A PREDICTIVE CODING METHOD

FOR LOSSLESS COMPRESSION OF IMAGES

A

Dissertation

Submitted

in partial fulfillment

for the award of the Degree of

Master of Technology

in Department of Computer Science & Engineering

(With specialization in Software Engineering)

Supervisor Name:

Ms.Savita Shiwani

PC M.Tech. (CS/SE/IC)

Submitted By:

Anil Mishra

M.Tech(SE)

Enrolment No.:SGVU111515746

Department of Computer Science & Engineering

Suresh Gyan Vihar University

Mahal, Jagatpura, Jaipur

June 2013

Candidate's Declaration

I hereby that the work, which is being presented in the Dissertation, entitled "**A Predictive Coding Method for Lossless Compression of Images**" in partial fulfillment for the award of Degree of "Master of Technology" in Deptt. of Computer Science & Engineering with specialization in Software Engineering **and submitted to the Department** of Computer Science & **Engineering,** Suresh Gyan Vihar University is a record of my own investigations carried under the Guidance of Ms.Savita Shiwani, Department of Computer Science & Engineering.

I have not submitted the matter presented in this Dissertation any where for the award of any other Degree.

Anil Mishra

M.Tech(SE)

Enrolment No.: SGVU111515746

Counter Singed by

Ms.Savita Shiwani

DETAILS OF CANDIDATE, SUPERVISOR (S) AND EXAMINER

Name of Candidate: ……. Roll No.

……

Deptt. of Study:

…….

Enrolment No.

……...……

Thesis Title:

……...…

……...

Supervisor (s) and Examiners Recommended **(with Office Address including Contact Numbers, email ID)**	
Supervisor	**Co-Supervisor**
Internal Examiner	

1	2	3

Signature with Date

Programme Coordinator

Dean / Principal

Acknowledgements

This research project would not have been possible without the support of many people. First and foremost among all, I would like to express my sincere gratitude to my supervisor, Ms.Savita Shiwani who was abundantly helpful and offered invaluable assistance, support and guidance. I am also highly obliged to Mr.Pankaj Goswami for his invaluable support. He was also my supervisor in the absence of Ms.Savita Shiwani. All my friends need special mention and thanks for their involvement in all my discussion and problems, related to my work as well as personal.

I wish to express my love and gratitude to my beloved family for their understanding, inspiration, and endless love, throughout the duration of my studies.

Anil Mishra
M.Tech(SE)

Abstract

Compression is a process, in which the given size of data is compressed to a smaller size. Storing and sending images to its original form can present a problem in terms of storage space and transmission speed. Compression is efficient for storing and transmission purpose. Based on the reconstruction quality, compression can be lossy or lossless. Images used for biomedical research or astrophysical categorization must retain all the quality of the original image because machine analysis can be performed on these images to find very specific details that the human eye cannot detect. If these images are to be transmitted or stored, lossless image compression is needed. In the first part of this thesis, we propose a comparative study, based on literature survey, between the prediction based methods and the transform based methods. From information theoretic point of view, prediction can be explained as an estimation of some unknown quantity from known observation. In prediction, the value of a current pixel is being predicted from some of the previous pixels. These pixels are already stored in the memory when the image is scanned. Subtracting the predicted pixel value from the current pixel value at the same spatial location, we get the error image. This error image is entropy coded. But, in the transform based method, spatial image pixel values used to convert into transform coefficient values. It produces as many coefficients as there are pixels in the image. These coefficients can then be compressed more easily because the information is statistically concentrated in just a few coefficients. These coefficients are quantized and the quantized values are entropy coded.

But, in literature survey, we found that for lossless compression, prediction based methods are better than the transform based methods in terms of complexity.
The second part of the thesis described a new lossless adaptive prediction

based algorithm for continuous tone images. In continuous tone images, spatial redundancy exists. Our approach is to develop a new backward-adaptive prediction technique to reduce the spatial redundancy in a image. The new prediction technique known as Modified Gradient Adjusted Predictor (MGAP) is developed. MGAP is based on the prediction method used in Context based Adaptive Lossless Image Coding (CALIC). An adaptive selection method which selects the predictor in a slope bin in terms of minimum entropy improves the compression performance.

Contents

CONTENTS

CONTENTS

CONTENTS

List of Figures

LIST OF FIGURES

List of Tables

1

Introduction

1.1 Elementary Concepts

1.1.1 Digital Image

Digital image is basically a two-dimensional (2-D) data matrix which consists of a finite number of rows and columns of pixels (picture elements). Let R denote the height or number of rows and C denote the width or number of columns of an image. A 2-D digital image as shown in Fig. 1.1 can be represented by a two-dimensional matrix as shown in (1.1).

$$\mathbf{I=I(i,j)} \quad \begin{matrix} I(0,0) & I(0,1) & & I(0,C-1) \\ I(1,0) & I(1,1) & & I(1,C-1) \\ \vdots & \vdots & \vdots & \vdots \\ I(R-1,0) & I(R-1,0) & \ldots & I(R-1,C-1) \end{matrix} \tag{1.1}$$

Where the first co-ordinate $i(0 \leq i < R)$ denotes row index along the vertical direction and the second co-ordinate $j(0 \leq j < C)$ denotes the column index along the horizontal direction. I(i, j) denotes the value of the pixel located in row i and column j, or more precisely, at position (i, j). By convention I(0, 0) is taken to be the top left corner of the image and the I(R-1, C-1) is bottom right corner of the image.

Resolution of such an image is written as $R \times C$.

The number of bits used to represent a single pixel is called the bit depth or color depth. If an image is b bits per pixel (bpp), it is also called a b-bit image. A b-bit image can represent 2^b different gray levels or colors. Three typical types of images are:

1. INTRODUCTION

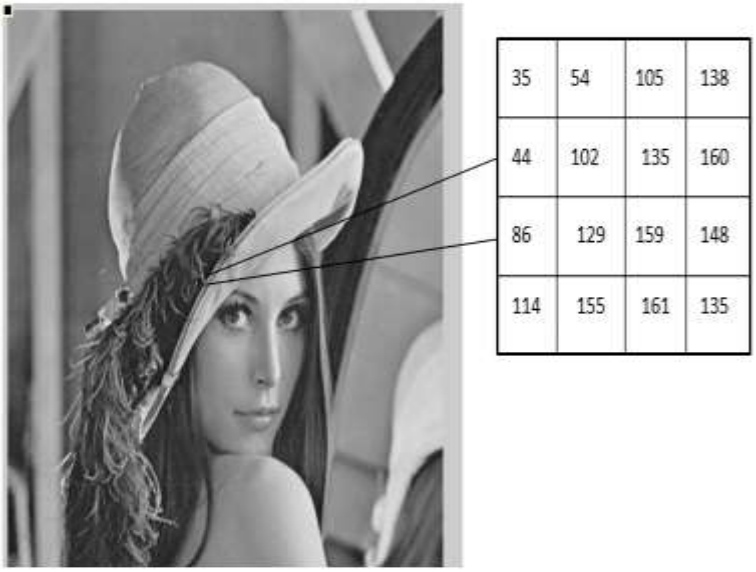

Figure 1.1: Digital Image (Lena Image)

1. 1-bit bi-level images: A bi-level image contains two different colors which are usually black and white. A bi-level image uses bit 0 to represent one of the two colors and uses bit 1 to represent the other color. Bi-level images are also referred to as binary images as shown in Fig. 1.2 or black and white images in the literature.

2. 8-bit gray scale images: An 8-bit gray-scale image contains up to 256 different shades of gray with values from 0 to 255. Sometimes one can interpret the pixel values of gray-scale images as indices to colors in a color palette. Pixel values of gray-scale images as shown in Fig. 1.1 are regarded as shades of gray from black to white.

3. 24-bit true color images: A 24-bit color image as shown in Fig. 1.3 in RGB color space consists of three gray-scale components of red, green, and blue; each of which is represented by 8 bits. An image in RGB space can be converted into other color spaces such as YCbCr and YUV. In YCbCr or YUV space, luminance (brightness) information is stored as a single component Y, and chrominance (color) information is stored as two different components: C (b) and C(r) for YCbCr space; U and V for YUV space.

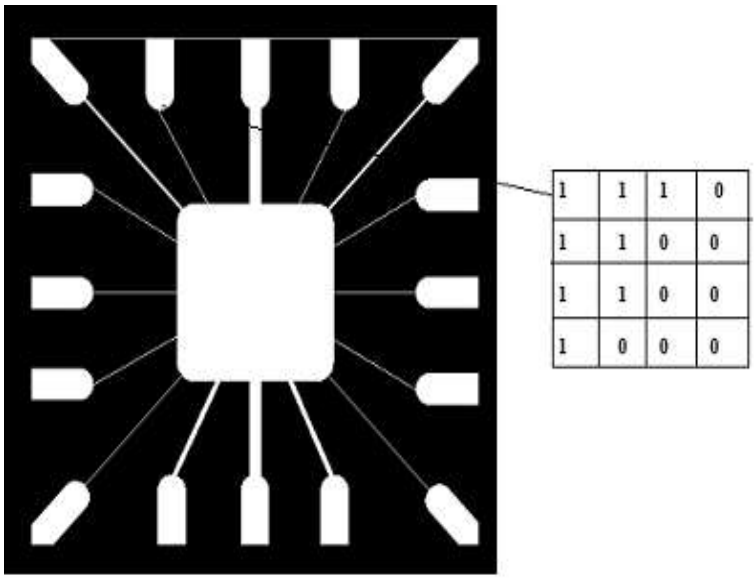

Figure 1.2: Binary Image

Figure 1.3: Color Image

1.1.2 Histogram of an Image

A histogram plots the relative frequency of each pixel value that occurs in a grayscale image. Fig. 1.4 shows the intensity histogram for the image from Fig. 1.1. The histogram provides a convenient summary of the intensities in an image, but is unable to convey any information regarding spatial relationships between pixels.

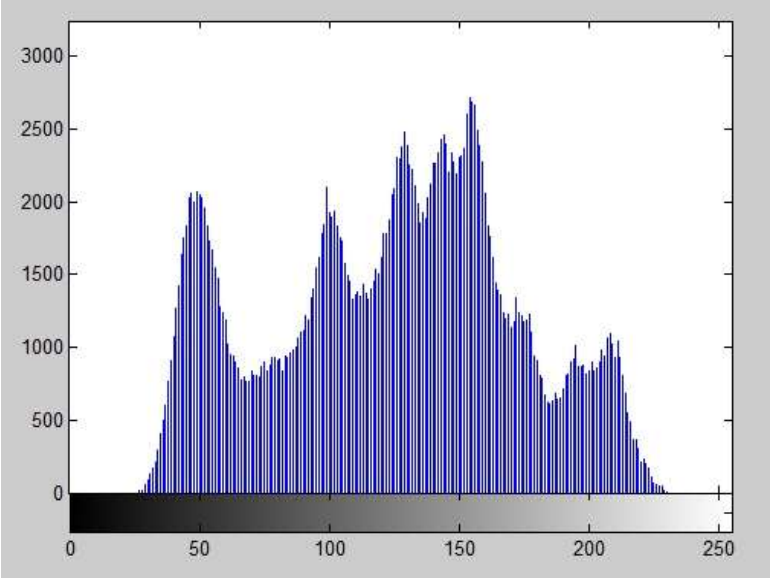

Figure 1.4: Histogram of the Lena Image

1.1.3 Entropy of an Image

If the probabilities of occurrence of data elements are known, then variable sized codes can be generated to minimize the number of bits for representing the data. There is, however, a limit to this minimization, which is known (in terms of information theory) as Entropy [1].

Given M random variables α_1, α_2, ..., α_m. If these variables have probabilities of Occurrence $p_1 = p(\alpha_1)$, $p_2 = p(\alpha_2)$,. , $pM = p(\alpha_M)$; then the entropy E is given by the following relation:

$$E = -\sum_{K=1}^{M} pk \log_2(pk) \qquad (1.2)$$

Entropy is a measure of amount of information in the given data according to the

Probability distribution of the alphabet. It defines the minimum number of bits required to encode the data [1].

Suppose there are M = 8 random variables r_1, r_2,...,r_8, having the same probability of occurrence; i.e., $p_1 = p_2 = ... = p_8 = 1/8$. Then using (1.3)

$$E = -\sum_{K=1}^{8} \frac{1}{8} \log_2 \frac{1}{8} \qquad (1.3)$$

On the other hand, if $p_1 = 1$, $p_2 = p_3 = ... = p_8 = 0$, then the entropy is

$$E = 0$$

The entropy of M random variable can range form 0 to $\log_2 M$.

Entropy is a measure of the degree of randomness of the set of random variables. The least random case is when one of the random variables has probability 1 so that the outcome is known in advance and H = 0. The most random case is when all events are equally likely. In this case $p_1 = p_2 = ... = p_M = 1/M$ and $H = \log_2 M$.

Entropy gives a lower bound on the average number of bits required to code each input symbol; in case of images, it gives the average number of bits required to code each input symbol; If the probabilities of occurrence of each input symbol is known to be p_1, p_2,..., p_M , then we are guaranteed that it is not possible to code them using less than

$$E = -\sum_{K=1}^{M} pk \log_2(pk) \qquad (1.4)$$

bits on the average.

1.1.4 Neighbors of a Pixel

A pixel P at co-ordinates (x, y) in an image has 8 neighbors surrounding it as shown in Fig. 1.5. Four of these neighboring pixels informally called TOP, BOTTOM, LEFT, and RIGHT are adjacent to it. These pixels have co-ordinates (x, y − 1), (x, y + 1), (x − 1, y), and (x + 1, y) respectively. These pixels are called the 4-neighbors of the pixel or N_4. These four pixels are at distance 1 from pixel P i.e., the distance from the centre of pixel P at (x, y) to the centre of either of these pixels is 1. The rest of the 4 neighbors of P have diagonal corners touching P. These pixels informally

Called TOP-LEFT, TOP-RIGHT, BOTTOM-LEFT, and BOTTOM-RIGHT have co-ordinates (x − 1, y − 1), (x + 1, y − 1), (x − 1, y + 1), and (x + 1, y + 1) respectively.

These pixels are called diagonal neighbors of P or N_D. The diagonal neighbors are at a distance of $\sqrt{2}$ from P . N_4 and N_D together are called N_8 neighbors of P [2].

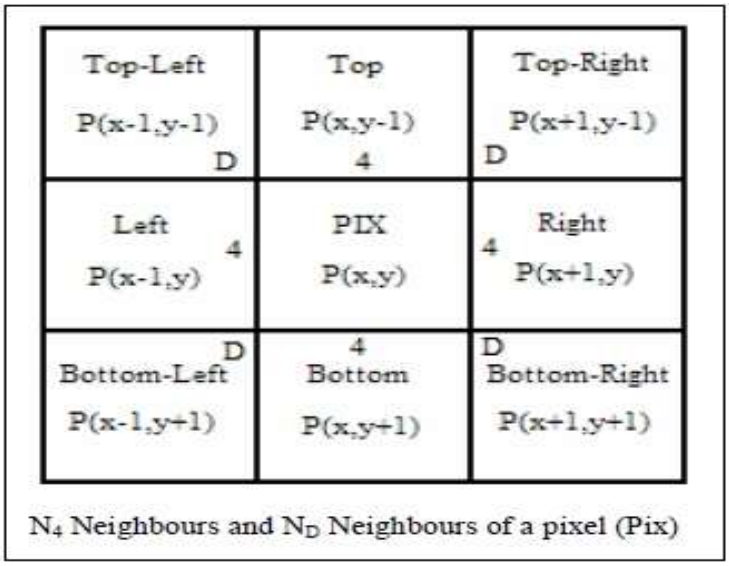

Figure 1.5: Neighbors of a Pixel

1.1.5 Raster Scan Order

An image consisting of R rows and C columns, if scanned one row at a time from top to bottom, and each row scanned from left to right is referred to as raster scan as depicted in Fig. 1.6. This is the order of scanning which is used in CRT (Cathode Ray Tube) monitors, where the electron gun focuses the beam at one spot at a time, starting from the top-left corner. The gun goes from left to right pixel by pixel and at the end of the first line moves to the leftmost pixel of the second line and again goes from left to right. Moving in this order, when all the rows are drawn the scan is complete. This order of scanning is also used by most of the image processing programs which filter the image pixel by pixel starting from top-left corner pixel and finishing at bottom-right corner pixel.

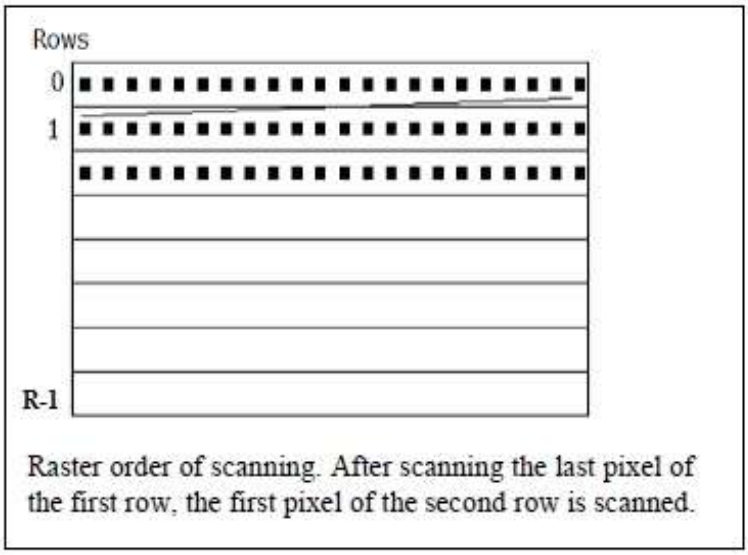

Figure 1.6: Raster Scan Order of the Image

1.2 Data Compression

Data is represented as a combination of information and redundancy. Information is the portion of data that must be preserved permanently in its original form in order to correctly interpret the meaning or purpose of the data. Redundancy is that portion of data which can be removed when it is not needed or can be reinserted in order to interpret the data when needed. Most often, the redundancy is reinserted in order to generate the original data in its original form. A technique to reduce the redundancy of data is defined as Data Compression. The redundancy in data representation is reduced in such a way that it can be subsequently reinserted to recover the original data, which is called as decompression of the data.

