# LOSSLESS AND NEARLY LOSSLESS DIGITAL VIDEO CODING

submitted by

Guruge Charith Kanchana Abhayaratne

for the degree of

Doctor of Philosophy

of the

University of Bath

2002

**COPYRIGHT**

Attention is drawn to the fact that copyright of this thesis rests with its author. This copy of the thesis has been supplied on the condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the prior written consent of the author.

This thesis may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purposes of consultation.

Signature of Author ..........................................................

Guruge Charith Kanchana Abhayaratne

# ABSTRACT

In lossless coding, compresssion and decompression of source data result in the exact recovery of the individual elements of the original source data. Lossless image / video coding is necessary in applications where no loss of pixel values is tolerable. Examples are medical imaging, remote sensing, in image/video archives and studio applications where tandem- and trans-coding are used in editing, which can lead to accumulating errors. Nearly-lossless coding is used in applications where a small error, defined as a maximum error or as an rms error, is tolerable. In lossless embedded coding, a losslessly coded bit stream can be decoded at any bit rate lower than the lossless bit rate. In this thesis, research on embedded lossless video coding based on a motion compensated framework, similar to that of MPEG-2, is presented. Transforms that map integers into integers and embedded source coding, which are the main ingredients of lossless embedded coding are discussed in the contexts of intra frames, which are similar to still images and non-intra frames, which contain motion compensated prediction errors. The lifting concept, which forms the integer wavelet transforms, and the intrinsic properties of the block orthogonal transforms, such as the Discrete Cosine (DCT), the Discrete Sine (DST) and the Walsh-Hadamard (WHT) are used to design the integer versions of the N-point DCT, DST and WHT, where N is any integer power of two. Furthermore, the design and the use of transforms with spatially adaptive numbers of vanishing / preserving moments, which are suitable for non-intra frames, and non-linear transforms are presented. The current and prospective embedded coding scannning methods are analysed and an adaptive quad tree splitting (AQS) based scanner is presented. The performance of the above transforms for both types of frames is analysed using the zeroth order entropy values and the coded bit rates, achieved by Embedded Lossless Image Coding (ELIC), which is based on AQS and efficient context modelling. In addition to the above experiments, the use of a transform in coding highly decorrelated non-intra frames is also investigated. Finally, the components discussed above are integated together to analyse the importance of motion compensation in lossless video coding and the robustness of embedded decoding at quasi-lossless decoding in an assymetric codec, where a “Group Of Pictures” (GOP) structure based motion prediction is involved.

# ACKNOWLEDGEMENTS

I wish to thank Professor Don Monro for his supervision of this work, Tandberg Television Ltd. for their sponsorship for the first three years, Dr. Adrian Evans for providing me with an external project to fund for the fourth year, Ms. Rosemary Ainsworth and Miss Dorcas Mumford for their excellent services as the Signal and Image processing Group (SIPG) secreteries, Mrs. Heather Kellaway for employing me at the Bath University Computing Services (BUCS) help desk and collegues at the SIPG and the BUCS help desk.

# Contents

# List of Figures

# List of Tables

# Statement of Originality

The author considers the following elements of this work form an original contribution to embedded lossless and nearly lossless image and video compression literature.

- Chapter 2
  - Literature review.
- Chapter 3
  - The boundary treatment for lifting.
  - Design and implementation of the N-point Integer Walsh Hadamard Transform (IWHT).
  - Design and implementation of the N-point Integer Discrete Cosine Transform (IDCT).
  - Design and implementation of the N-point Integer Discrete Sine Transform (IDST).
  - Relationship betweem the IDCT$_N$-II and the IDST$_N$-II.
  - Design and implementation of non-linear integer tarnsform with median prediction and updation (INLT3) based on quincunx splitting.
  - Complete tree wavelet packet transform analogy in block transforms.
  - Anlaysis of the performance of the above transforms with various N values and scales for lossless coding of intra frames.
- Chapter 4
  - Analysis of the cost of embedding.
  - Weighted bit plane (WBP) sliding and depth limiting to obtain effective bit space.
  - Analysis of scanning schemes for the significance switching mask coding.

  - Adaptive quad tree splitting (AQS) of the significance switching mask coding.

- Chapter 5
  - Gradient oriented prediction and maximum likelihood based context model for switches.
  - Use of bits from the previous two bit planes for the context model for data refining.
  - The ELIC quantiser.
  - Analysis of the lossless coding performance of the above transforms with ELIC for intra frames.
  - Incorporating near-lossless quantisation into lifting steps for near-lossless image coding.

- Chapter 6
  - Analysis of the wavelet and the other transforms on residuals
  - Discovery that a single wavelet scale is sufficient for residual coding.
  - Spatially adaptive lifting for non-intra frames - Optimal prediction approach.
  - Spatially adaptive lifting for non-intra frames - Adaptive interpolation approach.
  - Analysis of the performance of integer transforms with various N values and scales for non-intra farmes.
  - Discovery that at lossless bit rates, encoding residuals without using a transform outperforms encoding with any other integer tansform due to the high decorrelation resulted from motion compensated prediction.

- Chapter 7
  - Comparison of Motion-JPEG-LS, Motion-ELIC and ELViC performance.
  - Analysis of the effect of the motion compensation process on quasi-lossless decoding in a frame-wise embedded coding framework.

# Chapter 1

# Introduction

Digital video sequences in uncompressed formats require excessive storage capacity and huge transmission bandwidth. Therefore compression of digital video sequences is necessary for efficient storage and transmission. However, coding at high compression factors causes loss of visual quality and information of the original video sequences. There are many applications in which no loss of either visual or pixel value information is tolerable. Examples are studio quality digital video archives, inter studio video transmission and coding of medical and astronomical sequences, in which exact pixel recovery for all frames is required. Further, lossless video coding is vital in studio applications in order to prevent accumulation of the quantisation effects from repetitive encoding and decoding processes performed in programme production.

This thesis investigates lossless and nearly lossless digital video coding techniques. This chapter introduces four topics, namely video coding, image coding, lossless coding and nearly lossless coding, of relevance to this thesis.

## 1.1 Digital Video Coding

### 1.1.1 Digital video

A digital video sequence is a collection of pictures, also called frames, spaced at fixed time intervals. In a colour video sequence, each frame consists of three components, which can be either red-green-blue (RGB primaries) or luminance and two chrominance

(YC$_b$C$_r$ format) components. The luminance (Y) component is a monochrome image containing the structural information of the frame. The two chrominance (C$_b$ and C$_r$) components contain colour hue and saturation information of the frame. RGB format is used in displaying, whereas YC$_b$C$_r$ format is the colour space recommended by CCIR-601 for coding and transmission [1]. RGB colour space can be converted into YC$_b$C$_r$ format as in equation 1.1.

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} A \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix}$$
$$where,\ A = \begin{pmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{pmatrix} \tag{1.1}$$

Three types of chrominance sampling formats relative to the luminance are used in video coding. These are labelled as 4:4:4, 4:2:2 and 4:2:0. In 4:4:4 format, the same sampling grid for all three components is used. In 4:2:2 format, the chrominance is sampled 2:1 horizontally but not vertically. The 4:2:0 format has the chrominance sampled 2:1 both horizontally and vertically.

Each component of a frame is a two-dimensional (2-D) signal, which can be represented by a matrix. The elements of the frame matrix are called as pixels. Therefore, a video sequence can be considered as a three-dimensional (3-D) signal for each spectral (colour) component.

Each pixel of each spectral component is stored in 8 bit units. The basic bit rate parameter is defined as bits per pixel (bpp) for each frame or as bits per second (bps), by considering the third dimension, time length, of the sequence. The bps is the more common metric for video.

$$bpp = \frac{Total\ Bits}{Total\ Pixels} \tag{1.2}$$
$$bps = bpp \times (Total\ Pixels) \times (Frames\ per\ second) \tag{1.3}$$
$$Total\ Pixels = Frame\ Height \times Frame\ Width \times No.\ of\ Frames \tag{1.4}$$

### 1.1.2 Digital video coding

Digital video coding or compression is concerned with reducing the number of data storing units (bps) used to represent given information content in a video sequence. In addition to inter pixel spatial redundancy within a frame, video sequences contain high temporal (inter frame) redundancy, which is usually exploited in video coding algorithms by coding some frames using motion compensated prediction with reference to previously coded frames.

There are a few video coding standards, each catering for different requirements, in use at the moment [1, 2]. Examples are MPEG-1, MPEG-2, MPEG-4, H.261, H.262, H.263, MPEG-7 and MPEG-21. MPEG-1 has been optimised for non-interlaced video at bit rates of 1.2 to 1.5 Mbit/s, whereas MPEG-2 has been targeted for higher bit rate, for example 10 Mbit/s, applications such as DVD (Digital Versatile Disk) storage and digital television broadcasts. MPEG-4 was started with the emphasis on very low bit rate, for example less than 64 kbit/s, applications and was developed into an object based video codec in order to accommodate other multimedia requirements such as coding of multi view point scenes. MPEG-7 and MPEG-21, the standards under development at the moment, are being generated on previous standards, MPEG-1/-2 and -4, in order to address other multimedia requirements. MPEG-7 is mainly concentrated on describing the multimedia content that supports some degree of interpretation of the information's meaning, which can be passed onto, or accessed by, a device or a computer code [3], while MPEG-21 is concerned with defining a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities [4]. The ITU-T standard H.261 is on digital video coding for digital transmission over ISDN, where the bit rates are in the range of 64 to 1920 kbit/s. ITU-T also adopted MPEG-2 under the generic name H.262 for telecommunication applications, while H.263 codec is concerned with very low bit rates, such as lower than 64 kbt/s, video coding for mobile network applications.

Single layer MPEG-2 was used as the framework for the research presented in this thesis. Therefore, the elements of video coding are discussed below with reference to the MPEG-2 video coding layer.

#### 1.1.2.1 Single-layer MPEG-2 Codec

In MPEG-2 video codec [2], each video sequence is divided into groups of pictures (gop). Each gop consists of frames of three different types, namely I (Intra frames), P

(Predictive coded frames) and B (Bidirectionally predictive-coded frames) types (Figure 1.1). A gop is defined by two parameters, namely N and M. N is the gop length, which is also the distance between two I frames. M is the distance between the P frames or I frame and the following P frame within a gop. I type frames are considered as still images and coded without any reference to other frames. P and B type frames, collectively called non-intra frames, are coded with reference to previously coded frames, hence exploiting the temporal redundancy. A P frame is predicted from the nearest preceding I or P frame using forward prediction, whereas a B frame is predicted from the nearest I and P frames either preceding, following or both using either forward prediction, backward prediction or interpolating both forward and backward prediction. Hence, these prediction errors are coded in non-intra frames.

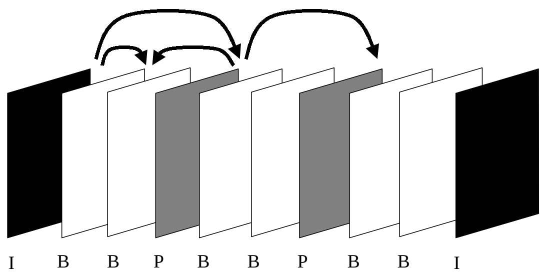


Figure 1.1: An example for a group of pictures (gop).

