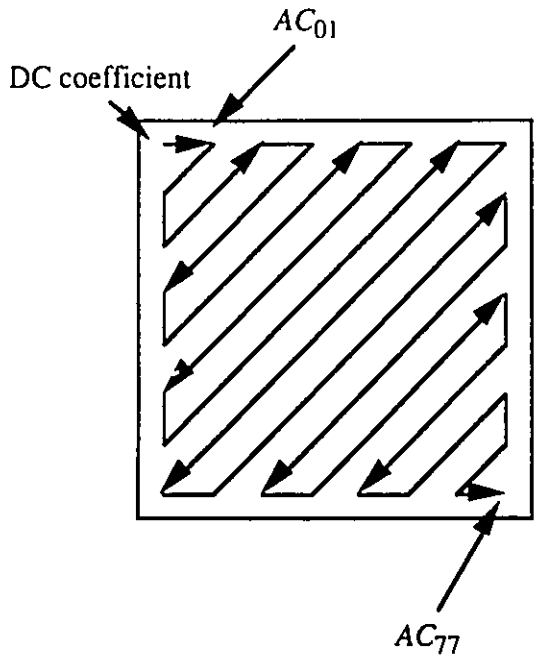


Figure 2.14 Zig-zag sequence of AC coefficients

Reconstruction of the image is done by decoding the coefficients using the entropy decoder. After that, the de-quantization is done by simply de-normalizing the quantized coefficients by the step size,

$$F^{Q'}(u,v) = F^{Q}(u,v) * Q(u,v) \tag{2.36}$$

Then, each 8x8 block is passed through the inverse discrete cosine transform (IDCT) so that the information is transformed back to the spatial domain using:

$$f'(x,y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)\, C(v)\, F'(u,v)\, \cos\left[\frac{(2x+1)u\pi}{16}\right] \cos\left[\frac{(2y+1)v\pi}{16}\right] \tag{2.37}$$

where, $C(t) = \begin{cases} 1/\sqrt{2} & for \quad t = 0 \\ 1 & otherwise \end{cases}$,

$F'(u,v)$ = de-quantized DCT coefficient, and

$f'(i,j)$ = the reconstructed image data

Finally, an inverse *level shift* operation is performed so that all pixel values are translated back to unsigned integers.

2.4.2 Lossless Mode

The lossless compression mode of JPEG uses a DPCM based algorithm instead of a DCT based one because the latter was difficult to define as a practical standard against which encoders and decoders could be independently implemented [5]. The encoder predicts the value of each pixel using the statistics found in the neighborhood pixels. Fig.2.15 shows how this works. Each pixel x is predicted from its neighborhood pixels a, b and c using some predetermined prediction law. JPEG specifies seven different prediction laws, which range from no prediction at all to second order prediction functions. The prediction difference is losslessly coded using either Huffman or arithmetic coding. Decoding is simply the inverse of this process.

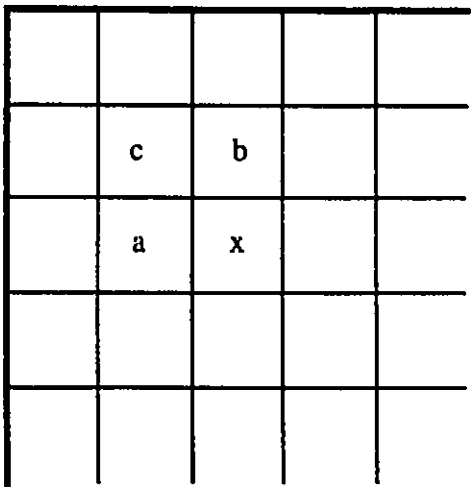


Figure 2.15 Neighborhood prediction of pixel x

Lossless codecs typically produce compression ratios of about 2:1 [36] for color images with moderately complex scenes. Such performance is surprisingly close to the other state-of-the-art lossless compression techniques, considering the simplicity of the process.

2.7 Summary

In this chapter, we have presented an overview of the basic techniques for image compression that are available. We started out by reviewing some of the fundamental concepts of information theory, namely entropy and rate distortion, as applied to image compression. This was followed by a review of some of the more important lossless compression techniques, such as Huffman, Arithmetic, Run-length, and Lempel-Ziv-Welsh (LZW) coding. We then described some of the major lossy compression techniques, like Prediction, Transform, Vector quantization, and Subband coding, followed by a brief description of Fractal block coding, an emerging compression technique. Finally, we presented the two basic modes of operation of JPEG, which has become a standard for still image compression.

Each of the compression techniques described above is appropriate for a certain kind of image application. For example, Run-length encoding works well on binary images but fails on gray-tone images. In order to improve performance, it is possible to use a combination of these methods. However, a combination of coding techniques increases complexity and is generally difficult to implement. Most of the techniques are based on the LSE and visual error criteria and therefore experience a performance degradation. The lossy techniques presented here are not appropriate for compressing range images. We will address this issue later on.

The major issue facing image compression is the representation of image data that can be coded optimally. Bearing this in mind, we have proposed a novel representation for 3-D images, which we describe in the next chapter.

Chapter 3

A Range Image Compression Algorithm

In this chapter, we present the details of our algorithm for the compression of range images. First, we present the outline of the algorithm. Then, we introduce various scanning techniques, such as Peano scanning and raster scanning. This is followed by a description of the algorithms main building block, the Fan redundancy finder. Encoding of the non-redundant information is then explained. The encoder uses Huffman coding; its usage is justified and details of the formation of its fixed encoding table are given. At the receiving end, the workings of the decoder is first explained. Then, we present the interpolation of the decoded data. Finally, we outline the issues involved in the de-scanning of the decoded information.

3.1 Outline of the algorithm

Our algorithm is designed for compressing range images. Its fidelity criterion is maximum error, unlike that of JPEG and most other methods. The main advantages are its simplicity and the ease with which it is implemented. Its complexity is of the order N, i.e. O(N), where N is the number of pixels to be processed and it requires very little

memory to operate. Our goal was to develop a method for range image compression which meet the following requirements:

- be at or near the state-of-the-art with regard to compression ratios.
- be applicable to practically any kind of digital range image, regardless of its size, complexity, or statistical properties.
- have tractable computational complexity so that software and hardware implementation is viable and economical.

It exploits both spatial and intensity redundancies present in range images. As we mentioned in chapter one, the boundaries of objects, in range images, are sharply defined. Our method preserves these edges, even at high degradation levels, which is often not the case with the methods presented in chapter two.

In contrast with JPEG, our algorithm has a single mode of operation. We do not classify our method into lossy or lossless modes: the method is lossless when the tolerance is set to zero. Like all lossy algorithms, compression ratios increase with higher tolerances.

The other major difference with still image compression methods resides in the control of the quality factor. The user controls the quality of the compression by specifying the maximum error, instead of using the mean-squared error (MSE). Thus, the user has perfect control over the compression error. With the methods presented in chapter two, there is no direct control over the MSE; one has to select a quality parameter and then verify the result. Then, the parameter has to be adjusted so that the desired tolerance level is reached. These iterations are costly in terms of computing requirements.

Fig. 3.1 shows the key processing steps of our algorithm. The scanner serializes pixels so that they are presented in sequential order to the other processing modules.

The encoder consists of a redundancy finder and an entropy encoder. The redundancy finder uses an algorithm called "Fan" to isolate non-redundant points (NRPs) in the image data. The algorithm transforms each line of the image into a more compact representation;

image data points are thereby approximated by straight line segments, taking care not to let the approximation fall outside of the user-specified error range. The non-redundant points are the extremities of these straight lines. The non-redundant information, which consists of the NRPs and the unidirectional distance between them, or run-lengths, is then fed into the entropy encoder. Here, they are encoded using Huffman coding. It is possible to encode the data using other coding schemes, such as Arithmetic or LZW coding. We have used Huffman coding because of its simplicity and better performance than others.

The decoder consists of the entropy decoder and a linear interpolator. The entropy decoder uses Huffman coding to decode the encoded data set received from the transmitter. It uses two fixed Huffman tables to do so, one for NRPs and for run lengths. The decoded data is then forwarded to the interpolator where it is reconstructed. The interpolator linearly interpolates the values of the pixels between two NRPs. The number of pixels to be interpolated between them is specified by their run length.

The De-scanner, which receives the reconstructed data, then converts it back to the raster format for post processing.

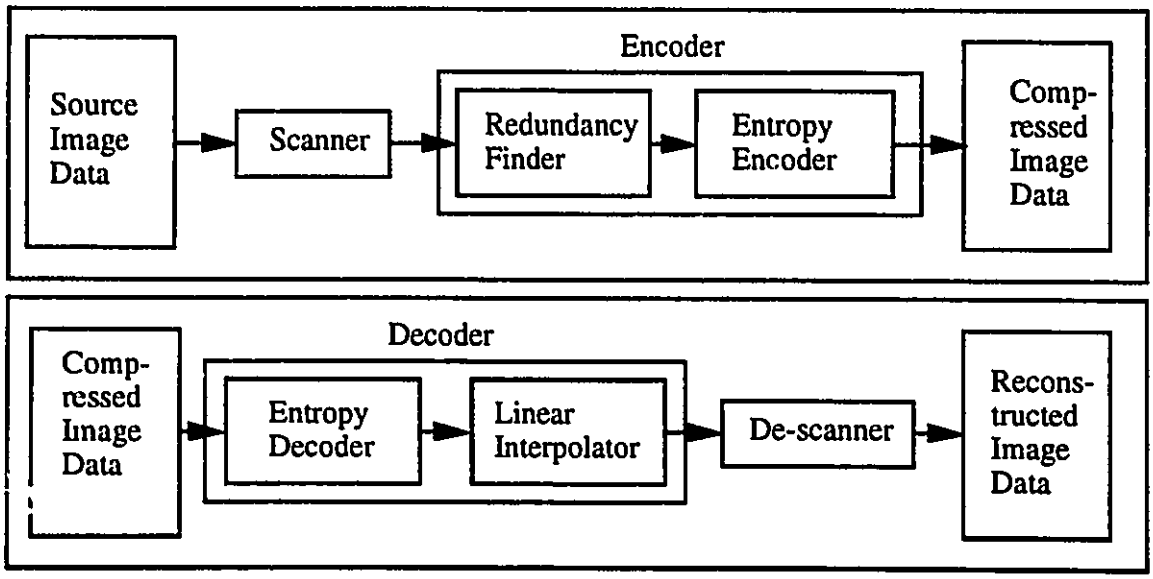


Figure 3.1 The algorithms processing steps

3.2 Scanning

Since our algorithm represents image data with straight lines, there is a need to serialize the pixel data. This is achieved with proper scanning techniques. Fig. 3.2 describes the structure of the scanner.

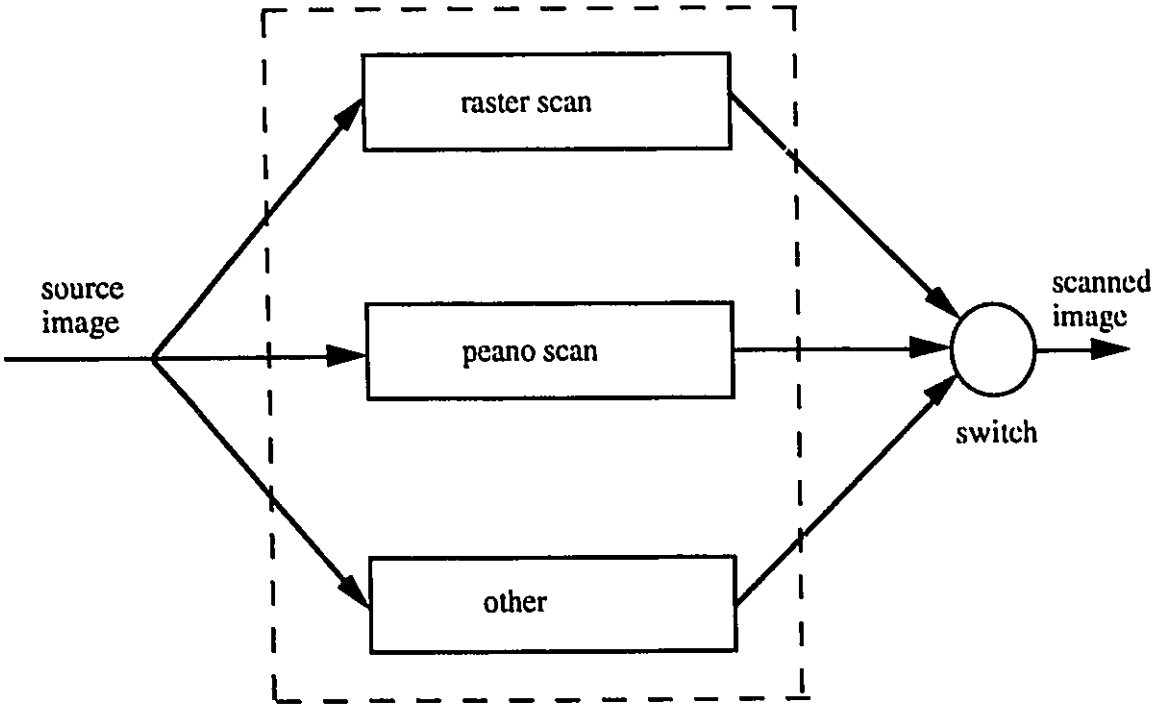


Figure 3.2 Structure of the scanner

We have employed various scanning techniques in the operation of the algorithm. Raster scanning and Peano scanning, however, are the premier techniques used in this thesis; this by no means limits the use of other scanning methods which may be more meaningful for certain applications. The scanning techniques used here are outlined below.

3.2.1 Raster Scanning

Raster scanning, which is also known as TV scanning, has been historically used in the majority of applications involving images because of its simplicity in implementation. Its use in electron tubes as well as in state-of-the-art scanners is commonplace. In raster scanning, lines in a source image are scanned from left to right, first line being on top of the image, as shown in fig. 3.3. Lines scanned this way are known as raster lines.

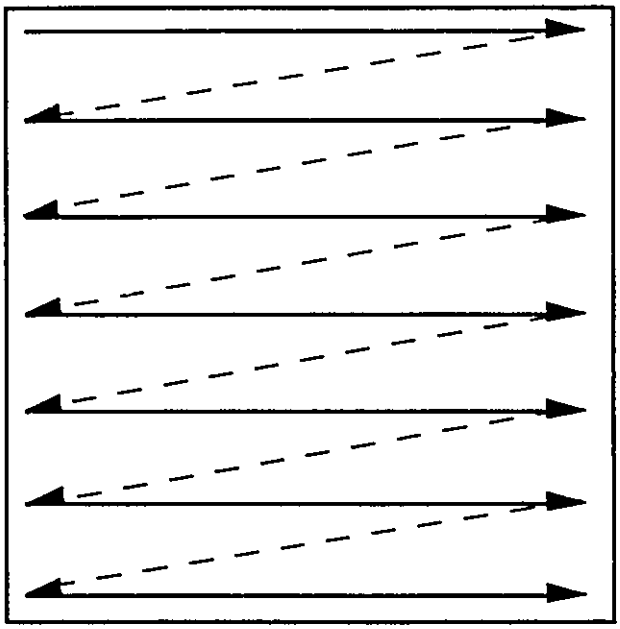


Figure 3.3 Raster scanning

3.2.2 Peano Scanning

Notwithstanding the pros of raster scanning, we may wonder if some other scanning technique might be more advantageous. Peano scanning is precisely such a scheme. It is a

fractal scanning method derived from the Peano curve which was developed by Peano, in the 18th century [68,69]. It has the following two advantageous properties [70] :

- it respects neighborhood relationships better than raster scanning: pixels that are close on the image plane tend to be close in the sequence
- its generating algorithm is recursive and therefore allows the implementation of a hierarchy

We will see, in chapter four, that the use of Peano scanning improves performance considerably. Fig. 3.4 gives some examples of Peano scanning. The squares in the figure are explored in the order with which they are numbered. Fig. 3.5 gives an idea of what the Peano curve corresponding to the Peano scannings of fig. 3.4 look like.

The procedure for generating Peano curves is as follows. The image to be scanned is first divided into equal squares, $4^j \times 4^j$ pixels, where j=1,2,3... The first 3 orders are shown in fig. 3.3-3.4. In order to go from order-j to order-j+1, each section of the order-j image is split into four smaller equal area sections. The order in which the four new sub-squares are scanned depends upon how the squares in the order-j image were scanned. For example, to go from order-1 to order-2, all four square sections of the order-1 image are divided so that a total of 16 square sections are formed in the image of order-2. Starting from the top-left corner of the order-2 image, we realize that the four sections formed from section 1 of the order-1 image can only be scanned in the clockwise (CW) direction. This is because section 2 of the order-1 image lies beneath section 1 of the same image and it is here that we must end up after we have scanned the four sub-squares of section 1; scanning in the counter-clockwise (CCW) direction will lead us into section 4. Having entered section 2 of the order-1 image, we realize that since we must go to section 3 of the same image next, we can only scan in the CW direction; scanning in the CCW direction will lead us out of the image. The image is traversed in this fashion until all the squares are explored. Thus, it is possible to recursively divide the image into smaller subsections.

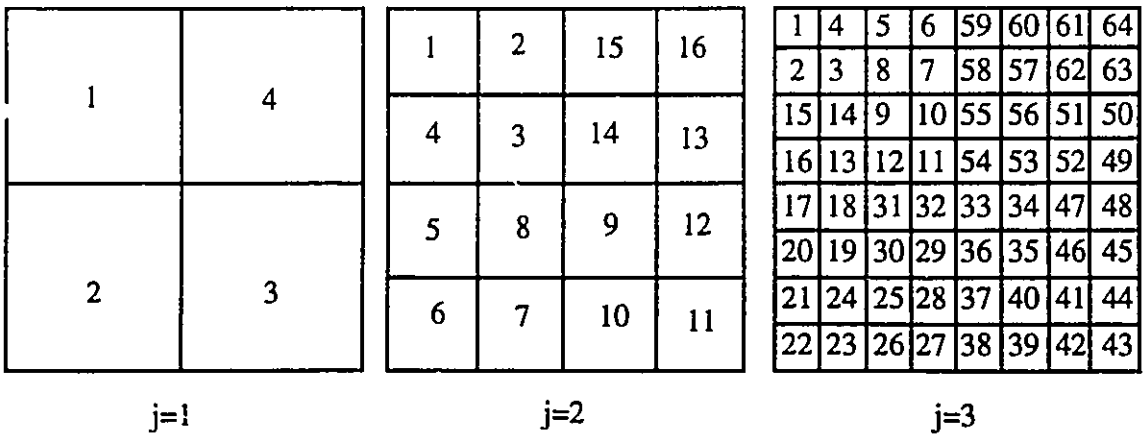


Figure 3.4 Peano scanning

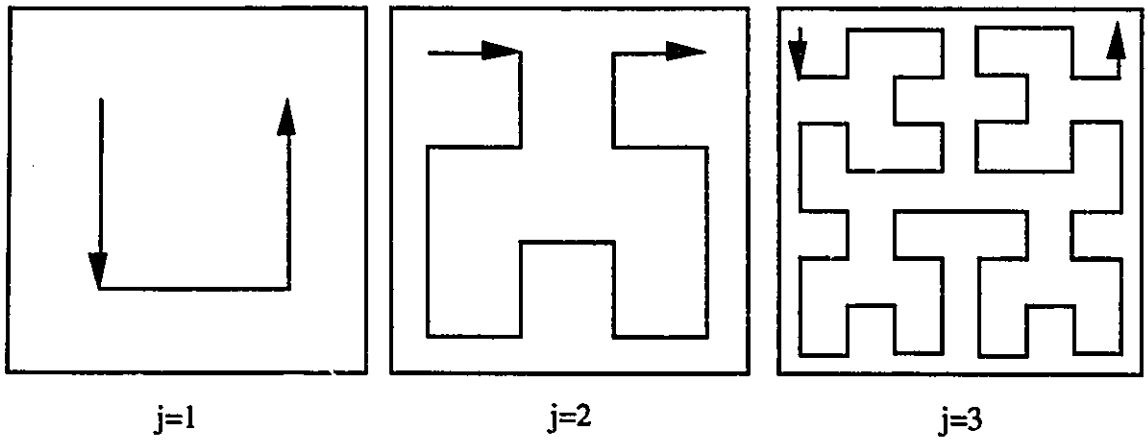


Figure 3.5 Peano curves

The major drawback of this scheme is that it requires a square image of size 4^j x4^j pixels in order to work properly. However, it is possible to lift this restriction by increasing the size of the image to that of the next order-j image and scanning it appropriately. Fig. 3.6 show how this is done. Fig. 3.7. illustrates the Peano curves involved. Fig. 3.6-3.7 (a) shows a raster scanned 3x4 image. Its size is increased to that of an order-2 image, i.e. 4x4, and is Peano scanned, as shown in fig. 3.6-3.7 (b). Whenever the scan goes outside of the original image, it is ignored and is, thus, not recorded. Fig. 3.6-3.7 (c) shows the

final Peano scan on the image. This scanning procedure, though not optimal, is adequate for our purpose. In fact, it still outperforms raster-scanning!