“Compression is the art or science of representing information in a compact form". The use of compression is because of the economic needs to save space in storage media and to save channel bandwidth in communication. For example, if there is a color image of size 512 × 512 and full color depth is 24 bits (8 bits red, 8 bits green, and 8 bits blue), then the uncompressed image needs a storage space of more than 786K bytes (512 × 512 × 24), which is a big data size for cost effective transmission and storage. Therefore, to reduce the image size, compression is necessary. Similarly, A

television-quality low-resolution color video of 30 frames per second with each frame containing 640 × 480 pixels (24 bits per color pixel) needs more than 210 megabits per second of storage. As a result, a digitized one-hour color movie would require approximately 95 gigabytes of storage. The storage requirement for upcoming high-definition television (HDTV) of resolution 1280 × 720 at 60 frames per second is far greater. A digitized one-hour color movie of HDTV-quality video will require approximately 560 gigabytes of storage. Transmission of these digital signals through limited bandwidth communication channels is even a greater challenge and sometimes impossible in its raw form. Although the cost of storage has decreased drastically over the past decade due to significant advancement in microelectronics and storage technology, the requirement of data storage and data processing application is growing explosively to outpace this achievement. Data redundancy is the central concept in image compression and can be mathematically defined as

$$\text{Image} = \text{Information} + \text{Redundant Data} \tag{1.5}$$

Given n_1 and n_2 denoting the information carrying units in two data sets that represents the same information/image. The relative data redundancy R_D of the first data set, n_1, is defined by:

$$R_D = 1 - \frac{1}{C_R} \tag{1.6}$$

Where C_R refers to the compression ratio and is defined by

$$C_R = \frac{n^1}{n_2} \tag{1.7}$$

If $n_1 = n_2$, then $C_R = 1$ and $R_D = 0$, indicating that the first representation of the information contains no redundant data. A typical compression ratio around 10 or (10:1) indicates that 90 ($R_D = 0.9$) of the data in the first data set is redundant. Image compression and coding techniques explore three types of redundancies:

1. Coding Redundancy
2. Interpixel Redundancy
3. Psycho visual Redundancy

Coding Redundancy: A code is a system of symbols (i.e., bytes, bits) that represent information. Each piece of information is represented by a set of code symbols. The gray level histogram of an image can be used in construction of codes to reduce the data used to represent it. Given the normalized histogram of a gray level image,

$$P_r(r_k) = \frac{n^k}{n} \tag{1.8}$$

where, k = 0, 1, 2.... , L-1. r_k is the pixel values defined in the interval [0,1] and $P_r(k)$ is the probability of occurrence of r_k. L is the number of gray levels. n_k is the number of times that k^{th} gray level appears, and n is the total number of pixels. Average number of bits required to represent each pixel is given by:

$$L_{avg} = \sum_{k=1}^{L-1} l(r_k) P_r(r_k)$$

where, $l(r_k)$ is the number of bits used to represent each value of r_k. For example, an 8-bit gray level image has the following gray level distribution as shown in Table 1.1

Table 1.1: Example of Coding Redundancy

r_k	$p_r(r_k)$	Code1	$l_1(r_k)$	Code2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

The average number of bits used for fixed 3-bit code:

$$L_{avg} = \sum_{k=0} l1(rk) P_r(r_k) = 3 \sum_{k=0} p_r(r_k) = 3.1 = 3 \text{ bits}$$

The average number of bits used for fixed variable-length code in this particular example:

$L_{avg} = \sum_{k=0}^{7} l2(rk)^{P}r(rk) = 2(0.19)+2(0.25)+3(0.16)+4(0.08)+5(0.06)+6(0.03)+6(0.02)$
= 2.7 bits

The compression ratio: $C_R = \frac{3}{2.7} = 1.1$

The relative data redundancy: $R_D = 1 - \frac{1}{1.1} = 0.09 = 10$

In this example, the suggested variable-length coding gets rid of the 10 percent redundant data of the fixed 3-bit code.

In this, data compression is achieved by assigning fewer bits to more probable gray levels than the less probable gray levels.

This type of coding is always reversible and usually implemented using look-up tables. Examples of image coding schemes that explore coding redundancy are the Huffman codes and the Arithmetic coding technique.

Interpixel Redundancy: This type of redundancy sometimes called as spatial redundancy, interframe redundancy, or geometric redundancy. It exploits the fact that an image very often contains strongly correlated pixels. In other words, large regions whose pixel values are the same or almost the same. This redundancy can be explored in several ways, one of which is by predicting a pixel value based on the values of its neighboring pixels. In order to do so, the original 2-D array of pixels is usually mapped into a different format, e.g., an array of differences between adjacent pixels. If the original image pixels can be reconstructed from the transformed data set, the mapping is said to be reversible.

Examples of compression techniques that explore the interpixel redundancy include: Constant Area Coding (CAC), (1-D or 2-D) Run-Length Encoding (RLE) techniques, and many predictive coding algorithms such as Differential Pulse Code Modulation (DPCM).

Psycho visual Redundancy: Many experiments on the psychophysical aspects of human vision have proven that the human eye does not respond with equal sensitivity to all incoming visual information; some pieces of information are more important than others. The knowledge of that particular type of information are more or less relevant to the final human user have led to image and video compression techniques that aim at eliminating or reducing any amount of data that is psycho visually redundant. The end result of applying these techniques is a compressed image file, whose size and quality are smaller than the original information, but whose resulting quality is still acceptable for the application at hand. The loss of quality that ensues as a byproduct

of such techniques is frequently called as quantization, to indicate that a wider range of input values is normally mapped into a narrower range of output values through an irreversible process. In order to establish the nature and extent of information loss, different fidelity criteria (some objective, such as root mean square (RMS) error; some subjective, such as pair wise comparison of two images encoded with different quality settings) can be used. Most of the image coding algorithms in use today exploit this type of redundancy, such as Discrete Cosine Transform (DCT) based algorithm at the heart of the JPEG encoding standard.

1.3 Basics of Image Coding

The block diagram of a lossless coding system is shown in Fig. 1.7. The encoder takes an image as input and generates a compressed bit stream as output. The decoder takes the compressed bit stream as input and recovers the original uncompressed image as output. In general, the encoder and decoder can each be viewed as consisting of three main stages. The decoder performs the inverse operation of the encoder. The operation of a lossless image encoder can be grouped into three stages:

1. Source Transformation
2. Quantization
3. Symbol Encoding

Source transformation applies a reversible (one-to-one) transformation to the input image data. The purpose of this stage is to convert the input image data X (n) into a form X (n) that can be compressed more efficiently. It transforms the input data into a format that facilitates reduction of interpixel redundancies. For this purpose, the selected transformation can aid in reducing the data correlation (interdependency, redundancy), alter the data statistical distribution, and/or pack a large amount of information into a few data samples or sub band regions. Typical transformations include differential/predictive mapping; transforms, such as the Discrete Cosine Transform (DCT); sub band decompositions, such as wavelet transforms; and color space conversions, such as conversion from the highly correlated RGB representation to the less correlated luminance-chrominance representation. Single or combination of the

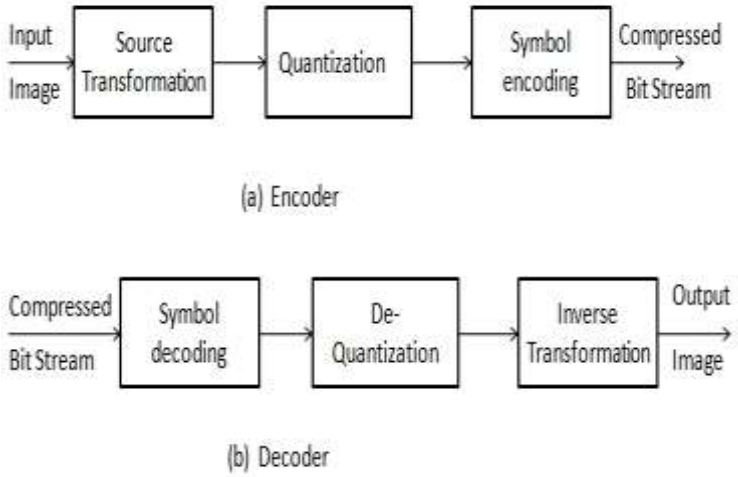

Figure 1.7: Data Compression Model

above transforms can be used at this stage. For example, an RGB color image can be transformed to its luminance-chrominance representation followed by DCT or sub band decomposition followed by predictive/differential mapping. In some applications (e.g., low power), it might be desirable to operate directly on the original data without incurring the additional cost of applying a transformation; in this case, the transformation could be set to the identity mapping.

1.3.1 Transform based Method

Transform coding is used to convert the spatial image pixel values to transform coefficient values. This produces as many coefficients as the pixels in the image. These coefficients can then be compressed more easily because the information is statistically concentrated in just a few coefficients. This principal is called as transform coding. After that, the coefficients are quantized and the quantized values are entropy encoded. In addition, it is more likely that the smaller coefficients can be coarsely quantized or deleted (lossy coding) without doing visible damage to the reproduced image. Many types of transforms have been tried for picture coding. For example, Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), and recently, Discrete Wavelet Transform (DWT).

In transform based method we start with the set of correlated data items. For example, p= (12,10,8,10,12,10,8,11).
Apply the DCT to them and find that it results in eight coefficients: 28.6375, 0.571202, 0.46194, 1.757, 3.18198, -1.72956, 0.191342, -0.308709.
These can be fed to the Inverse Discrete Cosine Transform (IDCT), transformed by it, to precisely reconstruct the original data. But, in the transform based method our goal is to compress the data by quantizing the coefficients. We quantize the coefficients to 28, 0, 0, 2, 3, -2, 0, 0, and apply the IDCT.
The reconstructed sequence is: 11.236, 9.62443, 7.66286, 9.57302, 12.3471, 10.0146, 8.05304, 10.6842.
So, we can say that DCT based encoders are always lossy, because round off errors are inevitable in the DCT steps.

1.3.2 Prediction based Method

In general, the prediction is formed by a linear combination of m previous samples:

$$\hat{I}_n = round[\sum_{i=1}^{m} a_i I_{n-i}] \tag{1.10}$$

Where, m is the order of the linear predictor, round is a function to denote the rounding or nearest integer operation, and a_i are the prediction coefficients. In the predictive coding technique, the value of a current pixel is being predicted from some of the previous pixels. These pixels are already stored in the memory when the image is scanned. Like in raster scan, in order to predict a pixel X(i, j), we have already scanned the pixels of the above row and pixels to the left. Alternatively, we can say that for 2-D images, the prediction is a function of the previous pixels in a left-to-right, top-to-bottom scan of an image as shown in Fig. 1.8.
Based on the number of pixels involved, we define the order of predictor.

1. First-order predictor:

$$X(r, c) = X(r, c - 1)$$

2. Second-order predictor:

$$X(r, c) = 1/2[X(r, c - 1) + X(r - 1, c)]$$

3. Third-order predictor:

$$\hat{X}(r, c) = X(r, c-1) - X(r-1, c-1) + X(r-1, c)$$

$$\hat{X}(r, c) = 0.75X(r, c-1) - 0.5X(r-1, c-1) + 0.75X(r-1, c)$$

In predictive coding, we use the fact that the intensity value of a pixel is likely to be similar to that of the surrounding pixels. Using prediction, we will eliminate the interpixel redundancies. Instead of coding an intensity value, we predict the value from the values of nearby pixels. The difference between the actual and predicted value of that pixel is coded. There is a high probability that the difference will be small. The predictive coding system consists of an encoder and a decoder, each containing an identical predictor. The predictor transforms a set of high-entropy but correlated values into a set of low-entropy but less correlated values. For example, there is a 4×4blockofanimagepixelsasshowninFig.1.9(a).
Topredictthepixelatposition
X(2, 2) and onward, we will store in memory the pixels of first row and first column. To predict the pixels of an image block, we will use the predictor

$$\hat{X}(r, c) = X(r, c-1) - X(r-1, c-1) + X(r-1, c).$$
$$\hat{X}(2, 2) = X(2, 1) - X(1, 1) + X(1, 2) \hat{X}(2, 2) = 46 - 45 + 89$$

Now, the error image will be calculated as,

$$e(r, c) = X(r, c) - \hat{X}(r, c)$$

so the error will be

$$e(2, 2) = 64 - 89 = -26$$

These steps are explained in Fig. 1.9. In the decoder side, we will use the same predictor and the original image will be reconstructed as

$$X1(r, c) = \hat{X}1(r, c) + e(r, c)$$

The reconstruction process is shown in Fig. 1.10. Quantizer: It reduces the accuracy of the transformation output in accordance with some pre-established fidelity criteria. It reduces the psycho visual redundancy in the image. In quantization, high rate pixel intensities are mapped into a relatively small number of symbols. This operation is

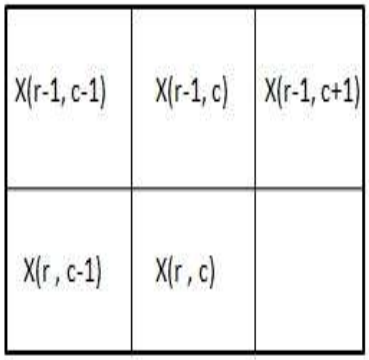

Figure 1.8: Predictor Structure

Irreversible and it is lossy. The conversion can operate on individual pixels (scalar quantization) or groups of pixels (vector quantization). So, in the lossless compression method we will not use quantization. Lossless Symbol Coding: This stage generates a binary bit-stream by assigning binary codeword's to the input symbols. The first two stages: source transformation and quantization can be regarded as preprocessing stages for mapping the data into a form that can be more efficiently coded by this lossless coding stage. It assigns the shortest code to the most frequently occurring symbols.

1.4 Compression Methods

Coding techniques are crucial for the effective transmission or storage of data-intensive visual information. In fact, a single uncompressed color image or video frame with a medium resolution of 500 × 500 pixels would require 100 seconds for transmission over an Integrated Services Digital Network (ISDN) link having a capacity of 64,000 bits per second (64 Kbps). The resulting delay is intolerably large considering that a delay as small as 1 to 2 seconds is needed to conduct an interactive "slide show" and a much smaller delay (on the order of 0.1 second) is required for video transmission or playback.

Although a CD-ROM device has storage capacity of a few gigabytes, its average data-read throughput is only a few Megabits per second (about 1.2 Mbps to 1.5 Mbps for the common 1 × read speed CLV CDs). As a result, compression is essential for the storage and the real-time transmission of digital audiovisual information, where large amount of data must be handled by devices having a limited bandwidth and storage

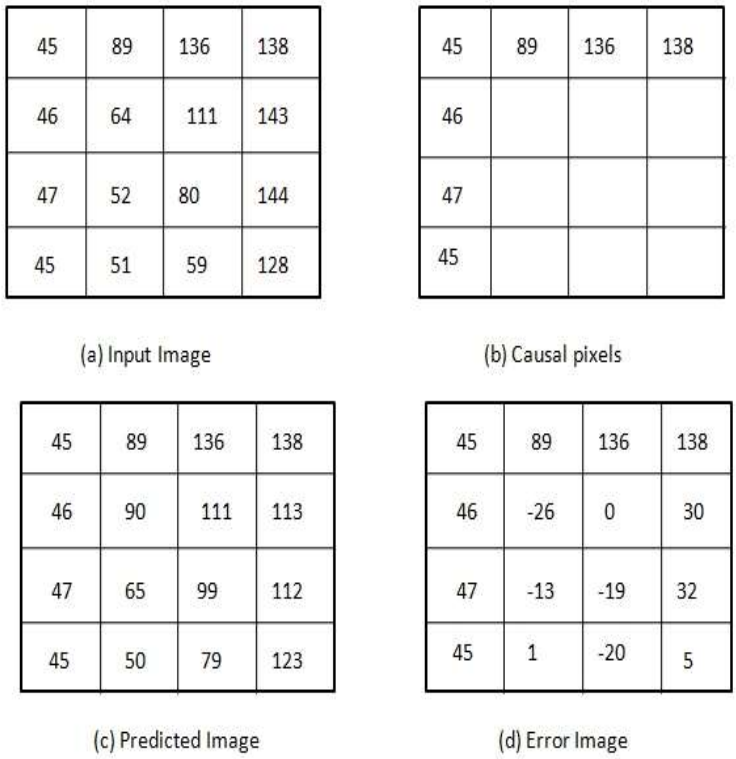

Figure 1.9: Steps to get Prediction Error

capacity. Basically, there are two types of compression methods:

1.4.1 Lossy Compression Methods

The goal of lossy compression is to achieve the best possible fidelity given an available communication or storage bit rate capacity or to minimize the number of bits representing the image signal subject to some allowable loss of information. In this way, a much greater reduction in bit rate can be attained as compared to the lossless compression, which is necessary for enabling many real time applications involving the handling and transmission of the audiovisual information. The function of compression is often referred to as coding, for short.