In MPEG-1/-2, each frame is divided into macro blocks, which are considered as the basic building blocks of an MPEG frame. A macro block consists of a $16 \times 16$ size luminance block together with chrominance blocks, the size of which is determined according to the chrominance format. The macro blocks in intra frames are divided into $8 \times 8$ sub blocks and each sub block is coded as in JPEG (Joint Picture Expert Group) [5] image coding. These coding models contain the Discrete Cosine Transform (DCT), the Human Visual System (HVS) based quantisation tables and entropy coding based on pre-defined variable length codes.

The coding strategy for the macro blocks in non-intra frames is decided by considering the amount of real motion, thus the difference between the block to be coded and the reference blocks within the search window in reference frame(s). These predictions are based on the motion compensation, thereby usually described using a vertical motion vector and a horizontal motion vector for each macro block. These vectors correspond to the displacements that give the best match between the macro block to be coded and the corresponding displaced region in the reference frames. The commonly used criterion for finding the best match are the minimisation of the mean absolute distortion

(MAD) or the mean square error (MSE). The motion vectors are transmitted to the decoder as part of the bit stream. The macro blocks in a P frame are categorised as either I-type, where no real motion is predicted, or P-type, where forward motion is predicted. Likewise, the macro blocks in a B frame are categorised into any of three types, which include the above mentioned two types and a B-type, where forward-backward motion is predicted from the preceding and following frames. In both P and B frames, the macro blocks classified as I-type are coded as those in intra frames. For the macro blocks classified as P-type and B-type, the prediction residuals in each block are coded in a similar manner to I-type coding using quantisation parameters and variable length codes different from those used in I-type blocks.

## 1.2 Image Coding

Digital image coding or compression is concerned with reducing the number of data storing units (bpp) used to represent a given information content in a digital image. Image coding is possible because images in uncompressed formats contain high data redundancy. In digital image compression processes, three basic data redundancies, namely inter pixel redundancy, psychovisual redundancy and coding redundancy can be identified and are successfully exploited.

### 1.2.1 Image coding model

The basic image compression model consists of three stages (Figure 1.2). They are listed below.

1. Transform/Inverse Transform.
2. Quantisation/Dequantisation.
3. Entropy Coding/Entropy Decoding.

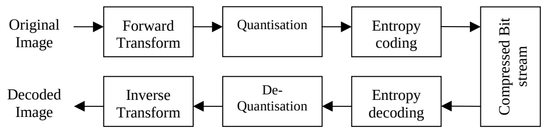


Figure 1.2: The basic image compression/decompression model.

In the transform stage, the main goal is to decorrelate the original image data, so that the original signal (image) energy is redistributed among only a small set of transform coefficients. This decorrelation eliminates the inter-pixel redundancy, thereby, providing a representation that can be coded more efficiently. The zeroth order entropy of the transformed coefficients is much lower than that of the original image. The transforms used in coding are reversible, so that the original image can be recovered by the inverse transform, provided no quantisation has been performed on the forward transform coefficients. So theoretically, the forward/inverse transform dual is a lossless process.

The DCT and the Discrete Wavelet Transforms (DWT) are the most commonly used transforms in current image codecs. With each transform having its own merits and demerits, the wavelet transforms are more widely used in image coding for their superior performance over the DCT. The criteria are discussed in sections 1.2.2 and 1.2.3.

The second stage, quantisation / dequantisation is the process that leads to the lossy compression. In the quantisation section, psychovisual redundancy in the image is reduced by throwing away unwanted bits from the transform coefficients. This leads to high compression ratios and distortion in image fidelity. The third stage, entropy coding, determines the number of bits required to represent a particular image at a given quantisation level. The combined process of entropy coding and entropy decoding is lossless. It maps the quantised transform coefficients into a bit stream using variable length codes, thus exploiting the coding redundancy.

### 1.2.2 The Discrete Cosine Transform (DCT)

An N-point DCT consists of N real basis vectors with cosine functions. There are four types of DCT, namely DCT-I, DCT-II, DCT-III and DCT-IV, derived according to the varying boundary conditions [6]. The DCT-II form, originally presented in [7], is commonly known as the DCT in signal processing research and widely used in image and video coding.

The DCT-II for a one dimensional (1-D) data sequence, $x$, of length $N$ and its inverse are defined as in equations 1.5 and 1.6 respectively.

$$X(k) = \epsilon_k \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n) \cos(\,(2n+1)\frac{\pi k}{2N}\,) \tag{1.5}$$

$$x(k) = \epsilon_k \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} X(n) \cos(\,(2n+1)\frac{\pi k}{2N}\,) \tag{1.6}$$
$$for\ k = 0, 1, \ldots, N-1$$
$$where,\ \epsilon_k = \begin{cases} \frac{1}{\sqrt{2}} & if\ k = 0\,, \\ 1 & else. \end{cases}$$

The 2-D DCT is performed by applying the 1-D DCT on rows and columns separately. The DCT is an orthogonal transform. The DCT possesses good variance distribution which leads to efficient energy compaction in the transform domain. When the input data is assumed to be a first order Markov process, the energy compaction efficiency of the DCT approaches that of the Karhunen-Loève Transform (KLT), of which the basis functions are input dependent [8, 9].

The DCT is performed on smaller blocks of the image, normally 8×8 sized blocks because the basis functions of the DCT are not spatially localised. As quantisation errors can be spread throughout the block, this can result in visible errors at image edges and at the block boundaries. Consequently, the DCT based coded images are characterised by blocky artefacts.

### 1.2.3 The wavelet transforms

The wavelets are localised waves. Once the "mother wavelet" $\Psi(t)$ is defined, the mother wavelet can be scaled and translated to obtain a family of other wavelets, which can be defined as

$$\Psi_{(a,b)}(t) = \frac{1}{\sqrt{a}}\,\Psi\left(\frac{t-a}{b}\right), \tag{1.7}$$

where a $> 1$ is the change of scale and $b \in \mathcal{R}$ is the translation in time [6, 10]. Therefore, in a wavelet transform, any finite energy signal can be represented by a linear combination of wavelets $\Psi_{(a,b)}(t)$. Then, the wavelet coefficients $f_{(a,b)}$ for the input signal $f(t)$ are defined as in equation 1.8.

$$f_{(a,b)} = \int_{-\infty}^{\infty} \Psi_{(a,b)}(t)\, f(t)\, dt \tag{1.8}$$

The above defined is the Continuous Wavelet Transform (CWT). The Discrete Wavelet Transform (DWT) is used in image coding applications. In DWT, the mother wavelet is translated and dilated by discrete values, in most cases by a power of 2 (dyadic).

Thus equation 1.7 becomes

$$\Psi_{(a,b)}(t) = 2^{a/2}\, \Psi(2^a t - b). \tag{1.9}$$

In current literature, there are two well known methods, namely filter banks and lifting, for implementing the DWT. The more widely used method is the filter bank approach. The filter bank approach consists of two filter banks, one each for the analysis (forward transform) and the synthesis (inverse transform) (Figure 1.3). In the analysis filter bank, the input signal is decomposed into two channels using a low pass filter (H0) that corresponds to averaging the input signal with a scaling function and a high pass filter (H1) that corresponds to detailing the input signal with the wavelet $\Psi_{(a,b)}(t)$, both followed by a decimator (to down sample the filtered data by a factor of 2) in each channel. In the synthesis filter bank, the transformed coefficients are interpolated (up sampled by a factor 2) and then convoluted with the filters F0 and F1, the coefficients of which are obtained from H1 and H0 respectively in order to eliminate aliasing.

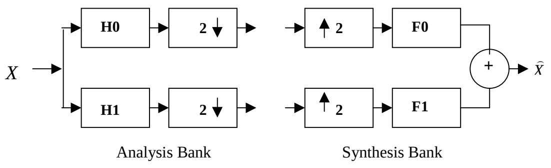


Figure 1.3: The filter bank approach for DWT.

The second approach is using the lifting scheme (Figure 1.4). In the forward transform, the input signal is decomposed into two subsets of odd ($d$) and even ($s$) samples. This process is called the lazy wavelet transform. Then the primal ($P$) and dual ($U$) lifting functions are operated repeatedly on the two subsets resulted from the lazy wavelet transform, in order to obtain the wavelet transform coefficients with the required number of vanishing moments [11]. The ($s$) and ($d$) after lifting correspond to the low and high passed signals respectively. More about lifting scheme is discussed in section 3.1.

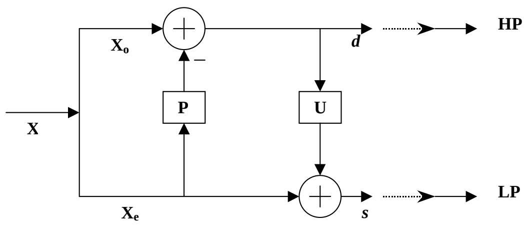


Figure 1.4: The lifting approach for DWT.

The high pass channel in the forward transform decomposes the input signal into details, where the signal behaviour is more localised in the spatial domain and wide band in the frequency domain. On the other hand, the low pass channel separates the regions of high statistical spatial correlation, from the original signal, thereby filtering the components more localised in the frequency domain (narrow band) and wide spread in the spatial domain. Thus, the wavelet transform coefficients represent two frequency sub bands, namely, a low frequency sub band with highly smooth spatial information and a high pass sub band with details that mainly correspond the wide band noise and the edges in the input signal.

The 2-D wavelet transform is achieved by performing the 1-D DWT separately on rows and columns of the 2-D signal (image). It can be performed in any order (either starting with row wise or column wise) as these two operations are two separable processes. This produces four sub bands, namely LL, LH, HL and HH (where L and H stand for low pass and high pass respectively). The LL sub band represents the original signal in half resolution and contains smooth spatial data with high spatial correlation. The HH sub band consists of details caused by noise and the edges in the image. The HL and LH sub bands consist of vertically and horizontally oriented high frequency details respectively. Most of the image energy is concentrated in the LL sub band. A hierarchy of wavelet coefficients can be obtained by applying the 2-D transform to the LL sub band of the current scale repeatedly up to 4 or 5 scales, (Figure 1.5) provided the original signal dimensions agree with the down sampling in the process.

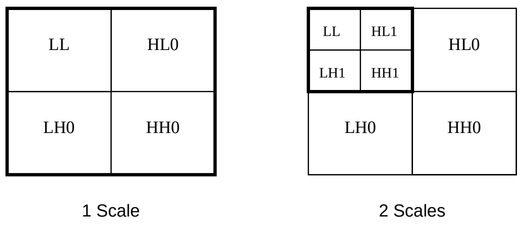


Figure 1.5: The wavelet transform operation.

The Wavelet transforms can be designed as orthogonal or biorthogonal. Wavelets also support an efficient energy compaction in the coefficients. Wavelet coefficients can be modelled into hierarchical trees using the multiresolution property of the wavelet coefficients. The multiresolutional structure of the wavelet transformed coefficients can be used to develop efficient quantisation algorithms, [12, 13, 14] that help to achieve a better picture quality than in DCT methods. Furthermore, spatial and quality wise scalability can easily be incorporated into the wavelet transforms based techniques.