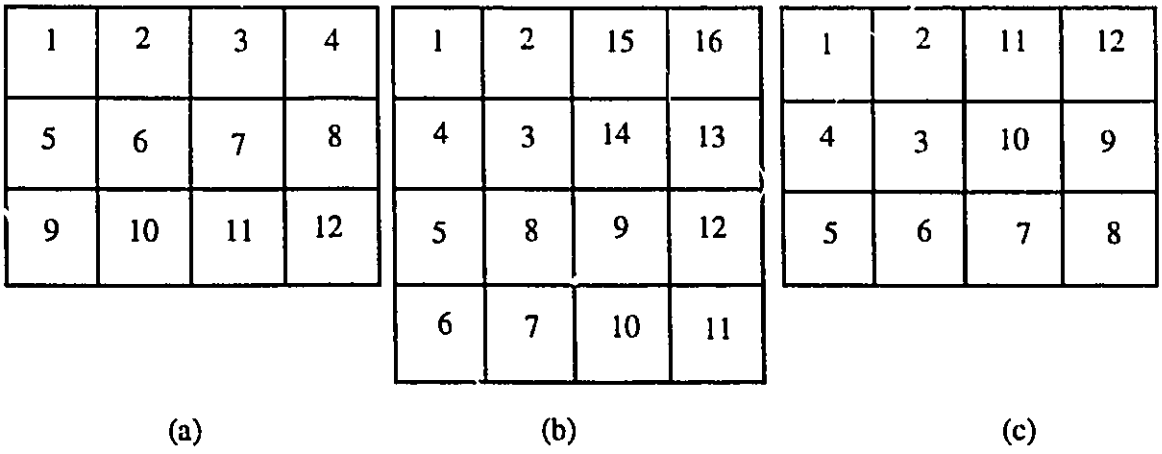


Figure 3.6 Construction of Peano scans on non-standard images

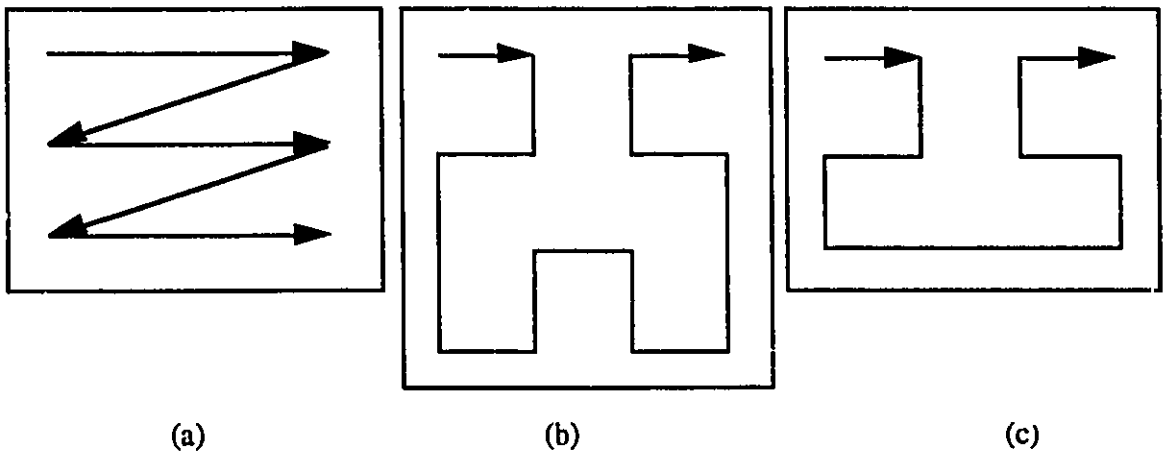


Figure 3.7 Peano curve construction of non-standard size images

3.2.3 Other scanning techniques

There are certainly many other scanning techniques which can be used. The main condition is to use a plane-filling curve so that all the pixels are explored. Another scanning technique, which we call horizontal-chain scanning, that could be used is depicted

in fig. 3.8. The reason for using this technique is to increase the possibility of correlation between pixels located at the end of lines, something which raster scannings ignore. Some other scanning techniques include vertical scanning, which is the vertical counterpart of raster scanning, and vertical-chain scanning, which is the vertical cour. erpart of horizontal-chain coding.

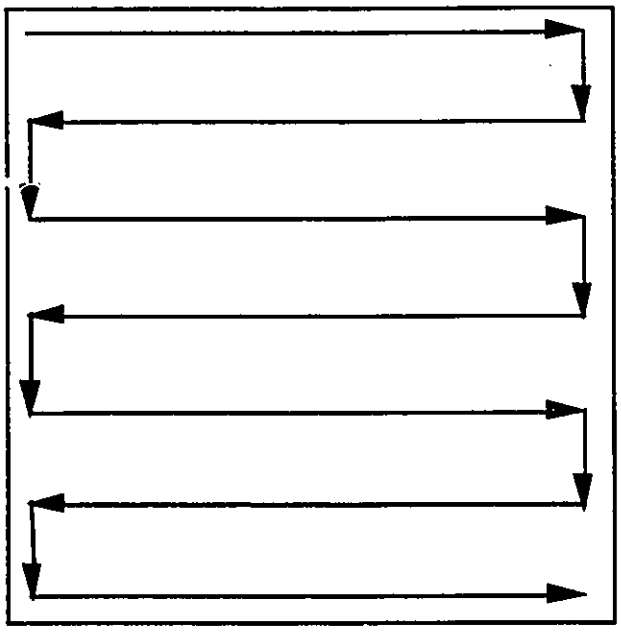


Figure 3.8 Horizontal-chain scanning technique

Although this scanning technique may seem appealing, its performance is worse than the raster scan because it tends to introduce jitters in the vertical contours of an image. However, in terms of compression ratios, the two scans are almost identical.

Image data, having been properly scanned, is then fed into the redundancy finder. The Peano scan introduces complexity into the system but at the same time improves its performance. It is up to the user to choose between complexity and performance.

It should be noted that range imaging sensors generally acquire data with a raster scan. This implies that we can feed the redundancy finder directly from the sensor if raster

scanning is to be used. Peano scanning, on the other hand, require that images be stored in memory before compression can be done.

3.3 Redundancy Finder

3.3.1 Unidirectional redundancy finder

The redundancy finder is the heart of our algorithm. It utilizes the spatial properties found in the source image in order to compress it. Its goal is to approximate image data points in the source image to obtain straight-line segments. The straight-line representation may be exact or approximate; the user has to specify its quality. Obviously, a lower quality will yield a better compression ratio and vice versa. One of the interest of our method resides in the control of this approximation. The quality parameter, ε, is a tolerance: the approximation maximum absolute error. The approximation method, called *Fan* by Gardenhire[71], who first introduced it, works as shown in Figs. 3.9-10, on a discrete signal f(k).

Assume that point k is the starting point and is, thus, a non redundant point (NRP). To determine whether point k+1 is a NRP, points k and k+1 are used to create a fan that starts at k. At point k+1, one of the sides of the fan is at $f(k+1)+\varepsilon$, while the other is at $f(k+1)-\varepsilon$. The fan, thus created, is extended to point k+2. If point k+2 lies outside the fan, as shown in fig. 3.9, then point k+1 becomes a NRP and the process is repeated with point k+1 being the starting point. In this figure, points k and k+1 are both NRPs.

If, however, point k+2 lies within the fan, as is depicted in fig. 3.10, then it is redundant and point k is retained as the starting point. A new fan is therefore created using points k and k+2.

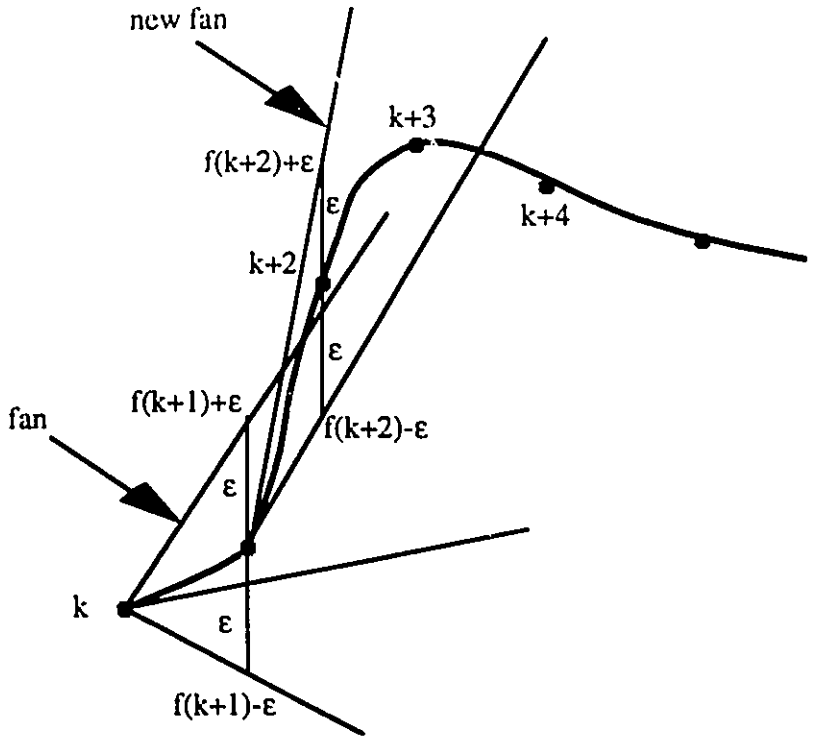


Figure 3.9 Operation of the Redundancy finder. NRP found at point k+1;

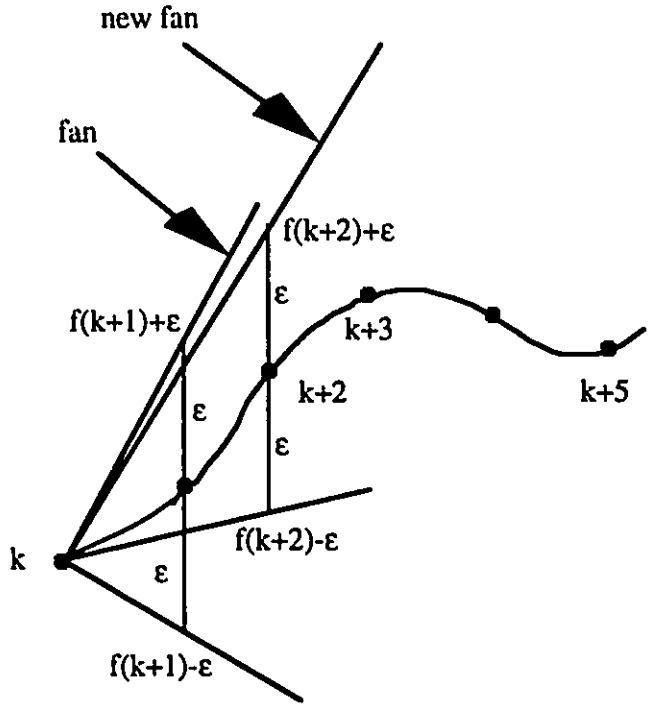


Figure 3.10 Operation of the Redundancy Finder. Redundant point found at point k+1 and k+2;

This fan is compared with the previous one and the sides of the fan that make it narrower are kept, in such a way that the new fan at k+2 is always inside of the fan at k+1. If point k+3 lies outside this new fan, point k+2 becomes a NRP and the process starts over with point k+2 as the starting point. If point k+3 lies within the new fan, point k+2 also becomes redundant and new and narrower fans are created until a NRP is found. In fig.3.10, point k+1 and k+2 are both redundant.

The fan algorithm can, therefore, be described as follows:

1. Initialize the first point of the image as a NRP, and the starting point.
2. loop (until end of image data is reached)

   (a) create a fan by using points k and k+1 such that the two tips of the fan at point k are at $f(k+1)+\varepsilon$ and $f(k+2)-\varepsilon$.

   (b) Extend the limits of the fan till point k+2.

   (c) If (point k+2 falls outside the two limits of the fan), then

   -assign point k+1 as a NRP

   -initialize point k+1 as the starting point

   -goto step 2

   Else

   -assign point k+1 as a redundant point

   -create new fan using points k and k+2

   -compare the new fan with the old one; keep the narrower version

   -goto step 2 (c)

4. Assign the last point in the image data as a NRP.

In this way, all the image data points are sequentially approximated by straight line segments, as shown in Fig. 3.11. All the NRPs and all run-lengths between each one of the NRPs are then stored and sent to the entropy encoder, which is described in the next section. The redundancy finder, thus, produces two sets of data, namely NRPs and

runlengths, as shown in fig. 3.12. These two sets of data, in essence, represent the image since most of the spatial redundancy is removed during the process.

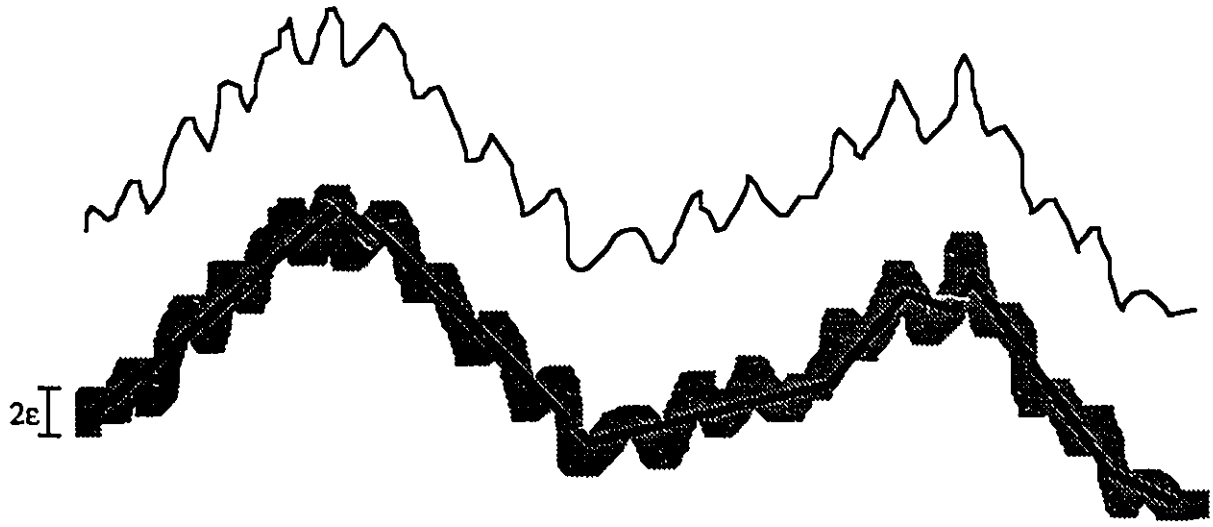


Figure 3.11 Approximation of the image data points by straight line segments

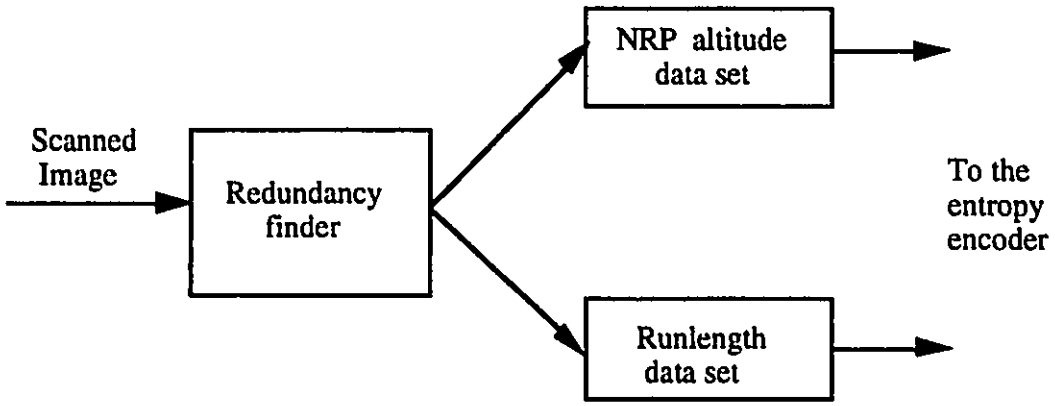


Figure 3.12 Output of the Redundancy finder

It is interesting to note that the redundancy finder exploits the spatial redundancy in the source image in only *one direction* - the direction in which it has been scanned. Therefore, it can be said that the redundancy finder is unidirectional. For example, if the image has been raster scanned, the redundancy finder would remove spatial redundancy found only in

the *horizontal* direction. It would be worth the effort, however, to investigate the possibility of removing the spatial redundancy in other directions as well, thus exploiting the 2-D nature of the image plane.

3.3.2 Two-directional redundancy finder

The two-directional redundancy finder uses the unidirectional redundancy finder in two sequential steps. Initially, it reduces the data set in one direction, say horizontal, forming a set of NRPs and runlengths. Then, it aligns the NRPs systematically and reduces the actual NRPs themselves, in the other direction, i.e. vertically. In other words, it exploits vertical redundancy after it has successfully exploited horizontal redundancy.

This task is accomplished by first lining up the NRPs found in each line of the source image data, during the initial step, side by side, as shown in fig. 3.13. The runlengths between them are removed so that they can be grouped together.

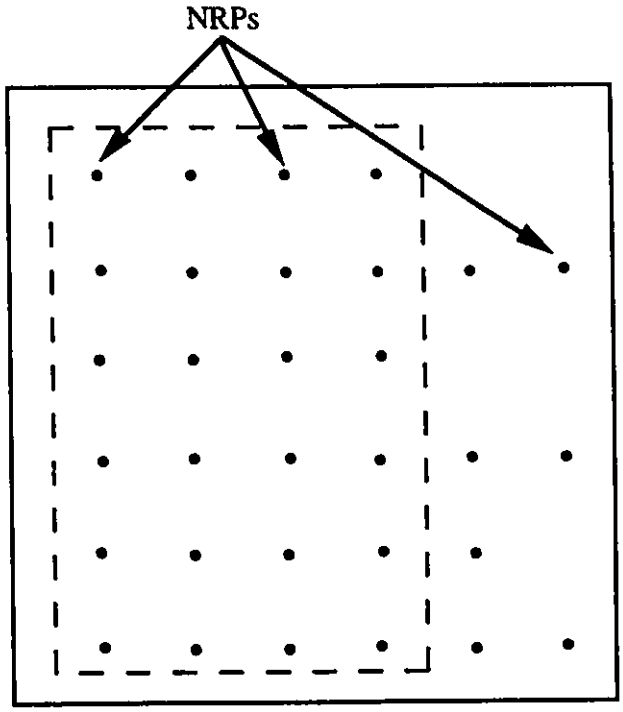


Figure 3.13 NRP alignment in a two-directional redundancy finder

Generally, each line of the image will have a different number of NRPs; the number of NRPs can vary from a minimum of zero to a maximum of the length of the image line size. For the sake of reduced complexity, we allow only those *vertical* lines of the image that have a maximum number of NRPs in them to pass through the unidirectional redundancy finder in the vertical direction. The rest of them are left untouched. The reduced set of NRPs, thus formed, are then encoded appropriately, as explained in the next section. It should be noted, though, that this process produces two sets of runlengths but only one set of NRPs.

Another issue facing the two-directional redundancy finder is the choice of ε in the second stage of the process. In Appendix A, we show that if ε_1 is used in the first step and ε_2 in the next step, then an overall maximum absolute error of $\varepsilon_1+\varepsilon_2$ will result. A value of $\varepsilon/2$ can be used in both steps in order to give an overall maximum absolute error of ε. Unfortunately, a value of $\varepsilon/2$ results in a lot more NRPs being produced in step one, thus reducing compression in the first step.

3.3.3 Multi-directional redundancy finder

The multi-directional redundancy finder is used in conjunction with Peano scanning. It is basically the same as the unidirectional redundancy finder except that it utilizes the multi-directional characteristics of the Peano curve.

3.4 Encoding

The encoding process depends on the type of scanning used and on the type of redundancy finder used on the source image. This has led us to classify the encoding process into three classes. Unidirectional encoding corresponds to the use of the unidirectional redundancy

finder on images scanned in one direction. This encoding class is the simplest of the three and can be used in applications sensitive to codec complexity.

Two-directional encoding, on the other hand, is used whenever the two-directional redundancy finder is required. Its usage is primarily for applications which can tolerate an extra amount of coder complexity in order to get an improvement in performance.

The third class of the encoding process is multi-directional encoding. It utilizes the Peano scanning technique to improve performance, even though it is at the expense of codec simplicity. The coder complexity, though, is not as high as that found in two-directional encoding. We will see in chapter 4 that this increased coder complexity is well worth the effort since the amount of performance increase is much higher than the added codec complexity.

The redundancy finder forwards two sets of information to the entropy encoder, as were described in fig. 3.12. They are the following:

- Run lengths
- NRP altitudes

There is no direct physical relationship between these two sets of data. We, therefore, encode them separately. We tried the three most promising error-free encoding techniques: Huffman, Arithmetic and Lempel-Ziv-Welsh (LZW) coding.

We recall from chapter two that Huffman coding is a minimum-redundancy, prefix code technique which allocates lesser number of bits to higher probability symbols in a symbol set. Arithmetic coding, on the other hand, is an optimal variable length coding scheme that encodes data symbols by creating a code string which represents a fractional value on the number line between 0 and 1. LZW coding takes substrings of input characters and maps them into fixed length output codes such that redundancy occurs whenever particular substrings appear frequently in the input sequence.