1.4.2 Lossless Compression Methods

The goal of lossless image compression is to represent an image signal with the smallest possible number of bits without loss of any information, thereby speeding up transmis-

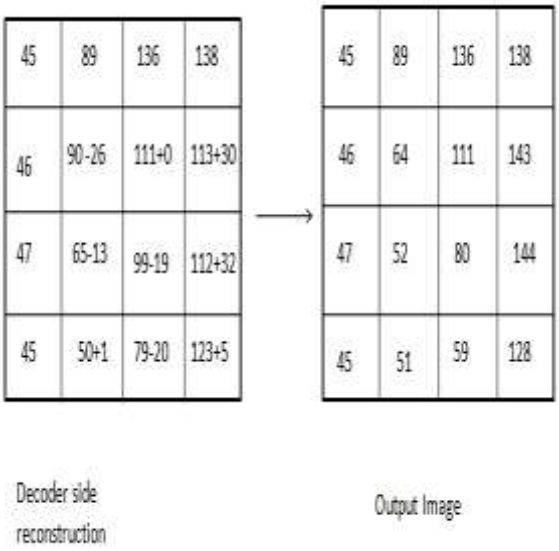

Figure 1.10: Reconstruction of the Image

sion and minimizing storage requirements. The number of bits representing the signal is typically expressed as an average bit rate (average number of bits per sample for still images and average number of bits per second for video). Lossy compression methods give high compression ratio up to 50:1 or more, while maintaining good perceptual quality of the reconstructed image. On the other hand, lossless compression methods generally do not give a very good compression ratio, but the reconstructed image is exact replica of the original image.

1.5 Variable Length Entropy Coding

Entropy, a notion first introduced by Shannon [1], is a measure of information. There is high amount of information in an event if the probability of the occurring of the event is low and vice versa. As an example, suppose you receive a phone call in January from a friend of yours residing in Delhi. He says that the weather is very cold here. This sentence really does not carry much information, as at that time of the year you do expect the weather to be cold there. However, if a top tennis player is beaten by a Wimbledon wildcard in the first round of the tournament, all the sports channels will talk about the unexpected news, as there is high amount of information in it.

Shannon [1] showed that the average number of bits necessary to encode a memory-less source, without loss, cannot be lower than the entropy of the source. However, the zeroth order entropy does not take the memory among the source symbols into account. When memory is present in a source, dependencies between the symbols need to be exploited to achieve the maximum compression. In this case, the achievable lossless bit rate is governed by high-order entropy which is less than zeros order entropy. The entropy coding which is used in compression is basically of two types:

- Fixed Length Coding: In the fixed length coding, all codeword's have the same length (number of bits). ASCII, the most widely used code for representing text in computer systems, is a fixed length code. For example: A-000, B-001, C-010, D-011, E-100, F-101, etc.

- Variable Length Coding: In the variable length coding number of digits which are used to encode letters varies. For example: A-0, B-00, C-110, D-111, E-1000, F-1011

For example, if we want to encode symbols "ABAADECF" through a fixed length code, we need 24 bits. But, in variable length code we need 19 bits. Reducing the number of symbols used to encode a message can be quite important. If the message is going to be transmitted through a computer network, the amount of time required for the transmission will be proportional to the number of symbols required to encode it. If we can reduce the number of symbols in the encoding by 10%, then the message can be transmitted 10% more quickly. So, the variable length coding is preferred in case of compression.

The term entropy coding in this thesis refers to use of a variable length code to lossless represent a sequence of symbols from a discrete alphabet. A practical entropy code must be uniquely decodable, so that there is only one possible sequence of codeword's for any unique input sequence. Currently, the following popular entropy coding techniques, such as, Huffman coding, Run Length coding, Golomb-Rice coding, and Arithmetic coding are used in modern compression systems and standards.

1.5.1 Huffman Coding

Huffman encoding [3] is an example of entropy encoding which is based on statistical methods. Given the character that must be encoded, together with the probability

of their occurrences, the Huffman encoding algorithm determines the optimal code using the minimum number of bits. Hence, the length (number of bits) of the coded characters will differ. The most frequently occurring characters are assigned to the shortest code words. A Huffman code can be determined by successively constructing a binary tree, whereby the leaves represent the characters that are to be encoded. Every node contains the relative probability of occurrence of the characters belonging to the sub-tree beneath the node. The edges are labeled with the bits 0 and 1. The following brief example illustrates the process:

1. The letters A, B, C, D, and E are to be encoded and have relative probabilities of occurrence as follows:
   P (A) = 0.16, P (B) = 0.51, P (C) = 0.09, P (D) = 0.13, P (E) = 0.11.

2. The two characters with the lowest probabilities, C and E, are combined in the first binary tree, which has the characters as leaves. The conditional probability of their root node CE is 0.20. The edge from node CE to C is assigned a 1 and the edge from CE to E is assigned a 0. This assignment is arbitrary, therefore, different Huffman codes can result from the same data.

3. Nodes with the following relative probabilities remain: P (A) = 0.16, P (B) = 0.51, P (CE) = 0.20, P (D) = 0.13. Now, the two nodes with the lowest probabilities are D and A. These nodes are combined to form the leaves of a new binary tree. The combined probability of the root node AD is 0.29. The edge from AD to A is assigned a 1 and the edge from AD to D is assigned a 0. If root nodes of different trees have the same probability, then trees having the shortest maximal path between their root and their nodes should be combined first. This keeps the length of the code words roughly constant.

4. Nodes with the following relative probabilities remain (AD) = 0.29, P (B) = 0.51, P (CE) = 0.20. Now, the two nodes with the lowest probabilities are AD and CE. These are combined into a binary tree. The combined probability of their root node ADCE is 0.49. The edge from ADCE to AD is assigned a 0 and the edge from ADCE to CE is assigned a 1.

5. Two nodes remain with the following relative probabilities' (ADCE) = 0.49, P (B) = 0.51. These are combined to a final binary tree with the root node

ADCEB. The edge from ADCEB to B is assigned a 1 and the edge from ADCEB to ADCE is assigned a 0.

6. Figure 1.11 shows the resulting Huffman code as a binary tree. The result is the following code words, which are stored in a table:
w(A) = 001, w(B) = 1, w(C) = 011, w(D) = 000, w(E) = 010.

Such a table could be generated for an image. The same table must be available for both encoding and decoding. If the information of an image can be transformed into a bit stream, then a Huffman table can be used to compress the data without any loss.

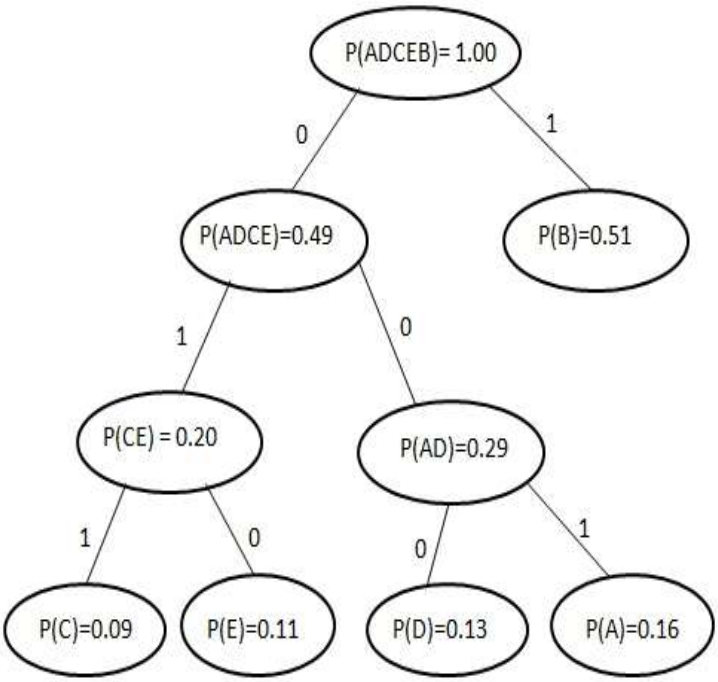

Figure 1.11: Huffman Coding

1.5.2 Run Length Coding

In run length encoding, if a byte occurs at least four consecutive times, then the number of occurrences is counted. The compressed data contains this byte followed by a special flag, called M −byte, and the number of its occurrences. The exclamation mark ! can be

defined as this M − byte. A single occurrence of this exclamation mark is interpreted as M−byteduringthedecompression;twoconsecutiveexclamationmarksareinterpreted as an exclamation mark occurring within the data.
An example of a run length encoding:
Uncompressed data: ABCCCCCCCCDEFGGG
Compressed data: AB C!8DEFGGG

1.5.3 Arithmetic Coding

Arithmetic coding [4]-[5] is a form of variable-length entropy encoding, used in lossless data compression. Arithmetic coding differs from other forms of entropy encoding, such as Huffman coding. Rather than seperating the input into component symbols and replacing each with a code, Arithmetic coding encodes the entire message into a single number, a fraction n where ($0.0 \leq n < 1.0$). The idea behind Arithmetic coding is to have a probability line, $0 - 1$, and assign to every symbol a range in this line based on its probability. The higher the probability, the higher range which assigns to it. Once we have defined the ranges and the probability line, it start to encode symbols, every symbols defines where the output floating point number lands. Example:

1. Start with an interval [0, 1), divided into sub-intervals of all possible symbols to appear within a message. Make the size of each subinterval proportional to the frequency at which it appears in the message.

Symbol	Probability	Interval
a	0.2	[0.0, 0.2)
b	0.3	[0.2, 0.5)
c	0.1	[0.5, 0.6)
d	0.4	[0.6, 1.0)

2. While encoding a symbol, "zoom" into the current interval, and divide it into sub-intervals like in step (1) with the new range. Example: suppose we want to encode "abd". We zoom into the interval corresponding to "a", and divide up

that interval into smaller sub-intervals like before. We shall now use this new interval as the basis of the next symbol encoding step.

Symbol	New "a" interval
a	[0.0, 0.04)
b	[0.04, 0.1)
c	[0.1, 0.102)
d	[0.102, 0.2)

3. Repeat the process until the maximum precision of the machine is reached, or all symbols are encoded. To encode the next character "b", we use the "a" interval created before and zoom into the subinterval "b", and use that for the next step. This produces: and lastly, the finally result is:

Symbol	New "b" interval
a	[0.102, 0.1216)
b	[0.1216, 0.151)
c	[0.151, 0.1608)
d	[0.1608, 0.2)

Symbol	New "d" interval
a	[0.1608, 0.16864)
b	[0.16864, 0.1804)
c	[0.1804, 0.18432)
d	[0.18432, 0.2)

4. Transmit some number within the latest interval to send the codeword. The number of symbols encoded will be stated in the protocol of the image format, so any number within [0.1608, 0.2) will be acceptable.

To decode the message, a similar algorithm is followed, except that the final number is given and the symbols are decoded sequentially from that.

Like Huffman coding, Arithmetic coding is optimal from an information theoretical point of view. Therefore, the length of the encoded data is also minimal. Unlike Huffman coding, Arithmetic coding does not code each symbol separately. Each symbol is instead coded by considering all prior data. Thus, a data stream encoded in this fashion must always be read from the beginning. Consequently, random access is not possible. In practice, the average compression rate achieved by Arithmetic and Huffman coding is similar.

Differences in compression efficiency arise in special cases, for example, in digitized graphics consisting primarily of a single (background) color. Arithmetic coding is better suited than Huffman coding in this case. This is always the case if symbols occur in the input data stream with such a frequency that they have a very small information content. These can be encoded using less than one bit, whereas in Huffman coding each symbol requires at least one bit.

1.5.4 Golomb-Rice Coding

In 1965, Golomb [6] developed a very simple coding scheme for sources with a geometric probability distribution, that is, of the following form:

$$P(n) = (1 - p_0)\, p0,\ n \geq 0,\ 0 < p_0 < 1, \qquad (1.11)$$

where p_0 is constant. For example, if p_0 denotes the probability of zero, then P (n) is the probability of a run-length of n zeros. Given a positive integer m, the Golomb code G_m encodes a non-negative integer n as follows. Divide n by m. Let q be the quotient and r be the remainder of this division (that is, n = mq + r). Then, the G_m code for n is the concatenation of the unary representation of q (that is, q zeros followed by a single one) with the modified binary representation of r (that is, using codeword's of length $\lfloor \log_2 (m) \rfloor$ bits for $r < 2^{\lceil \log 2 (m) \rceil} - m$ and $\lfloor \log_2 (m) \rfloor$ bits otherwise).

For example, let m = 3 and n = 14. Dividing 14 by 3 yields q = 4 and r = 2. The unary representation of 4 is 00001 and the binary representation of 2 is 10. Thus, the G_3 code of 14 is given by concatenating 00001 and 10, that is, 0000110.

For the geometric probability distribution, a G_m code is optimal if m is given by

$$m = [-(\log(1 - pb)/pb]$$

Later, Rice addressed the problem of coding both negative and non-negative values and rediscovered a special case of Golomb codes when m is a power of 2, that is $m = 2^k$, with very simple encoding and decoding procedures. For $m = 2^k$, the quotient of n/m is given by shifting to the right by k bits the binary representation of n, and the remainder of n/m is given by the last k bits of n. Thus, the Rice code for n is given by concatenating the unary representation of q with the k least significant bits of n.

For example, let k = 2 (m = 4) and n = 22. The binary representation of 22 is 10110. 10110 shifted to the right by two bits (which corresponds to dividing 22 by 4) yields 101 = 5, which has a 000001 unary representation. The two least significant bits of 10110 are 10. Thus, for k = 2, the Rice code of 22 is 00000110. Since both Golomb and Rice made the observation about the simplicity of generating codes when $m = 2^k$, these special codes are also referred to as Golomb-Rice codes.

The Golomb-Rice codes were designed for coding non-negative values. If a source has negative values in a two-sided geometric distribution, then one can remap all the values to non-negative values. This also requires a good estimate of the center of this two-sided distribution. Golomb-Rice codes have been used lately in a number of image compression schemes. This is due to the observation that prediction error in image coding tend to behave as two-sided geometric distribution centered at zero.

1.6 Performance Measurement

Appropriate performance metrics are required to evaluate the performance of a specific image compression scheme. An image compression algorithm can be evaluated in many different ways according to different requirements. Major compression metrics include compression ratio, compression speed, computing complexity, memory and storage complexity, objective and subjective quality of the reconstructed image, etc. The most common metric of performance measure of an image compression scheme is the compression ratio, which is defined by:

$$\text{Compression Ratio} = \frac{\text{Original Image size in bits}}{\text{Compressed Image size in bits}} \qquad (1.12)$$

i.e., the ratio of the number of bits to represent the original image data to the number of bits to represent the compressed image data. The original image size used in this

thesis does not include any image format overhead or byte alignment overhead. For an image of size W × H and bit depth b, the original image size is simply calculated by the following formula:

$$\text{Original Image size in bits} = W \times H \times b \tag{1.13}$$

On the contrary, the compressed image size counts all header or tail overhead needed to reconstruct the original image. For example, if an image of size 512 × 512 with 8 bits per pixel is compressed to 8192 bytes, the compression ratio will be 32 : 1 or 32. Another way to evaluate image compression performance is to use compression rate in the unit of bits per sample or bits per pixel (bpp), which is defined as the average number of bits used to represent a single sample (pixel). The compression rate is given by:

$$\text{Compression Rate} = \frac{\text{Compressed Image size in bits}}{\text{Number of pixels}} \tag{1.14}$$

In the above example, we say that the original rate is 8 bpp and the compression rate Is $\frac{8192\times 8}{512\times 512} = 0.25$

Note that a larger value of compression ratio or a smaller value of compression rate indicates better compression performance of a compression scheme.

1.7 Objective of the Proposed Thesis

Uncompressed images normally require a large amount of storage capacity and transmission bandwidth. For example, a 24-bit true color high-definition television (HDTV) image with size 1920 × 1080 needs approximately 6 megabytes (6 Mbytes) of storage space. On the other hand, various types of redundancy exist in images, such as, spatial redundancy (or interpixel redundancy), coding redundancy, psycho visual redundancy, etc. The primary goal of image compression is to minimize the number of bits required to represent the original images by reducing the redundancy in images, while still meeting the user-defined quality requirements. The core issue in image compression is to design efficient and effective compression schemes.

In recent years, a lot of research have been done in lossless compression of gray-scale images. However, existing gray-scale image compression schemes either do not provide the lossless compression or not comparable to CALIC in terms of their compression performance and complexity. CALIC (Context-based, Adaptive, Lossless Image Codec),

which is widely accepted as the state-of-the-art lossless compression algorithm for grayscale image compression. Therefore, this thesis will emphasize on the development of an efficient algorithm for lossless compression of images which performs better than CALIC in terms of compression ratio. We have employed two methods to achieve our objective: prediction and context modification.

1.8 Problem Statement

Images require large amount of disk space for storage purpose. Compression of images will help in storing and in transmitting. In some image classes, any loss at the time of reconstruction is unacceptable. For example, because of the legal and regulatory issues, compression of medical images should be lossless. In this case, decoder should also be simple. Because of the above given reasons, we have proposed a lossless compression algorithm in this thesis. This algorithm performs better than the existing lossless compression algorithms like JPEG-LS and CALIC.

1.9 Thesis Outline

Chapter 2: This chapter described a survey on lossless image compression methods.
Chapter 3: This chapter proposed a modified GAP algorithm (MGAP) which gives minimum entropy as compared to GAP and MED.
Chapter 4: This chapter deals with the conclusions and future research work
Chapter 5: References.

2

A Survey on Lossless Image Compression Methods

2.1 Overview

Recent past has seen a lot of contributions in the area of lossless image compression from several researchers. In literature, broadly two types of methods are present for lossless compression of the images:

1. Prediction based Methods
2. Transform based Methods

Current prediction based methods are generic in nature and consists of three main steps as mentioned below:

1. Decor relation of images by subtracting predicted pixels, i.e., prediction
2. Estimation of current pixel context, which is found by measuring some of the properties of its causal neighborhood pixels, i.e., Context Modeling
3. Context based entropy coding of prediction errors, i.e., Entropy Coding

These above mentioned steps form the basic structure of the state-of-the-art lossless image coding methods. Some lossless image coding methods are described in [9]. There are some state-of-the-art lossless image coding algorithms which used different prediction method to predict the neighboring pixels.