### 1.2.4 Quantisation and embedded coding

The dynamic range of the transform coefficients are narrowed by the quantisation process, thus achieving high compression. Quantisation can be either scalar (uniform) quantisation or vector quantisation. In scalar quantisation, a single coefficient $C$ is divided by a quantisation factor $Q$ and rounded to the nearest integer in order to obtain the quantised coefficient $T(C)$ (equation 1.10).

$$T(C) = \mathit{Round}\left(\frac{C}{Q}\right) \tag{1.10}$$

In vector quantisation, the coefficient set is divided into 1-D or 2-D blocks, that are also called vectors, and a code book is used to find a pattern for each block. The approximated pattern for each block is coded using a lookup value in the code book. The code book can be adaptive and implemented dynamically or fixed, the latter being predefined using a training data set.

The codec can be designed as spatially and quality wise scalable. In a spatially scalable codec, the image can be decoded at smaller resolutions than the original dimensions, whereas in a quality wise scalable codec, the image can be first decoded as a degraded version of the original and then updated progressively up to the targeted bit rate or to the required image fidelity. These features can be achieved by employing progressive and embedded coding respectively, in the quantiser.

**Definition 1.1 (Embedded Coding)** *In* embedded coding*, a signal (image) is coded at a bit rate $R_i$ in a such way that the bit streams for all the other lower bit rates ($R_0 < R_1 < \cdots \leq R_i$) are progressively embedded within the bit stream for $R_i$ and ($D(R_0) > D(R_1) > \cdots \geq D(R_i)$), where $D(R)$ is the signal distortion at rate $R$.*

The embedded coding quantisers employ scalar quantisation. With embedded coding, the coded image bit stream can be decoded at any other lower bit rate. Examples are Embedded Zero tree coding of Wavelet coefficients (EZW) [13], Set Partitioning in Hierarchical Trees (SPIHT) [14], Compression of Reversible Embedded Wavelets (CREW) [15] and Embedded Predictive Wavelet Image Coder (EPWIC) [16], which is based on a conditional probability model developed by joint sub band statistics. More about embedded quantising in embedded lossless coding is discussed in chapter 4.

### 1.2.5 Entropy coding

The zeroth order entropy $H0$ for a symbol set of M different symbols considering a memoryless source can be computed as in equation 1.11.

$$H0 = -\sum_{i=0}^{M-1} p_i \log_2 p_i \tag{1.11}$$

$p_i$ *is the probability of the* $i^{th}$ *symbol.*

Two widely used entropy coding methods are Huffman coding and arithmetic coding [17]. Normally, these coding schemes assume the source as a memoryless model and compress it to the zeroth order entropy of the data stream according to the probability density distribution of the data symbols in the input stream [18]. However, significant gains can be obtained by exploiting the redundancy in the input stream. This is normally achieved by: run length coding followed by the entropy coding, by using adaptive entropy coding, by context based entropy coding or by combining two or more of the above.

In adaptive coding, the symbol probabilities up to the most recently coded symbol are considered in the probability density distribution in computing the probability of the current symbol. The probability, $p_i$, of the current symbol $x_i$ in equation 1.11 is the conditional probability conditioned with the previous $i-1$ symbols.

$$p_i = p(X_i = x_i | X_0 = x_0, \ldots, X_{i-1} = x_{i-1}). \tag{1.12}$$

With adaptive entropy coding, no probability information has to be sent to the decoder. Adaptive arithmetic coding, presented in [17, 18], uses this concept. Adaptive entropy coding provides better coding performance and avoids multiple scans through the input symbol stream. Further, it can avoid the use of pre-built coding tables that are used for entropy coding in the current JPEG [5] and MPEG standards.

To compress data using adaptive entropy coding methods, a model of the data stream is required. This model needs to achieve correct prediction of the probability of the incoming symbol and the probability estimations need to deviate from a uniform distribution [18]. Finite context models based entropy coding helps to achieve such models. In context based entropy coding, probability for each incoming symbol is calculated based on the probability distribution function of a coding context in which the symbol appears. The finite contexts are usually determined by the statistical modelling of the previously coded symbols.

### 1.2.6 Image quality measurement

The most common method of comparing the decoded image, $\tilde{I}$, with the original image, $I$, of N×M dimensions is to use the peak signal-to-noise ratio (psnr), which is based on the root mean square (rms) error of the two images as in equations 1.13 and 1.14.

$$psnr = 20 \log_{10} \left( \frac{255}{rms\ error} \right) dB \tag{1.13}$$

$$rms\ error = \sqrt{\frac{1}{N \times M} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} ( I(x,y) - \tilde{I}(x,y) )^2} \tag{1.14}$$

## 1.3 Lossless and Nearly Lossless Coding

As seen in previous sections, image/video compression reduces the volume of data by discarding some, if not all, irrelevant data while maintaining an information content with a reasonable fidelity. Usually in visual media coding, one may reduce one or more of spatial (inter pixel and inter frame), spectral (within different spectral bands), psycho visual and coding redundancy types, while maintaining image visual quality. All coding techniques can be categorised into two groups, namely lossy and lossless coding, according to the effect of their compression.

The image coding model described in section 1.2 and the macro block coding described in section 1.1 fall into the lossy coding category, as quantisation is involved in these coding models. However, there are some applications where no loss of exact pixel values is tolerated. Lossless coding is used in such applications. The definition of lossless coding, adopted from JPEG-LS [19], the current lossless image coding standard is set out below.

**Definition 1.2 (Lossless Coding)** *In* lossless coding*, compression and decompression of a image / video sequence results in the exact recovery of the individual pixel values of the original image / video sequence.*

In lossless coding, only spatial and coding redundancies are exploited. The disadvantage in lossless coding is that only modest compression ratios can be achieved, whereas in lossy coding, higher compression ratios can be achieved at the expense of the image perceivable quality and exact pixel values.

Preserving the total dynamic range of the pixels is not very important and an exact recovery is not vital in many applications, where further image analysis based on pixel values is not performed. As a consequence, several near exact recovery coding techniques have been developed to eliminate the disadvantages of lossless coding. These nearly lossless coding methods can be categorised into two classes depending on their operation. They are

1. Near-lossless coding.
2. Quasi-lossless coding.

The definition of near-lossless coding, adopted from JPEG-LS [19], is as below.

**Definition 1.3 (Near-Lossless Coding)** Near-lossless coding *is a lossy encoding and decoding process, in which the output of the decoding process is such that each reconstructed pixel of image / video sequence differs from the corresponding one in the input to the encoder by not more than a pre specified value $\delta$.*

With near-lossless coding, the accuracy of the decoded pixel values, which is in the range of $\pm\delta$, is known, so that the accuracy of the pixel value based computations can be determined. When $\delta$=0, a near-lossless codec performs losslessly.

Quasi-lossless coding includes high bit rate lossy encoding processes, where the distortion due to quantisation is small. There is no distinct definition for quasi-lossless coding. In this research, lossy codec performance up to a bit rate that is about half of the corresponding lossless bit rate can be considered as quasi-lossless coding. Examples are lossy image coding with no quantisation, but the transform coefficients rounded (or truncated) to integers and visually lossless coding techniques.

More about recent work on lossless and near lossless coding can be found in the chapter on literature survey (Chapter 2).

### 1.3.1 Lossless image and video coding

As seen in section 1.1, colour images consist of three spectral bands. For colour images in RGB format, lossless coding can be performed for the three bands separately. This avoids exploiting inter spectral redundancies. The colour transformation in equation 1.1 cannot be used in lossless coding because the conversion back to RGB space is not lossless, due to rounding effects of the decimal values caused in the colour conversion. Therefore, a reversible colour space transformation as in equations 1.15 and 1.16 can be used [20].

$$\begin{aligned} &RGB \rightarrow YC_bC_r \\ Y &= \left\lfloor \frac{R+2G+B}{4} \right\rfloor \\ C_b &= B-G \\ C_r &= R-G \end{aligned} \tag{1.15}$$

$$\begin{aligned} &YC_bC_r \rightarrow RGB \\ G &= Y - \left\lfloor \frac{C_r+C_b}{4} \right\rfloor \\ R &= C_r+G \\ B &= C_b+G \end{aligned} \tag{1.16}$$

$\lfloor . \rfloor$ is the downward rounding operation.

In lossless video coding, the chroma sampling format of 4:4:4 is usually considered as other sampling formats (4:2:0 and 4:2:2) cause losses due to chroma sub-sampling.

### 1.3.2 Visually lossless coding

In entertainment video and digital picture applications, where the end user is the human eye, the knowledge about accuracy on the exact pixel value recovered is not vital at all. For such applications, a visually (perceptually) lossless image/video coding technique is more useful than lossless or near-lossless codecs, which code information containing visual redundancy. Further reduction of bit rates, and thereby higher compression ratios can be achieved by coding up to a visually lossless level.

Usually, the compression is said to be visually lossless when a compressed image cannot be distinguished from its original [21]. The author in [22] has stated that the term "visually lossless compression" can be used only if the reconstructed image looks like the original when they are compared in a two alternative forced choice test and the preference for one image over the other is statistically insignificant. Another definition is to keep the probability of detecting an error at each pixel just below the visual threshold [23].

Traditional image coding techniques are usually based on optimising rate distortion, where distortion is measured mathematically using the rms error metric. Visible distortions caused by quantisation in lossy coding / decoding are inappropriate for entertainment video and digital picture applications. This has resulted in the research into visual aspects of image / video coding algorithms. The errors due to quantisation and non-integer transforms may begin as mathematical differences, but end up as a visual difference once the image is displayed. The main goal of the perceptual coding research has been to determine the degree to which these mathematical differences become visible and thereby to formulate visually lossless coding, or to minimise the visual distortions for a given bit rate.

The perceptual coding research to determine a perceptually lossless compression is mainly based on two methods, namely the Human Visual System (HVS) models and the psychovisual tests [24]. In the HVS models the visual process involved in the perception of images are modelled using the fundamental theoretical and empirical knowledge of the HVS, whereas psychovisual testing involves the use of subjects (potential viewers) to assess the quality of the images.

### The human visual system (HVS) and the model

The HVS has a number of fundamental properties that have often been studied and modelled. The three main fundamental properties of these, in the order in which they occur in the HVS and therefore the order in which they appear in the HVS model [23] are: luminance sensitivity, frequency sensitivity and signal content sensitivity.

Luminance sensitivity corresponds to the subjective brightness of the image, which is known to be a non linear function of the light intensity incident on the eye and normally modelled by a logarithmic model in the amplitude non-linearity component of the HVS model [25]. Frequency sensitivity corresponds to the HVS's sensitivity to spatial changes of luminance levels in an image [23], which is modelled as the contrast sensitivity function (CSF) in the HVS model [25]. Signal content sensitivity corresponds to the sensitivity of the HVS to the signal content of an image. This is normally modelled as contrast masking functions, where the noise is masked by the underlying image content [23].

#### Psychovisual testing

The psychovisual tests use potential viewers, under highly controlled conditions to measure the visual quality of images. The conditions for psychovisual testing recommended by the ITU-T can be found in the Rec. 500-4 [26]. This recommendation sets the viewing conditions for the assessments, with reference to the distance from the screen, peak luminance of the screen, room illumination, number of assessors per monitor, display brightness and contrast and the nature of the viewing room. Further, it is recommended that the test images should be presented in a pseudo random sequence.

### 1.3.2.1 Visual quality metrics

As mentioned earlier, the traditional image quality measure, the rms error metric, does not measure the visual quality of the compressed image. Recently, there has been some published research on measuring the visual image quality. One such method is to compute the picture quality scale (PQS) [27] over the full range of image quality defined by the subjective mean opinion score (MOS). This involves measuring the properties of visual perception for both global features and localised disturbances. Another example is the wavelet visible difference predictor [23], which computes visible difference in wavelet image coding based on the multiple channel models of the HVS [25, 28, 29], and their relationship to the wavelet transform.