For high resolution images, such as medical images, Huffman and arithmetic coding outperform LZW, though they themselves perform equally well [72]. This is also the case with higher resolution images, such as 15-bit range images. We shall see in the next chapter that arithmetic coding outperforms Huffman coding by only a mere 2-3%. Furthermore, since arithmetic coding encodes all of the data set together in order to achieve the highest coding gain, decoding can take place only after all of the data symbols have been encoded and transmitted. Huffman coding, on the other hand, encodes each symbol separately, thus allowing the decoder to begin decoding as soon as the encoded symbols become available. This can be a crucial factor in some range image applications such as in progressive image transmission. Because of these reasons, we have chosen Huffman coding as the coding scheme to encode the two sets of data points.

Fig. 3.14 shows the block diagram of the encoding process.

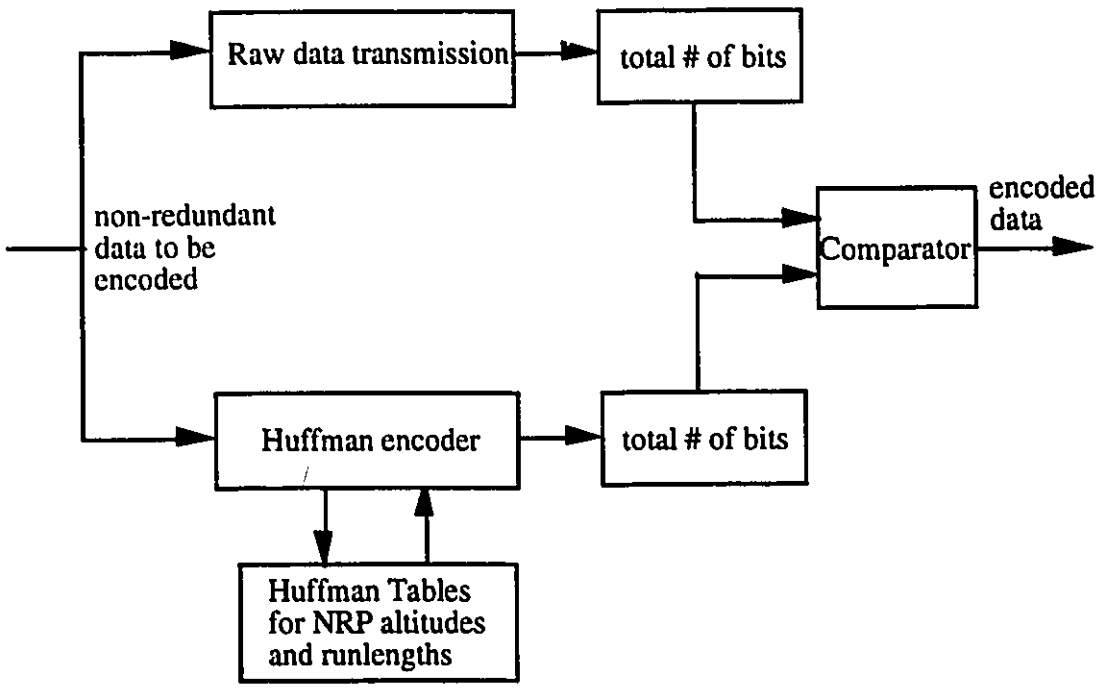


Figure 3.14 Encoder classification

The encoder also includes a raw image transmitter which transmits the data without encoding it. The reason for including this is explained in the next two sections. We will now describe the process with which the runlengths and NRPs are actually encoded.

3.4.1 Runlength Encoding

Runlengths were defined earlier in this chapter as the unidirectional distance between nonredundant points. When the tolerance level, ε, is set to zero, most of the runlengths in a typical image will have a distance of *zero* since the vast majority of adjacent pixels have unequal values. Furthermore, the frequency with which runlengths of one or more appear will decrease monotonically. This implies that the probability of occurrence of a certain run-length size decreases with increasing run-length sizes. Fig. 3.15, which is a histogram of the runlengths in various range images for ε=0 depicts this situation.

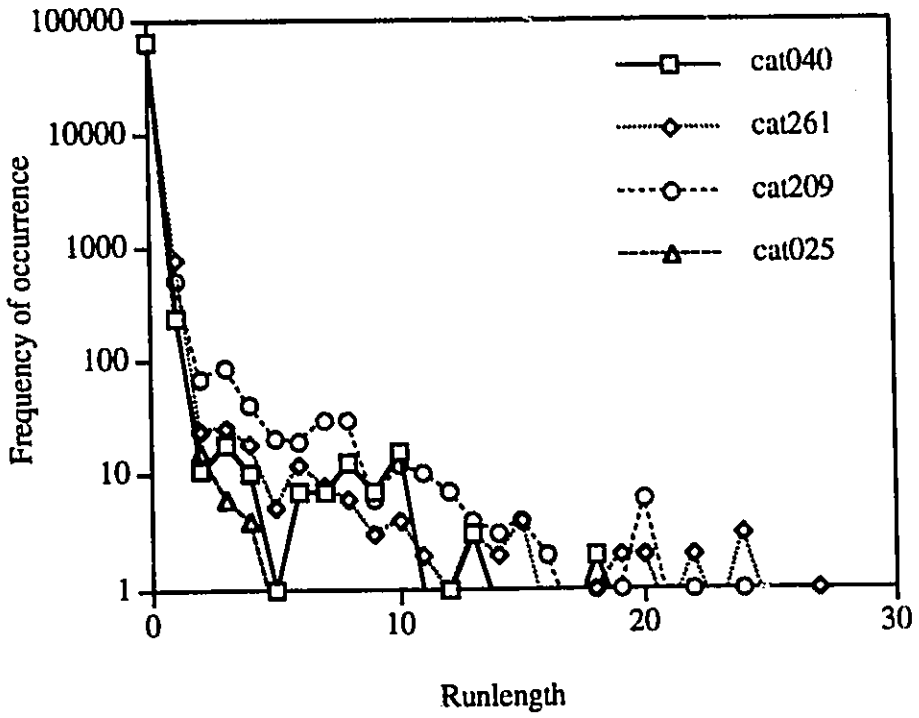


Figure 3.15 Histogram of runlengths for ε=0

When ε is greater than zero, we would expect the occurrence of longer runlengths to increase and that of the shorter runlengths to decrease as the distortion increases, because more and more pixels would become redundant.

However, the probability of occurrence of shorter runlengths would still be higher than the probability of occurrence of longer ones. Fig. 3.16 shows this on a typical range image. We observe from this figure that as ε increases the histogram of runlengths becomes flatter. Once a flat histogram is reached, Huffman coding, or any other coding scheme, would fail and it would, therefore, by preferable to directly transmit the raw data instead of attempting to encode it. Our method takes this into account and compares the number of bits required to transmit the data using Huffman encoding and raw data transmission, deciding in favor of the scheme that requires lesser number of bits. It should be noted, though, that the value of ε beyond which Huffman fails varies from one image to another.

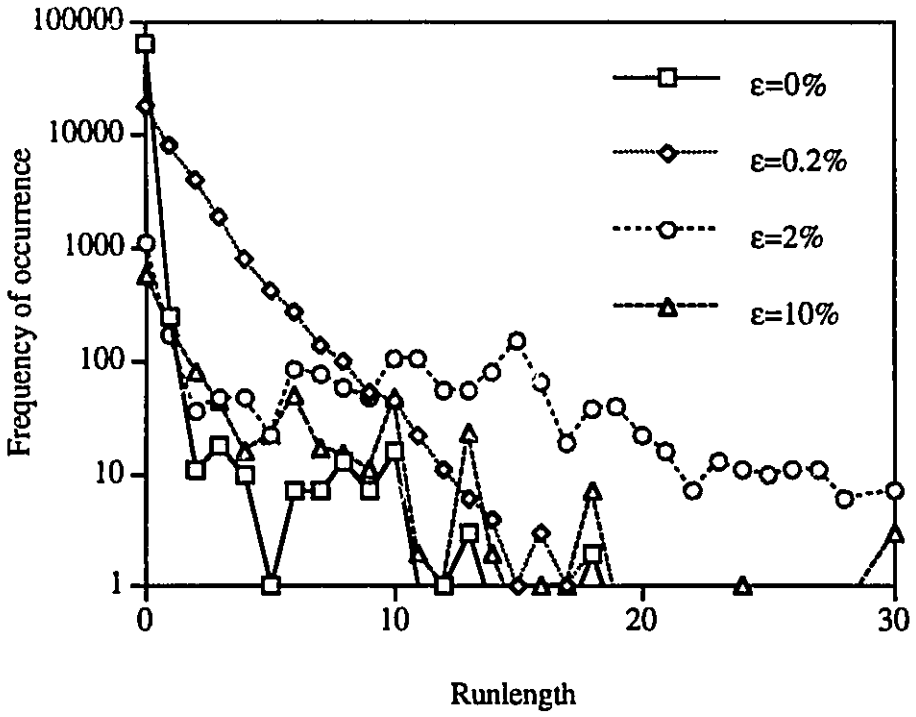


Figure 3.16 Histogram of runlengths for the lossy mode of operation

As can be seen from figs. 3.15-16, the probability of occurrence of runlengths is roughly a monotonically decreasing function of runlengths size. Encoding it using a fixed Huffman tree is, therefore, adequate. The construction of the tree, though, is a critical task.

We created a Huffman tree using eight diverse and dramatically different range images, but will encourage users to define their own Huffman tree according to their application. Fig.3.17 depicts the histogram we used to create the Huffman tree for runlengths.

Note that all three encoding classes defined earlier in this section use the same procedure for runlength encoding. Furthermore, two-directional encoding uses the same runlength Huffman tree to encode *both* sets of runlengths.

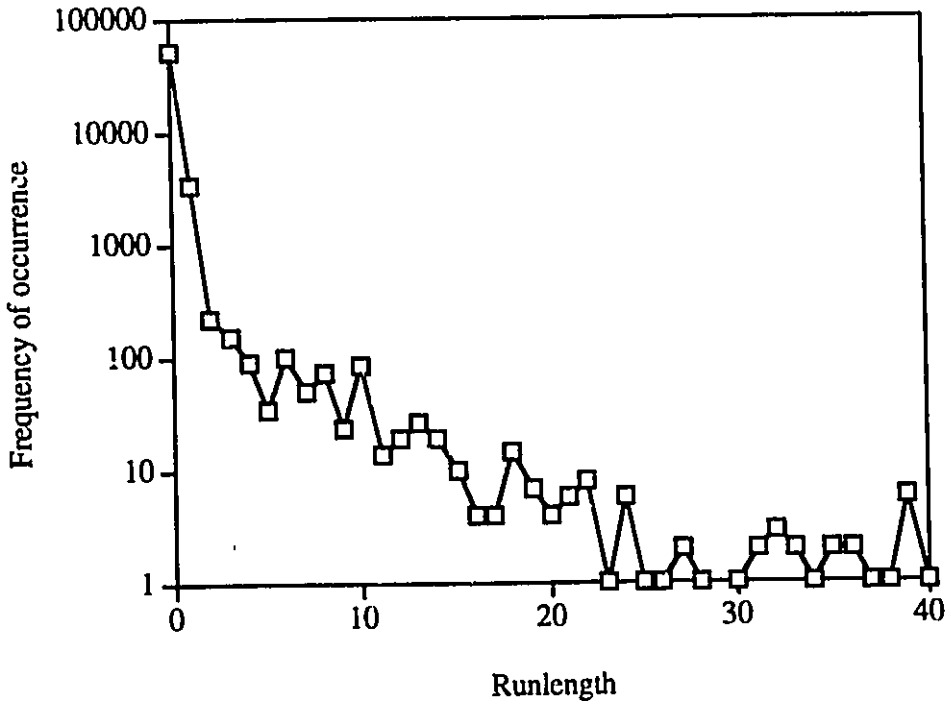


Figure 3.17 Histogram of runlengths used to create a fixed Huffman tree

3.4.2 NRP Altitude Encoding

Encoding nonredundant points is a considerably more difficult task than encoding runlengths. This is because the histogram of NRPs is spread out and the data set is quite uncorrelated. The raw NRPs, therefore, have high entropy; this is to be expected because of the fact that the points are nonredundant! Hence, encoding the actual nonredundant point data set does not give good yield. It is, therefore, imperative that another encoding technique be found.

In the lossless mode of operation, adjacent NRPs generally correspond to adjacent pixels in the actual image because runlengths are mostly of length zero. This implies that coding the differences between the intensity of adjacent NRPs might give a better performance. In fact, this technique, which is also known as *differential encoding* [47], is frequently used in such situations. Fig. 3.18 shows the histogram of a differentially encoded set of NRPs for the range image cat040 (shown in fig. 4.1).

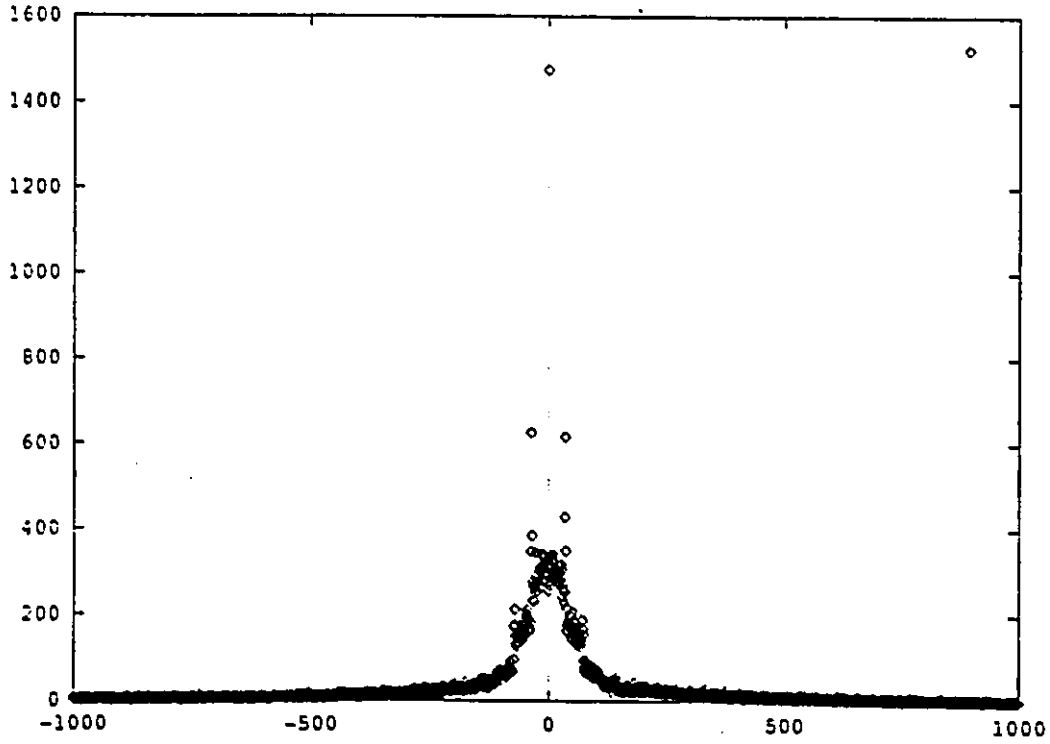


Figure 3.18 Histogram of differences of NRPs for cat040. ε=0

The histogram shows that the dynamic range has been significantly reduced, from 15 bits to 9 bits. Variance is higher for the NRPs than it is for the differences, which indicates that the entropy encoder will experience an improvement in the performance. We can see that the histogram resembles a double exponential, as it is expected to [47], since it represents pixel-to-pixel differences.

On the other hand, when ε is greater than zero, the runlength between two adjacent NRPs increases as distortion is introduced in the image, resulting in lesser spatial correlation between adjacent NRPs. However, for small amounts of distortion, the spatial correlation between adjacent NRPs is high enough to still warrant the use of differential encoding. As distortion increases, though, the correlation between adjacent NRPs decreases resulting in a decrease in the frequency of small pixel intensity differences, as shown in fig. 3.19.

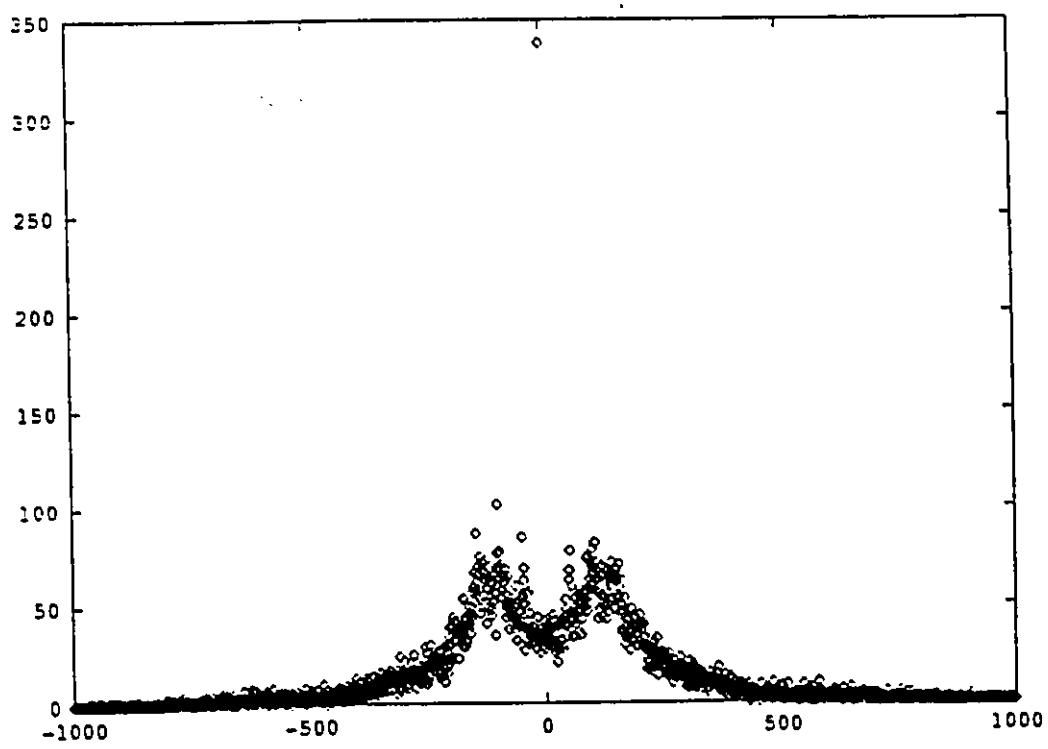


Figure 3.19 Histogram of differences of NRPs for cat040. ε=0.25%

This implies that the probability of occurrence of smaller differences decreases and larger differences increases with increasing distortion, and vice versa. We observed that the two peaks in fig. 3.19 corresponds to the maximum absolute error value. The probability of occurrence of differences between the two peaks is lower because of the fact that the differences in this region are less than the maximum absolute error value specified.

The histogram of differences behaves in a manner similar to the histogram of runlengths, i.e. as ε increases, the histogram flattens out causing Huffman coding to fail. When this happens, we no longer encode the data with Huffman coding; instead, we transmit the data in its original raw form.

A problem that arises at this point is regarding the Huffman table. Transmitting the table will result in higher coding gain but because of the high overhead involved in table transmission, compression will be much lower. Creating a fixed Huffman table, on the other hand, will result in higher compression but lower coding gain. Fortunately, a tradeoff can be made between coding gain and compression by using a Huffman table that is amenable to both. As is the case with runlength Huffman tables, creating a fixed table is application dependent. We have used the same diverse variety of range images that were used to create the table for runlengths to create the Huffman table for NRPs. The histogram used to create the table is shown in fig. 3.20.

It is possible to create another fixed Huffman table that corresponds closer to fig. 3.19. In fact, it is possible to construct many fixed Huffman tables and choose the one amongst them whose histogram has a higher correlation with the image histogram to be compressed. However, we observed that the performance improvement in doing so is not substantial and is, thus, not worth the effort.

The encoder, therefore, encodes the NRPs differentially, using Huffman encoding. However, it does so differently for the different encoding classes. For unidirectional encoding, we observe that a strong correlation exists between neighboring NRPs in the direction perpendicular to the scanning direction. This spatial coherence in images is

expected since it arises from the spatial coherence of the physical surface being imaged [73].