2.2 JPEG-LS

Marcelo J. Weinberger, Gadiel Seroussi, and Guillermo Sapiro [7] presented a lossless compression algorithm, called LOCO-I (LOw COmplexity LOssless COmpression). LOCO-I is the core algorithm used in the ISO/ITU standard for lossless and near lossless compression of continuous tone images.

JPEG-LS is based on the ideas used for LOCO-I compression method. JPEG-LS examines several of the previously-seen neighbors of the current pixel, uses them as the context of the pixel. It is based on a simple fixed context model. It uses the context to predict the pixel and to select a probability distribution out of several such distributions, and uses that distribution to encode the prediction error with a special Golomb code.

2.2.1 Median Edge Detector (MED) (Prediction Method)

The prediction in JPEG-LS is based on the causal template depicted in Fig. 2.1, where X denotes the current pixel and N , W , N W , and N E, are neighboring pixels in the North, West, North-West, and North-East direction respectively. JPEG-LS limit its image buffering requirement to one scan line.

Ideally, the value guessed for the current pixel X should depend on N, W, N W, and NE through an adaptive model of the local edge direction. While our complexity constraints rule out this possibility, some form of edge detection is still desirable. In LOCO-I/JPEG-LS, a fixed predictor performs a primitive test to detect vertical or horizontal edges. The predictor used in JPEG-LS is known as Median Edge Detector (MED).

In MED predictor, each pixel is processed in a raster scan order. The MED predictor generates a predicted value for each processed pixel. By subtracting the predicted value from the original value we get the prediction error. The prediction error value is then computed and entropy coded.

The prediction is based on the causal template of three neighborhood pixels, as shown in Fig. 2.1, where X represents the current pixel (i.e., the pixel being predicted), and N, W, and NW, represent the pixel values at the spatial location North, West, and North-West, respectively. The MED predictor uses the past data of X in the causal template, i.e., N , W , and N W , to check whether any vertical or horizontal edge can

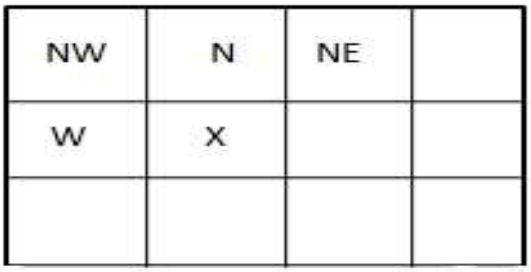

Figure 2.1: Causal Template used by JPEG-LS

be detected. When a vertical edge is detected to the left of the current pixel, the MED predictor tends to pick N as the predictive value. It tends to predict W as the predictive value when a horizontal edge above the current pixel is detected. However, when there is neither a horizontal edge nor a vertical edge, MED predicts X as $N + W - N W$. This is under an assumption that the pixel intensity variation in horizontal and also in vertical direction follows the equality conditions as described below,

Variation in horizontal direction: $N W - N = W - x$

$$\Rightarrow xN + W - N W$$

Variation in vertical direction: $N W - W = N - x$

$$\Rightarrow xN + W - N W$$

The above method of prediction is mathematically described as follows,

if($N W \geq \max(N, W)$) then
$x=\min(N,W)$
else if($N W \leq \min(N, W)$) then
x=max(N,W)
else
$x=N+W-NW$
endif

The guessed value is seen as the median of three fixed predictors, N , W , and N +

W −NW. Combining these interpretations, the predictor was renamed during the standardization process as "Median Edge Detector" (MED).
The predicted value is subtracted from the original pixel value to get the prediction error, as follows:

$$e_i = X_i - \hat{X}_i \quad (2.1)$$

This can be noted that it is the prediction errors that are transmitted to the decoder and decoder reconstructs the original pixel, X, doing a reverse operation.

2.2.2 Context Modeling

For context determination the following steps should be followed:

1. Calculate the three gradient values, i.e., the context that conditions the encoding of the current prediction residual in JPEG-LS is built out of the differences D_1 = NE−N, D_2 = N −NW, and D_3 = NW −W. These differences represent the local gradients, thus capturing the level of activity (smoothness, edginess) surrounding a sample, which governs the statistical behavior of the prediction errors.

2. Compare the three gradients D_i to certain parameters and calculate three region numbers Q_i. Each region number Q_i can take one of the nine integer values in the interval [- 4, + 4], so there are 9 × 9 × 9 = 729 different region numbers.

3. Map the region numbers Q_i to an integer Q in the interval [0, 364]. The tuple (0, 0, 0) is mapped to 0, and the 728 remaining tuples are mapped to [1, 728/2 = 364], such that (a, b, c) and (- a, - b, - c) are mapped to the same value.

The integer Q becomes the context for the current pixel X. After determining the context Q, it corrects the prediction, based on the quantity SIGN (determined from the signs of the three regions Q_i), the correction values C[Q] and parameter M AXV AL. The algorithm used for prediction correcting is mathematically described below:

```
if (SIGN = +1) then x = x + C[Q]
 else
x= x−C[Q]
```

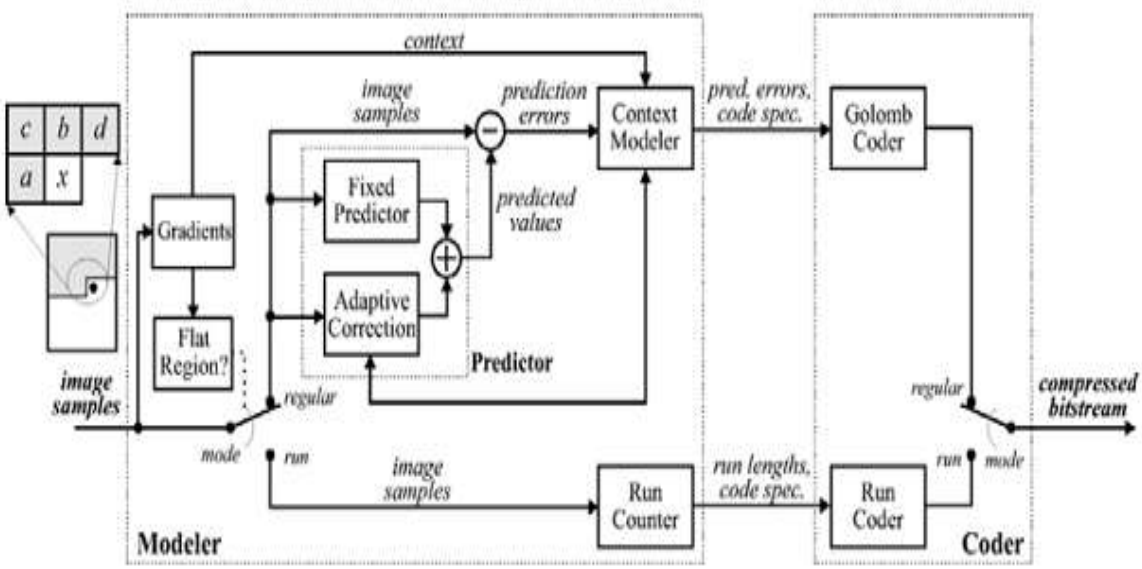

Figure 2.2: Block Diagram of JPEG-LS

if (x > M AXV AL) then x = M AXV
AL
else
if (x < M AXV AL) then x = M AXV AL

2.2.3 Entropy Coding

JPEG-LS uses Golomb-Rice coding method for entropy coding. The block diagram for JPEG-LS encoder is given in Fig. 2.2.

2.3 Context based Adaptive Lossless Image Coding (CALIC)

CALIC [10] is a one-pass coding scheme that encodes and decodes in raster scan order. It uses the previous scan lines of coded pixels to do the prediction and form the context. The continuous-tone mode basically has four major components

1. Gradient Adjusted Predictor (GAP): GAP tries to use the context gradient information to predict the intensity of current pixel, x, which result in a x. This is the prediction step.

2. Context Selection and Quantization: This step try to further remove the correlation between the prediction errors of GAP step by condition the error onto

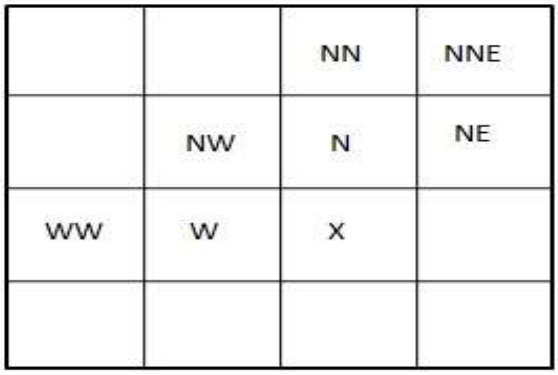

Figure 2.3: Causal Template used by GAP

different context error energies. The quantization of context error energies results in totally 8 different energy levels.

3. Context Modeling of Prediction Errors: Error modeling try to classify the errors into different texture catalogs and then use the corresponding sample means e to further adjust x, which results in a x.

4. Entropy Coding of Prediction Errors: After the whole modeling prediction either Huffman coding or Arithmetic coding can be applied. The frame structure of CALIC is shown in Fig. 2.3.

2.3.1 Gradient Adjusted Predictor (GAP)

The GAP predictor, which is employed in CALIC, adapts itself to the gradients of horizontal and vertical edges. The GAP tries to detect how rapidly the edge is changing around pixel x and then by classifying the tendency of edge changing into sharp, normal, and weak edge. It gives different weight for the neighborhood pixels for a linear prediction of pixel x. For a pixel belonging to a slope bin, the predictor associated with that bin is used to predict the pixel x.
First CALIC defines d_v as the criteria for vertical direction edge and d_h as the criteria for horizontal direction edge. d_v and d_h are computed as:

$$d_v = |(W - N\,W)| + |(N - N\,N)| + |(N\,E - N\,N\,E)|$$

$$d_h = |(W - W\ W)| + |(N - N\ W)| + |(N - N\ E)|$$

where, d_v and d_h are estimates of the gradients of the intensity value near pixel x in the horizontal and vertical direction. The slope can be estimated as:

$$S = d_v - d_h$$

With d_v , d_h, and value of S, CALIC defines the linear gradient adjusted prediction as given in the following pseudo codes:

```
if (S>80) x = W //Sharp Horizontal Edge
else if (S< - 80) x = N //Sharp Vertical Edge
else
x=(W +N)/2+(NE−NW)/4
if (S>32) x = ( x+W)/2 //Horizontal Edge
else if (S>8) x = (3 x+W)/4 //Weak Horizontal Edge
else if (S< −32) x = ( x+N)/2 //Vertical Edge
else if (S<-8) x = (3 x+N)/4 //Weak Vertical Edge
```

The predictor coefficients and thresholds given in the pseudo code were arbitrary chosen. They are based on some experiments and round to the number that has good relationship with 2, so that it is easy to construct fast hardware implementation.

2.3.2 Coding Context

The GAP step only removes part of redundancy in the image. It is observed that the variance of prediction errors e = x − x strongly correlated the smoothness of the image around the predicted pixel x. A further modeling of this correlation is needed to remove more redundancy of these smoothness areas. CALIC defines an error energy estimator as

$$\Delta = d_h + d_v + 2|e_w|$$

where $e_w = x_w - x_w$. The advantage of having Δ is that, we can condition the error distribution on Δ, so that, the prediction error can be separated into different variance classes. Therefore, the entropy coding of errors using estimated conditional probability P (e/Δ) improves coding efficiency over merely using P (e). It is easy to image that in

an image, large amount of Δ value will occur according to different context. We need to quantize Δ to achieve a good trade-off of compression effect and efficiency. CALIC selects an 8-levels quantizer to quantize Δ. For instance, the Q is the quantizer of Δ, i.e., Q:Δ → 0, 1, 2 .... 7, based on the criteria of minimizing the conditional entropy of the errors. CALIC got a best performance partition Δ of the following level thresholds:

$$T = 5,15,25,42,60,85,140 \tag{2.2}$$

The 8-level quantize for the error energy actually can help small images to generate enough samples for context modeling to learn P (e/Q(Δ)) quickly in adaptive entropy coding. Meanwhile, it also saves a lot of memory during entropy coding.

2.3.3 Context Modeling for Adaptive Error Feedback

Even GAP is a well constructed predictor, it still can not capture the feature for different contexts. In order to refine our GAP prediction result, CALIC tries to model the context of the prediction error so that the higher order structures such as texture patterns and local activity in the image can be exploited. This context modeling can give us further compression gain in the sense of removing more redundancies.

It is easy to imagine that in an image, we can have a lot of combinations of possible contexts. They can lead to the sparse context (modeling dilution) or high model cost problem. CALIC tries to use binary relationship (larger or smaller) between context pixels to form the context C. Furthermore, the first order statistic of context model which is the error mean e(C) will be used as a feedback to further refine GAP prediction so that the whole prediction becomes a nonlinear predictor.

2.3.3.1 Context Formation

To avoid context dilution problem, CALIC carefully selects the context elements as combining 144 context textures with 4 context error energies. This selection is an empirical one so that it can be further refined. But actually from the compression effect of CALIC, we can see it works gracefully. Here is the error modeling context formation method:

$C = [x_0, ...x_6, x_7]$

= [N , W , N W , N E, N N , W W , 2N − N N , 2W − W W]

C is then quantized to an 8 bit binary number B = $b_7b_6b_5\ldots\ldots b_0$ using the prediction value x as the threshold, namely, ($0 \leq k < K = 8$):

$$b_k = \begin{cases} 0 \text{ if } x_k \geq x \\ 1 \text{ if } x_k < x \end{cases}$$

This one byte texture patterns in the modelling context indicate the behavior of e. Since the variability of neighboring pixels also influences the error distribution, we combine the quantized error energy with the quantized texture pattern to form compound modelling contexts denoted by C(δ, β). Here, again to avoid the context dilution problem, we further divide the quantized error energy by 2 to get $\delta = \Delta/2$. This scheme can be viewed as a product quantization of two independently treated image features: spatial texture patterns and energy of prediction errors. So, the final context formation can be described as: $C(\delta, \beta) = l_1l_0|b_7b_6\ldots\ldots b_0$ or $b_7b_6\ldots\ldots b_0|l_1l_0$, where ($l_1l_2$) is the binary value of δ, e.g., 00, 01, 10, 11. This C(δ, β) seems to have $2^{10} = 1024$ different values. But actually, only 576 different values are possible. If the prediction value x lies between I_n and I_{nn}, then the condition $2I_n - I_{nn}$ is either always 1 or always 0. So, the quantized result of I_n, I_{nn}, and $2I_n - I_{nn}$ only can have 6 different values instead of 8. The same situation applies to I_w, I_{ww}, and $2I_w - I_{ww}$ as well. So, the total possible values of C (δ, β) are $4 \times 6 \times 6 \times 4 = 576$.

2.3.3.2 Error Feedback

To utilize the context information to generate a positive feedback error, CALIC uses error conditional expectations within each compound context by using the corresponding sample means ê (δ, β) for different compound contexts. Computing ê (δ, β) only involves accumulating the error terms in each compound context and maintaining a count on the occurrence of each context. For each compound context, we count the number of occurrences. CALIC only use 1 byte to store the occurrence count. Whenever the count reaches a predefined value of 128, it scales down the count, but leaves the ê (δ, β). Besides being a technique to reduce the memory use, rescaling also has the widely recognized beneficial side effect of the observed data. It is an inexpensive way of adapting the context error model to time varying sources. Indeed, the scaling technique is slightly improved compression for CALIC. The conditional mean of context modeling ê (δ, β) is

the most likely prediction error in a given compound context C(δ, β). CALIC corrects the bias in GAP prediction step by a error feedback to get better prediction, namely x= X +ê(δ,β). CALIC actually uses the new prediction error e=x−x, instead of e=x−x,todothecontextbasederrormodelling.Therefinedpredictionwillbe more accurate. Conceptually, we ultimately have a two-stage adaptive prediction scheme via context modeling of errors and error feedback.

2.3.4 Entropy Coding of Prediction Errors

After the previous prediction part, the prediction error needs to be encoded by entropy coding method to achieve real compression.

2.3.4.1 Error Sign Flipping

Since the decoder also can generate the same context error mean ê(δ, β), CALIC first checks whether the context error mean is negative or not. If yes, −ϵ is encoded; else +ϵ is encoded. This step will sharpen the conditional probabilities; hence will reduce the conditional entropy. The basic idea is to sharpen the probability distribution so that smaller variance can be achieved.

2.3.4.2 Remapping Errors

Another technique CALIC applies to entropy coding is error remapping, which means it tries to remap the errors to a probability decreasing order so that entropy coding will be benefited. Although prediction errors can potentially take on 2^{z+1} possible values that range from $(-2^z + 1)$ to $(+ 2^z - 1)$, they can be mapped into the range 0 to 2^{z-1} using the constraint that the value must fall into the interval ⌊−I,2z −1⌆ I], where z is the number of bits in intensity resolution.