However, these quality metrics are not yet commonly used in the compression community due to their computational complexity.

## 1.4 Summary

The basic concepts of four coherent topics, namely video coding, image coding, lossless and nearly lossless coding and visually lossless coding were introduced in this chapter. The next chapter will present published research relevant to the topic of this thesis.

# Chapter 2

# Literature Review

Image and video coding algorithms can be either lossy or lossless. Lossy image and video coding techniques with low bit rate coding have been the preferred coding methods in multimedia and internet applications. Therefore, most of the published research has been on lossy coding techniques. However, research on lossless image/video coding and visually lossless coding has been highly considered in some applications due to the reasons discussed in the previous chapter. This chapter discusses the published research on lossless image coding, near-lossless image coding and lossless video coding. The organisation of this chapter is as follows. Sections 2.1-2.3 report and discuss the published research on above topics respectively. Finally, in section 2.4 the research objectives and their necessity for this thesis is presented followed by a thesis outline.

## 2.1 Lossless Image Coding

In lossless coding, the inter pixel redundancy of the image and the coding redundancy of the decorrelated symbol stream are exploited in order to achieve compression. The lossless coding techniques that have been presented in the current published literature can be grouped into three main categories. They are as follows.

1. Prediction based methods.
2. Lossy coding followed by lossless coding of the residuals.
3. Transforms that map integers into integers (integer transforms) based methods.

### 2.1.1 Prediction based methods

The predictive coding based on a causal template has been the most commonly used lossless image coding method due to its simplicity and efficiency. The philosophy underlying the predictive image coding techniques is to remove inter pixel redundancy by predicting the current pixel from the previously coded (i.e. using a causal template) neighbouring pixels and coding the prediction error losslessly. The probability distribution of the prediction error can be modelled as a double sided geometric distribution with narrow peaks centred at zero and long tails, thus resulting in significantly lower zeroth order entropy than that of the original image.

In the lossless mode of the previous JPEG standard [5] (JPEG-Lossless), a predictor set which uses three neighbouring pixels (N, W and NW as in Figure 2.1) to predict the current pixel X has been employed. Then the prediction error for each pixel is entropy coded using a Huffman table which is nearly identical to that used for DC coefficient coding in the JPEG baseline codec.

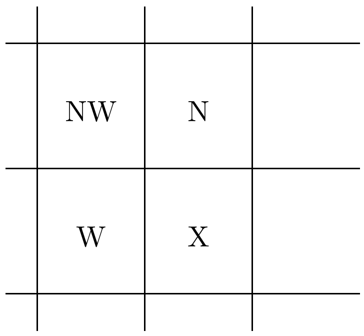


Figure 2.1: JPEG-Lossless mode prediction template.

However, such linear and fixed predictive techniques are far from being powerful enough to provide a good prediction. Therefore, during the past few years, predictive coding techniques have been developed into the combination of predictive coding based on adaptive statistical context modelling of images [30, 31] and the entropy coding of the prediction error using estimated probability density functions (PDF) conditioned on the contexts in which the pixels are observed [18]. Two such examples are the Context based Adaptive Lossless Image Codec (CALIC) [31, 32, 33] and the LOw COmplexity LOssless COmpression (LOCO-I) [31, 34].

The CALIC algorithm, which puts heavy emphasis on image data modelling, uses a Gradient Adjusted Predictor (GAP) which adapts the prediction according to the local gradients, thereby using a non-linear predictor which adapts to varying source statistics. The GAP classifies the gradient of the current pixel X according to the estimated gradients in the neighbourhood (Figure 2.2), which is wider than that used in JPEG-Lossless, and chooses the appropriate predictor according to the classification [32, 33]. Then the prediction error is context modelled and entropy coded.

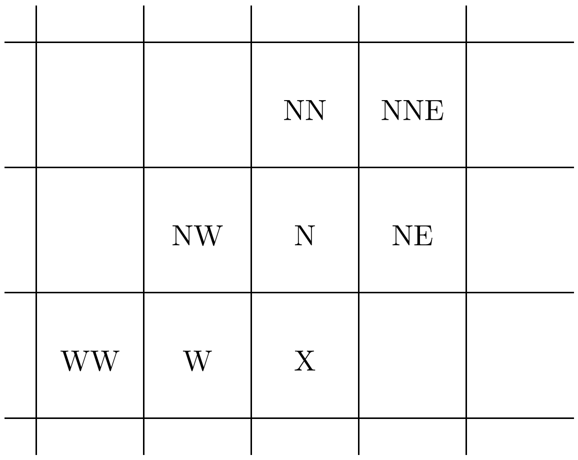


Figure 2.2: The GAP prediction template.

The LOCO-I algorithm uses the Median Edge Detection (MED) predictor, which adapts in the presence of local edges [34]. The MED classifies the edge type of the current pixel X by using a causal template, as in JEPG-Lossless, and one of three predictors are used according to the classified edge type. The MED prediction error for each pixel is coded using a context model, determined by quantised gradients, followed by Golomb-Rice coding [35] for entropy coding. The LOCO-I algorithm was recently standardised as the new lossless and near lossless image coding standard, JPEG-LS [19, 36]. More about recent developments in context based prediction techniques used in lossless image coding research leading to JPEG-LS standardisation can be found in [31].

The other predictive techniques used in lossless image coding include adaptive L-predictors (linear combinations of ordered statistics) based on finite state machine context selection [37], a prediction model based on backward adaptive recognition of local texture orientation (BAROLTO) followed by a Poisson statistical model for error coding [38], prediction based on fuzzy switching of a set of linear predictors followed by arithmetic coding [39] and prediction based on adaptive median FIR filter followed by error mapping and context modelling for entropy coding [40].

The sequential prediction techniques discussed above have been extended into sub band prediction based methods that result in multiresolutional decorrelation. The most famous example is the hierarchical interpolation (HINT) [41, 42] algorithm used in reversible medical imaging applications. In HINT, image pixels are classified into five different types using a 4×4 base block (Figure 2.3) and firstly, the $\diamond$ type pixels are coded using DPCM. Then the $\triangle$ pixels are linear interpolated using four $\diamond$ type pixels and the error is variable length coded (VLC). Then the $\bullet$ pixels are predicted by interpolating the $\diamond$ and $\triangle$ pixels followed by the $\times$ and $\star$ pixel prediction. In all the above steps, the prediction is rounded and the prediction error, which is an integer, is coded using VLC.

| | | | | |
|---|---|---|---|---|
| $\diamond$ | $\star$ | $\bullet$ | $\star$ | $\diamond$ |
| $\star$ | $\times$ | $\star$ | $\times$ | $\star$ |
| $\bullet$ | $\star$ | $\triangle$ | $\star$ | $\bullet$ |
| $\star$ | $\times$ | $\star$ | $\times$ | $\star$ |
| $\diamond$ | $\star$ | $\bullet$ | $\star$ | $\diamond$ |

Figure 2.3: HINT pixel classification.

Similar interpolative sub band prediction techniques have been developed for lossless coding. Examples are a prediction technique based on a four channel filter bank [43] and an interleaved hierarchical prediction [44] which splits the non separable interpolation process into two cascaded directional steps. The author in [45, 46] compared different interpolation methods for the prediction in four sub band splitting as in quincunx splitting and has concluded the median-FIR based interpolation as the best decorrelator.

The sub band interpolative methods followed by context based residual coding has not achieved better lossless compression performance compared to that of sequential prediction techniques discussed earlier. However, progressive coding / decoding can be incorporated into these sub band interpolative methods.

The two types of prediction based techniques discussed above, namely, sequential and interpolative, have used different entropy coding techniques such as Huffman tables in JPEG-lossless, Golomb-Rice in LOCO-I, DPCM and VLC in HINT and statistical context modeling, to encode the prediction residuals. As in JPEG-LS, the current lossless standard, most of the methods employ context modelling of the error prior to entropy coding. A minimum entropy clustering method, where a set of vectors are clustered using a minimum entropy criteria as in classical vector quantisation, has also been used in literature [47]. It has also been recorded that such a method has significantly outperformed a single entropy coder based lossless coding.

### 2.1.2 Lossy coding followed by residual coding

The second approach is using lossy compression followed by entropy coding of the error due to lossy coding. In one such lossless coding example, filter banks based wavelet transforms were used for lossy compression [48], where wavelet coefficients were encoded using a variable block size segmentation and a directional prediction scheme. The residual error due to the finite precision arithmetic was coded using adaptive arithmetic coding. A similar approach using zero tree coding for the lossy coding part, thereby adding the embedded property to the codec, has been presented in [49, 50].

The compression ratios achieved by this approach are comparable with those of JPEG-Lossless. This approach is more computationally expensive than the predictive coding techniques due to the inverse wavelet transform, which has to be performed in the coder in order to find the error caused by the finite precision arithmetic in the lossy coding part. Therefore, this approach is inferior to the predictive coding approach in lossless coding.

### 2.1.3 Integer transforms based methods

Although transforms are theoretically designed to be lossless, when implemented on computer hardware or software, no perfect reconstruction can be achieved due to finite precision arithmetic in computer operations and the large dynamic range of the transform coefficients. However, the image transformations can be modified in order to achieve integer coefficients with a finite dynamic range. Therefore such transforms that map integers into integers can be used for lossless image coding.

The early examples include lossless implementations of Walsh Hadamard transform (WHT) [51, 52] and DCT [53]. The WHT comprises combinations of $+1$ s and $-1$ s, followed by a normalising factor. If the normalising factor is ignored, the transform can be realised by additions and subtractions of the input signal elements. In [51, 52], the redundancy in additions and subtractions have been removed by incorporating a lossless quantisation scheme. The compression ratios achieved using such lossless WHT were better than those from JPEG-Lossless. Since the WHT is an orthogonal transform, the embedded coding feature has also been added to the codec using the lossless WHT.

The DCT based lossless coding methods involve factoring the un-normalised DCT matrix into upper and lower triangular matrices with unit diagonals. Since the DCT is an orthogonal transform, the DCT matrix is unitary. Any unitary matrix can be factorised into two triangular matrices (Upper and Lower triangular) [54] which in turn can be used to obtain lossless DCT coefficients. Such a method for the 8-point DCT has been introduced in [53]. The lossless DCT, which has also been used for embedded coding has shown the lossless and rate distortion performances similar to those of the lossless WHT [53].

Recently, the wavelet transforms that map integers into integers have also been implemented [55]. Any discrete wavelet transform, or two channel filter bank with finite filters, can be decomposed into a finite set of simple filtering steps called lifting steps [55, 56, 57]. This decomposition is normally done by factoring the polyphase matrix of the wavelet or filter bank into elementary matrices using the Euclidean algorithm [55, 58]. With this, the normal lattice structure associated with the wavelets and filter banks is converted to a ladder structure, with which the rounding operations can be employed into lifting steps in order to achieve integer coefficients [6, 55]. The beauty of the lifting scheme is that it can be used in biorthogonal wavelets, which is a non-unitary system, as well [11]. Various lifting factorisations have been introduced in literature [56, 59, 60]. The S+P transform (**S** Transform + **P**rediction) [61], which was obtained by introducing an additional predictive lifting step on the S transform [56], is also an example for integer wavelets. More about integer wavelet transforms using lifting and their applications on lossless image coding is discussed in section 3.1.