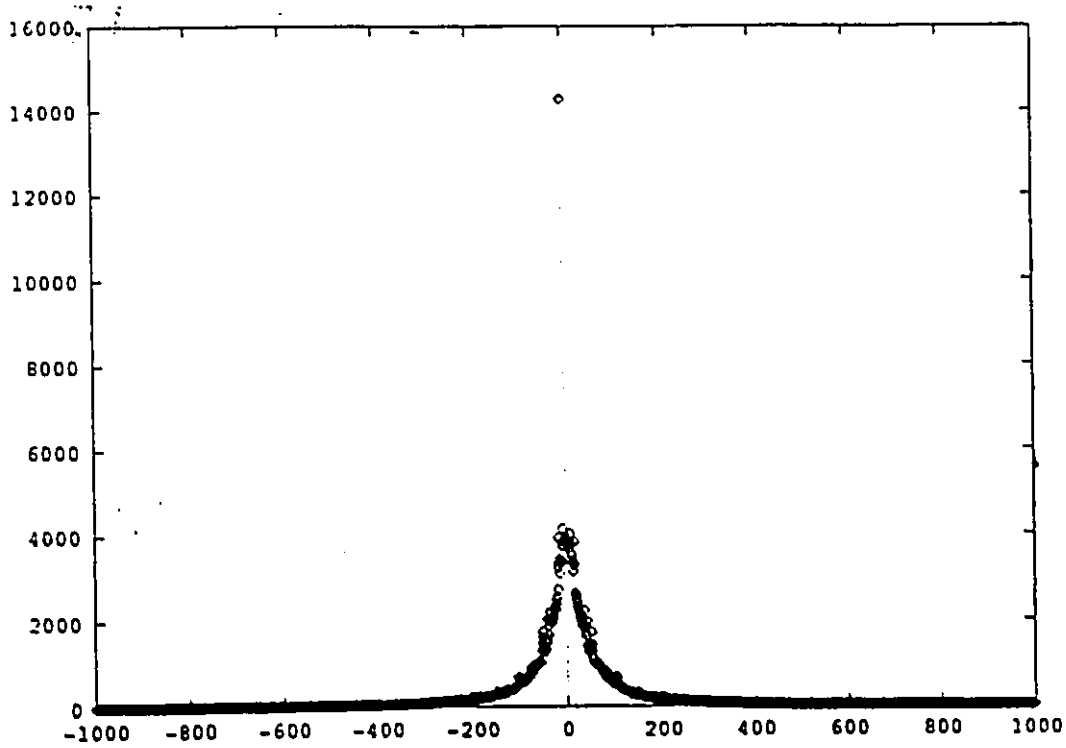


Figure 3.20 Histogram of differences of NRPs used to create a fixed Huffman tree

Having already removed the pixel-to-pixel coherence in one direction using the unidirectional redundancy finder, only the pixel-to-pixel coherence in the other direction remains. Taking advantage of that, the encoder computes the difference between NRPs in adjacent lines. Before doing this, however, the NRPs are arranged line-by-line after having the unidirectional distance between them removed. This implies introducing an "End-of-line" (EOL) symbol in the symbol set. This ordering generally results in the number of NRPs in each line being different. This scenario is depicted in Fig. 3.21.

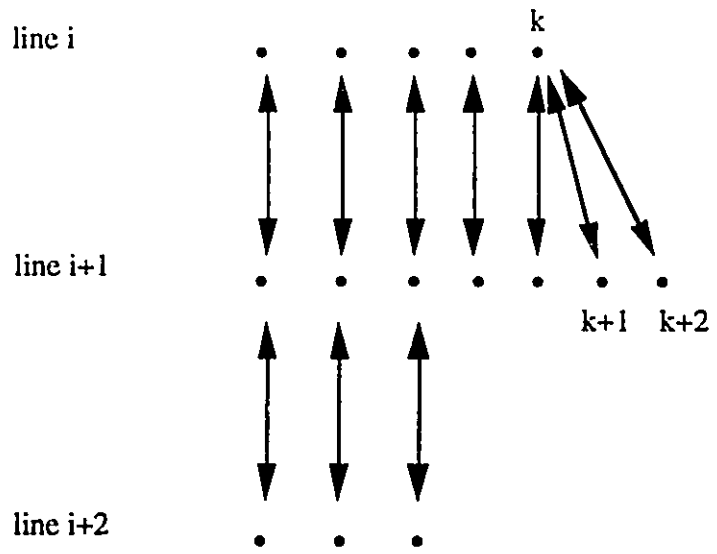


Figure 3.21 Difference encoding in unidirectional encoding

Since line i+1 is longer than line i, subtracting the points in the latter from the points in the former on a one-to-one basis will cause errors. To overcome this, the encoder subtracts the last point in line i from the extra points in line i+1, i.e. it subtracts point k from point k+1 and from point k+2. Even though the first line is not differentially encoded, it is nevertheless still coded using the differential coding tree.

Two-directional encoding is very similar to unidirectional encoding. After the first step, the NRPs are lined in the same way as they are in unidirectional encoding. After the second step, the NRPs, which are lesser in number, are grouped together in the same manner as they had been after the first step. Then, difference encoding is done in the same fashion as that depicted in fig. 3.21.

In multi-directional encoding, the difference between "adjacent" Peano scanned NRPs is taken and encoded, as shown in fig. 3.22. The encoding process is actually simpler than the other two encoding techniques. All the points, except the first, are differentially encoded and no EOLs symbols are needed. It should be noted, however, that the

complexity of the Peano scanned system lies not in the coder but in the scanner and de-scanner.

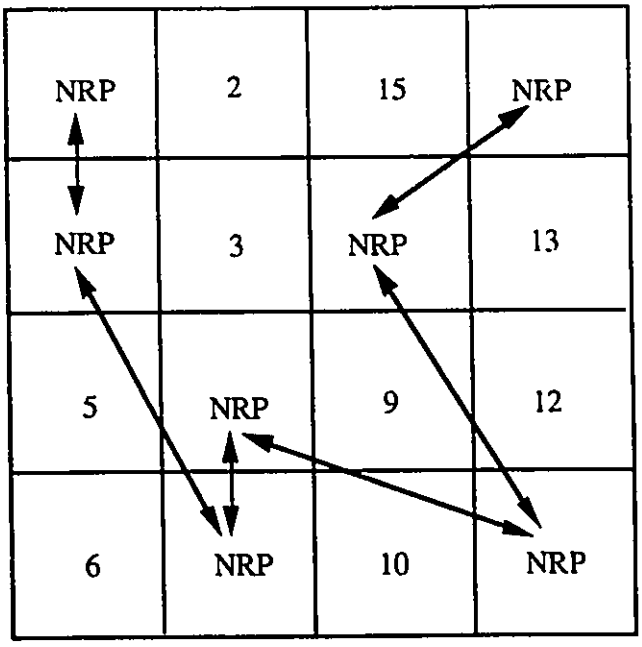


Figure 3.22 Difference encoding in Peano scanned images

3.5 Decoding

The entropy decoder, as shown in Fig. 3.23, receives the encoded data, which includes:

- encoded bit stream of run-lengths
- encoded bit stream of NRPs and CR's

It decodes the runlengths simply by traversing the runlength Huffman tree. In two-directional decoding, though, it is required to decode two sets of runlengths. The decoder, therefore, decodes each set separately. Both sets use the same tree for decoding.

The NRPs are also decoded similarly. When multi-directional encoding has been done, decoding is straightforward. It is performed by simply looking up the encoded word in the

Huffman tree. In the other two cases, though, the decoder also has to decipher EOL symbols, as well as take care of decoding the first line of the data set, which is not differentially encoded.

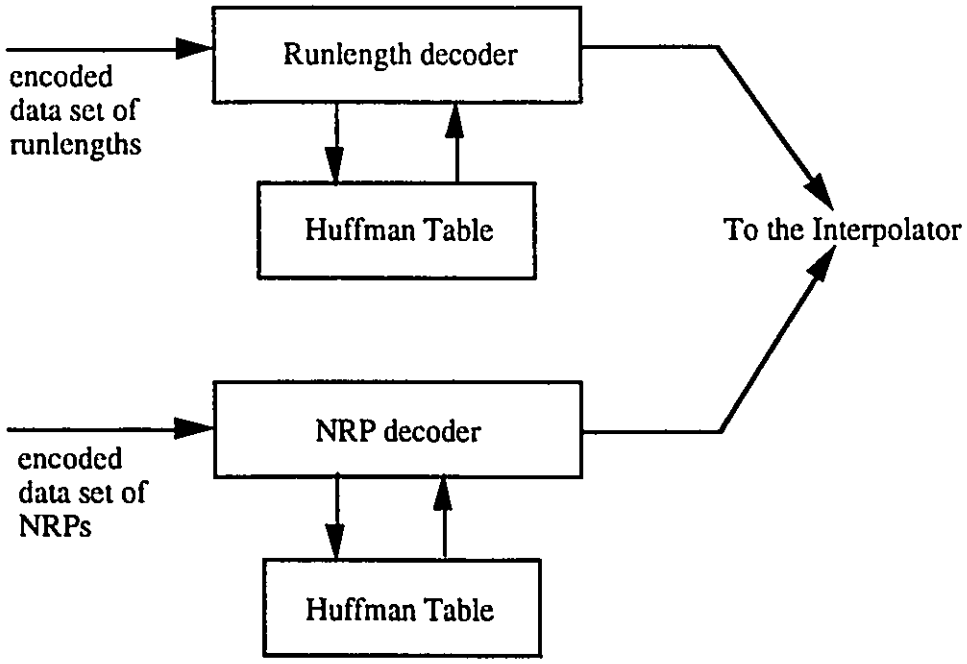


Figure 3.23 Block diagram of the decoder

After decoding both sets of information, the decoder forwards them to the interpolator, which is described in the next section. It should be noted that the encoding stage does not introduce any errors in the system. Information loss occurs only in the redundancy finder.

3.6 Interpolation

The interpolator receives two sets of information from the entropy decoder:

- one or two sets of run-length data blocks

• NRP data block

Both sets of data are stored in buffers before interpolation can begin. Fig. 3.24 shows how the interpolator works. Assume that points k and k' are NRPs with a run-length of 6. Then, six points are interpolated between them. Thus, point k' becomes point k+7. Similarly, if point k" is also a NRP with a run-length of 2, then this implies that two points are placed between the NRPs k' and k". This process is repeated until all the data points have been reconstructed.

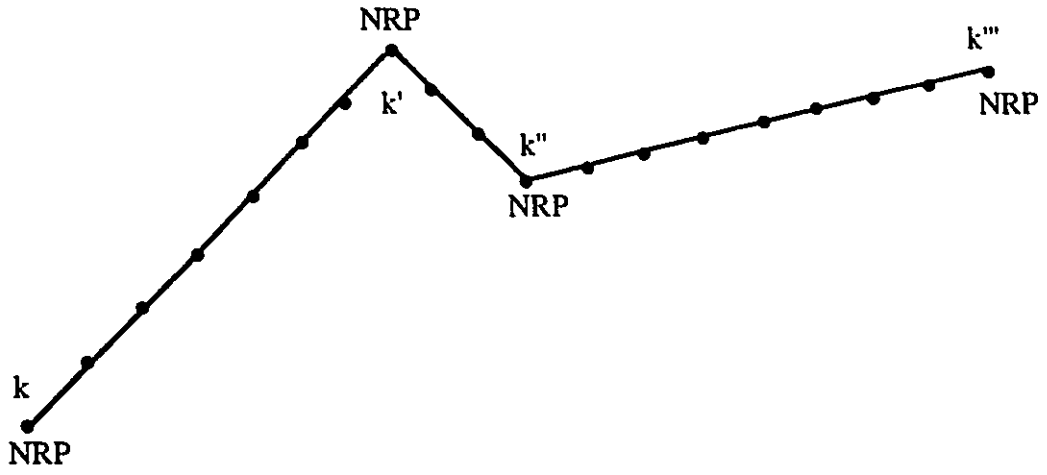


Figure 3.24 Interpolation of the NRPs using runlengths

When unidirectional scanning is used, the interpolator must first align the NRPs properly, before interpolation can begin. It does this with the help of EOL symbols. When two-directional decoding is needed, the interpolator must also take into account the fact that not all lines were passed through the redundancy finder in step two of the two-directional process. Therefore, it must align the NRPs accordingly.

The interpolator, having reconstructed the image, then passes on the information to the last element in the chain -- the de-scanner.

3.7 De-scanning

The de-scanner reverses the action of scanning. Its structure is depicted in fig. 3.25. After the compressed image has been reconstructed by the interpolator, there is a need to de-scan it so that the pixels are appropriately positioned on the image plane.

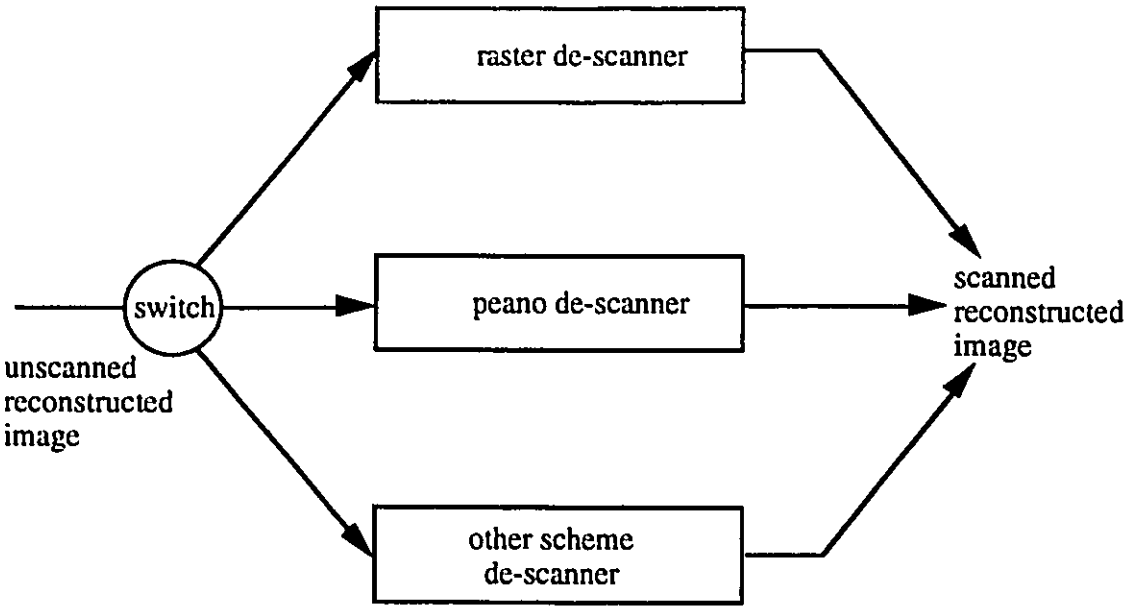


Figure 3.25 Structure of the de-scanner

When Peano scanning is used, the de-scanner computes the Peano scanning of order-j such that the reconstructed image fits inside it. When the size of the image is of $4^j \times 4^j$, which is the standard Peano curve size, the de-scanner computes the Peano curve quite effortlessly using the recursive Peano curve generating algorithm. However, when this is not the case, the de-scanner has to find an appropriate Peano curve pattern using the technique described in section 3.2.

After having found the appropriate Peano scan, the de-scanner simply "stretches" the image data points and stores them into rasterlines. As was pointed out earlier, the scanner-

de-scanner combination is a module that introduces complexity in the Peano scanning scheme. However, it is clear that the complexity is not enormous and can, therefore, be acceptable for the majority of applications.

3.8 Summary

In this chapter, we presented the details of our range image compression algorithm. We started out by giving an outline of the algorithm. This was followed by an introduction to the various scanning techniques that can be used in the algorithm, such as Peano and raster scanning. Then, we presented a description of the algorithms main building block, the redundancy finder. The workings of the unidirectional, two-directional, and multi-directional redundancy finder were explained. We then explained the encoding of the non-redundant information. The justification for using Huffman coding was given and the development of the fixed Huffman tree was described. At the receiver's end, the workings of the decoder was explained first. Then, the interpolation of the decoded data was presented. Finally, we presented the issues involved in the de-scanning of the decoded information.

The choice of the scanning scheme is application dependent. Unidirectional scanning, such as the popular raster scanning, can be used for compression with low system complexity. Two-directional scanning can increase performance by removing redundancy found in two orthogonal directions but at the cost of added system complexity. Peano scanning, however, increases performance at very little added system complexity.

The use of differential encoding is brought about due to the high pixel-to-pixel correlation found in neighborhood pixels. We choice a fixed Huffman tree because of the relative invariance in the overall structure of the histogram of the nonredundant information.

The encoding of NRPs and runlengths is a lossless process; however, it is certainly possible to quantize the data so as to achieve higher compression, in which case the encoding process would be lossy. However, quantization at this stage may well mean a loss of control over the tolerance, ε.

The decoding process is straightforward; fixed Huffman trees are used to do so. Even though a Huffman tree is provided, it may be more efficient to create them individually depending on application. However, performance does not increase significantly with other fixed Huffman trees. Interpolation and de-scanning produce reconstructed images which can then be processed accordingly.

In the next chapter, we describe the performance of the many facets of the algorithm described here. Finally, we show a comparison between the algorithm and JPEG, the still image compression standard.

Chapter 4

Performance Analysis

In this chapter, we present the performance analysis of the FAN range image compression algorithm. First, we describe the range images on which performance analysis was made. Then, a brief comparision between Huffman and arithmetic coding is given. Various system performances, such as unidirectional, two-directional and Peano systems, are then outlined. In unidirectional systems, the performance of our algorithm on scanning schemes such as raster, vertical, and horizontal-chain techniques, are described. In the two-directional system, we describe the algorithms performance on images after they have passed through the two-directional redundancy finder. In Peano system, performance is evaluated on Peano scanned images. Finally, we present the results of the comparison of our algorithm with JPEG, the still image compression standard.

4.1 Range image database

For this study, we have selected four range images from the database of range images available at the National Research Council of Canada. The images were carefully chosen so that they encompass the whole range of diversity found in 3-D imaging. Figs. 4.1-4.4 display these four images in two formats.