2.3.4.3 Histogram Tail Truncation

In order to save entropy coding space as much as we can, CALIC actually will utilize different size of entropy coding model to encode the prediction error based on different context error energies δ. CALIC said, even after error remapping, an alphabet size of 2^z is still unnecessarily large. Large errors occur with diminishing frequency or not at all. But they still occupy spots in code space. For instance, for δ = 0, over 99%

of the error population is within the range [8, 8] of the error histogram. If an error histogram of size 2^z is used, it will significantly reduce the efficiency of entropy coding

by distorting the underlying error statistics. So, CALIC truncates the tail of the error histogram that is used to estimate P (e|δ), and use an escape mechanism to code the errors beyond the truncated code range, if they occur. Specifically, CALIC limits the size of each conditional error histogram to some value N_δ based on the error energy level $1 \leq \delta < L$, such that a large majority of errors to be coded under the coding context δ falls into the range of the N_δ largest entries of the δth error histogram. By doing a lot of experiments, CALIC got a good table sizes for the eight coding contexts as following:

$$(N_0...N_7) = (18, 26, 34, 50, 66, 82, 114, 256) \quad (2.3)$$

The whole idea is actually similar to the post office functioning wherein different boxes are kept for different parcels. Since we do not know the parcel size for the time being, so we first used to classify them by their estimated weight. Then if the parcel is oversize for the prescribed box of its estimated weight level, a larger box will be used. The block diagram for CALIC encoder is given in Fig. 2.4.

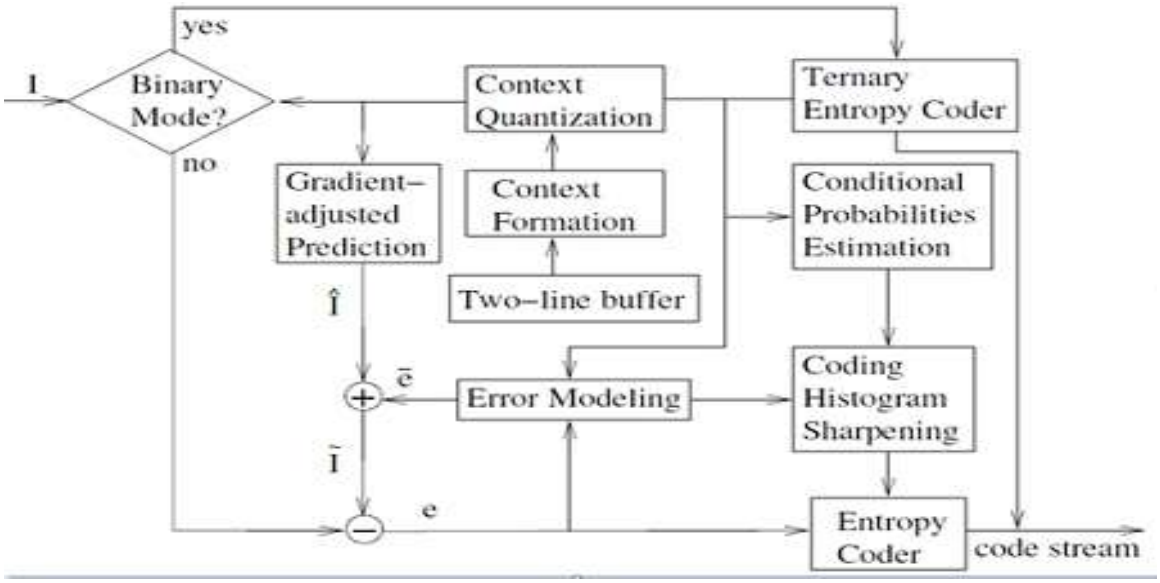

Figure 2.4: Block Diagram of CALIC

2.4 TMW

B. Meyer and P. Tischer proposed a general purpose lossless grayscale image compression method, known as TMW [11]. It is based on the use of linear predictors and implicit segmentation. In this algorithm, the compression process is split into an analysis step and a coding step. In the first step, called as image analysis step, a set of linear predictors and other parameters suitable for the image is chosen. These are included in a way that minimizes the length of the encoded image and is then used by an arithmetic coder, which codes the samples in a second pass (coding stage). The chosen parameter set has to be considered part of the encoded image and has to be stored or transmitted alongside the result of the coding stage. Three different kinds of predictors are used

1. Pixel-predictors, that predict a pixel value based on the pixel values of its causal neighbors.
2. Sigma-predictors, that predict the magnitude of a pixel-predictor's prediction error based on the magnitude of that pixel-predictor's prediction errors for the causal neighbors.
3. Blending-predictors, that predict how well suited a particular pixel-predictor is to predict a pixel value, based on how well the pixel-predictor performed on the causal neighbors.

The parameters of the resulting model are the weights of the predictors.
TMW uses linear pixel-predictors of the form:

$$x = \sum_{i=1}^{M} w_i P\, v_i \tag{2.4}$$

where M being the number of causal neighbors used for the prediction and $P\, v_i$ is the pixel value of the i-th causal neighbor. w_i are the model parameters determined during the image analysis stage and their values are included in the first part of the encoded message. One of the key idea of TMW is to use not just one but multiple such pixel-predictors for each pixel. The idea behind this is that the correlation characteristics of a pixel with its causal neighbors typically are not constant over the whole image.

Trying to model all pixels with the same predictor would necessarily result in that predictor being sub-optimal for at least some areas of the image. TMW has a greater computational complexity than CALIC; the results are in any case surprising because of the following facts:

- Linear predictors are known to be not much effective in capturing the fast transitions in image edges.
- Global optimization seemed unable to improve substantially the performance of the lossless image compressor.
- CALIC was thought to achieve a data rate extremely close to the real entropy of the image.

2.5 Adaptive Linear Prediction Coding (ALPC)

ALPC is based on adaptive linear prediction and consists of two main steps, pixel prediction and entropy coding [12]. The input image is scanned from top to bottom, left to right, and the intensity of each pixel P IX(x, y) is predicted according to the weighted sum of its neighbors (or context) and rounded to the nearest integer value. Figure 2.5 shows the pixels that form the context of P IX(x, y). The context has a fixed shape and only the weights are allowed to change.

$$\mathbf{P}IX(x, y) = \text{int}(w_0 * P\,IX(x, y-2) + w_1 * P\,IX(x-1, y-1) + w_2 * P\,IX(x, y-1) + w3 * P\,IX(x+1, y-1) + w_4 * P\,IX(x-2, y) + w_5 * P\,IX(x-1, y))$$

After the prediction, an error ERR(x, y) (prediction error or residual) is calculated by subtracting the current pixel from its prediction as given below:

$$ERR(x,y) = PIX(x, y) - P\,IX(x, y) \tag{2.5}$$

and finally the prediction error is entropy coded and sent to the decoder.
If we encode the image in raster-scan order, with a top to bottom, left to right scan, the context will be composed of previously encoded pixels and the prediction error is sufficient enough for the decoder to make a reconstruction of the original pixel value. During the encoding process, the weights $w_0, \ldots, w_5$ are adaptively changed and optimized on a per pixel basis. Our intent is to determine the predictor's weights such that they are able to model local characteristics of the image being encoded. After several

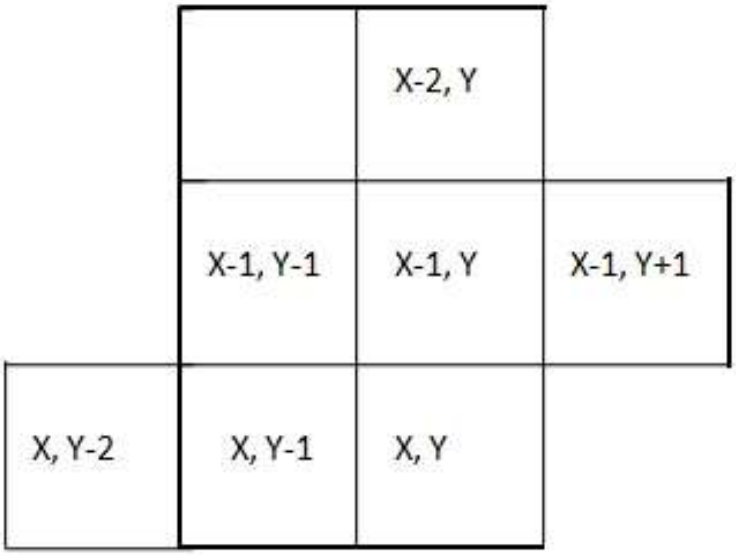

Figure 2.5: Context of the Pixel PIX(x,y) in ALPC

experiments, we decided to determine the predictor by optimizing the energy of the prediction error inside a small window of radius R_p centered on P IX(x, y), $W_{x,y}$ (R_p) as shown in Fig. 2.6.

$$\min_{w}0,...,w5\ ^{E(x,\ y)\ =\ \min}w0,...,w5\sum_{PIX(x',y')\in Wx,y(Rp)}(ERR(x',\ y'))^2 \qquad (2.6)$$

Using a window of previously encoded pixel, we can use a backward prediction scheme and the encoder has no need to send any side information to the decoder. On the other hand, the backward prediction has a well-known major drawback, i.e., poor performance in the presence of edges. The radius R_p of the window $W_{x,y}$ (R_p) as given in Fig. 2.6, is one of the essential features of this algorithm. Its size affects the prediction quality because if R_p is too small, only a few samples are in the window and the predictor "overspecializes" making big errors in presence of edges. On the other hand, too many samples in the window (R_p too big) tend to generate predictors that are not specific enough to remove local variations in the image. In this, the author decided to keep R_p constant and equal for all the images.

To improve the prediction, the optimization is performed only on a subset of samples collected in the window. The rationale is that we want the predictors' weights to be

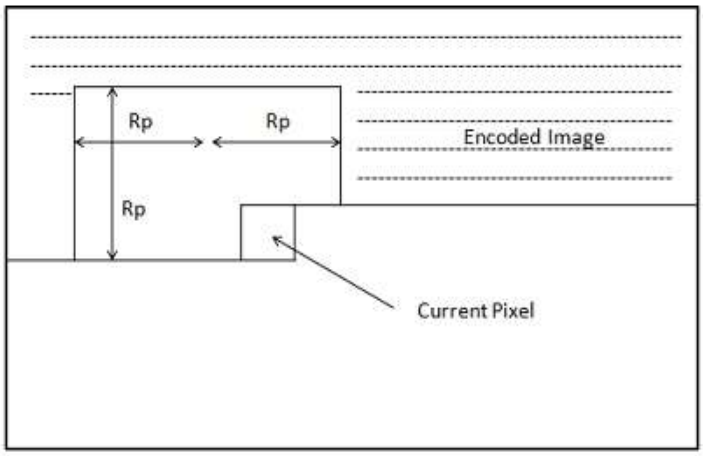

Figure 2.6: Frame Structure used in ALPC

Representative of the relation existing between the context and the pixel being encoded. By discarding the samples that have a context "too different" from the one of the current pixel, we can specialize the prediction and follow fine periodic patterns in the window. At each step of the optimization, while the difference between the previous and the current errors is smaller than a fixed threshold, the weights w_i of the predictors are changed. When only a few samples are in the window, for example when P IX(x, y) is close to the top or to the left border, a default fixed predictor is used in the prediction. $P_{def} = w_0 = 0$, $w_1 = -1$, $w_2 = 1$, $w_3 = 0$, $w_4 = 0$, $w_5 = 1$.
Adaptive linear prediction generates a skewed Laplacian distribution, centered at zero. It used Arithmetic encoder for the entropy encoding.

2.6 Edge Directed Prediction

In this method, we need to consider the N nearest causal neighbors in the prediction [13]. The ordering of the causal neighbor for n = 12, is given in Fig. 2.7.

$$X(n) = \sum_{k=1}^{N} a(k) * X(n-k) \tag{2.7}$$

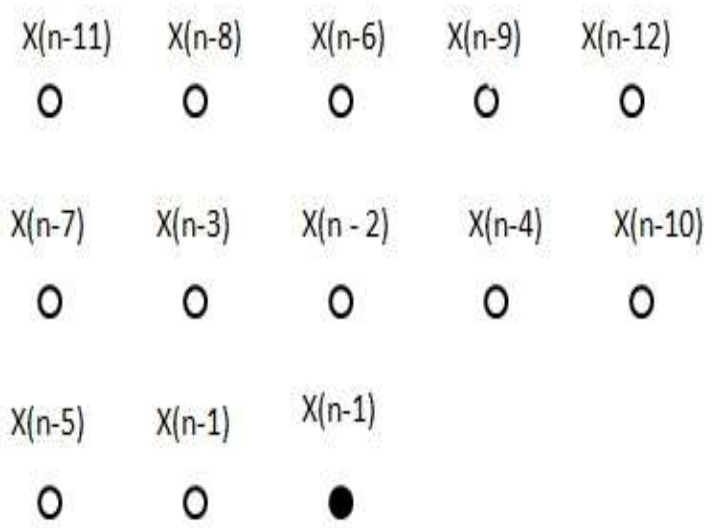

Figure 2.7: Ordering of the Causal Neighbors

LS-based adaptive prediction schemes provide an attractive alternative to achieve the orientation adaptation and approximate the optimal prediction. Instead of detecting the edge or estimating the orientation explicitly, LS-based approaches locally optimize the prediction coefficients inside a causal window (called "training window"). A convenient choice of the training window is the double-rectangular window, which containsM=2T(T+1)causalneighborsasshowninFig.2.8.Edge-directedproperty refersto the dominant role played by the pixels around an edge in the LS optimization process, which gives the name "Edge Directed Prediction"(EDP). To reduce the computational complexity of the edge directed predictor the author proposed that the prediction co-efficient optimized for a pixel around an edge are often also suitable for its neighbors along the same edge. The set of optimal predictors for an edge is the subset of the set of optimal predictors for the smooth regions. Therefore, the prediction coefficients optimized for an edge can be stored and repeatedly used until the scanning reaches the next edge event. In other words, the author wanted to perform the LS optimization on an edge-by-edge basis rather than on a pixel-by-pixel basis. The author proposed that if the amplitude of the prediction residue e(n) = X(n) − X(n) is beyond a pre-selected threshold th, the LS optimization is activated to update the prediction coefficients;

otherwise employ the stored coefficients to predict the next pixel.

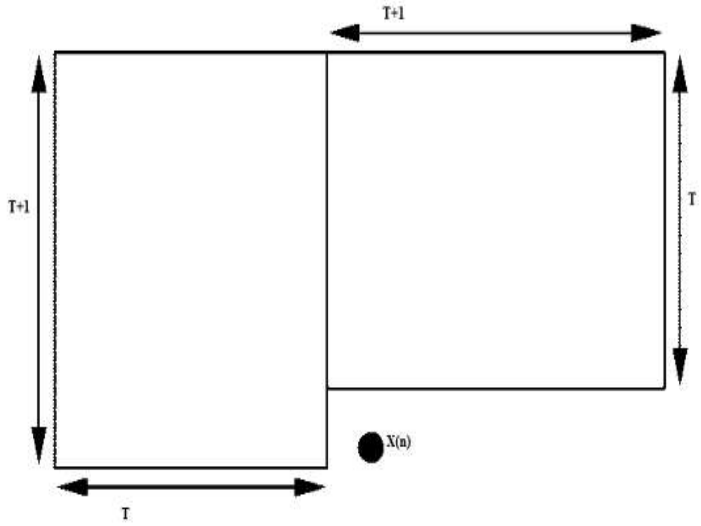

Figure 2.8: Training Window used to optimize the Prediction Coefficients

2.7 Discrete Cosine Transform based Coding

The Discrete Cosine Transform (DCT) is a technique for converting a signal into elementary frequency components. The DCT is related to the optimal KL transform in terms of energy compaction and there are fast DCT algorithms in 1 and 2 dimensions. These facts made DCT the most popular transform for image compression. The forward $2-D$ DCT is defined in [14]-[15]. It is widely used in image compression. The rapid growth of digital imaging applications, including desktop publishing, multimedia, teleconferencing, and high-definition television (HDTV) have increased the need for effective and standardized image compression techniques. The emerging standards are JPEG, for compression of still images; MPEG, for compression of motion video; and CCITT H.261 (also known as $P \times 64$), for compression of video telephony and teleconferencing. All three of these standards employ a basic technique known as the Discrete Cosine Transform (DCT). It is developed by Ahmed, Natarajan, and Rao [14]. The following is the general overview of the DCT process:

1. The image is broken into 8×8 blocks of pixels.

2. Working from left to right, top to bottom, the DCT is applied to each block.

3. Each block is compressed through quantization.

4. The array of compressed blocks that constitute the image is stored in a drastically reduced amount of space.

5. When desired, the image is reconstructed through decompression, the used process is known as the Inverse Discrete Cosine Transform (IDCT).

2.8 Discrete Wavelet Transform based Coding

In 1976, Crosier, Esteban, and Galand devised a technique to decompose the discrete time signals. In the same year, Crochiere, Weber and Flanagan did a similar work on coding of speech signals. They named their analysis scheme as sub-band coding. In 1983, Burt defined a technique very similar to sub band coding and named it pyramidal coding which is also known as multiresolution analysis. Later in 1989, Vetterli and Le

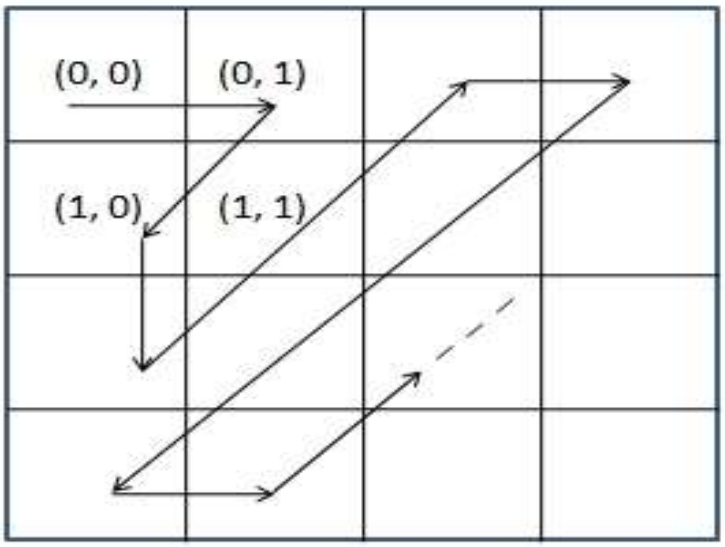

Figure 2.9: Zigzag Order in DCT

Gall made some improvements to the sub band coding scheme, removing the existing redundancy in the pyramidal coding scheme.
Filters are one of the most widely used signal processing functions. Wavelets can be realized by iteration of filters with rescaling. The resolution of the signal, which is a measure of the amount of detailed information in the signal, is determined by the filtering operations and the scale is determined by up-sampling and down-sampling operations.