These integer wavelets have been used in lossless image coding by using the separable lifting steps for the rows and columns of the image, and entropy coding of the coefficients [20, 62, 63, 64]. Three methods have been considered to encode the wavelet coefficients in the published literature.

The first method was encoding the coefficients using a two pass Huffman coding optimised for each image [63].

In the second method, a Magnitude-Set Variable Length Integer (MS-VLI) was defined and the Magnitude-Set(MS) information was arithmetic coded conditioned to the mean MS of the causal neighbours appearing in a raster scan [63]. This method has been improved by using rigorous context modelling based on pixels from the parent and sibling sub bands, and thus exploiting the inter sub band dependencies of the coefficients [64]. Further, the use of context based arithmetic coding in this method has been demonstrated in [65].

The third method was using an embedded or progressive fidelity transmission algorithm to encode the wavelet coefficients. The embedded coding algorithms, SPIHT [14] and CREW [15], have been used in lossless image coding algorithms presented in [63] and [20] respectively for progressive coding of integer wavelet coefficients.

The results published in [63] show the superiority of the second method over the first and the third. But the third method, using embedded coding, enables the coded bit stream to be decoded at any lower bit rate, thereby making the codec versatile for both lossy and lossless modes of operations.

#### 2.1.3.1 Context modelling

As seen in above lossless coding examples, it is evident that employing a context modelling process to choose a prediction context [19, 36, 37] or a coding context [18, 30, 31, 33, 39, 49], in which the current pixel or the symbol appears, has improved the efficiency of the predictor or the entropy coder or both. A brief literature survey on context modelling can be found in section 5.1.1.

### 2.1.4 Summary

As reported in the literature [31], predictive coding has been the preferred data decorrelating method for lossless image coding as they have obtained the best compression ratios out of the three coding techniques considered. The current lossless coding standard, JPEG-LS, is also based on a predictive lossless coding technique. The sequential predictive coding techniques discussed in section 2.1.1 were based on local predictive techniques (using a causal template). The high pass bands in transform coding using filter banks or lifting are analogous to prediction error due to global predictions. Therefore, integer wavelet transforms provide a global prediction which uses the neighbour pixels from both sides of the pixel concerned (using a non causal template). The advantage of using integer wavelet transforms or any other integer orthogonal transform is due to the progressive coding / decoding by fidelity or by resolution features that can be incorporated into the lossless bit stream through embedded coding techniques. Furthermore, using sub band interpolative prediction methods, which possess lower computational complexity than the integer transforms, also enables progressive coding / decoding. This is the main advantage of using integer transforms or hierarchical splitting based techniques over sequential predictions based techniques. Recently however, there has been an instance of incorporating a rate control mechanism based on visual perception into JPEG-LS, so that it can also be used as a quasi-lossless coder [66].

## 2.2 Near-Lossless Coding

According to Definition 1.3 on page 13, in near-lossless coding, a maximum pixel error $\pm\,\delta$ that any reconstructed pixel is allowed to have due to lossy coding can be specified by the user at the time of coding. Coding is optimised so that the decoded image pixels differ from those in the original by not more than this pre-specified value.

Near-lossless coding can easily be incorporated into predictive lossless coding techniques discussed in section 2.1.1. This is normally performed by quantising the prediction error. For example, in JPEG-LS a near-lossless compression with maximum error value $\delta$ is achieved by quantising the prediction error values ($\epsilon$) as below [19].

$$Q(\epsilon) = Sign\,(\epsilon)\,\left\lfloor \frac{|\epsilon| + \delta}{2\,\delta + 1} \right\rfloor \tag{2.1}$$

$Q(\epsilon)$ is the quantised error value $\epsilon$. The reconstructed error value $\hat{\epsilon}$ is obtained as below.

$$\hat{\epsilon} = Q\,(\epsilon) \times (\,2\,\delta + 1\,) \tag{2.2}$$

In the encoder, the reconstructed error value is used to reconstruct the actual pixel values, which are used in context modelling and prediction processes, so that both encoder and decoder are synchronised to each other. For small $\delta$ values, this type of near-lossless coding has shown superior PSNR results compared to traditional lossy coding methods [36].

The quantiser in equation 2.1 quantises the prediction error uniformly. There are a few instances of near-lossless coding methods that attempt to maximise the rate achievable for a set of pixels rather than using uniform quantising. For example, rather than quantising as in equation 2.1, trellises were constructed describing all possible quantised prediction error sequences that yield reconstructed images meeting the near-lossless requirement and the trellises with minimum entropy were selected to encode as in [67, 68, 69].

In lossless image coding methods using integer transforms, it is unpractical to encode the transform coefficients to a near-lossless criterion in the transform domain. Therefore, the pixel values were normally pre-quantised in a similar manner as above prior to the integer transform and embedded coding. Such a method using integer wavelet transforms has been introduced in [70]. For each pixel $x$, the quantised pixel value $l$ is obtained as below.

$$l = \left\lfloor \frac{x + \delta}{2\,\delta + 1} \right\rfloor \tag{2.3}$$

At the decoder, pixel values are reconstructed as below after the inverse transform.

$$\hat{x} = l \times (2\delta + 1) \tag{2.4}$$

Although the compression ratios achieved with this method were inferior to those of predictive techniques, embedded features can be incorporated to near-lossless coding with this method. However, this method is not very efficient for the near-lossless parameter value, $\delta$, higher than 3 [70].

Recently, a near-lossless coder with successive refinement capability based on the near-lossless parameter $\delta$ has also been published [71]. This coder provides a near-lossless to lossless progressive coder using a successive refinement method based on pre defined $\delta$ values, for example, $\delta_n > \delta_{n-1} > \cdots > \delta_1 > \delta_0 = 0$. This embedded stream can be decoded at the $\delta$ values defined at the coding end.

## 2.3 Lossless Video Coding

Only a few attempts on lossless video coding methods have been published thus far. The performance of predictive techniques for lossless coding of image sequences of 24 bits per pixel in RGB domain has been investigated in [72], where an adaptive prediction scheme that exploits temporal and spectral correlation in a 3-D colour signal was presented. Those results show that significant advantages can be gained by reducing temporal correlation using motion compensation based on a modified block matching. Moreover, the authors have shown that further advantage can be achieved by considering inter band 3-D predictions among colour bands, rather than coding each band individually.

In another example, a 3-D version of CALIC [32, 33], has been used to exploit inter band correlation in multispectral lossless coding [73]. Also in this case, the authors have shown that considering temporal and inter band dependencies would improve lossless coding results rather than by applying 2-D techniques on individual frames.

Further, there has been an example of extending the modified HINT [41] into 3-D by considering inter frame sub sampling for generalised recursive interpolation (GRINT) [74] for lossless coding of medical image sequences.

In another lossless video coding example, a DCT based lossless video coding scheme compatible with MPEG-2 was introduced [75]. In this method, a modified DCT, followed by a lossless quantisation scheme was designed heuristically exploiting the periodic structure of the coefficients. The main advantage of this type of coding is that the video sequences coded with this algorithm can be decoded with an MPEG-2 decoder for a lossy version.

In summary, from the above examples, it can be concluded that the 3-D predictive coding methods and the modified integer transforms based techniques have already been considered for lossless coding of 3-D data. Further, they have shown that incorporating motion compensation in prediction schemes would improve the lossless video coding.

## 2.4 Thesis Outline

In this chapter, the existing research on the topics relevant to this research was presented. However, little research into lossless video coding has been published. Basically, 3-D predictive methods and modified transforms have been used in lossless video coding. Further, in [72], the authors have shown that higher compression can be achieved by incorporating motion compensation into prediction schemes.

In this research, motion compensation based lossless video coding is considered. The motion compensation framework used in MPEG-2 is used in this research. This breaks this research into two parts, namely, lossless coding techniques that can be used for intra frames and those that can be used for non-intra frames. In entertainment video applications, the exact pixel value recovery is not vital where the end user is a human viewer. In that case, employing embedded to lossless coding provides the added advantage of decoding at lower bit rates from the lossless bit stream. Further embedded decoding facility enables quick previewing or a quick inter-studio transmission via a low bandwidth channel from a losslessly coded master copy.

The first part of this research is mainly concerned with embedded lossless and nearly lossless coding of intra frames (still images). This is normally achieved by employing a transform that maps integers into integers and an embedded quantiser. Although the use of integer wavelet transforms in lossless image coding has been reasonably demonstrated, the use of other transforms has not been considered extensively. This is mainly due to the non availability of integer implementations of such transforms with generic block sizes. In Chapter 3, Integer wavelet transforms are introduced and the integer versions of the DCT, the DST and the Walsh Hadamard (WHT) are derived using their intrinsic properties and the lifting concepts. Integer forms of the above mentioned block transforms are designed for generic block sizes, so that the most suitable transform size for lossless coding can be found by experiment. Further, the concepts of hierarchical non linear sub band splitting techniques are used to derive a new set of integer non-linear transforms for lossless coding experiments.

In Chapter 4, the concept of embedded coding and its framework are introduced. The existing scanning techniques of transform coefficients for embedded coding are analysed and a novel scanning scheme, Adaptive Quadtree Splitting, is introduced based on the integer wavelet coefficients.

Chapter 5 investigates the performance of the Embedded to Lossless Image Coder (ELIC) using the integer transforms discussed in Chapter 3 on intra frames (still images). Further, near-lossless coding performance using pre-quantisation and novel in-transform (online) near-lossless quantisation methods are evaluated. Finally, the rate distortion performance in quasi-lossless decoding is discussed.

Chapter 6 focusses on lossless coding of the motion compensated prediction residuals in non-intra frames. The characteristics of the residuals are presented and the use of the integer transforms presented in Chapter 3 is investigated. Finally, the embedded quantiser ELIC is used to investigate the performance of the integer transforms on residuals and the best lossless coding strategy for the non-intra frames.

Chapter 7 combines the research presented in the previous chapters together. The embedded lossless video coder is introduced and its lossless and nearly lossless coding / decoding performances are analysed, mainly considering the embedded decoding performance at quasi-lossless bit rates in an asymmetric video coding framework. Finally, Chapter 8 concludes this thesis with the conclusions of the findings from this research and possible future work.

### 2.4.1 Test images and video sequences set

The test image set used in this research include 5 images, namely, Gold Hill, Barbara1, Barbara2, Boats and Black board. All images were of the dimensions of 576×704. The images were in grey scale and consisted of 8 bits per pixel. Four sequences with different motion characteristics were used as the test sequences. They were Claire (a talking head), Mobile (horizontal, vertical and rotational object motion coupled with camera motion), Unicycle (vertically moving texts on a moving background) and Kiel harbour (zooming in). All the sequences were in grey scale (Y component only- 8 bits per pixel) and in CIF size (288×352).

# Chapter 3

# Integer Transforms

Transforms that map integers into integers, also known as integer transforms, are the main component of transforms based lossless image coding. The main objective of this chapter is to introduce and design such integer transforms that may be useful in lossless coding of intra frames and non-intra frames. In this chapter, the design, properties and usage in lossless image coding of integer transforms, namely, the Integer Wavelet Transforms (IWT), the Integer Walsh Hadamard Transform (IWHT), the Integer Discrete Cosine Transform (IDCT), the Integer Discrete Sine Transform (IDST) and an Integer Non-Linear Transform (INLT) based on quincunx sub band splitting and median filtering are discussed. The rest of the chapter is organised as follows: Sections 3.1-3.5 present the introduction, design and the lossless performance of the above transforms respectively. Finally, section 3.6 compares and discusses the performance of those transforms on lossless image coding.