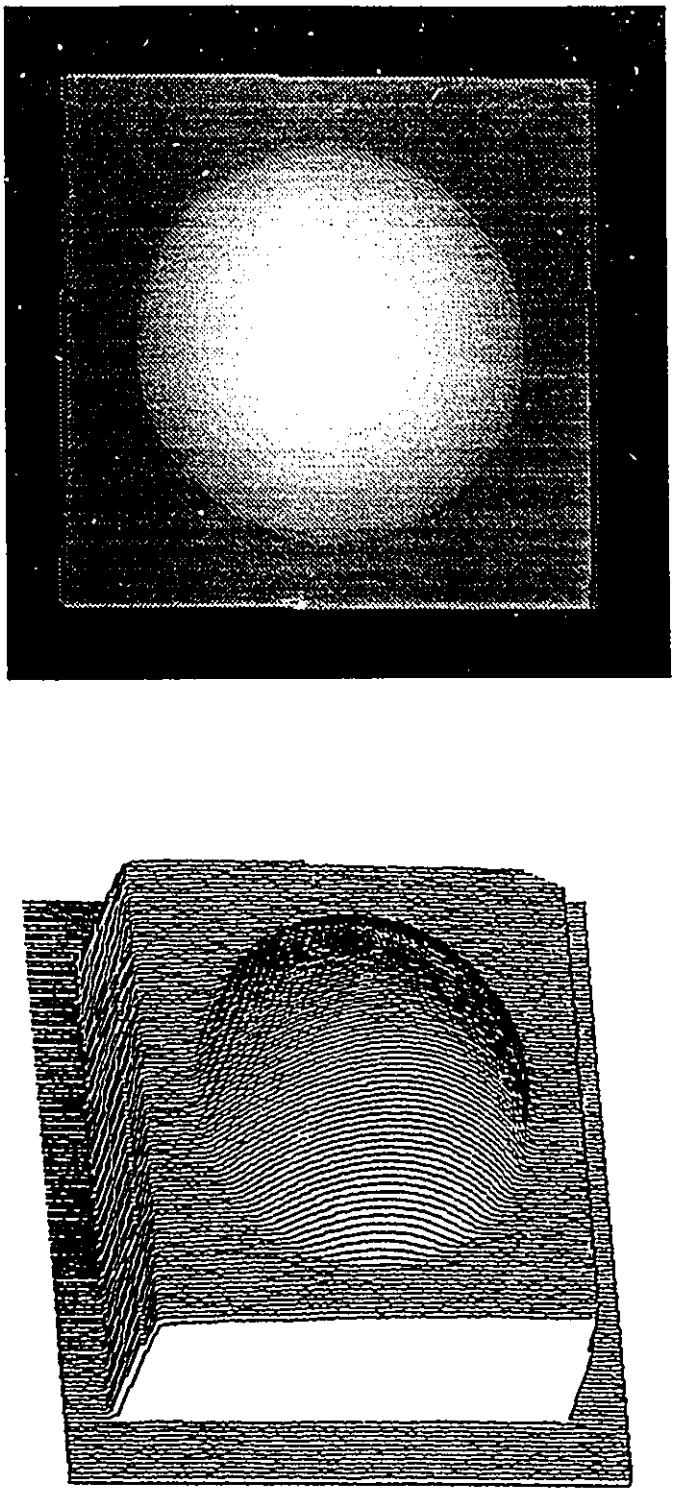


Figure 4.1 2-D and 3-D representation of image cat040

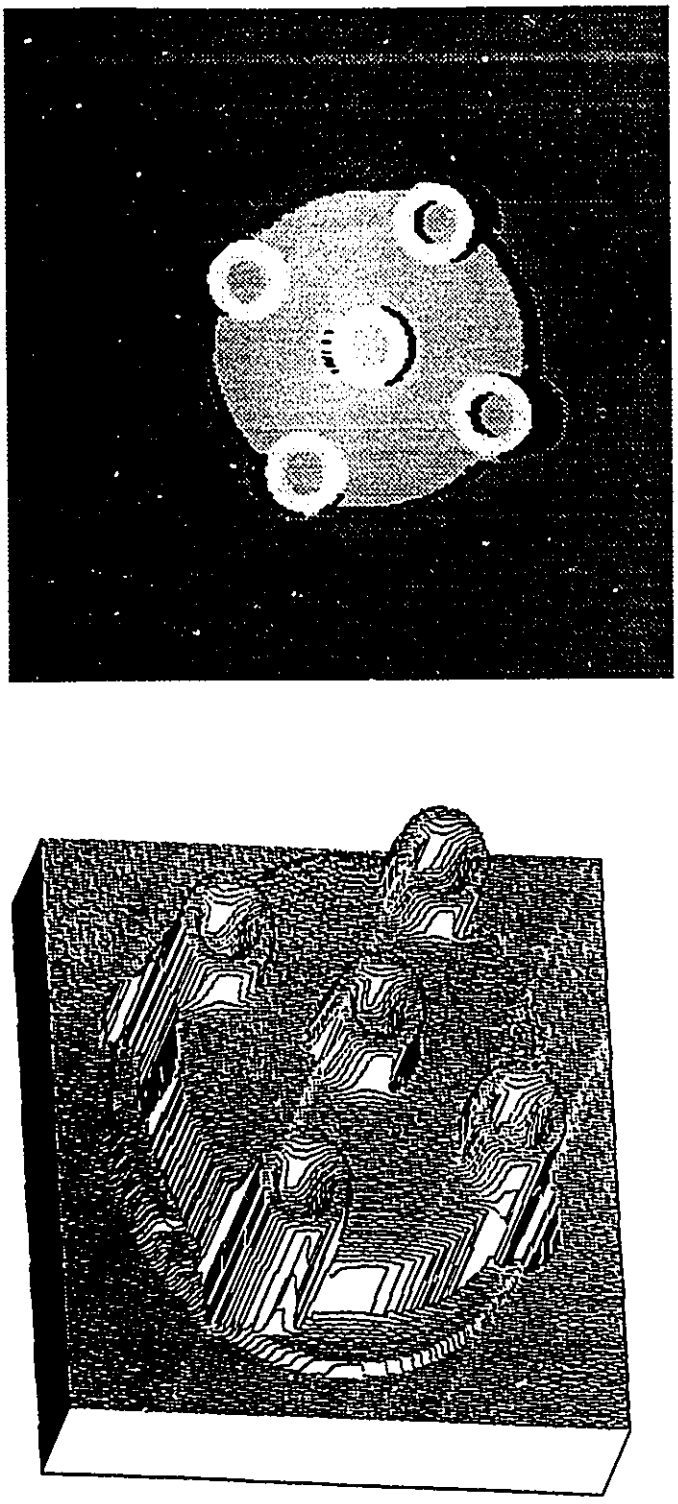
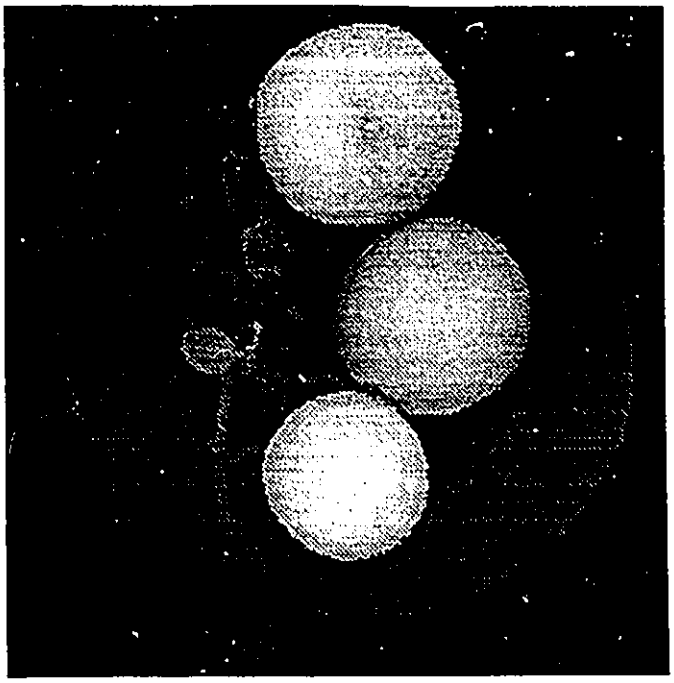


Figure 4.2 2-D and 3-D representation of image cat261



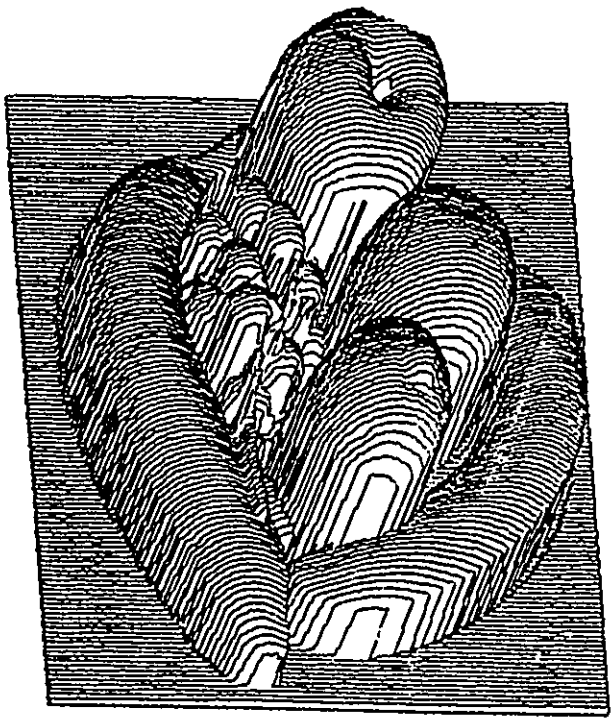


Figure 4.3 2-D and 3-D representation of image cat209

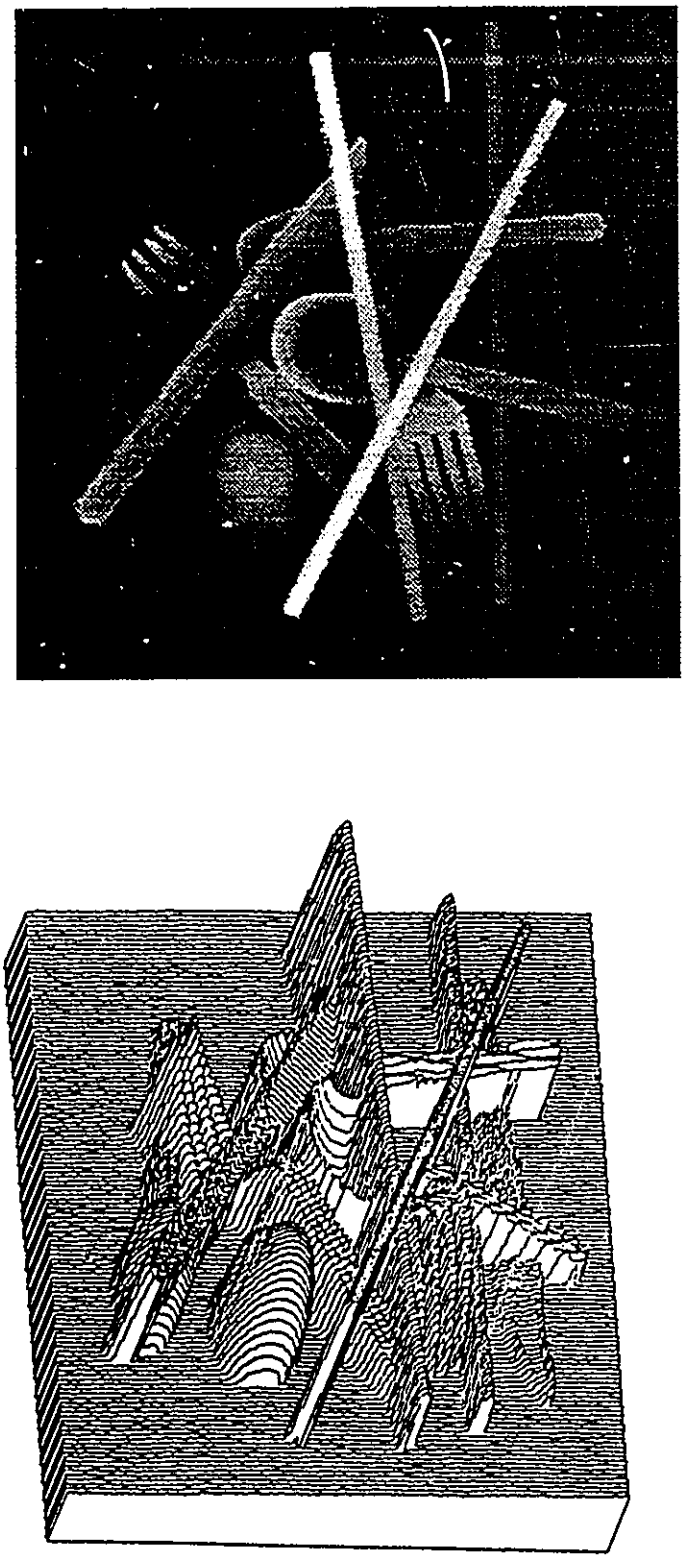


Figure 4.4 2-D and 3-D representation of image cat025

The first representation of the image is two dimensional. It depicts the top view of the image. The second representation, on the other hand, is the image's contour map, or three dimensional view. Fig. 4.1 depicts a semi-spherical object (cat040). The image is quite simple; however, it has many slopes in it and forms a good test on algorithms based on line approximation, such as the algorithm we are proposing. Fig. 4.2 depicts a distributor cap (cat261). This image is also simple, even though it is more complex than the semi-spherical object because of its many edges. However, because of the smaller information covered area of the image, it may give misleading performance information in terms of compression.

Fig. 4.3 describes a collection of fruits (cat209). This image is more complex than the other two because it contains objects with various shapes and sizes. The image forms a good test because of its ability to test an algorithms performance on objects with a variety of curvatures. Finally, fig. 4.4 is a cluster of various utensils (cat025). The vast amount of information found on its surface makes it very complex, and therefore, ideal for testing compression algorithms. The image is no doubt the most complex of the images used in the analysis.

The range images depicted above are of high resolution (15-bit representation); the gray-level values range from 0 to 32000. Their size is 256-by-256 pixels and the unit employed for maximum absolute error, ε, is the percentage of dynamic range.

4.2 Huffman Vs. Arithmetic coding

We mentioned in chapter 3 that the performance of both Huffman and arithmetic coding schemes are almost identical, especially on high resolution images such as medical images. Table 4.1 compares the two techniques on two different raster-scanned range images. Fig.4.5 shows the rate distortion curves for both methods on these images.

tolerance	Huff:cat040	Arith:cat040	Huff:cat261	Arith:cat261
0%	1.40:1	1.44:1	1.64:1	1.69:1
0.2%	2.41:1	2.49:1	7.16:1	7.31:1
0.5%	5.76:1	5.87:1	10.78:1	11.09:1
2%	16.18:1	16.51:1	18.38:1	19.11:1
4%	20.96:1	21.78:1	22.07:1	22.87:1
7%	24.64:1	25.87:1	25.27:1	26.15:1
10%	29.98:1	30.56:1	27.19:1	28.01:1
15%	37.13:1	38.24:1	31.79:1	33.06:1
20%	39.84:1	40.43:1	37.72:1	38.47:1
30%	48.79:1	50.24:1	43.59:1	44.89:1
40%	57.2:1	58.78:1	45.60:1	47.24:1
50%	78.41:1	80.56:1	46.18:1	47.56:1
75%	250.26:1	252.77:1	61.81:1	64.03:1

Table 4.1 Compression ratios of the FAN algorithm using Huffman and arithmetic coding

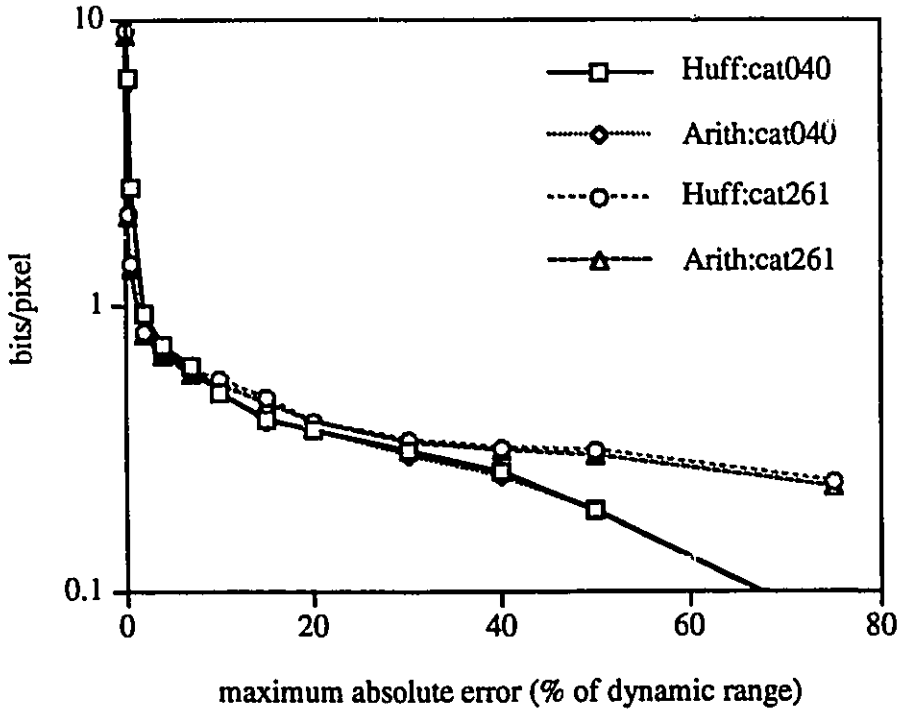


Figure 4.5 Rate distortion curves for various 15-bit range images

We observe that arithmetic coding is only slightly better then Huffman coding in terms of compression ratios, i.e. it gives a 2-3% improvement! Such a minutesimal improvement in performance does not warrant its use over Huffman coding, especially since it prevents the decoder from decoding until all of the data has been encoded and transmitted. We have, therefore, used Huffman coding in all our analysis.

4.2 Unidirectional system analysis

As mentioned earlier in this thesis, there are many unidirectional scanning techniques available to the user. We have, however, employed only three techniques because we believe that they cover the majority of applications. They are raster, vertical and horizontal-chain scanning systems. Performance on these systems is described below.

4.2.1 Raster scanning system

Table 4.2 outlines the performance of the algorithm on various raster scanned images. Fig.4.6 shows the rate distortion curves for these images. We observe that there is quite a large jump in the compression ratio with a slight increase in distortion. This is to be expected because of the inherent nature of the method. Note that image cat261 initially gives higher compression compared to image cat040 even though it is more complex. This is due to the lesser information content size of the image, as can be observed form the image itself.

In order to appreciate the performance of the algorithm on these images, it is imperative to study the reconstructed image carefully. Fig. 4.7 shows the reconstructed image of cat261 for an error of about 7%. Fig. 4.8 is the error image; it has been scaled so that the error is proportional to the intensity value in the image.

tolerance	cat040	cat261	cat209	cat025
0%	1.40:1	1.64:1	1.41:1	1.35:1
0.2%	2.41:1	7.16:1	3.13:1	1.78:1
0.5%	5.76:1	10.78:1	4.76:1	3.00:1
2%	16.18:1	18.38:1	7.78:1	4.68:1
4%	20.96:1	22.07:1	9.68:1	6.25:1
7%	24.64:1	25.27:1	11.27:1	8.10:1
10%	29.98:1	27.19:1	12.28:1	9.87:1
15%	37.13:1	31.79:1	14.33:1	12.80:1
20%	39.84:1	37.72:1	15.86:1	16.08:1
30%	48.79:1	43.59:1	18.21:1	22.64:1
40%	57.2:1	45.60:1	18.56:1	33.29:1
50%	78.41:1	46.18:1	22.21:1	45.87:1
75%	250.26:1	61.81:1	26.06:1	107.92:1

Table 4.2 Compression ratios of the FAN algorithm on various raster-scanned range images

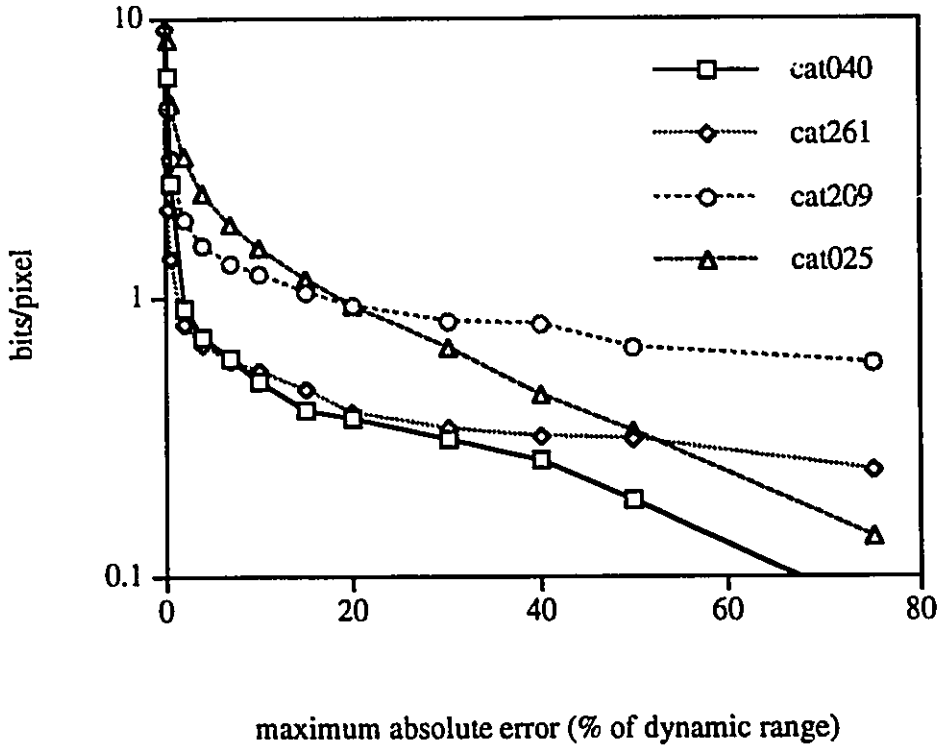


Figure 4.6 Rate distortion curves for various 15-bit range images

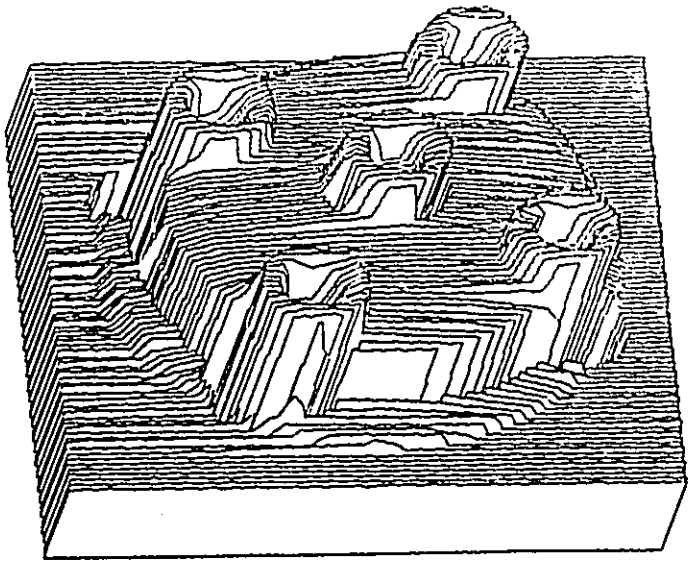


Figure 4.7 Surface plot of the reconstructed raster scanned image cat261. ε=7%

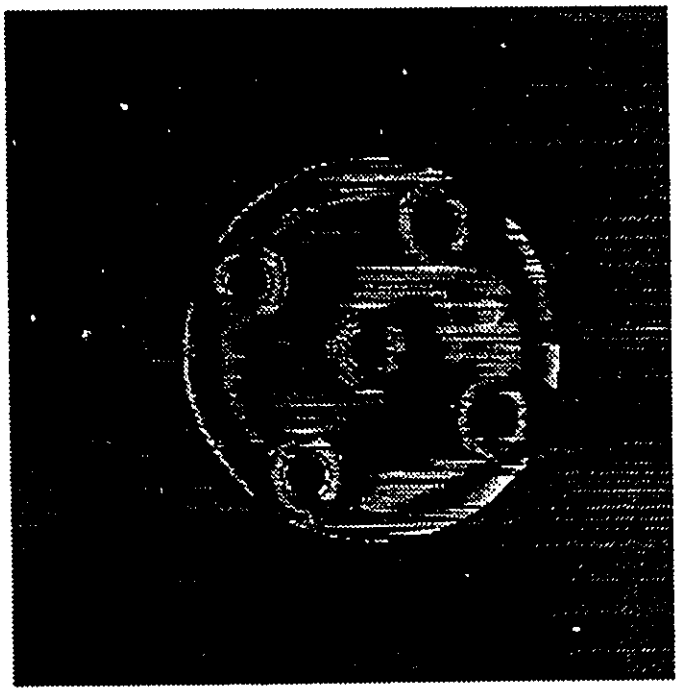


Figure 4.8 Error image for ε=7%

Close examination of these two figures reveals streaks. These streaks arise due to the interpolation of NRPs and are inherent to the raster scanning scheme. It should be noted, though, that these streaky errors are always within the user specified tolerance level. However, if there is some processing to be done after the image is reconstructed, the raster scanning scheme is not appropriate because the compression artifacts are structured. It would be preferable to have unstructured or pseudo-random artifacts because they are more similar to random noise.