2.8.1 Wavelet Transform

The DWT is computed by successive low-pass and high-pass filtering of the discrete time domain signal as shown in Fig. 2.10. This is called the Mallet algorithm or Mallet-tree decomposition. Its significance is in the manner it connects the continuous-time multi-resolution to discrete-time filters. In Fig. 2.11, the signal is denoted by the sequence x[n], where n is an integer. The low-pass filter is denoted by G_0, while the high-pass filter is denoted by H_0. At each level, the high-pass filter produces detail information, d[n], while the low-pass filter associated with the scaling function produces coarse approximations, a[n].
At each decomposition level, the half band filters produce signals spanning only half

the frequency band. For example, if a signal has a maximum of 1000 Hz component, then half band low-pass filtering removes all the frequencies above 500 Hz. This doubles the frequency resolution as the uncertainty in frequency is reduced by half. The unit of frequency is of particular importance at this time. In discrete signals, frequency is expressed in terms of radians. In accordance with SyQuest's rule, if the original signal has the highest frequency of 2ω radians, then it now has the highest frequency of $\frac{2}{}$ radians. It can now be sampled at a frequency of ω radians, thus discarding half the samples with no loss of information. This decimation by 2 halves the time resolution as the entire signal is now represented by only half the number of samples. Thus, while the half band low-pass filtering removes half of the frequencies and thus halves the resolution, the decimation by 2 doubles the scale.

With this approach, the time resolution becomes arbitrarily good at high frequencies, while the frequency resolution becomes arbitrarily good at low frequencies. The filtering and decimation process is continued until the desired level is reached. The maximum number of levels depends on the length of the signal. The DWT of the original signal is then obtained by concatenating all the coefficients, a[n] and d[n], starting from the last level of decomposition.

JPEG-2000 also uses the wavelet transform to decompose the data in the image into sub-bands. These sub-bands can be coded more efficiently than the original data due to their statistical properties. Even though the wavelet transform used by JPEG-2000 is one dimensional (1D-DWT) in nature, we obtain a two dimensional discrete wavelet transform (2D-DWT) by applying the 1D-DWT across the rows and columns of the image as a product separable transform.

1D-DWT gives a sequence x[n], we can obtain the 1D-DWT by filtering and sub sampling as depicted in Fig. 2.10. The input data is initially filtered using an analysis filter bank consisting of a high-pass filter, h[n] and low-pass filter, g[n]. The filtered signals are then sub sampled by a factor of 2 to give $s_0[n]$ and $d_0[n]$, the low-pass and high-pass sub-band sequences respectively.

$$s_0[n] = \sum_{k=-\infty}^{\infty} x[k]h[2n - k] \tag{2.12}$$

$$d_0[n] = \sum_{k=-\infty}^{\infty} x[k]g[2n - k] \tag{2.13}$$

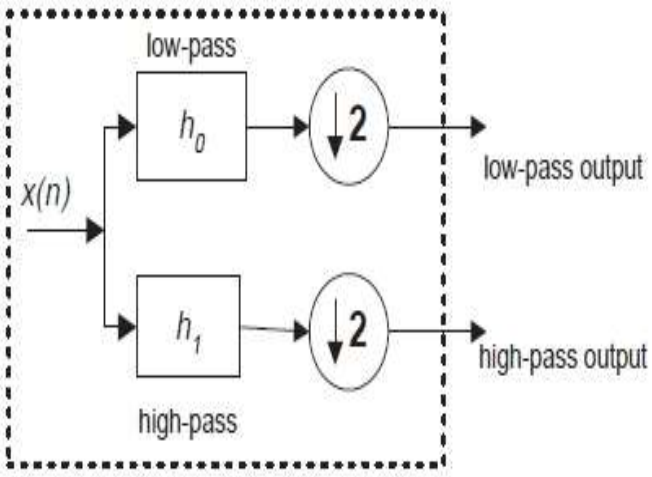

Figure 2.10: 1D- DWT Structure

The sub-sampling helps to maintain the sample rate constant. Thus, the forward discrete wavelet transform (DWT) decomposes a 1D sequence to give two sequences with half the number of samples in the original sequence.

2D-DWT in JPEG-2000, the number of levels of the 2D-DWT performed is implementation dependent. Figure 2.11 shows the filter bank structure used in each level of the pyramidal decomposition of an image. The filters h[n] and g[n] are one dimensional filters. We get four sub-bands (LL, LH, HL, and HH) from one level of the 2D-DWT. The LL sub band contains the low level details of the image. In the next level, the 2D-DWT of the LL sub band is obtained and this is repeated in each succeeding level. Figure 2.13 shows the reconstruction of the original signal from the wavelet coefficients.Basically, the reconstruction is the reverse process of decomposition. The approximation and detail coefficients at every level are up-sampled by two, passed through the low-pass and high-pass synthesis filters, and then added. This process is continued through the same number of levels as in the decomposition process to obtain the original signal. The Mallet algorithm works equally well if the analysis filters, G_0 and H_0, are exchanged with the synthesis filters, G_1 and H_1.

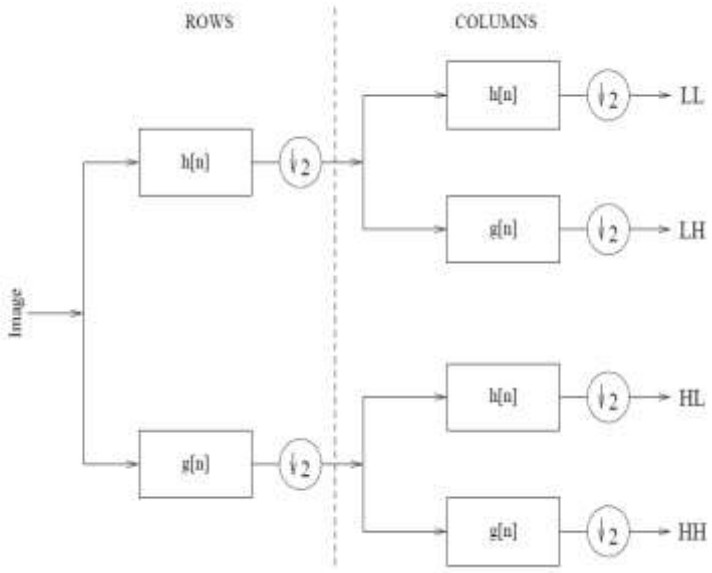

Figure 2.11: Filter Bank Structure used in the Wavelet Decomposition of an Image

2.8.2 Conditions for Perfect Reconstruction

In most of the Wavelet Transform applications, it is required that the original signal be synthesized from the wavelet coefficients. To achieve perfect reconstruction, the analysis and synthesis filters have to satisfy certain conditions. Let, $G_0(z)$ and $G_1(z)$ be the low-pass analysis and synthesis filters, respectively, and $H_0(z)$ and $H_1(z)$ be the high-pass analysis and synthesis filters, respectively. Then the filters need to satisfy the following two conditions as given in (2.14) and (2.15),

$$G_0(-z) \cdot G_1(z) + H_0(-z) \cdot H_1(z) = 0 \tag{2.14}$$

$$G_0(z) \cdot G_1(z) + H_0(z) \cdot H_1(z) = 2z^{-d} \tag{2.15}$$

The first condition implies that the reconstruction is aliasing-free and the second condition implies that the amplitude distortion has the amplitude of one. It can be observed that the perfect reconstruction condition does not change if we switch the analysis and synthesis filters.

There are a number of filters which satisfy these conditions. But not all of them give

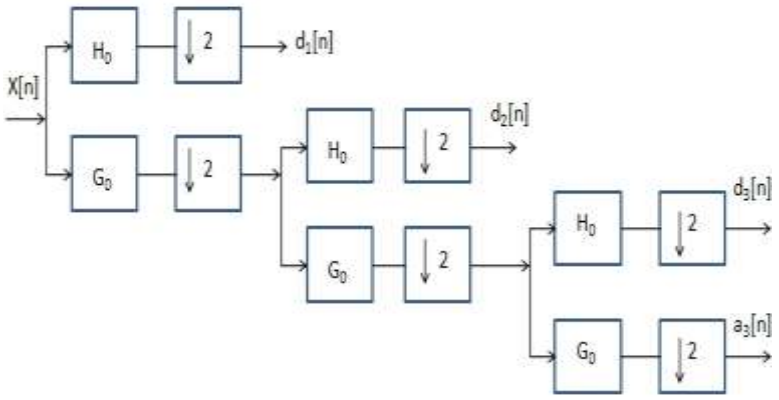

Figure 2.12: Wavelet Decomposition

accurate Wavelet Transforms, especially when the filter coefficients are quantized. The accuracy of the Wavelet Transform can be determined after reconstruction by calculating the signal to noise ratio (SNR) of the signal. Some applications like pattern recognition do not need reconstruction, and in such applications, the above conditions need not apply.

2.8.3 Integer Wavelet Transform

In many applications (e.g., image compression and processing), the input data consists of integer samples. In addition, the storage and encoding of integer number is easier compared to the floating point numbers. Unfortunately, all of the above transforms assume that the input samples are floating point values. They return floating point values as wavelet coefficients, even if the input values actually were integer. Rounding off the floating point values to integer values does not help because then we will loose the perfect reconstruction feature. Fortunately the lifting scheme can easily be modified to a transform that maps integers to integers and that is reversible, and thus allows a perfect reconstruction. This will be done by adding some rounding-operations at the expense of introducing a non-linearity in the transform [15]-[16].

The basic idea behind the lifting scheme is to exploit the correlation among the signal

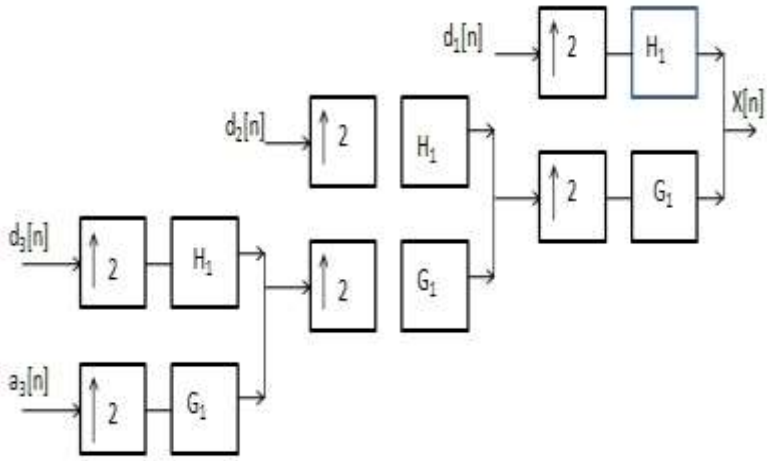

Figure 2.13: Reconstruction in Wavelet Transform

samples. The correlation is typically local in space and frequency, i.e., adjacent samples and frequencies are more correlated than ones that are far apart [17]. A typical lifting scheme consists of the three steps: Split, Predict, and Update [18].

Split: Let x[n] be the input signal. The input is first spited into even and odd indexed samples. The even indexed samples are $x_e[n] = x[2n]$ and the odd indexed samples are $x_o[n] = x[2n + 1]$. Typically the two sets are closely correlated. Thus, a prediction of a value in one set can be made by using the other set.

Predict: The odd indexed samples $x_o[n]$ can be predicted by a predictor P from the neighboring even indexed samples $x_e[n]$. The predictor P (x_e) is a linear combination of neighboring even indexed samples:

$$P(x_e)[n] = \sum_i P_i x_e[n + i] \qquad (2.16)$$

This is the "predict" step. This predictor need not be exact and the prediction error or detail d[n] is:

$$d[n] = x_0[n] - P(x_e)[n] \qquad (2.17)$$

Here, d[n] has high frequency coefficients after the wavelet transform. We can expect the first order entropy of d[n] to be smaller than that of $x_o[n]$, if the predictor P

is accurate. The operation of obtaining a prediction and the detail is known as the "lifting" scheme.

Update: For the even indexed samples $x_e[n]$, an "update" step by an update operator U is used to obtain smoothed values s[n] with the use of the detail:

$$s[n] = x_e[n] + U(d)[n] \tag{2.18}$$

where U (d)[n] is a linear combination of neighboring d[n] values:

$$U(d)[n] = \sum_i u_i d[n+i] \tag{2.19}$$

This is a second "lifting" step. s[n] has low frequency coefficients after the wavelet transform. The lifting scheme is invertible, thus no information is lost. The reconstruction algorithm follows the same structure as the lifting scheme, but in reverse order, i.e., $x_e[n]$ is first recovered by:

$$x_e[n] = s[n] - U(d)[n] \tag{2.20}$$

and then $x_o[n]$ can be obtained by:

$$x_o[n] = d[n] + P(x_e)[n] \tag{2.21}$$

One example of the IWT is the S transform [22], which is an integer version of the Haar transform. The forward S transform is:

$$\begin{aligned} d[n] &= x[2n+1] - x[2n] \\ s[n] &= x[2n] + \lfloor d[n]/2 \rfloor \end{aligned} \tag{2.22}$$

and the inverse S transform is:

$$\begin{aligned} x[2n] &= s[n] - \lfloor d[n]/2 \rfloor \\ x[2n+1] &= d[n] + x[2n] \end{aligned} \tag{2.23}$$

The S + P transform proposed by Said and Pearlman [18] performs a S transform first,

linearly predicts the low-pass coefficients $s_{1,l}$ and generates the high pass coefficients. The following is the forward S + P transform:

$$\begin{aligned} d^{1,l} &= s0,2l+1 - s_{0,2l} \\ s_{1,l} &= s_{0,2l} + \lfloor d^{1,l}/2 \rfloor \\ d_{1,l} &= d^{1}_{,l} + \lfloor {}^{8}(s_{1,l-1} - s_{1,l}) + {}^{8}(s_{1,l} - s_{1,l+1} + \lfloor {}^{8}d^{1}_{,l+1} + 2 \rfloor \end{aligned} \tag{2.24}$$

The inverse S + P transform is:

$$\begin{aligned} {}^{d1}_{,l} &= d1,l - \lfloor {}^{8}_{-}(s_{1,l-1} - s_{1,l}) + {}^{8}_{-}(s_{1,l} - s_{1,l+1} + \lfloor {}^{8}\,d^{1}_{,l+1} + {}^{2}_{-} \rfloor \\ s_{0,2l} &= s_{1,l} - \lfloor d^{1}_{,l} /2 \rfloor \\ s_{0,2l+1} &= d^{1}_{,l} + s0,2l \end{aligned} \tag{2.25}$$

Other examples of the IWT are given in [22]:

1. A (2,2) transform:

$$\begin{aligned} d[n] &= x[2n+1] - \lfloor {}^{2}_{-}(x[2n] + x[2n+2]) + {}^{2}_{-} \rfloor \\ s[n] &= x[2n] + \lfloor {}^{4}_{-}(d[n-1] + d[n]) + {}^{2}_{-} \rfloor. \end{aligned} \tag{2.26}$$

2. A (2+2,2) transform:

$$\begin{aligned} d^{(1)}[n] &= x[2n+1] - \lfloor {}^{2}_{-}(x[2n] + x[2n+2]) + {}^{2}_{-} \rfloor s[n] = x[2n] + \lfloor {}^{4}_{-}(d^{(1)}[n-1] + d^{(1)}[n]) + \\ {}^{1}_{-}2d[n] &= d^{(1)}[n] - \lfloor {}^{16}_{--}(-s[n-1] + s[n] + s[n+1] - s[n+2]) + {}^{2}_{-} \rfloor \end{aligned} \tag{2.27}$$

3. A (4,4) transform:

$$\begin{aligned} d[n] &= x[2n+1] - \lfloor {}^{196}_{--}(x[2n] + x[2n+2]) - {}^{1}_{-}16(x[2n-2] + x[2n+4]) + {}^{2}_{-} \rfloor s[n] = x[2n] + \\ \lfloor {}^{32}_{-}(d[n-1] + d[n]) &- {}^{32}_{-}(d[n-2] + d[n+1]) + {}^{2}_{-} \rfloor \end{aligned} \tag{2.28}$$

These IWTs are separable, thus they can be used on images row-wise and column-wise in any order. One level of the IWT generates 4 sub-bands, LL, LH, HL, and HH, as shown in Fig. 2.11.

2.8.4 Embedded Zero tree Wavelet (EZW) Coding

The EZW [19] algorithm was one of the first algorithms to show the full power of wavelet-based image compression. It was initiated by Shapiro, who combined the bit plane coding approach with the tree structure to code the wavelet coefficients of an image.

The EZW encoder is based on the progressive encoding to compress an image into a bit stream with increasing accuracy. This means that when more bits are added to the stream, the decoded image will contain more detail. Progressive encoding is also known as the embedded encoding, which explains the E in EZW. EZW is useful in progressive transmission of information, since the algorithm automatically selects those important coefficients form the background of the image while the less important coefficients add details to the image. The advantages of EZW are:

1. The algorithm may be stopped at any point to fit a given bit budget.
2. A lower quality image may be reconstructed with the limited amount of information on hand, obtained either from a limited bit budget or from disruption in the transmission channel.