## 3.1 The Integer Wavelet Transforms (IWT)

The wavelets are translates and dilates of a fixed function known as the mother wavelet. As mentioned in section 1.2.3, the wavelet transforms can be implemented either using filter banks or using lifting steps. In the lifting method, a filter bank operation is split into a finite sequence of simple filtering steps by performing lifting steps. This corresponds to the factorisation of the polyphase matrix, corresponding to the filter bank into elementary matrices [55, 56, 57, 59, 60, 76]. Lifting has been used in constructing both orthogonal and biorthogonal wavelets [11].

### 3.1.1 The lifting scheme

The idea of lifting was originated as a method of building the second generation wavelets, where the wavelets are not necessarily translates and dilates of the mother wavelet as in the first generation wavelets [76]. The first generation wavelets were constructed with the aid of Fourier transform techniques. But construction of wavelets using the lifting approach does not use Fourier transform techniques. Lifting is entirely a spatial method. The lifting concept can be shown in a block diagram as in Figure 3.1.

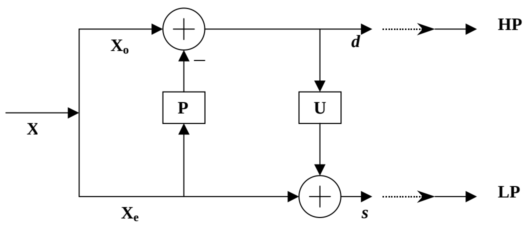


Figure 3.1: Lifting Block Diagram.

The first step of lifting is to separate the original sequence ($X$) into two sub sequences containing the odd indexed samples ($X_o$) and the even indexed samples ($X_e$). This sub sampling step is also called the lazy wavelet transform.

$$X_o : \quad d_i \leftarrow \quad x_{2i+1} \tag{3.1}$$

$$X_e : \quad s_i \leftarrow \quad x_{2i} \tag{3.2}$$

$$for\ i = 0 \ldots (L/2-1),$$

$$where\ L\ is\ the\ signal\ length.$$

Then the lifting steps, dual lifting (P) and primal lifting (U), are performed on these two sequences. The two sets $X_o$ and $X_e$ are closely correlated. So a predictor $P(\,)$ can be used to predict one set from the other. In this prediction step, which is also called dual lifting, the odd indexed samples are predicted using the neighbouring even indexed samples and the prediction error (detail) is recorded replacing the original sample value, thus providing in-place calculations.

$$d_i \leftarrow \quad d_i - P(s_A) \tag{3.3}$$

$$where,\ A = (\,i - \lceil N/2 \rceil + 1, \ldots, i + \lfloor N/2 \rfloor\,)$$

$$N\ is\ the\ number\ of\ dual\ vanishing\ moments\ in\ d.$$

N is the number of dual vanishing moments, which set the smoothness of the P function. This prediction step is also similar to the prediction performed in state-of-the-art lossless coding methods. But in lifting, prediction is only based on the pixels in the same row. In predictive lossless coding, the prediction is based on a causal mask, whereas in lifting, pixels on either sides of the pixel to be predicted are also used in the prediction mask. This is possible due to the sub sampling step performed through the lazy wavelet.

In the second lifting step, primal lifting (U), the even samples are replaced with smoothed values using the update operator $U(\,)$ on previously computed details. The $U(\,)$ operator is designed to maintain the correct running average of the original sequence, in order to avoid aliasing.

$$\begin{aligned} s_i &\leftarrow s_i + U(d_B) \qquad (3.4)\\ where,\ B &= (\,i - \lfloor \tilde{N}/2 \rfloor, \ldots, i + \lceil \tilde{N}/2 \rceil - 1\,)\\ &\tilde{N}\ is\ the\ number\ of\ real\ vanishing\ moments. \end{aligned}$$

The $U(\,)$ operator preserves the first $\tilde{N}$ moments in the $s$ sequence. The lazy wavelet is lifted to a transform with required properties (number of vanishing moments in analysis and synthesis filters as in the filter bank approach) by applying the dual and primal lifting pair of operations one or more times. Finally, the output streams are normalised using the normalising factor $'k'$.

$$d_i \leftarrow d_i \times 1/k \qquad (3.5)$$

$$s_i \leftarrow s_i \times k \qquad (3.6)$$

The output from $s$ channel after dual lifting steps provides a low pass sub band, whereas the output from $d$ channel, after dual lifting steps provides a high pass sub band. The inverse transform is obtained by reversing the order and the sign of the operations performed in the forward transform.

### 3.1.2 Integer wavelets

The above transforms and the filter bank based transforms based on filter banks, result in floating point values as wavelet coefficients. In many applications the input data contains integers. An integer version of transforms is important for lossless compression. Rounding the floating point coefficients to obtain integers does not provide much help as it loses the perfect reconstruction. But the lifting scheme can be easily modified to achieve a transform that maps integers to integers, while maintaining the perfect reconstruction property [56]. This is achieved by rounding off the outputs after $P(\,)$ and $U(\,)$ operators. This rounding off is normally implemented by adding a factor $\frac{1}{2}$ to the outputs of $P(\,)$ and $U(\,)$ operators and subsequently truncating them downward, before subtracting from or adding to the signal channels. Thus, the equations 3.3 and 3.4 become,

$$d_i \leftarrow d_i - \left\lfloor P(s_A) + \tfrac{1}{2} \right\rfloor \tag{3.7}$$

$$s_i \leftarrow s_i + \left\lfloor U(d_B) + \tfrac{1}{2} \right\rfloor \tag{3.8}$$

These rounding operations cause non-linearity in the transform.

The lifting steps are derived by factoring the polyphase matrix, which corresponds to the filter bank, into the elementary matrices. This can be performed in several ways using the Euclidean algorithm [55], which is commonly used to find the greatest common divisor in two polynomials [58]. The most common method of factoring is to achieve a unit normalising factor $'k'$, so that the integer property of the coefficients is preserved. This is normally performed by ignoring the $\sqrt{2}$ normalising factor involved in filter bank design from the polyphase matrix prior to factoring. According to [55], the normalising operator can be replaced by introducing four additional lifting steps into the lifting scheme (see page 38 for the lifting steps for $k = \sqrt{2}$). More discussion on treatment on normalising can be found in Section 4.2.1.1.

### 3.1.3 Tests performed with integer wavelets

The wavelets designed and implemented by the lifting scheme are normally identified with the notation $(N, \tilde{N})$, where $N$ and $\tilde{N}$ are the number of vanishing moments in the dual and primal lifting steps respectively, as opposed to the number of filter taps in the filter bank implementation. The wavelets with different combinations of vanishing moments, namely (2,2), (4,2), (4,4), (2+2,2), (2,4), (6,2), (all from [56]), (1,1) (also known as S transform and the same as Haar wavelet) and S+P [63] were used in the following experiments for lossless image coding. The lifting steps for those integer wavelets can be found in Appendix A.

#### 3.1.3.1 The treatment at signal boundaries

The $P()$ and $U()$ operators use a template centred on the signal component to be predicted and updated respectively. These symmetric templates cannot be used at the signal boundaries, thus, requiring a special treatment at the boundaries.

The same problem arises in filtering operations in the filter bank realisation of the wavelets. In filter bank realisation, three types of boundary treatments, namely zero padding, circular extension and symmetric extension [77], are commonly used. In zero padding, the signal is padded with zeros according to the filter length, whereas in circular extension, the signal is repeated according to the filter length, so that the circular convolution operations can be performed. In the symmetric extension method, the signal is reflected at the boundary on a half or whole point, according to the symmetry and the length of the filter. It is widely believed that the symmetric extension method provides the best performance in lossy image coding [6, 77].

The same three extension methods can be employed in the lifting approach. A new boundary treatment for lifting by changing the symmetry of the prediction and update templates near boundaries has been presented in [76]. In this method, since the templates are not placed centering the component to be predicted or updated, the weights of the components in the template have to be recomputed according to the interpolation polynomial generated by the required vanishing moments of the wavelet. In this research, a symmetric extension based method, as shown below, was used as the boundary treatment, due to its simplicity and better performance in lossy compression.

In P lifting :

$$\cdots\ s[i-1] \qquad s[i]\ d[i]\ s[i+1] \qquad s[i+2]\ \cdots$$

For example, $d[i]$ is predicted using $s[i-1]\ldots s[i+2]$. The prediction templates at $i = 0$, $i = L-1$ and $i = L-2$, where L is the sub sample length, using this boundary treatment are as below.

For $i = 0 : \{s[2]\ s[0]\ s[1]\ s[2]\}$
For $i = L-1 : \{s[L-2]\ s[L-1]\ s[L-1]\ s[L-2]\}$
For $i = L-2 : \{s[L-3]\ s[L-2]\ s[L-1]\ s[L-3]\}$

In U lifting :

$$\cdots\ d[i-2] \qquad d[i-1]\ s[i]\ d[i] \qquad d[i+1]\ \cdots$$

For example, $s[i]$ is updated using $d[i-2]\ldots d[i+1]$. The prediction templates at $i = 0$, $i = 1$ and $i = L-1$, where L is the sub sample length, using this boundary treatment are as below.

For $i = 0 : \{d[1]\ d[0]\ d[0]\ d[1]\}$
For $i = 1 : \{d[2]\ d[0]\ d[1]\ d[2]\}$
For $i = L-1 : \{d[L-3]\ d[L-2]\ d[L-1]\ d[L-3]\}$

In this method, the neighbours nearest to the component to be predicted / updated are considered in the prediction / update templates.

#### 3.1.3.2 The zero-order entropy values

All the transforms were applied on the images in the test image set (in section 2.4.1) up to 5 levels of iterations. The weighted average zero-order entropy values (in bpp), $R0$, calculated using equation 3.9 by considering the zero-order entropy value $H0$ for individual sub bands using the equation 1.11 (on page 11), are as in Table 3.1.

$$R0 = \frac{1}{N}\sum_{i=0}^{3\times s} N_i\, H0_i \tag{3.9}$$

$$\begin{aligned}&where, N\ is\ the\ total\ pixels\ in\ the\ image,\\ &s\ is\ the\ number\ of\ scales,\\ &N_i\ is\ the\ total\ pixels\ in\ the\ i^{th}\ sub\ band,\\ &H0_i\ is\ the\ zero-order\ entropy\ in\ the\ i^{th}\ sub\ band.\end{aligned}$$

| | (2,2) | (4,2) | (4,4) | (2+2,2) | (2,4) | (6,2) | S | S+P |
|---|---|---|---|---|---|---|---|---|
| Gold Hill | 4.705 | 4.702 | 4.702 | 4.694 | 4.715 | 4.718 | 5.038 | 4.759 |
| Barbara1 | 4.958 | 4.810 | 4.787 | 4.808 | 4.951 | 4.767 | 5.487 | 4.876 |
| Barbara2 | 5.066 | 5.024 | 5.008 | 5.024 | 5.062 | 5.030 | 5.453 | 5.041 |
| Boats | 4.234 | 4.195 | 4.192 | 4.183 | 4.241 | 4.197 | 4.643 | 4.269 |
| Black Board | 3.888 | 3.886 | 3.878 | 3.870 | 3.893 | 3.897 | 4.172 | 3.974 |
| Average | 4.570 | 4.523 | 4.513 | 4.516 | 4.572 | 4.522 | 4.959 | 4.584 |

Table 3.1: Weighted zero-order entropy values in bpp for IWT.