4.2.2. Vertical scanning system

Vertical scanning is often used in image compression applications which contain images comprising of vertical objects. This is because vertical objects can be exploited using the redundancy found in their shape. Of course, the same can be said for horizontal scanning systems, such as raster scanning techniques, which can efficiently exploit horizontal objects.

tolerance	cat040	cat261	cat209	cat025
0%	1.39:1	1.62:1	1.38:1	1.30:1
0.2%	2.25:1	7.18:1	3.38:1	1.96:1
0.5%	6.01:1	10.78:1	5.16:1	3.78:1
2%	19.66:1	16.24:1	8.69:1	7.26:1
4%	25.38:1	19.97:1	10.52:1	9.84:1
7%	28.57:1	23.14:1	12.24:1	12.41:1
10%	32.37:1	25.16:1	13.55:1	14.44:1
15%	39.14:1	29.64:1	15.46:1	17.55:1
20%	40.61:1	34.77:1	17.44:1	20.80:1
30%	40.87:1	40.23:1	21.21:1	33.64:1
40%	43.16:1	43.92:1	21.91:1	52.68:1
50%	54.81:1	45.56:1	26.19:1	85.21:1
75%	254.47:1	66.18:1	32.01:1	147.74:1

Table 4.3 Compression ratios of the FAN algorithm on various vertically-scanned range images

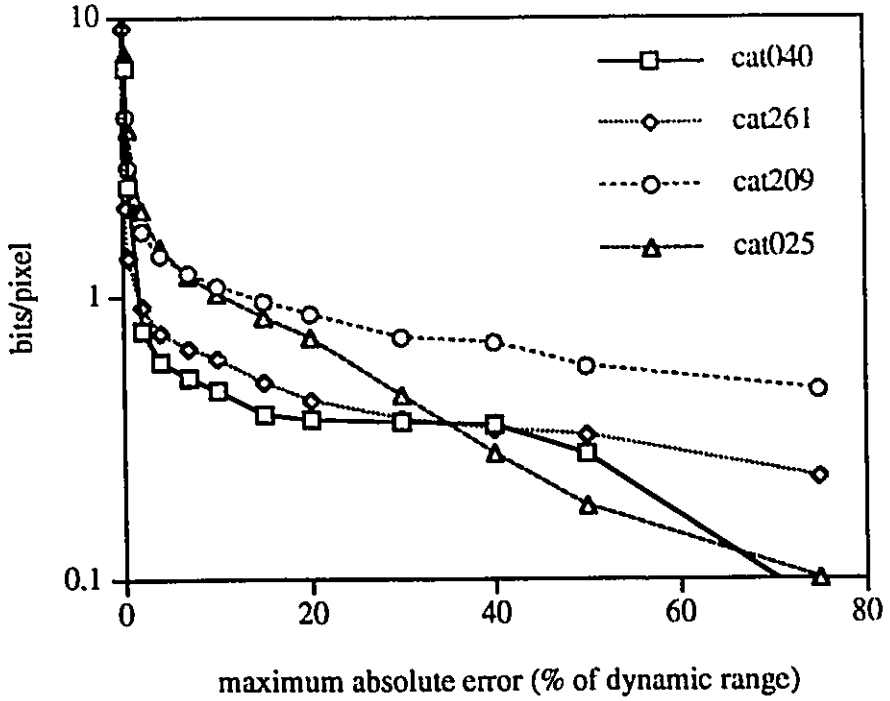


Figure 4.9 Rate distortion curves for vertically scanned images

However, since images generally do not contain objects of a particular orientation, we would expect no dramatic change in performance between vertical scanning and raster scanning systems.

Table 4.3, which highlights the compression ratios for the same images as above, and fig.4.9, which shows the corresponding rate distortion curves, verify our hypothesis.

4.2.3 Horizontal-chain scanning system

Horizontal-chain scanning, as well as vertical-chain scanning, is used in applications where a high degree of pixel-to-pixel correlation exists between pixels located near the image

boundary or frame. For example, if the surface on which an object (to be scanned using a range imaging sensor) is placed has a tilt or slope, then distance measurements on both extremes of the slope will be different. Therefore, it would be beneficial for the scanning technique to exploit this fact in order to achieve better compression ratios. However, since the technique is not all that different from raster scanning, or vertical scanning, we would expect to achi. . about the same performance. This is illustrated in table 4.4, which is a performance evaluation on different range images, and in fig. 4.10, which is the corresponding rate distortion curve.

tolerance	cat040	cat261	cat209	cat025
0%	1.08:1	1.21:1	1.10:1	1.05:1
0.2%	2.18:1	6.64:1	2.80:1	1.59:1
0.5%	5.45:1	10.25:1	4.36:1	3.05:1
2%	15.33:1	16.36:1	7.15:1	4.83:1
4%	17.63:1	19.36:1	8.74:1	6.29:1
7%	21.18:1	22.27:1	10.18:1	8.09:1
10%	25.18:1	23.95:1	11.28:1	9.80:1
15%	33.87:1	28.55:1	13.18:1	12.58:1
20%	37.21:1	34.09:1	14.71:1	15.63:1
30%	38.22:1	38.42:1	16.80:1	22.36:1
40%	59.52:1	39.44:1	16.98:1	33.08:1
50%	80.94:1	41.73:1	20.66:1	46.31:1
75%	263.83:1	54.08:1	26.06:1	108.32:1

Table 4.4 Compression ratios of the FAN algorithm on various horizontal-chain scanned range images

All of the three unidirectional techniques performed almost equally well, as we expected. Taking the simplicity of the system into consideration, the results obtained were more than adequate. However, all of them contained streaks in the reconstructed image which cause post-processing problems. Unfortunately, this is inherent to the unidirectional scanning scheme and cannot be avoided.

We will see in the next section that higher compression can be achieved using a technique that removes redundancy in two orthogonal directions.

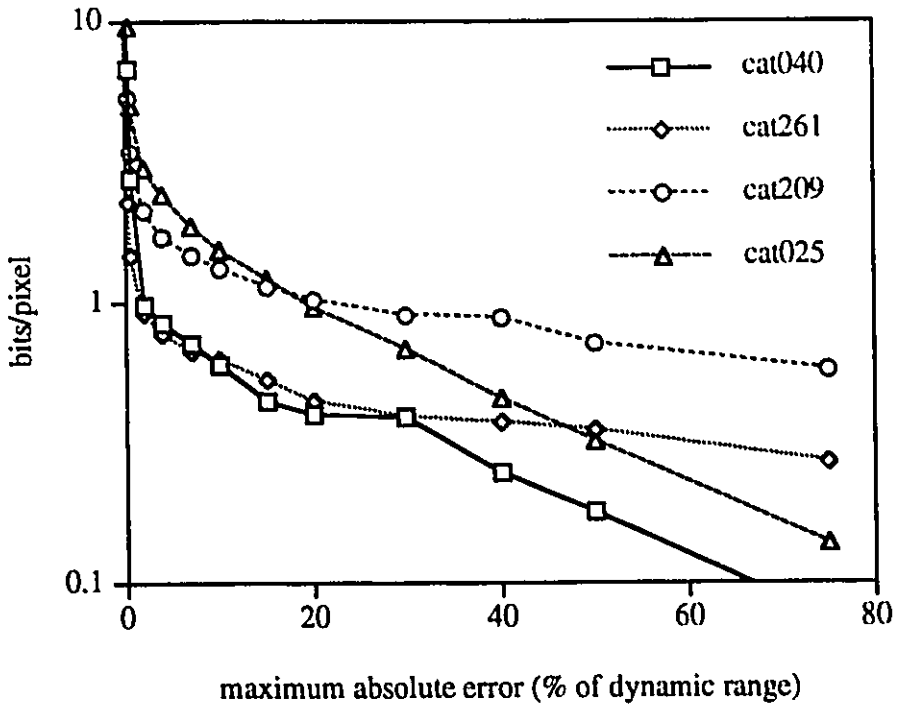


Figure 4.10 Rate distortion curves for horizontal-chain scanned images

4.3 Two-directional system analysis

In an effort to increase the algorithms performance, we attempted to exploit the 2-D nature of the image plane. It is a two-pass process that removes redundancy in images in two orthogonal directions. Two error values, ε_1 and ε_2, are specified, one for each direction. The unidirectional system is then a special case of the two-directional method, i.e. the case when the maximum absolute error in the second direction is set to zero, while the first one

is set to the maximum allowed value. Table 4.5 outlines its performance when both error parameters, ε_1 and ε_2, are set to half the value of the specified maximum tolerance, ε. Fig.4.11 depicts the systems rate distortion curves. Raster scanning has been used for this analysis, though it is not mandatory to do so.

tolerance	cat040	cat261	cat209	cat025
0%	1.55:1	1.77:1	1.52:1	1.43:1
0.2%	2.62:1	7.86:1	3.47:1	1.92:1
0.5%	6.55:1	11.64:1	5.19:1	3.24:1
2%	17.96:1	20.03:1	8.67:1	5.11:1
4%	23.48:1	24.49:1	10.64:1	6.75:1
7%	27.84:1	28.31:1	12.60:1	8.67:1
10%	32.98:1	30.18:1	13.52:1	10.75:1
15%	40.47:1	34.97:1	15.47:1	14.08:1
20%	43.82:1	41.11:1	17.44:1	17.65:1
30%	44.18:1	47.94:1	19.69:1	24.13:1
40%	62.91:1	49.61:1	20.42:1	35.99:1
50%	90.17:1	51.23:1	23.97:1	48.57:1
75%	271.28:1	67.81:1	28.43:1	114.39:1

Table 4.5 Compression ratios of the two-directional FAN algorithm on various raster scanned range images

It is clear from the table and the figure above that using the two-directional compression system gives an average of about 10% increase in the amount of compression. This, unfortunately, is not a tremendous improvement, given the amount of complexity introduced into the system. We will see in the next section that Peano scanning, which is a multi-directional scanning system, improves the performance of the algorithm dramatically without adding that level of complexity.

It should be noted that changing ε_1 to half the value of ε causes more NRPs to be found in the first step of the procedure. Altering the error values, ε_1 and ε_2, to different permutations brings about a change in the number of NRPs found but does not change the system performance dramatically. For example, by making ε_1 equal to two-third of ε and

ε_2 one-third of ε, the performance does not change by much. This is to be expected since there is a finite amount of redundancy found in any one direction of an image which limits the amount of compression possible. Appendix A outlines the proof that $\varepsilon=\varepsilon_1+\varepsilon_2$.

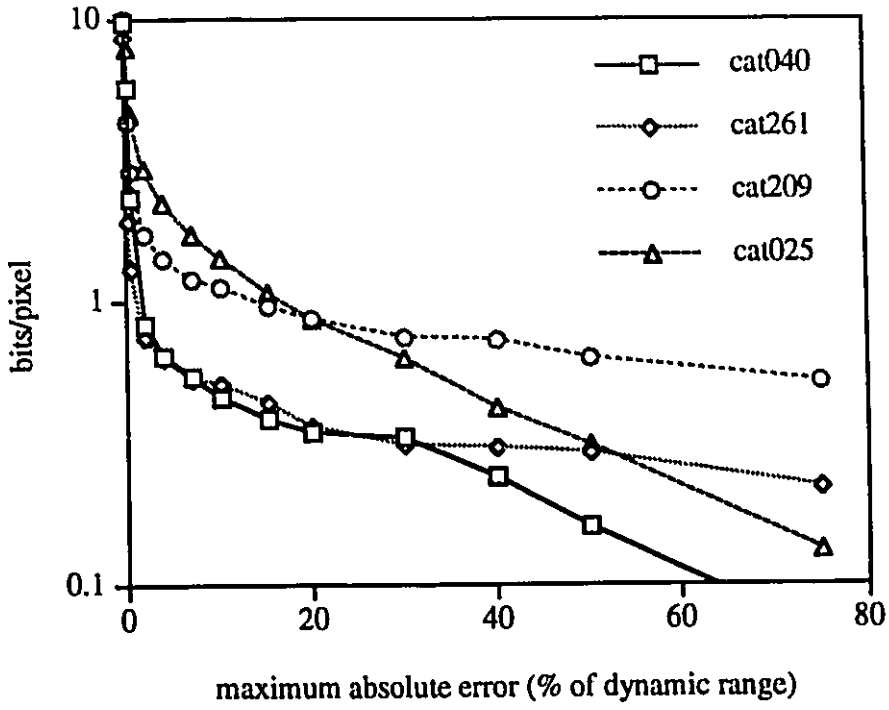


Figure 4.11 Rate distortion curves for two-directional compression systems

4.4 Peano scanning analysis

We saw that two-directional systems gave about a 10% performance improvement over unidirectional systems. It can, therefore, be assumed that some multi-directional systems would give even higher performance. Peano scanning, which is a fractal-based multi-directional system, is a scanning technique that is reputed to have a higher pixel-to-pixel

correlation than a raster scanning technique. The performance of the scheme is shown in Table 4.6. Fig. 4.12 depicts the corresponding rate distortion curves.

tolerance	cat040	cat261	cat209	cat025
0%	1.44:1	1.71:1	1.53:1	1.35:1
0.2%	2.19:1	6.90:1	2.94:1	1.98:1
0.5%	4.45:1	10.76:1	4.07:1	3.26:1
2%	16.49:1	19.06:1	8.89:1	5.70:1
4%	25.53:1	24.85:1	13.8:1	8.57:1
7%	32.83:1	32.20:1	18.53:1	12.03:1
10%	38.85:1	39.62:1	22.23:1	14.98:1
15%	45.21:1	49.37:1	27.00:1	20.43:1
20%	58.86:1	59.48:1	31.62:1	26.78:1
30%	65.84:1	82.28:1	40.02:1	45.18:1
40%	102.44:1	93.91:1	44.72:1	72.44:1
50%	140.03:1	98.29:1	53.46:1	103.43:1
75%	5461.33:1	136.57:1	71.58:1	196.69:1

Table 4.6 Compression ratios of the FAN algorithm on various Peano scanned range images

It is clear from the table and figure above that Peano scanning produces results which are generally better than unidirectional or two-directional systems. This is because Peano scanning exploits spatial redundancy in the image much better than other schemes do. Fig.4.14, which shows the Peano scanned reconstructed image of cat261 for a maximum absolute error of about 7%, verifies this. It should be noted that the Peano scanned reconstructed image does not produce streaks, as did the raster scanned reconstructed image, shown in fig. 4.7. The streaks were caused by the long raster lines in the raster scan. Peano curves, on the other hand, do not have such long lines, thus keeping this effect to a minimum. In fact, the Peano scanned reconstructed image is visually pleasing and better for post processing.

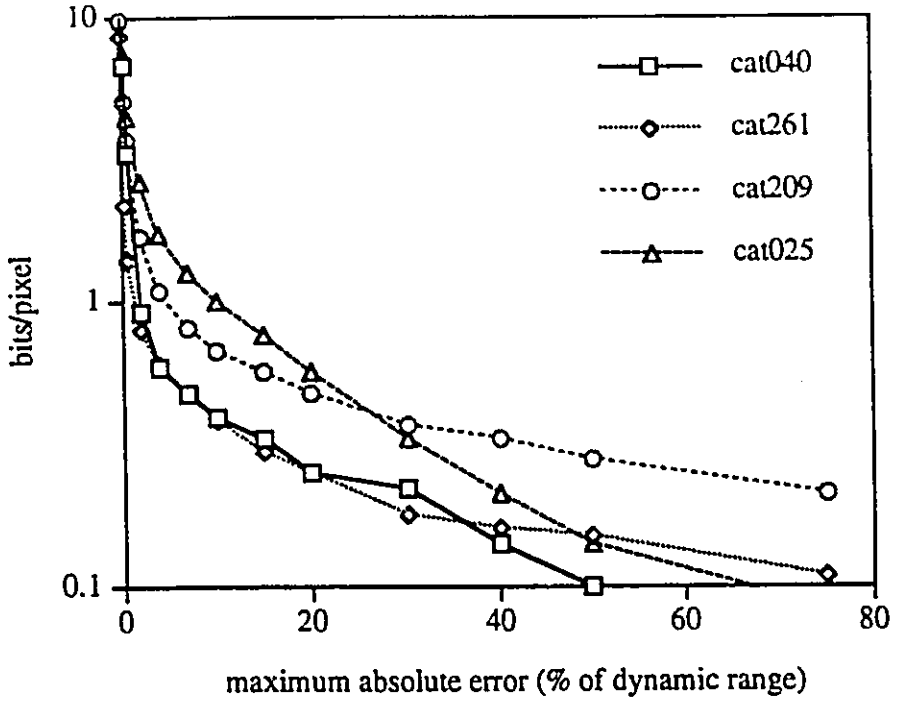


Figure 4.12 Rate distortion curves for various 15-bit Peano scanned range images

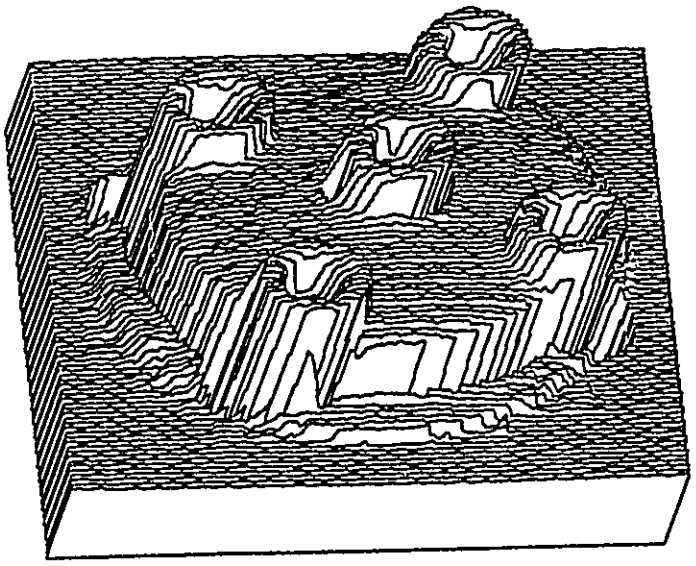


Figure 4.13 Surface plot of the reconstructed Peano scanned image cat261. ε=7%

The error image produced by Peano scanning for the same maximum absolute error is shown in fig. 4.14. Compared to fig. 4.8, which shows the error image for a raster scanned system, we observe that the Peano scanned error image displays much more "randomness", i.e. the error seems less structured than that found in the raster scanned system. If we look closely at fig. 4.14, we can see the Peano scanning itself! In fig. 4.8, we saw straight streaks formed due to the interpolation of NRPs. In fig. 4.14, we see the same thing but since the scan is multi-directional, we observe the fractal path instead.

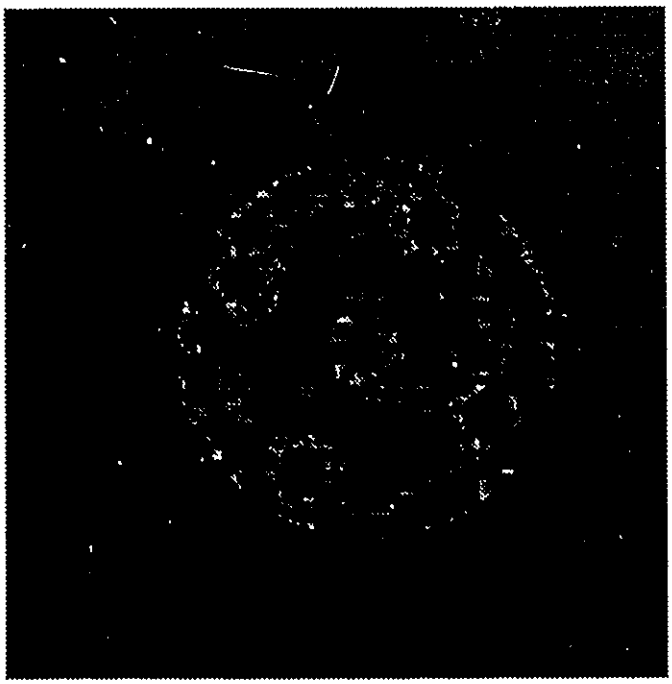


Figure 4.14 Error image of cat261 for Peano scanning. ε=7%

The images upon which performance analysis was made were all of size 256-by-256 pixels. Recall from the previous chapter that Peano scanning is a plane filling scanning technique that requires that the image be of size $4^j \times 4^j$, where j=1,2,3.. Hence the images, which were of order-4, performed well. The question that remains to be answered is how images of other sizes would be accommodated? This is an important question that needs to be addressed since one of the goals of the FAN algorithm is that it should be able to

work on all images, regardless of their size. A possible technique that could be used to solve this problem was presented in chapter three. Results of the technique are presented in fig. 4.15-16. Fig. 4.16 is the reconstructed image of fig. 4.15.