The EZW is an entropy coder that makes use of the correlation between sub-bands of different resolutions. There are four symbols used in the EZW. A zero-tree-root (ZTR) designates an insignificant wavelet coefficient and all of its descendants. It is the symbol that produces the largest amount of compression. A POS/NEG symbol is a large-valued (significant) with a positive/negative sign. An IZ represents an insignificant wavelet coefficient (at a given threshold) with at least one significant descendant. IZ is the symbol that reduces coding efficiency. For every pass, a threshold is chosen against which all the wavelet coefficients are measured. If a wavelet coefficient is larger than the threshold, it is encoded and removed from the image; if it is smaller, it is left for the next pass. To arrive at a perfect reconstruction, the process is repeated after lowering the threshold, until the threshold has become smaller than the smallest coefficient to be transmitted. When all the wavelet coefficients have been visited, the threshold is lowered and the image is scanned again to add more detail to the already encoded image. This process is repeated until all the wavelet coefficients have been encoded completely or another criterion has been satisfied (maximum bit rate for instance).

2.8.5 Set Partitioning in Hierarchical Trees (SPIHT) Coding

The SPIHT algorithm is a highly refined version of the Embedded Zero-tree Wavelet (EZW) algorithm. It was designed and introduced by Said and Pearlman for still image compression [20]. It can perform better at higher compression ratios for a wide variety

of images than EZW. Set partitioning refers to the way the quad-trees partition the wavelet transform values at a given threshold. It provides the good image quality, high PSNR, especially for color images. It is optimized for progressive image transmission. It produces a fully embedded coded file. It uses simple quantization algorithm. It is a method of coding and decoding the wavelet transform of an image.
By coding and transmitting information about the wavelet coefficients, it is possible for a decoder to perform an inverse transformation on the wavelet and reconstruct the original image. A useful property of SPIHT is that the entire wavelet does not need to be transmitted in order to recover the image. Instead, as the decoder receives more information about the original wavelet transform, the inverse transformation yields a better-quality reconstruction (i.e., a higher PSNR) of the original image. It is completely adaptive algorithm which can code to exact bit rate or distortion protection. In this algorithm, pixels are evaluated in turn to see if they are larger than the threshold; if not, these pixels are considered insignificant. If a parent and all of its descendants are insignificant, then the coder merely records the parent's coordinates. Since the children's coordinates can be inferred from those of the parent, those coordinates are not recorded, resulting in a potentially great savings in the output bit stream. After locating and recording all the significant pixels for the given threshold, the threshold is reduced by a factor of two and the process repeats. By the end of each stage, all coefficients that have been found to be significant will have their most significant bits recorded.

2.8.6 JPEG-2000

JPEG-2000 [23] offers several new features and improvements over the older JPEG standard. Besides supporting both the lossless and the lossy compression, JPEG-2000 offers superior performance even at low bit rates. It has good error resilience properties, thus making it useful in Internet and mobile applications. One of the key features supported by JPEG-2000 is the region of interest coding scheme, which allows the regions of interest in the image to be reconstructed at the receiver side with lossless quality.
JPEG-2000 is a wavelet transform based codec and the various stages involved are shown in the block diagram given in Fig. 2.14.
The pre-processing stage in the JPEG-2000 codec consists of three sub stages, as shown

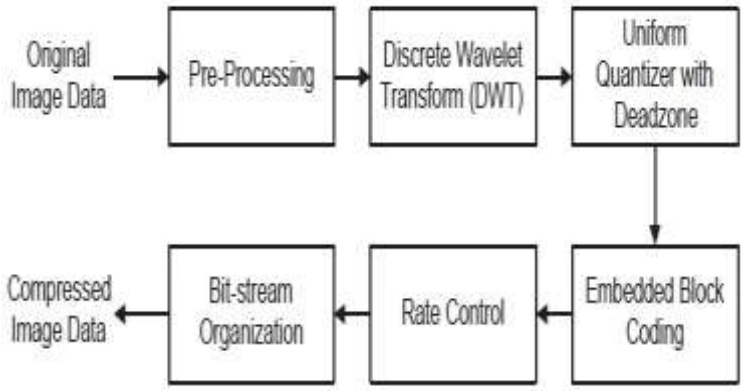

Figure 2.14: Block Diagram of JPEG-2000 Encoder

in the block diagram of Fig. 2.15.

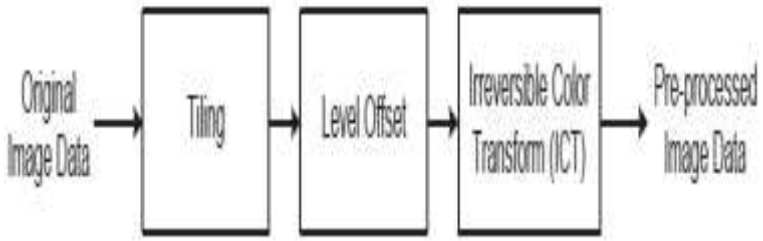

Figure 2.15: Preprocessing Substages

If the image to be encoded is larger than the memory available in the encoder, JPEG2000 allows optional tiling. In the tiling stage, the image is broken down to nonoverlapping tiles of smaller equal sizes.

Since JPEG-2000 uses high-pass filtering, the input data should have dynamic range around zero. Thus, the unsigned image values have to be offset by a dc value to get signed integers centered around zero. For example, in a 8-bit grayscale image, the

pixels have values in the range $0 - 255$ (0 to $2^8 - 1$). In the pre-processing stage, all the values are offset by -128 (-2^7). Thus, after level offset, the pixel values are in the range -128 to 127.

Quantization is a process to reduce the precision of the coefficients, thus helping us to obtain higher compression ratios. All the coefficients obtained after applying the wavelet transform are quantized using a uniform quantizer dead zone. In each sub-band b, all the coefficients are quantized using a basic step size δ_b. The quantizer step size for a particular sub-band is chosen depending on the significance of the sub-band in the final reconstruction of the image. After choosing the quantizer step size, the quantization rule given in (2.29) is applied to each coefficient in that sub-band.

$$Q(x) = \text{sign}(x)\lfloor |(x)|/\delta_b \rfloor \tag{2.29}$$

where, x is the input coefficient, δ_b is the quantizer step size, |x| is the absolute value of x, sign(x) is the sign of the coefficient x, $\lfloor x \rfloor$ represents the largest integer less than x and Q(x) is the output of the quantizer. The dead zone quantizer structure is shown in Fig. 2.16.

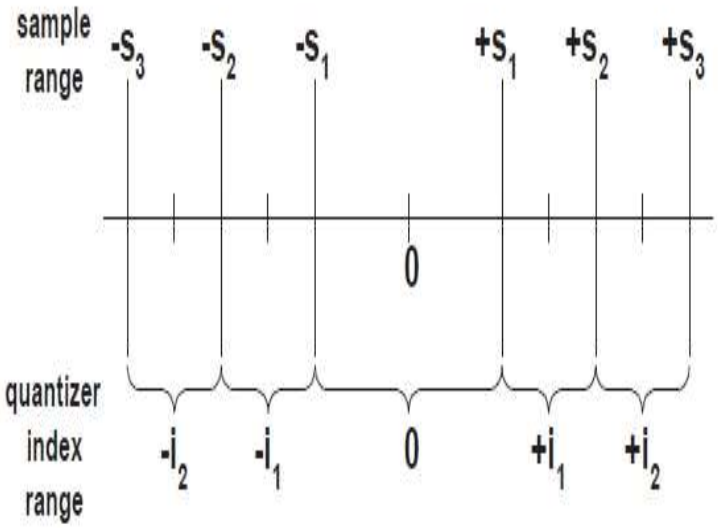

Figure 2.16: Uniform Quantizer with Dead Zone

The quantization range around 0 is $2\delta_b$, which is called as the dead zone. For example, let $x = -32.64$ and the quantizer step size be 10, then $Q(x) = -\lfloor 32.64/10 \rfloor = -3$. Any

value of x between -10 and +10 falls in the dead zone giving the quantizer output 0. One of the most important properties of the wavelet transform is that the energy present in the input signal is usually compacted in a very small number of wavelet coefficients. This compaction of energy can bring about a large degree of compression if we code only the high energy coefficients. The disadvantage of this method is that the position of the coefficient also need to be sent for proper decoding, and the position information will utilize a significant portion of the bit rate. JPEG-2000 uses a method known as embedded block coding to lower the cost of encoding the significant coefficient position. In JPEG-2000, each quantized sub-band is broken down into non-overlapping rectangles of equal size with height and width equal to the powers of 2 (e.g.: 64 × 64). These non-overlapping rectangles are called as code blocks. Each code block is then encoded separately using the embedded block coding algorithm explained in [23]. In this algorithm, each bit plane in a code block is encoded in three sub bit plane passes instead of single pass. The set of three passes generates an embedded code for each bit plane of the code block and is compressed using a context dependent binary arithmetic encoder. Thus, embedded codes generated by several code blocks for each bit plane from the quality layers. Each additional layer improves the quality of the image reconstructed at the receiver side.

The JPEG-2000 decoder does the opposite of the actions done by the encoder, giving the reconstructed image with losses or without losses depending on the encoding scheme followed. The code-stream at the receiver side is passed through the arithmetic decoder, inverse quantizer, inverse wavelet transform and inverse color transform blocks. The reconstructed image may not be lossless. The losses are introduced by the quantization of the wavelet coefficients.

2.9 Conclusions

In this chapter, we have discussed various lossless image compression methods. Some methods are based on prediction and some methods are based on transform. Though there are number of compression methods available, the need for improved performance demands newer and better techniques. We have also found that the transform based methods are progressive from lossy to lossless. So, these methods are more complex than prediction based methods.

3

Modified Gradient Adjusted Predictor (MGAP)

3.1 Introduction

Prediction is used to decor relate the image pixels with their neighborhood. In the literature survey [9], it has been shown that prediction is worthwhile in the state-of-the-art lossless image compression methods as JPEG-LS [7] and CALIC [10]. The literature survey has shown a number of ways for predicting the value of the current pixel based on their spatial correlation with neighboring pixels [9]. It is well known that if the data is decor related with the appropriate predictor, then the resultant structure of error residue will be independent and identically distributed [24]. This error residue can be encoded optimally into a binary sequence by using any of the standard variable length coding techniques like Huffman coding [3] or Arithmetic coding [4],[5]. If by any means we are able to decor relate the image pixels better than the existing prediction methods, which are used in the state-of-the-art method, then the error distribution will be more laplacian, i.e., peak at zero. This will lead to better compression performance as compared to existing state-of-the-art lossless compression methods like JPEG-LS [7] and CALIC [10].

In 2007, Anil Kumar Tiwari et. al. [25], explained a method for lossless compression of videos. In this paper, they proposed to find a LS-based predictor for each slope bin. When this predictor is associated with CALIC framework, there is a significant improvement in the compression ratio. At encoder side, LS based predictor has higher

3. MODIFIED GRADIENT ADJUSTED PREDICTOR (MGAP)

coding complexity. In 2008, Anil Kumar Tiwari et. al. [26] described a method for lossless compression of images. In this paper, they tried to find the LS-based predictor which are statistically optimal for each of the slope bin. In 2011, Aleksey et. al. [27] presented a method in which they described the prediction based on the edge detection and estimation of local gradients. So, in the recent past, to improve the compression performance, lots of prediction techniques have been used as described in the literature. Because of the above given reason, in this work, we are proposing a new prediction method which will predict the current pixels better than the previous existing methods as GAP [10] and MED [7]. GAP [10] and MED [7] are used as the prediction method in the state-of-the-art lossless compression method as in CALIC [10] and JPEG-LS [7], respectively.

In the proposed algorithm, we concentrate on one particular type of adaptive prediction which is performed by switching among a finite number of predictors. The decision of which predictor to use for a particular bin in an image, is based on the minimum entropy. The predictor will be obtained on an image basis.The predictor associated with the slope bins are obtained offline.

In the proposed work, we will make a pool of all the seven predictors which are used in GAP. For each slope bin in a given image, to find a representative predictor, we will switch among the given predictors and will find the predictor, which is giving the minimum entropy among seven. In this way, we will select the predictor that gives the minimum entropy in place of the default predictors used in GAP. Simulation results of the proposed Modified Gradient Adjusted Predictor (MGAP) algorithm have shown a significant improvement in the zero-order entropy as compared to the GAP and MED. Determining a best predictor for a particular slope bin, among the default predictors, for a given image is a computationally complex process. But, The computational complexity of the proposed method, excluding the training process, is of the same order as that of GAP. We have determined the best predictor for each slope bin in an image. We can construct a predictor map. If we store this predictor map during encoding, the best possible predictor for each slope is send to the decoder side. Therefore, at the decoder side it will perform faster than GAP. In sending the representative predictor, i.e., prediction map at the decoder side will include very less overhead. This represents the ideal forward adaptive prediction selection scheme.

3.2 Proposed Work

In case of the images, we observed that the characteristics of the images are different from one image class to others. For example, in case of the medical images as shown in Fig. 3.1, the region is very smooth, i.e., there is not much intensity variation in the neighboring pixels. But, in case of the texture images, there are lots of strong edges, i.e., the intensity of the neighboring pixels are changing rapidly as shown in Fig. 3.2. The GAP predictor has a default predictors for each slope bins as given in Table 3.1. If the image characteristics are changing from one image class to others then it is assumed that the predictors which are used to predict the image pixels should be different.

For example, as given in Table 3.2, we have selected the prediction context in case of

Table 3.1: Slope bin classification and associated predictor in case of GAP

Slope bin	Range	Predictor Coefficients			
		a(1)	a(2)	a(3)	a(4)
S1	S >80	1.0000	0.0000	−0.0000	0.0000
S2	S <−80	0.0000	1.0000	−0.0000	0.0000
S3	32 < S ≤ 80	0.7500	0.2500	−0.1250	0.1250
S4	8<S ≤32	0.6250	0.3750	−0.1875	0.1875
S5	−80 ≤ S < −32	0.2500	0.7500	−0.1250	0.1250
S6	−32 ≤ S < −8	0.3750	0.6250	−0.1875	0.1875
S7	−8 ≤ S ≤ 8	0.5000	0.5000	−0.2500	+0.2500

the finger print image named 101_1 (28). We have randomly selected this image from the image database (28). To predict the current pixel 181, we have calculated the slope with the help of calculating its d_v and d_h, as described in chapter 2. We have calculated the value of its slope after calculating its d_v -d_h. According to Table 3.1, current pixel's slope value lies in slope bin 3. If we use the default predictor, which is associated with the slope bin 3, the predicted value will be 169. In this case, the prediction error which will be calculated as $e_i = x_i - x_i$ will be 181 − 169, i.e., 12. However, if we predict the current pixel by its west pixel, the predicted value is 181 and the prediction error will be 181 − 181 i.e., zero. Therefore, we can say that if we predict these type of pixels with the west predictor as compare to the default predictor, then we will find more zeros, i.e., the prediction error distribution will be more laplacian (i.e., peak at zero).

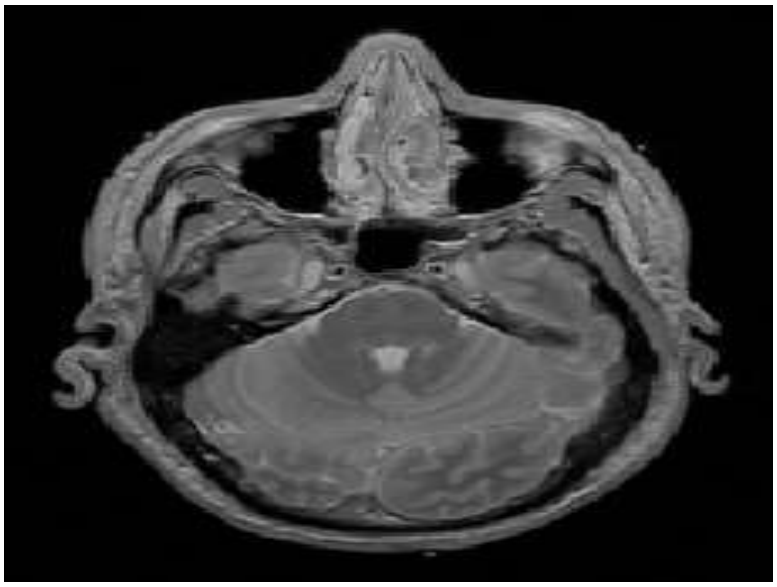

Figure 3.1: Medical Image

Similarly, as given in Table 3.3 to predict the current pixel, we have calculated the value of d_v and d_h, and found that this pixel's slope value lies in slope 5. If we predict the current pixel by its default predictor as given in Table 3.1, the predicted value will be 141. But, if we predict the current pixel by its north pixel, then the predicted value will be 145 and residual error value will be zero.

From the above examples, we can say that the default predictors which are associated with that slope bin in case of GAP, will always not predict the current pixel optimally. Different images can have different structures, which can be best exploited by one of the seven predictors for each bin. For each bin all seven predictors can be tried and which gives the minimum entropy will be used for prediction. This leads to simple adaptive predictive coding schemes.