From the above results, it is evident that the wavelets with a greater number of vanishing moments in the prediction lifting step provide better lossless performance compared to the wavelets with a fewer number of vanishing moments in the prediction step. This is due to the high inter pixel correlation present in natural images (as in intra frames in video) and the wider prediction masks that lead to better prediction. Further, it can be seen that the extra prediction step in the S+P transform and the (2+2,2) transform have resulted in better performance compared to the S transform and the (2,2) transform, respectively. On average, the (4,4) wavelet reported the lowest entropy values. Therefore, the (4,4) wavelet has been used as the preferred lifting steps for the succeeding intra frame research presented in this thesis. The performance of the above integer wavelets on residuals in non-intra frames are presented in section 6.3.

## 3.2 The Integer Walsh Hadamard Transform (IWHT)

The Walsh transform uses Walsh functions (WAL) as the basis vectors. The Walsh functions form an ordered set of rectangular waveforms with amplitude values of $\pm 1$ [78, 79]. They are defined by two arguments, namely a time period ($t$) and an ordering number ($n$) representing the sequency, which corresponds to the frequency in sinusoidal waves. The sequency number corresponds to the number of sign changes (or zero crossings) in a rectangular waveform.

The $\mathrm{WAL}_N(n\,,t)$ for an $N$-point Walsh function set is defined as in equation 3.10

$$WAL_N(n,t) = \begin{cases} 1\times \quad Sign\,(\cos\,(\,\frac{n}{2}\,,\,t+\frac{1}{2}\,)\,) & ;\quad n \quad even \\ 1\times \quad Sign\,(\sin\,(\,\frac{n+1}{2}\,,\,t+\frac{1}{2}\,)\,) & ;\quad n \quad odd \end{cases} \tag{3.10}$$
$$for\; n = 0,\ldots,N-1.$$
$$for\; t = 0,\ldots,N-1.$$

The Walsh functions are orthogonal [78]. Thereby,

$$\sum_{t=0}^{N-1} WAL_N(l\,,t)\,WAL_N(m\,,t) = \begin{cases} N & ;\quad l=m \\ 0 & ;\quad l\neq m \end{cases} \tag{3.11}$$

An orthonormal basis vector set can be obtained by dividing these Walsh functions by $\sqrt{N}$. Therefore, the $N$-point Walsh transform pair (forward and inverse) for 1-D data sequence, $x$, of length, N, is defined as in equations 3.12 and 3.13 respectively.

$$X(k) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} x(i)\,WAL_N(k\,,i) \tag{3.12}$$
$$for\; k = 0,1,\ldots,N-1.$$

$$x(i) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X(k)\,WAL_N(k\,,i) \tag{3.13}$$
$$for\; i = 0,1,\ldots,N-1.$$

### 3.2.1 The Walsh Hadamard transform

In the Walsh transform, the Walsh function set $WAL_N$ can be considered as the transform matrix. It is unitary and symmetric. The rows of the transform matrix are the basis vectors of $WAL_N$, which are ordered in the increasing sequency order. There are a few other ways of arranging the basis vectors. One such method is according to the Natural order (or Hadamard order) derived from successive Kroneckor products ($\otimes$) of the lower order Walsh (Hadamard) matrices while preserving the symmetry and the orthogonality [78]. This ordering is commonly known as the Walsh Hadamard transform. The lowest order Walsh Hadamard matrix $WH_N$ is the order of two ($N = 2$) as in equation 3.14.

$$WH_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3.14}$$

Thereby, the Walsh Hadamard matrix for the higher orders which are integer powers of two can be defined using Kroneckor products as in equation 3.15.

$$WH_N = WH_{N/2} \otimes WH_2 \tag{3.15}$$

This leads to the redefinition of the $N$-point discrete Walsh Hadamard transform pair as in equations 3.16 and 3.17 respectively.

$$X(k) = \sum_{i=0}^{N-1} x(i)\, WH_N(k\,, i) \tag{3.16}$$
$$for\; k = 0, 1, \ldots, N-1.$$

$$x(i) = \sum_{k=0}^{N-1} X(k)\, WH_N(k\,, i) \tag{3.17}$$
$$for\; i = 0, 1, \ldots, N-1.$$

### 3.2.2 The IWHT by lifting

The division by $\sqrt{N}$ in the $WAL_N$ functions (or the division by a series of $\sqrt{2}$ factors in the $WH_N$ matrices) causes non-integer values in the transformed coefficients, rounding of which loses the unitary nature, thus resulting in lossy reconstruction. The previous versions of integer Walsh Hadamard transforms have been implemented by ignoring the normalising factors and incorporating a lossless quantisation scheme [51, 52]. Further, they were designed heuristically only for the 8-point WHTs. In this section, a novel lossless Walsh Hadamard transform, which maps integers into integers, is introduced. Unlike the previous integer WHTs, this derivation incorporates the normalisation factors into the implementation and considers any block size, N, where N is an integer power of two.

The equation 3.16 can be represented in matrix form as in equation 3.18.

$$\mathbf{X} = \mathbf{WH_N}\ \mathbf{x} \tag{3.18}$$

This can be rewritten using equations 3.15 and 3.14 as in equation 3.19, which in turn can be written as in equation 3.20 using matrix partitions.

$$\mathbf{X} = \frac{\mathbf{1}}{\sqrt{\mathbf{2}}} \begin{bmatrix} \mathbf{WH_{N/2}} & \mathbf{WH_{N/2}} \\ \mathbf{WH_{N/2}} & -\mathbf{WH_{N/2}} \end{bmatrix} \mathbf{x} \tag{3.19}$$

$$\left[\begin{array}{c} \mathbf{X_1} \\ \hline \mathbf{X_2} \end{array}\right] = \frac{\mathbf{1}}{\sqrt{\mathbf{2}}} \left[\begin{array}{c|c} \mathbf{WH_{N/2}} & \mathbf{WH_{N/2}} \\ \hline \mathbf{WH_{N/2}} & -\mathbf{WH_{N/2}} \end{array}\right] \left[\begin{array}{c} \mathbf{x_1} \\ \hline \mathbf{x_2} \end{array}\right] \tag{3.20}$$

$$\textit{where, } \mathbf{X_1} = \mathbf{X}(0,\ldots,N/2-1),\ \mathbf{X_2} = \mathbf{X}(N/2,\ldots,N-1)$$
$$\textit{and } \mathbf{x_1} = \mathbf{x}(0,\ldots,N/2-1),\ \mathbf{x_2} = \mathbf{x}(N/2,\ldots,N-1)$$

The equation 3.20 can be simplified into the form in equation 3.22 using partitioned matrix multiplication.

$$\left[\begin{array}{c} \mathbf{X_1} \\ \hline \mathbf{X_2} \end{array}\right] = \frac{\mathbf{1}}{\sqrt{\mathbf{2}}} \left[\begin{array}{c} \mathbf{WH_{N/2}}\,\mathbf{x_1} + \mathbf{WH_{N/2}}\,\mathbf{x_2} \\ \hline \mathbf{WH_{N/2}}\,\mathbf{x_1} - \mathbf{WH_{N/2}}\,\mathbf{x_2} \end{array}\right] \tag{3.21}$$

$$\Longrightarrow \left[\begin{array}{c} \mathbf{X_1} \\ \hline \mathbf{X_2} \end{array}\right] = \left[\begin{array}{c} \mathbf{WH_{N/2}}\,\mathbf{U_1} \\ \hline \mathbf{WH_{N/2}}\,\mathbf{U_2} \end{array}\right] \tag{3.22}$$

$$\textit{where, } \left[\begin{array}{c} \mathbf{U_1} \\ \hline \mathbf{U_2} \end{array}\right] = \underbrace{\frac{1}{\sqrt{2}} \left[\begin{array}{c|c} \mathbf{I_{N/2}} & \mathbf{I_{N/2}} \\ \hline \mathbf{I_{N/2}} & -\mathbf{I_{N/2}} \end{array}\right]}_{\mathbf{A}} \left[\begin{array}{c} \mathbf{x_1} \\ \hline \mathbf{x_2} \end{array}\right] \tag{3.23}$$

$$\textit{where, } \mathbf{I_{N/2}} \textit{ is the Identity matrix of size } N/2$$
$$\textit{and } \mathbf{A} \textit{ corresponds to the } \mathbf{WH_2}.$$

The matrix A in equation 3.23 corresponds to a set of 2-point Walsh Hadamard matrices ($\mathbf{WH_2}$). The higher and the lower partitions of the right hand side of equation 3.22, which are the same as the original form in equation 3.18, can be rewritten using the relationship in equations 3.22 and 3.23 recursively until $N = 2$. Thereby, the Walsh Hadamard transform can be implemented using the $\mathbf{WH_2}$, applied recursively according to a binary partition tree in the higher and the lower partitions of the $\mathbf{WH_N}$ matrix. The pseudo code for implementing the $\mathbf{WH_N}$ is as in Figure 3.2.

```
function  WH_N (x, StartPosition, N)
{
   WH_2 (x, StartPosition, N)  % Do WH_2
   if (N>2)
   {
      WH_N (x, StartPosition, N/2);    % Lower half
      WH_N (x, StartPosition+N/2, N/2);   % Upper half
   }
}
```

Figure 3.2: The Pseudo code for forward $\mathbf{WH_N}$.