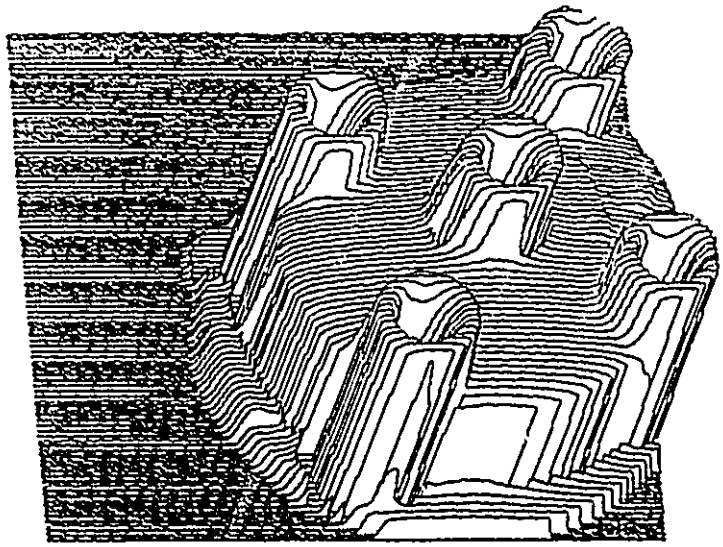


Figure 4.15 Surface plot of Peano scanned image of size 180X181 pixels

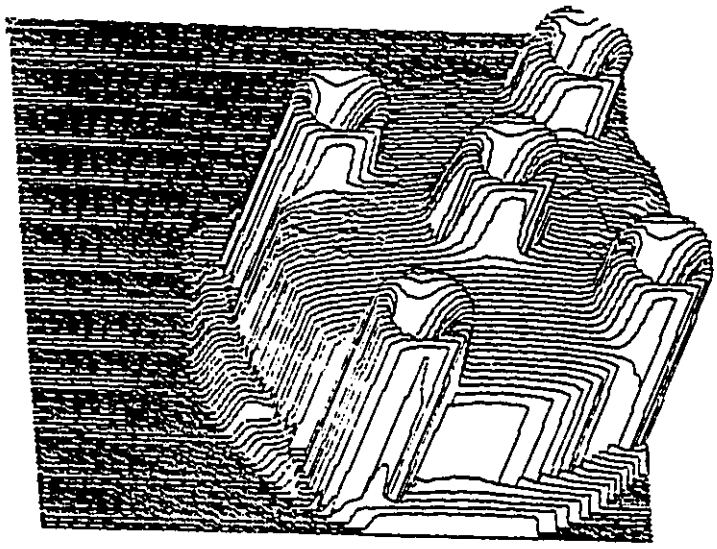


Figure 4.16 Surface plot of the reconstructed Peano scanned image of fig. 4.15. ε=7%

It can be seen from the figure that the technique employed to get around the problem of a fixed size Peano scanned image works very well. The technique results in compression ratios which are similar to those produced using order-j images.

4.5 Comparison with JPEG

It was pointed out at the outset that JPEG was not a suitable compression method for range images. This is because JPEG assumes a performance criterion which is inappropriate for range images. As was mentioned before, most range image applications require a maximum absolute error criterion, instead of a visual one which is inherent to JPEG [7]. This criterion requirement implies that range image compression algorithms be assessed according to maximum pixel intensity error. Thus, any useful comparison between JPEG and our algorithms must be based on the maximum absolute error criterion.

Figs. 4.17-20 show rate distortion curves on the four test range images comparing JPEG with Peano and raster scanned FAN compression systems. The range images have been scaled down to 8-bits for the comparison since existing JPEG packages work only on 8-bit resolution images. As can be seen from the figures, the curves are very similar to each other and are generally similar looking for all range images.

The JPEG used was that found in the *xv* Interactive Image Display software package for Unix (Version 3.00 by John Bradley, 1993) [74]. From the figures above it can be seen that there is an undefined region for JPEG. This is due to the limitations of xv as it uses a limited quantization table for JPEG. However, the JPEG plots can be extrapolated to give a compression performance for low values of maximum absolute error. Lossless JPEG, on the other hand, gives a theoretical compression ratio of about 2:1 [7], which is about the same as that given by the FAN algorithms.

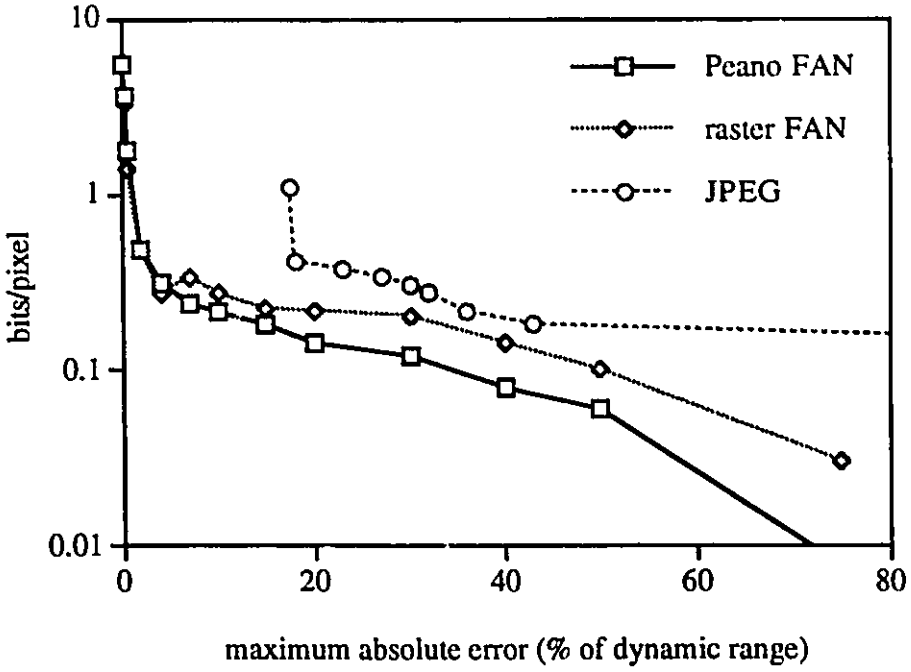


Figure 4.17 Comparison of JPEG with the FAN algorithm for image cat040

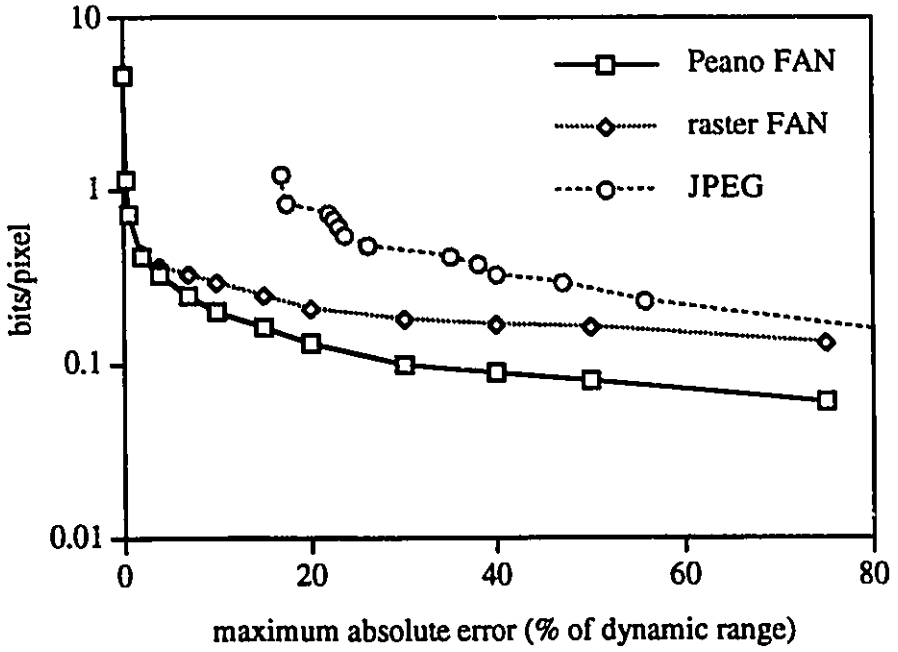


Figure 4.18 Comparison of JPEG with the FAN algorithm for image cat261

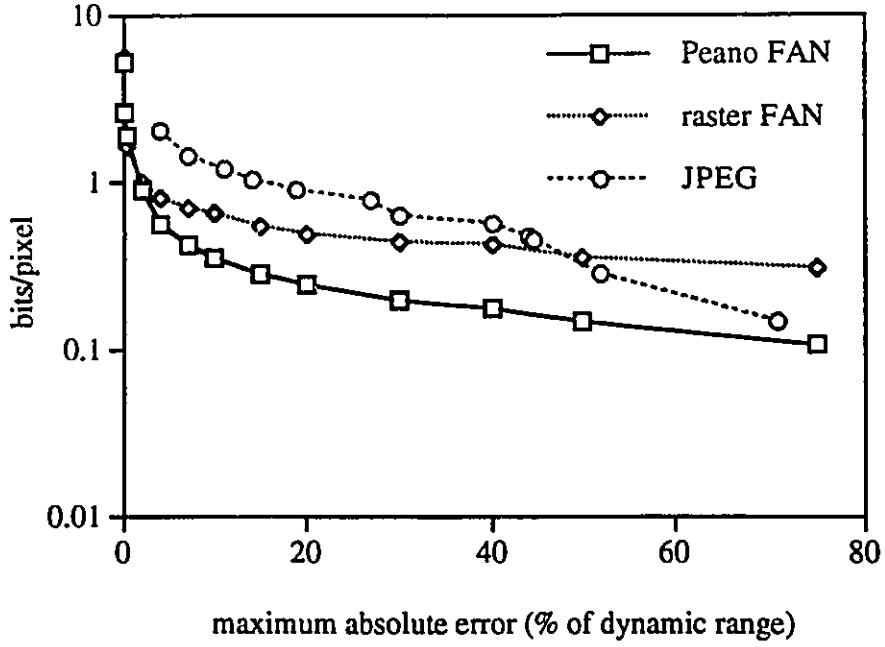


Figure 4.19 Comparison of JPEG with the FAN algorithm for image cat209

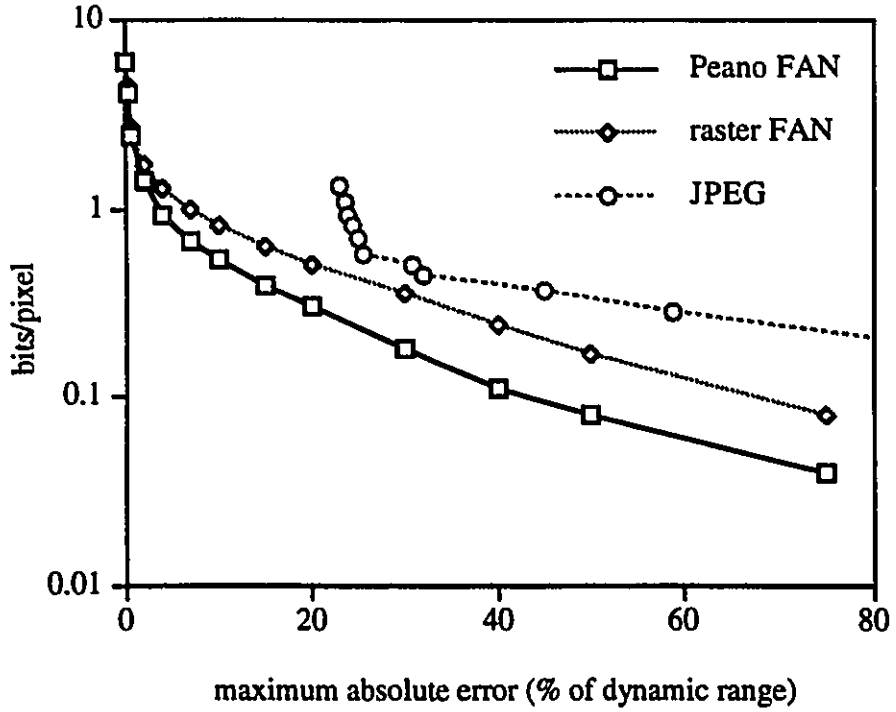


Figure 4.20 Comparison of JPEG with the FAN algorithm for image cat025

Fig. 4.21 shows the error image for JPEG for 18% error. Fig. 4.22 is the corresponding error image for the Peano scan algorithm. We can see that almost all of the error introduced by JPEG lies on the part of the image that has the most information content, whereas the error introduced by the Peano scanned FAN algorithm is more evenly spread out and less structured. Close examination of the error image also shows the Peano scanning pattern, especially in the lower right hand corner.

It is clear from these two figures that JPEG is an inappropriate technique for range image compression. While JPEG does try to minimize the error in most of the image, it fails to do so at all places. This implies that JPEG will give results which are visually pleasing but they will be unacceptable for a criterion such as maximum absolute error. For such a criterion, our algorithm clearly outperforms JPEG.

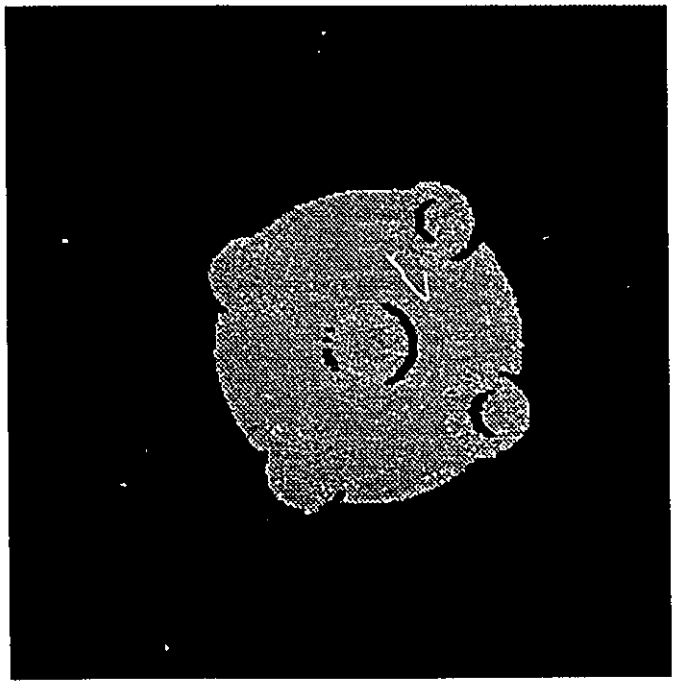


Figure 4.21 Error image of cat261 using JPEG. ε=18%

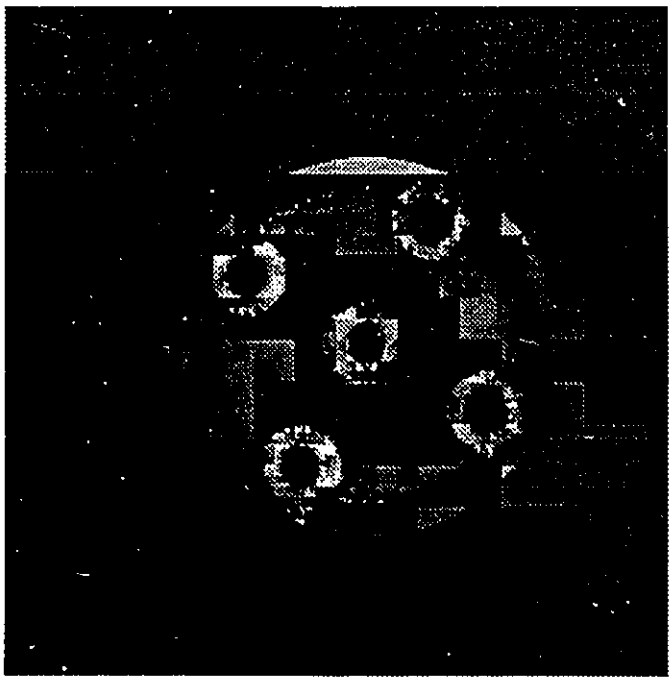


Figure 4.22 Error image of cat261 using Peano scan. ε=18%

4.6 Summary

In this chapter, the performance analysis of the FAN range image compression algorithm was presented. We started off by describing the range images on which performance analysis was made. Then, we compared Huffman and arithmetic coding schemes to give justification for using the former over the latter. We then outlined the performance of various FAN compression systems, such as unidirectional, two-directional and multi-directional systems. In unidirectional systems, we described the performance of the algorithm on scanning schemes such as raster, vertical, and horizontal-chain techniques. In the two-directional system, we described the algorithms performance on images after they have passed through the two-directional redundancy finder, which exploits redundancy in two orthogonal directions. In multi-directional systems, we evaluated the performance of

the algorithm using the Peano scanning scheme. Finally, we presented the results of the comparison of the raster and Peano FAN schemes with JPEG, the still image compression standard.

Chapter 5

Adaptive FAN algorithm

In the last chapter, we have described the performance of the FAN algorithm for a fixed value of ε, the user-specified tolerance level. We now consider the performance of our algorithm when ε varies within the image.

Instead of specifying a fixed tolerance level, the user may be more interested in specifying a *tolerance image*, i.e. an image of size equal to that of the test image where each pixel represents a maximum absolute error value instead of a distance measurement. There is, thus, a one-to-one correspondence between each test image pixel and each tolerance image pixel. This will allow the user to use varying tolerance levels corresponding to different segments in the test image. This is especially useful if some a priori information concerning the segments of the test image are available. For example, if certain segments of the test image are known to be noise-only and others to be signal-plus-noise, then using a different value of ε in those segments may result in better performance.

Fig. 5.1 depicts a tolerance image where half of the pixels are specified to be 0.15% of the dynamic range and the other half to be 0.3%. This tolerance image when used in conjunction with the test image cat040 in the adaptive mode of our algorithm results in the error image whose surface plot is shown in fig. 5.2.

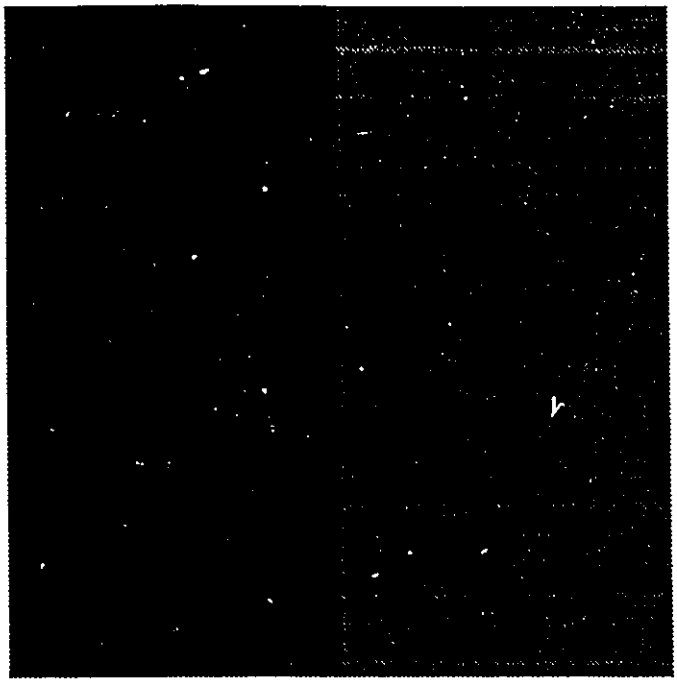


Figure 5.1 Tolerance image. The left half of the image is set to ε=0.15% and the right half set to ε=0.3%.

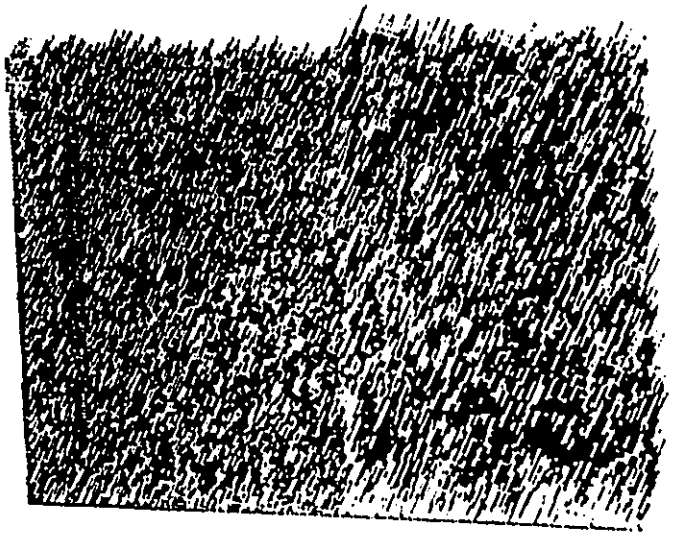


Figure 5.2 Surface plot of the error image corresponding to the tolerance image as shown in fig. 5.1 and test image cat040.