Table 3.2: Neighborhood pixels in case of finger print image

127	128	128	127
138	135	135	128
176	181	181	176
184	189	184	181

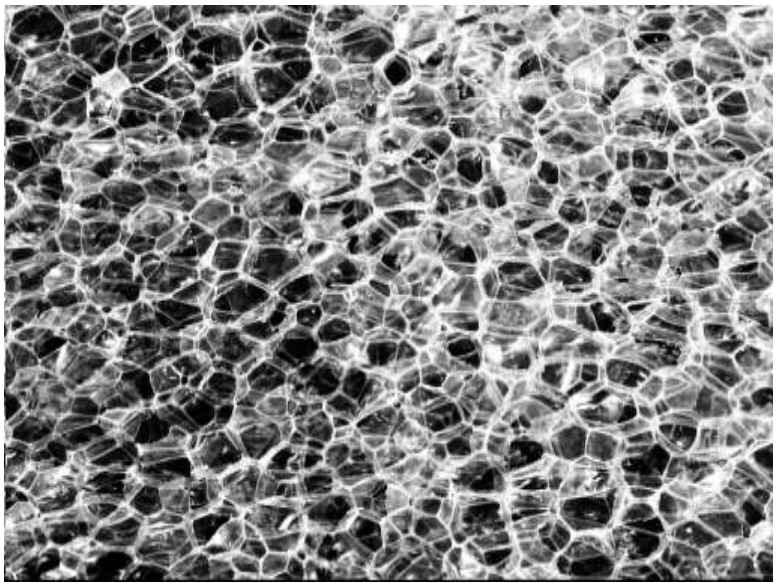

Figure 3.2: Texture Image

Table 3.3: Neighborhood pixels in case of finger print image

110	124	142	136
115	120	145	139
111	121	145	138
104	113	137	131

From the exhaustive search we are able to select the best predictor among the given predictors, which will work better than the default predictor. Determining the best predictor for a particular slope bin, among the default predictors, for a given image is a computationally complex. If we determine the best predictor for each slope bin in an image, we can construct a predictor map. If we store this predictor map during encoding, the best possible predictor (given the available predictors) can be send to the decoder side for reconstruction. This represents the forward adaptive prediction scheme. This will reduce the complexity at the decoder side.

The algorithm which is used for prediction in the state-of-the-art lossless image compression method, i.e., in CALIC is explained in chapter 2. In Table 3.1, the value of the slope bins and associated predictor for that particular bin is given.

To modify the GAP algorithm we proposed a new prediction method named as Modified

GAP (MGAP).

3.3 Modified Gradient Adjusted Predictor (MGAP)

The MGAP algorithm is expected to result in better prediction than that is obtained by using the conventional GAP. This intuition is based on the fact that the optimize set of predictors for object of the given image is expected to perform better than that can be obtained using default predictors of GAP. In other words, the main objective of MGAP is to find image-dependent representative predictor for each of the slope bins. We propose an algorithm to modify GAP named as M-GAP as follows.

1. We make a pool of all the seven predictors of GAP. These predictors are given in Table 3.1.
2. For a given image, select all the pixels belonging to a particular slope bin. The bin boundaries are given in Table 3.1.
3. Apply one-by-one all these seven predictors on the pixels identified in step 2 and find corresponding prediction error. In this way, we have seven set of source of prediction errors obtained due to application of each of the predictors.
4. Sources obtained in step 3. The entropy can be obtained using the following equation as:

$$H = -\sum_{i=1}^{n} P_i \log_2 (p_i) \qquad (3.1)$$

   where, p_i is the probability of the i^{th} symbol and n is the number of symbols in the source.
5. The predictor that results into minimum entropy is associated with the bin will be selected as a predictor for that particular bin.
6. Repeat the steps from 1 to 5 for all the seven slope bins.

The above process results into the optimal set of predictors, obtained from the set of GAP predictors, for the given image. We have shown a set of predictors for a finger print image 101 1 [28] in Table 3.4. It may be interesting to note that these coefficients are significantly different from the default predictors given in Table 3.1. Since predictors are

significantly different, the performance improvement is also expected to be significant.

Table 3.4: MGAP coefficients for finger print image_101 1 [28]

Slope bin	a(1)	a(2)	a(3)	a(4)
S1	1.0000	0.0000	−0.0000	0.0000
S2	0.000	1.0000	−0.0000	0.0000
S3	1.0000	0.0000	−0.0000	0.0000
S4	0.7500	0.2500	−0.1250	0.1250
S5	0.5000	0.5000	−0.2500	+.2500
S6	0.5000	0.5000	−0.2500	0.2500
S7	0.5000	0.5000	−0.2500	0.2500

We can also verify that in different class of images, which are selected randomly from the database, the MGAP predictor coefficents are different from the default coefficients given in Table 3.1.

Table 3.5: MGAP coefficients for aerial image 2.1.01 [29]

Slope bin	a(1)	a(2)	a(3)	a(4)
S1	0.625	0.3750	−0.1875	0.1875
S2	0.625	0.3750	−0.1875	0.1875
S3	0.625	0.3750	−0.1875	0.1875
S4	0.625	0.3750	−0.1875	0.1875
S5	0.625	0.3750	−0.1875	0.1875
S6	0.625	0.3750	−0.1875	0.1875
S7	0.625	0.3750	−0.1875	0.1875

3.4 Experimental Results

In this work, the performance of the proposed method is studied on different class of images like PCB, texture, aerial, finger print, cartoon and medical images. Its performance is compared against the MED and GAP method. From the results shown in the tables we can say that the proposed method performs consistently better than GAP and MED.

Table 3.6: MGAP coefficients for texture image 1.3.03 [30]

Slope bin	a(1)	a(2)	a(3)	a(4)
S1	0.7500	0.2500	−0.1250	0.1250
S2	0.5000	0.5000	−0.2500	0.2500
S3	1.000	0.0000	−0.0000	0.0000
S4	0.5000	0.5000	−0.2500	0.2500
S5	0.5000	0.5000	−0.2500	0.2500
S6	0.5000	0.5000	−0.2500	0.2500
S7	0.5000	0.5000	−0.2500	0.2500

Table 3.7: MGAP coefficients for PCB image

Slope bin	a(1)	a(2)	a(3)	a(4)
S1	0.2500	0.7500	−0.1250	0.1250
S2	1.000	0.0000	−0.0000	0.0000
S3	0.7500	0.2500	−0.1250	0.1250
S4	0.5000	0.5000	−0.2500	0.2500
S5	0.0000	1.0000	−0.0000	0.0000
S6	0.5000	0.5000	−0.2500	0.2500
S7	0.5000	0.5000	−0.2500	0.2500

In order to test our algorithm, we have used different class of images like PCB images, aerial, texture, finger print, and cartoon images. All the images are gray scale and are of different size. The images which are used for experimental results are randomly selected from the database.

The MGAP is only little more expensive than the original GAP. This is so because first it will find the most appropriate coefficients for each slope. After that these coefficients are used to predict the current pixel.

MGAP replaces the default predictors of GAP with image dependent predictors in such a way that the entropy is minimum. Not only in case of the medical images, we also have applied this MGAP algorithm in several other image classes like PCB, sky, aerial, texture, finger print, and cartoon images.

Table 3.8: Zero-order entropy on the PCB images and compared with MED and GAP

Image Name	Size	MED	GAP	Proposed
1	960 × 960	4.29	4.22	4.14
2	960 × 960	4.40	4.41	4.30
3	960 × 960	4.38	4.39	4.29
4	960 × 960	4.33	4.42	4.29
5	960 × 960	4.64	4.65	4.56
6	960 × 960	4.56	4.56	4.45
7	960 × 960	4.54	4.53	4.44
8	960 × 960	4.33	4.33	4.24
9	960 × 960	4.31	4.24	4.16
10	960 × 960	4.50	4.48	4.36
	Average	4.43	4.42	4.32

Table 3.9: Zero-order entropy on the aerial images and compared with MED and GAP

Image Name	Size	MED	GAP	Proposed
2.1.01	512 × 512	6.15	6.06	5.99
2.1.02	512 × 512	6.40	6.37	6.33
2.1.03	512 × 512	4.62	4.51	4.48
2.1.04	512 × 512	5.97	5.90	5.87
2.1.05	512 × 512	5.40	5.35	5.33
	Average	5.71	5.64	5.60

3.5 Conclusions

This chapter describes the predictive lossless image compression process. To reduce the statistical redundancy between the image pixels, we have proposed a new algorithm. This algorithm selects a candidate predictor for each of the slope bin which gives minimum entropy as compared to other predictors for that particular slope bin. The proposed algorithm gives better compression performance compared to the state-of-the-art methods like GAP and MED. We have observed that if there will be more number of predictors, and we have to select the candidate predictor among them, then it will give better results. However, if there are more number of predictors, then finding an appropriate predictor will be computationally complex process. From the results

Table 3.10: Zero-order entropy on the cartoon images and compared with MED and GAP

Image Name	Size	MED	GAP	Proposed
bear_2	512 × 512	3.74	3.69	3.70
buf f alo_2	512 × 512	4.69	5.11	4.67
elephant_2	512 × 512	3.25	4.02	3.24
goat_2	512 × 512	4.74	5.18	4.83
rino_2	512 × 512	4.88	4.83	4.82
	Average	4.26	4.57	4.25

Table 3.11: Zero-order entropy on the finger print images and compared with MED and GAP

Image Name	Size	MED	GAP	Proposed
101_1	300 × 300	4.40	4.33	4.25
101_2	300 × 300	4.54	4.48	4.43
101_3	300 × 300	4.25	4.15	4.08
101_4	300 × 300	3.91	4.00	3.83
101_5	300 × 300	4.69	4.55	4.50
	Average	4.36	4.30	4.22

it can be verified that the proposed algorithm on an average gives better or similar performance on a chosen set of images when compared with MED and GAP.

Table 3.12: Zero-order entropy on the texture images and compared with MED and GAP

Image Name	Size	MED	GAP	Proposed
1.2.12	512 × 512	7.16	7.15	7.12
1.2.13	512 × 512	6.63	6.63	6.61
1.3.01	512 × 512	5.72	5.63	5.59
1.3.02	512 × 512	5.58	5.46	5.42
1.3.03	512 × 512	5.42	5.17	5.12
	Average	6.10	6.00	5.97

3. MODIFIED GRADIENT ADJUSTED PREDICTOR (MGAP)

4

Conclusions and Future Work

4.1 Conclusions

In this thesis, an efficient predictive coding algorithm for images has been presented. In this work we attempted to find the statistically optimal predictor for each of the slope bins in terms of entropy. The predictor that results in the minimum zero order entropy is chosen to represent the slope bin. The proposed algorithm gives better compression performance on different class of images like PCB, texture, aerial, finger print, cartoon and medical images. The new prediction technique performs better than the prediction method used in CALIC and JPEG-LS. The second piece of works described a method for medical images. Mostly, medical images consist of a significant amount of background region which is very smooth while the foreground region i.e., object region consists of details. We separated out these two regions and apply different prediction algorithm on these two characteristically different regions. The proposed algorithm gives better compression performance as compared to MED and GAP. We also have described the context-based adaptive lossless image compression technique for mammogram images. We have used GAP as a structural predictor. After the prediction, it is corrected by context modeling derived by using the conditional probability. We have also modified the error energy bins. The complete algorithm gives higher performance as compared to state-of-the-art lossless image compression methods.

4.2 Future Work

Currently, In the proposed prediction algorithm our method uses an offline training process to obtain the representative predictor for each slope bin. Although the compression performance on images outside the training set is attractive. But the training is computationally a complex process. In future we will try to reduce the complexity in training process. In the future work, we also want to design some heuristic based on the characteristics of the image, to further refine the prediction error. So that,error pixels of the same context can be grouped into a similar bin. We expect that the method can adaptively do the context quantization without maintaining a preset threshold for error energy bins. There is still enough space to improve the code efficiency and reduce the computing and storage complexity. Finally, extending the work to the video compression would also be very interesting.

5. References

[1] C. E. Shannon, "A mathematical theory of communications: Parts I and II", Bell System Technology Journal, Vol. 27, pp. 379423, 623656, Octo- ber 1948.

[2] Rafael C. Gonzalez and Richard E. Woods, "Digital Image Processing", 3rd Edition (DIP/3e), Prentice Hall, 2008.

[3] D. A. Huffman, "A method for the construction of minimum redundancy codes", Proceedings of the IRE, Vol. 40, no. 9, pp. 10981101, September 1952.

[4] J. Rissanen, "Generalized kraft inequality and arithmetic coding", IBM Journal on Research and Development, Vol. 20, no. 3, pp. 198203, May 1976.

[5] I. H. Witten, R. Neal, and J. M. Cleary, "Arithmetic coding for data compression", Communications of the ACM, Vol. 30, no. 6, pp. 520540, June 1987.

[6] S. W. Golomb, "Run-length encodings", IEEE Transactions on Information Theory, Vol. IT-12, pp. 399401, July 1966.

[7] M. J. Weinberger, G. Seroussi, and G. sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS", IEEE Trans. Image Processing, Vol. 9, No.8, pp. 1309-1324, Aug, 2000.

[8] M. J. Weinberger, G. Seroussi, and G. Sapiro,"A Low Complexity, Context-Based, Lossless Image Compression Algorithm", Proc. of the Data Compression Conference, Snowbird, Utah, March, 1996.

[9] M. Alwani, and A. K.Tiwari,"A Survey on Lossless Image Compression Methods", IEEE, International Conference on Computer Engineering and Technology, April 2010, Chengdu, China.

5.REFERENCES

[10] X. Wu and N. Memon, "Context-based, adaptive, lossless image coding", IEEE Trans. Commun., Vol. 45, No. 4, pp. 437-444, Apr. 1997.

[11] B. Meyer and P. Tischer, "TMW- a New Method for Lossless Image Compression", Proc. of Picture Coding Symposium (PCS), Berlin, Germany, Oct, 1997.

[12] G. Motta, J. A. Storer, and B. Carpentieri, "Lossless image coding via adaptive linear prediction and classification", Proceedings of the IEEE, Vol. 88, No.11, pp. 1790-1796, 2000.

[13] Xin Li, Michael T. Orchard, "Edge Directed Prediction for Lossless Compression of Natural Images", IEEE Transactions on image processing, Vol. 10, issue 6, pp.813-817, June 2001.

[14] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform", IEEE Transactions on Computers, Vol. 23, no. 23, pp. 9093, January 1974.

[15] K. R. Rao and P. Yip, Discrete Cosine Transform - Algorithms, Advantages, Applications. New York: Academic Press, 1990.

[16] A. Bilgin, P. J. Sementilli, F. Sheng, and M. W. Marcellin, "Scalable image coding using reversible integer wavelet transforms", IEEE Transactions on Image Processing, Vol. 9, no. 11, pp. 1972 1977, November 2000.

[17] I. Daubechies and W. Sweldens, "Factoring wavelet transform into lifting steps", Journal of Fourier Analysis and Applications, Vol. 4, no. 3, pp. 247269, 1998.

[18] A. Said and W. A. Pearlman, "An image multiresolution representation for lossless and lossy image compression", IEEE Transactions on Image Processing, Vol. 5, pp. 13031310, September 1996.

[19] J. M. Shapiro, "Embedded Image Coding using zero trees of wavelet coefficients", IEEE transactions on signal processing, Vol. 41, No. 12, pp. 3445-3462, Dec 1993.

[20] A. Said and W. A. Pearlman,"A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", IEEE Trans. on Circuit and systems for video Technology, Vol. 6, no.3, pp.243-250, June 1996.

5.REFERENCES

[21] Michael D. Adams and rabab Ward,"Wavelet transform in JPEG 2000 standard", In Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 3, pp. 1749-1752. IEEE, May 2001.

[22] A. R. Calderbank, I. Daubechies, W. Sweldens, and B. L. Yeo, "Wavelet transforms that map integers to integers", Applied and Computational Harmonic Analysis, Vol. 5, no. 3, pp. 332369, July 1998.

[23] ITU-T, ITU-T Recommendation T.800: CD15444-1, V1.0, December2000. (JPEG2000).

[24] X. Wu, "Efficient and effective lossless compression of continuous tone images via context selection and quantization", IEEE Trans. Image Processing, Vol. 6, pp., 656-64, 1997.

[25] Anil Kumar Tiwari and R. V. Raja Kumar," Least-Squares Based Switched Adaptive Predictors for Lossless Video Coding", ICIP, Vol. 6, pp. 69-72, 2007.

[26] A. K. Tiwari and R. V. Raja Kumar, "A minimum entropy based switched adaptive predictor for lossless compression of images", Springer Journal on Signal, Image and Video Processing, Vol. 3, no. 4, December, 2009.

[27] Aleksej Avramovic and Salvica Savic,"Lossless Predictive Compression of Medical Images", Serbian Journal of Electrical Engineering, Vol. 8, no.1, pp. 27-36, Feb. 2011.

[28] http://bias.csr.unibo.it/fvc2004/download.asp

[29] http://sipi.usc.edu/database/database.php

[30] http://www.cs.rug.nl/imaging/databases/contour-database/contour-2.html

[31] N. Memon, K. Xuan and J. Cinkler, "Context-based lossless and near-lossless compression of EEG signals", IEEE Trans. Inform. Technol. Biomed., pp. 231238, 1999.

5.REFERENCES

[32] A. Singla, and A. K. Tiwari,"A Novel Method for Lossless Compression of Medical Images by Seperating Background and Object Regions", IEEE, International Conference on Advance Computing and Communication Technologies, Oct. 2010, Panipat, Haryana.

[33] Ravi Kumar, Shivprakash Koliwad and G.S. Dwarakish, "Lossless Compression of Digital Mammography using Fixed Block Segmentation and Pixel Grouping", Indian conf. on Computer Vision, Graphics and Image Processing, 2008.

[34] R. Ashraf and M. Akbar, "Absolutely lossless compression of medical images", 27th Annual Conference Proceedings of the IEEE Engineering in Medicine and Biology, 2005.

[35] http://peipa.essex.ac.uk/ipa/pix/mias/

[36] K. R. Namuduri, N. Ranganathan, Hooman Rashedi, "SVBS: A High-resolution medical image compression algorithm using slicing with variable block size segmentation", IEEE proceedings of ICPR,1996, pp 919-923.

[37] Y. Huang, H. M.Dreizen and N. P. Galatsanos," Prioritized DCT for compression and progressive tansmission of Images", IEEE trans. On Image Proc. (IP), Vo1.2, No.4, pp.477-487, 1992.