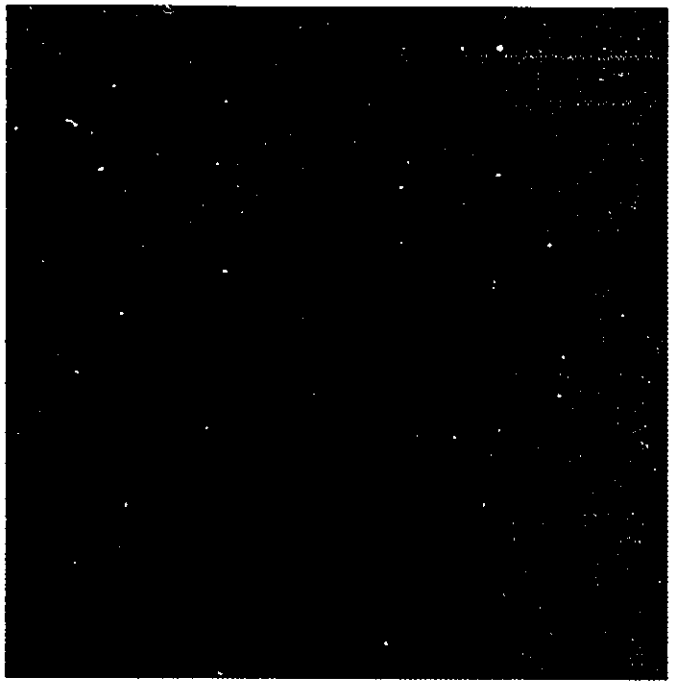


Figure 5.3 Tolerance image. From left to right, ε=0.06%, ε=0.12%,ε=0.18% and ε=0.25%

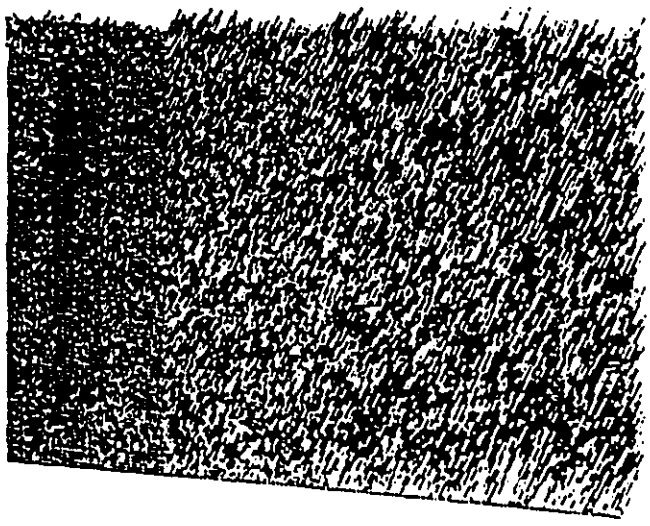


Figure 5.4 Surface plot of the error image corresponding to the tolerance image as shown in fig. 5.3 and test image cat040.

Similarly, fig. 5.3 shows another tolerance image in which the tolerance level increase from 0.06% to 0.25% in steps of 0.06%. Fig. 5.4 depicts the surface plot of its associated error image. We observe from the two error images that there is a sharp difference in error values in each segment of the image. However, the maximum absolute error specified in each segment of the image is always less than or equal to that specified in the tolerance image. Furthermore, the compression ratio achieved is always between that obtained when the lower and higher end of the specified tolerance level are used separately. This is to be expected since each segment of the test image produces a different number of NRPs and runlengths.

The tolerance images used test both the low-to-high and high-to-low transitions of the ε. This is due to the way the images are read into the program, i.e. they are read in as a sequence of pixels from the upper left hand corner to the lower right hand corner. It should be noted that neither transition causes any problems to the algorithms performance. Hence, any tolerance image can be used effectively.

Chapter 6

Conclusion and Future Work

6.1 Conclusions

In this thesis, we have proposed a new method to compress range images. The goal of this work was to develop a compression algorithm especially designed for range images. This is because other available techniques, such as JPEG, are not appropriate for range images due to their very nature. Keeping this in mind, we have developed a technique that is simple, fast, reliable, robust and cost effective.

The method is not sensitive to parameters such as image size, complexity, type, dynamic range or noise level or type. The only major parameter affecting the technique is ε, the user-specified tolerance level. The algorithm works such that the maximum absolute error is always less than or equal to ε. This tolerance criterion is appropriate for most range image applications, especially CAD/CAM.

Three different compression algorithms corresponding to different scanning techniques have been presented. Unidirectional scanning methods, such as those using raster scans, effectively remove redundancy in one direction. They are simple and efficient. The two-directional scanning method, on the other hand, exploits redundancy in two orthogonal directions. It improves performance over the unidirectional schemes but at the cost of added complexity. Of the three algorithms proposed, it is the by far the most complex.

The multi-directional scanning technique uses Peano curves to exploit redundancy in many directions. It is slightly more complex than the unidirectional methods but has a substantial performance improvement over them in terms of compression ratios. The Peano scheme has the best performance-to-complexity ratio of the three schemes and, therefore, we recommend its use over them.

Huffman coding with a fixed Huffman tree is used in the algorithms codec. However, any other coding scheme can be used in conjunction with the FAN redundancy finder. If Huffman coding is used, we would recommend users to construct their own Huffman table according to application.

The algorithms performance has been evaluated on four typical simple-to-complex range images. Furthermore, an in-depth comparison has been done between the algorithm and JPEG. For a mechanical criterion, namely maximum absolute error, which is a requirement for most range image applications, the proposed method significantly outperforms JPEG.

We have also presented an adaptive version of the algorithm. It gives the user greater flexibility to vary the tolerance level within the image, instead of keeping it fixed. The algorithm respects the varying tolerance value by always keeping the maximum absolute error within that specified by the user's tolerance image. The compression ratios resulting from such a scenario always fall between those found when the minimum and maximum tolerance levels are used separately. The adaptive FAN algorithm gives the user greater control over range image compression.

6.2 Future Work

The FAN range image compression algorithm has been designed to work in 1-D, i.e. it uses straight line approximations. However, by using a 2-D approximation structure, such as a square or a triangle, we may be able to improve system performance significantly.

This is because such a technique would exploit redundancy in both orthogonal directions of the image. Other techniques that could be tried are quadtree representation, Delaunay triangulation representation and morphological skeleton representation.

The extension of the algorithm to color range images is a possible research direction where higher compression ratios can be achieved. The color planes can be compressed either using existing compression techniques or by using a version of the FAN redundancy finder. Performance of the resulting algorithm can be studied in the presence of noisy images.

The long-term goal of this work is to integrate the new compression method into an emerging range image standard. The method possesses all the ingredients to succeed as a standard: simplicity, appropriateness, reasonable efficiency, and a seamless transition from the lossy to the non-lossy technique.

Finally, some architectural work in terms of hardware can be done for possible VLSI implementation of the algorithm.

REFERENCES

[1] M. Rabbani, "Selected Papers on Image Coding and Compression," *SPIE Milestone Series*, vol. MS-48, Bellingham, Washington, 1992.

[2] M. Rabbani, "Digital Image Compression Techniques," *SPIE Optical Engineering Press*, Bellingham, Washington, 1991.

[3] R. H. Gandhi, "3-Dimensional Pyramids for Video Compression", *MASc Thesis*, Ottawa-Carleton Institute for Electrical Engineering, Ottawa, Canada, 1993.

[4] W. K. Pratt, "Digital Image Processing", *Wiley-Interscience*, New York, 1978.

[5] G. K. Wallace, "Overview of the JPEG (ISO/CCITT) Still Image Compression Standard. Image Processing Algorithms and Techniques," *Proceedings of the SPIE*, vol. 1244, pp. 220-233, February 1990.

[6] A. K. Jain, "Image data compression: a review," *Proceedings of the IEEE*, vol. 69, no. 3, pp. 349-389, March, 1981.

[7] M. Kunt, A. Ikonomopoulos, M. Kocher, "Second-generation Image Coding Techniques," *Proceedings of the IEEE*, vol. 74, no. 4, pp. 549-574, 1985.

[8] P.J. Besl, "Active, Optical Range Imaging Sensors," *Machine Vision and Applications*, vol. 1, pp. 127-152, 1988.

[9] M. Rioux, L. Cournoyer, "The NRCC Three dimensional Image Data Files," *NRC Report 29077*, June 1988.

[10] M. Rioux, F. Blais, J. -A. Beraldin, P. Boulanger, "Range Imaging Sensors Developed at NRC Laboratories," *IEEE Proceedings of the Workshop on Interpretation of 3-D Scenes*, Austin, Texas, IEEE Computer Society, November 27-29, 1989, pp. 154-160.

[11] R. A. Jarvis, "A perspective on range finding for computer vision," *IEEE Transactions on Pattern Analysis and Machine Intellegence*, vol. PAMI-5, no. 2, pp. 122-139, 1983.

[12] W. F. Schreiber, "The measurement of third order probability distributions of television signals," *IRE Transactions on Information Theory*, vol. IT-2, pp. 94-105, September, 1956.

[13] W. K. Pratt, Ed., "Image Transmission Techniques," *Academic Press*, New York, 1979.

[14] C. E. Shannon, "The Mathematical Theory of Communication," Parts I and II, *Bell Systems Technical Journal*, vol. 27, pp. 379-423 and 623-656, 1948.

[15] R. G. Gallagher, "Information Theory and Reliable Communications," *John Wiley*, New York, 1968.

[16] T. Berger, "Rate Distortion Theory," *Prentice-Hall*, Engelwood Cliffs, New Jersey, 1971.

[17] L. D. Davisson, "Rate Distortion Thoery and Applications," *Proceedings of the IEEE*, vol. 60, pp. 156-164, July 1972.

[18] A. D. Wyner, "Fundamental Limits of Information Theory," *Proceedings of the IEEE*, pp. 239-251, February 1981.

[19] D. H. Kelly, "Effects of sharp edges on the visibility of sinusoidal gratings," *Journal of the Optical Society of America*, vol. 60, pp. 98-103, January 1980.

[20] A. Netravali and B. Prasada, "Adaptive Quantization of Picture Signals using Spatial Masking," *Proceedings of the IEEE*, vol. 65, pp. 536-548, April 1977.

[21] Z. L. Budrikis, "Visual Fidelity Criterion and Modelling," *Proceedings of the IEEE*, vol. 60, pp. 771-779, July 1972.

[22] J. L. Mannos and D. J. Sakrison, "The effects of a visual fidelity criterion on the encoding of images," *IEEE Transactions on Information Technology*, vol. IT-20, pp. 525-536, July 1974.

[23] V. R. Algazi, "The Psyco-Physics of Vision and their Relation to Picture Quality and Image Coding Limitations," *Acta Electronics*, vol. 19, no. 3, pp. 225-232, 1976.

[24] J. O. Lamb, "Distortion Criteria of the Human Viewer", *IEEE Transactions on Systems, Man and Cybernetics*, pp. 778-793, December 1979.

[25] J. A. Saghri, P. S. Cheatham, and A. Habibi, "Image Quality Measure based on a Human Visual System Model," *Journal of Optical Engineering*, pp. 813-818, July 1989.

[26] J. He, E. L. Deriniak, "Error-free Image Compression Algorithm using Classifying sequencing Techniques," *Applied Optics*, Vol. 31, no. 14, pp. 2554-2559, May 1992.

[27] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proceedings of the IRE*, pp. 1098-1101, September 1952.

[28] R. G. Gallagher, "Variations on a theme of Huffman," *IEEE Transaction on Information Theory*, vol. IT-24, no. 6, pp. 668-674, November 1978.

[29] J. G. Proakis, "Digital Communication," second edition, *McGraw-Hill*, 1989.

[30] A. K. Jain, "Fundamentals of Digital Image Processing," *Prentice-Hall*, Engelwood Cliffs, New Jersey, 1989.

[31] D. E. Knuth, "Dynamic Huffman Coding," *Journal of Algorithms*, vol. 6, pp. 163-180, 1985.

[32] J. S. Vitter, "Design and Analysis of Dynamic Huffman Coding," *Proceedings of the 26th Annual IEEE Symposium on Foundation of Computer Science*, Portand, Oregon, IEEE Computer Society, pp. 293-302, 1985.

[33] G. G. Langdon, "An Introduction to Arithmetic Coding," *IBM Jounal of Res. Devices*, vol. 28, pp. 135-149, March 1984.

[34] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, vol. 30, pp. 520-540, June 1987.

[35] J. Rissanen and G. G. Langdon, "Arithmetic Coding," *IBM Journal of Res. Devices*, vol. 23, no. 2, pp. 149-162, 1979.

[36] G. K. Wallace, "The JPEG Still-Picture Compression Standard", *Comm. of the ACM*, vol. 34, no. 4, pp. 30-44, April 1991.

[37] T. S. Huang, "Coding of two-tone Images," IEEE Transactions on Communications, vol. COM-25, no. 11, pp. 1405-1424, 1977.

[38] R. Hunter and A. H. Robinson, "International Digital Facsimile Coding Standards," *Proceedings of the IEEE*, vol. 68, no. 7, pp. 854-867, July 1980.

[39] P. A. Wintz, "Transform Picture Coding," *Proceedings of the IEEE*, vol. 60, no.7, pp. 809-820, July 1972.

[40] S. W. Golomb, "Run Length Encodings," *IEEE Transactions on Information Theory*, vol. IT-12, pp. 399-401, July 1986.

[41] T. A. Welch, "A Technique for High Performance Data Compression," *IEEE Computer Magazine*, pp. 8-19, June 1984.

[42] A. Lempel and J. Ziv, "On the complexity of finite sequences", *IEEE Trans. on Info. Theory*, vol. IT-22, no. 1, pp. 75-81, January 1976.

[43] J. Ziv and A. Lempel, "Compression of individual sequences via variable rate coding", *IEEE Trans. on Info Theory*, vol. IT-24, no. 5, pp. 530-536, September 1978.

[44] J. Makhoul, "Linear Prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, pp. 637-655, 1971.

[45] J. D. Markel and A. H. Gray Jr., "Linear Prediction of Speech," *Springer-Verlag*, New York, 1976.

[46] A. N. Netravali and J. O. Limb, "Picture coding: A review", *Proceedings of the IEEE*, pp. 366-406, March 1980.

[47] N. Jayant and P. Noll, "Digital Coding of Waveforms: Principles and applications to Speech and Video," *Prentice-Hall*, Engelwood Cliffs, New Jersey, 1984.

[48] A. Habibi, "Survey of Adaptive Image Coding Techniques," *IEEE Transactions on Communications*, vol. COM-25, no. 11, pp. 1275-1284, November 1977.

[49] A. Habibi and P. A. Wintz, "Image Coding by Linear Transformation and Block Quantization," *IEEE Transactions on Communication Technology*, vol. COM-19, pp. 50-63, February 1971.

[50] A. K. Jain, "A fast Karhunen-Loeve Transform for Finite Discrete Images," *Proceedings of the National Electronics Conference*, Chicago, Illinois, pp. 322-328, October 1974.

[51] H. C. Andrews and W. K. Pratt, "Transform Image Coding," *Proceedings of the Computer Processing Communications*, Polytechnic Press, New York, pp. 63-84, 1969.

[52] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," *IEEE Transactions on Computers*, vol. C-23, pp. 90-93, January 1974.

[53] M. Antonini, P. Mathiew, and I. Daubechies, "Image Coding using Wavelet Transforms," *IEEE Transactions on Image Processing*, vol. 1, no. 2, April 1992, pp. 205-20.

[54] W. Chen, H. Smith, "Adaptive Coding of Monochrome and Color Images," *IEEE Transactions on Communications*, vol. COM-25, no. 11, pp. 1285-1292, 1977.

[55] R. M. Gray, "Vector Quantization," *IEEE Acoustics, Speech and Signal Processing Magazine*, pp. 4-29, April 1984.

[56] N. M. Nasrabadi, R. A. King, "Image Coding using Vector Quantization: A Review," *IEEE Transactions on Communications*, vol. 36, no. 8, pp. 957-971, August 1988.

[57] Y. Linde, A. Buzo, R. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, pp. 84-95, January 1980.

[58] B. Ramamurthi, A. Gersho, "Classified Vector Quantization of Images," *IEEE Transactions on Communications*, vol. COM-34, no. 11, pp. 1105-1115, 1986.

[59] J. Kim, S. Lee, "A Transform Domain Classified Vector Quantizer for Image Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, no. 1, pp. 3-14, 1992.

[60] R. Aravind, A. Gersho, "Low-rate Image Coding with Finite-State Vector Quantization," *IEEE Conference on Acoustics, Speech and Signal Processing*, pp. 4.3.1-4.3.4, 1986.

[61] M. Goldberg, P. Boucher, S. Shlien, "Image Compression using Adaptive Vector Quantization," *IEEE Transactions on Communications*, vol. COM-34, pp. 180-187, February 1986.

[62] J. W. Woods, S. D. O'Neil, "Subband Coding of Images," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, no. 5, pp. 1278-1288, October 1986.

[63] H. Gharavi, and A. Tabatabai, "Subband Coding of Monochrome and Color Images," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 2, pp. 207-214, February 1988.

[64] A. E. Jacquin, "Image Coding based on a Fractal Theory of Iterated Contractive Image Transforms," *IEEE Transactions on Image Processing*, vol. 1., no. 1, pp. 18-30, January 1992.

[65] M. F. Barnsley, "Fractals Everywhere," *Academic Press*, New York, 1988.

[66] B. Mandelbrot, "The Fractal Geometry of Nature," *Freeman*, San Fransisco, CA., 1982.

[67] A. N. Netravali and B. G. Haskell, "Digital Pictures: Representation and Compression," *Plenum*, New York, 1989.

[68] B. Mandelbrot, "Fractals, form, chance, and dimensions",*W. H. Freeman and Co.*, San Franciso, 1977.

[69] V. V. Alexandrov, A. D. Polyakov, V. M. Latchinov, "Synthesis and applications of Peano curve multidimensional analog", *AFCET-IRIA Conf. on Pattern Recog. and Image Proc.*, Chatenay Malabry, France, Feb. 1978.

[70] J. C. Simon, J. Quinqueton, "On the use of a peano scanning in image processing", *Proceedings of the NATO Advanced Study Inst. on Dig. Image Proc. and Anal.*, Bonas, France, June 1978.

[71] L.W. Gardenhire, "Data Redundancy Reduction, the Key to AdaptiveTelemetry", *Proceedings of the National Telemetering Conference*, Los Angeles, Section 1-5, pp. 1-16, 1964.

[72] C.G.Boncelet Jr., J.R.Cobbs, A.R.Moser, "Error Free Compression of Medical X-Ray Images",*Visual Comm.and Image Processing*, Proceedings of the SPIE, vol. 1001, pp. 269-276, 1988.

[73] P.J.Besl, R,C.Jain, "Segmentation Through Variable-Order Surface Fitting", *IEEE Transactions on Pattern Analysis and Machine Intellegence*, vol. 10, no.2, pp.167-192, Mar. 1988.

[74] J. Bradley, "XV Interactive Image Display For The X Window System," *Shareware software*, Version 3.00, 1993.

Appendix A

Proof of worst case ε

Fig. A.1 shows a diagram of the possible range of values of the interpolated points $f'(x_1)$, $f'(x_2)$ and $f'(x_3)$ of an image scanned using the two-directional scanning system. Points $f(x_1)$, $f(x_2)$ and $f(x_3)$ are the actual image pixel values and ε_1 and ε_2 are the user-specified maximum tolerance level in the two directions.

If the interpolated points $f'(x_1)$ and $f'(x_3)$ are error-free, i.e. ε_1 =0, then the interpolated point in the other direction, $f'(x_2)$, will lie at point B in fig. A.1. Thus, the resulting maximum absolute error, ε, would be ε_2.

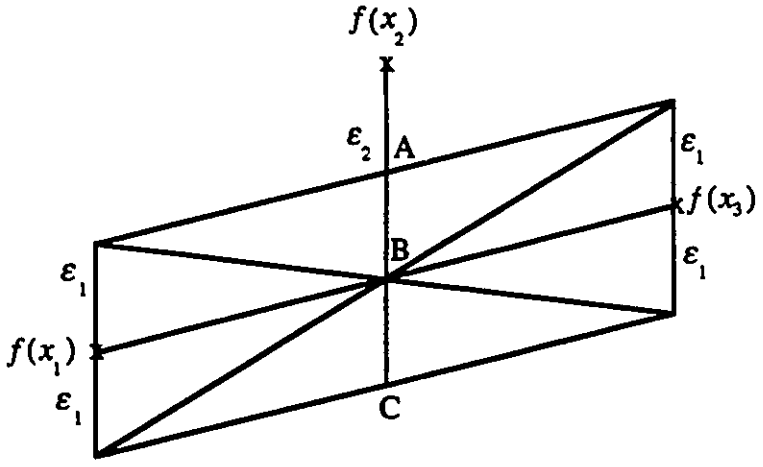


Figure A.1. Error variances of various points in a rasterline

However, if the interpolated points $f'(x_1)$ and $f'(x_3)$ are not error free, i.e. ε_1 is non-zero, then various scenarios can occur. Though there are many possible cases, only the extreme cases are outlined below.

Case 1:

If the interpolated points

$$\begin{aligned} f'(x_1) &= f(x_1) + \varepsilon_1 \\ f'(x_3) &= f(x_3) + \varepsilon_1 \end{aligned} \tag{A.1}$$

then the interpolated point $f'(x_3)$ will be found at point A in the figure. This implies that the maximum absolute error, ε, is

$$\varepsilon = \varepsilon_2 - \varepsilon_1 \tag{A.2}$$

Case 2:

If the interpolated points

$$\begin{aligned} f'(x_1) &= f(x_1) + \varepsilon_1 \\ f'(x_3) &= f(x_3) - \varepsilon_1 \end{aligned} \tag{A.3}$$

or

$$\begin{aligned} f'(x_1) &= f(x_1) - \varepsilon_1 \\ f'(x_3) &= f(x_3) + \varepsilon_1 \end{aligned} \tag{A.4}$$

then, the interpolated point $f'(x_3)$ will be found at point B and the maximum absolute error will, therefore, be

$$\varepsilon = \varepsilon_2 \tag{A.5}$$

Case 3:

If the interpolated points

$$f'(x_1) = f(x_1) - \varepsilon_1$$
$$f'(x_3) = f(x_3) - \varepsilon_1 \tag{A.6}$$

then the interpolated point $f'(x_3)$ lies at point C. The maximum absolute error is, therefore,

$$\varepsilon = \varepsilon_2 + \varepsilon_1 \tag{A.7}$$

Since ε_1 and ε_2 are both non-negative numbers, the worst case scenario is that depicted in Case 3. Hence, the worst case ε is the sum of the errors in the two directions.