#### 3.2.2.1 Integer implementation of $\mathbf{WH_2}$

The matrix $\mathbf{WH_2}$, the main building block of the $WH_N$ transform, is unitary. Any $2\times 2$ unitary matrix $(B = [\,a\ b\,;\ c\ d\,])$ with $b \neq 0$ can be factorised into a combination of upper and lower triangular matrices to form a ladder network [80], so that the rounding operations can be performed as in lifting steps. Moreover, the matrix $\mathbf{WH_2}$ is similar to the polyphase matrix of the S transform, the only difference being the symmetry in $\mathbf{WH_2}$. Therefore, a similar factorisation as in the S transform can be performed to obtain the lifting steps as below.

$$\mathbf{WH_2} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \underbrace{\begin{bmatrix} \sqrt{2} & 0 \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix}}_{\mathbf{K}} \underbrace{\begin{bmatrix} 1 & -\frac{1}{2} \\ 0 & 1 \end{bmatrix}}_{\mathbf{U}} \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix}}_{\mathbf{P}} \tag{3.24}$$

The matrix $\mathbf{P}$ corresponds to the prediction step (dual lifting) and the matrix $\mathbf{U}$ corresponds to the updating step (primal lifting) as in lifting terminology (section 3.1). They can be implemented in lifting steps with rounding operations as in equations 3.25 and 3.26. They are similar to the lifting steps in the S transform (See Appendix A equations A.3 and A.4), which is also the integer form of the Haar Wavelet.

$$\textit{From } \mathbf{P}: \quad x_2 \leftarrow -x_2 + x_1 \tag{3.25}$$

$$\textit{From } \mathbf{U}: \quad x_1 \leftarrow x_1 - \left\lfloor \frac{1}{2}(x_2) + \frac{1}{2} \right\rfloor \tag{3.26}$$

#### 3.2.2.2 Implementing the scaling matrix (K)

The scaling factor **K** can be incorporated in either of two ways. In the 1-D Walsh Hadamard transform, the matrix **K** can be replaced by additional four lifting steps as in equation 3.27 [55].

$$\begin{bmatrix} \sqrt{2} & 0 \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}}_{\mathbf{K3}} \underbrace{\begin{bmatrix} 1 & 1-\frac{1}{\sqrt{2}} \\ 0 & 1 \end{bmatrix}}_{\mathbf{K2}} \underbrace{\begin{bmatrix} 1 & 0 \\ \sqrt{2} & 1 \end{bmatrix}}_{\mathbf{K1}} \underbrace{\begin{bmatrix} 1 & \frac{1-\sqrt{2}}{2} \\ 0 & 1 \end{bmatrix}}_{\mathbf{K0}} \tag{3.27}$$

The lifting steps are as below.

$$From\ \mathbf{K0}: \quad x_1 \leftarrow x_1 + \left\lfloor \frac{1-\sqrt{2}}{2} x_2 + \frac{1}{2} \right\rfloor \tag{3.28}$$

$$From\ \mathbf{K1}: \quad x_2 \leftarrow x_2 - \left\lfloor \sqrt{2}\, x_1 + \frac{1}{2} \right\rfloor \tag{3.29}$$

$$From\ \mathbf{K2}: \quad x_1 \leftarrow x_1 - \left\lfloor (1-\frac{1}{\sqrt{2}})\, x_2 + \frac{1}{2} \right\rfloor \tag{3.30}$$

$$From\ \mathbf{K3}: \quad x_2 \leftarrow x_2 - x_1 \tag{3.31}$$

However, in a 2-D transform, the effect of scaling becomes a multiplication by a scaling mask, **KK**, recursively along a quad tree in the coefficient domain.

$$\mathbf{KK} = \begin{array}{|c|c|} \hline \times\, 2 & \times\, 1 \\ \hline \times\, 1 & \times\, \frac{1}{2} \\ \hline \end{array}$$

Therefore, in the 2-D transform, the use of matrix **K** can be avoided and the net effect of using it can be considered in the quantising and entropy coding stages, as will be discussed in Section 4.2.1.2.

#### 3.2.2.3 Another approach for integer WH₂

The $\mathbf{WH_2}$ matrix (equation 3.14) possesses a unit determinant and a non-zero $(0,1)^{th}$ element. Therefore, it can be factorised into a product of upper and lower triangular matrices as shown in eqation 3.32 [80].

$$\mathbf{WH_2} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \frac{\cos(\frac{\pi}{4})-1}{\sin(\frac{\pi}{4})} \\ 0 & 1 \end{bmatrix}}_{F2} \underbrace{\begin{bmatrix} 1 & 0 \\ \sin(\frac{\pi}{4}) & 1 \end{bmatrix}}_{F1} \underbrace{\begin{bmatrix} 1 & \frac{\cos(\frac{\pi}{4})-1}{\sin(\frac{\pi}{4})} \\ 0 & 1 \end{bmatrix}}_{F0} \tag{3.32}$$

The corresponding lifting steps with rounding operations for the integer realisation are as below.

$$\textit{From } \mathbf{F0}: \quad x_1 \leftarrow x_1 + \left\lfloor \frac{\cos(\frac{\pi}{4})-1}{\sin(\frac{\pi}{4})}\, x_2 + \frac{1}{2} \right\rfloor \tag{3.33}$$

$$\textit{From } \mathbf{F1}: \quad x_2 \leftarrow x_2 + \left\lfloor \sin(\frac{\pi}{4})\, x_1 + \frac{1}{2} \right\rfloor \tag{3.34}$$

$$\textit{From } \mathbf{F2}: \quad x_1 \leftarrow x_1 + \left\lfloor \frac{\cos(\frac{\pi}{4})-1}{\sin(\frac{\pi}{4})}\, x_2 + \frac{1}{2} \right\rfloor \tag{3.35}$$

Only three lifting steps are involved in this method, whereas five lifting steps are involved in the **KUP** factorisation method in sections 3.2.2.1 and 3.2.2.2.

#### 3.2.2.4 Output in sequency order

In this implementation, which is based on the Kroneckor product representation of the $\mathbf{WH_N}$, the output is not ordered in the increasing sequency order, but in the Natural order. But for applications such as embedded image coding, the output in increasing sequency order is important. An increasing sequency ordered output can be obtained by permuting the input. This is based on the symmetry of the $\mathbf{WH_N}$ matrix. The rows of the $WAL_N$ matrix have been permuted to the Natural order in obtaining the $\mathbf{WH_N}$. The columns of $\mathbf{WH_N}$ correspond to the input. Due to symmetry, the Natural order arrangement can be set off by rearranging the columns, which in turn corresponds to rearranging the input order of the data. The permutation index (`PermIndex`) for an N-point transform (where N is an integer power of two) can be computed as in Figure 3.3. Then the input sequence, $x(i)$, in sequency order is $x_s(i) = x(\texttt{PermIndex}(i))$ for $i = 0, \ldots, N-1$.

```
function [PermIndex]=GetIndex(N)
  if (N==2)
    PermIndex=[0 1];
  else
    PermIndex(1:2:N-1)=GetIndex(N/2);
    PermIndex(2:2:N)=N-1-PermIndex(1:2:N-1);
  end
```

Figure 3.3: MATLAB code for the permutation index.

### 3.2.2.5 Block diagram for integer Walsh Hadamard transform

The above presented integer implementation of the $\mathbf{WH_N}$ can be summarised in a block diagram as in Figure 3.4. The inverse transform can be implemented by reversing the recursive operations and the lifting steps by changing the sign and the order of operation of the lifting equations 3.25, 3.26 and the scaling factor equations. The signal flow diagram for the forward integer $\mathbf{WH_8}$ avoiding the scaling factors is as in Figure 3.5.

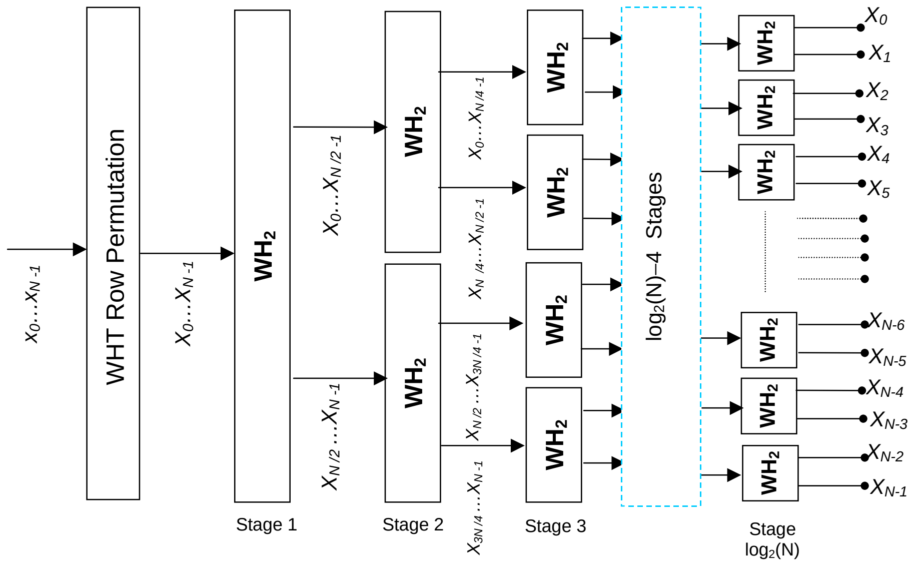


Figure 3.4: Block diagram for integer $\mathbf{WH_N}$.

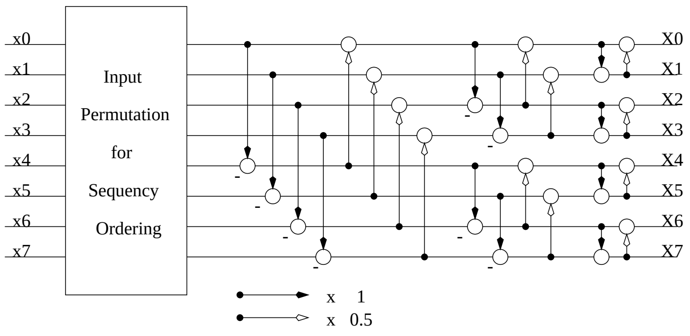


Figure 3.5: Signal flow diagram for $\mathbf{WH_8}$.

### 3.2.3 The zero-order entropy values

#### The block to tree structre rearrangement

The WHT$_N$ is performed on the partitioned image blocks of size N×N. According to this implementation, the WHT$_N$ is analogous to a 1-D wavelet packet transform using **WH$_2$** (which in turn is analogous to Haar transform) performing along a complete binary tree. The 2-D WHT is analogous to a separable 1-D wavelet packet transform applied to rows and columns respectively. Therefore, the transformed blocks can be reorganised into wavelet packet sub bands. This rearrangement is shown in Figure 3.7 as a pseudo code and in Figure 3.6 pictorially for N=4 on a 8×8 image. The number of scales in the corresponding wavelet packet transform is log$_2$(N) and the total number of packet sub bands is N$^2$. This rearrangement allows the wavelet coding techniques to be used for coding the IWHT coefficients.

WHT$_4$ Blocks

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $00_0$ | $01_0$ | $02_0$ | $03_0$ | $00_1$ | $01_1$ | $02_1$ | $03_1$ |
| $10_0$ | $11_0$ | $12_0$ | $13_0$ | $10_1$ | $11_1$ | $12_1$ | $13_1$ |
| $20_0$ | $21_0$ | $22_0$ | $23_0$ | $20_1$ | $21_1$ | $22_1$ | $23_1$ |
| $30_0$ | $31_0$ | $32_0$ | $33_0$ | $30_1$ | $31_1$ | $32_1$ | $33_1$ |
| $00_2$ | $01_2$ | $02_2$ | $03_2$ | $00_3$ | $01_3$ | $02_3$ | $03_3$ |
| $10_2$ | $11_2$ | $12_2$ | $13_2$ | $10_3$ | $11_3$ | $12_3$ | $13_3$ |
| $20_2$ | $21_2$ | $22_2$ | $23_2$ | $20_3$ | $21_3$ | $22_3$ | $23_3$ |
| $30_2$ | $31_2$ | $32_2$ | $33_2$ | $30_3$ | $31_3$ | $32_3$ | $33_3$ |

$$\Downarrow$$

WHT$_4$ Wavelet packet sub bands

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $00_0$ | $00_1$ | $01_0$ | $01_1$ | $02_0$ | $02_1$ | $03_0$ | $03_1$ |
| $00_2$ | $00_3$ | $01_2$ | $01_3$ | $02_2$ | $02_3$ | $03_2$ | $03_3$ |
| $10_0$ | $10_1$ | $11_0$ | $11_1$ | $12_0$ | $12_1$ | $13_0$ | $13_1$ |
| $10_2$ | $10_3$ | $11_2$ | $11_3$ | $12_2$ | $12_3$ | $13_2$ | $13_3$ |
| $20_0$ | $20_1$ | $21_0$ | $21_1$ | $22_0$ | $22_1$ | $23_0$ | $23_1$ |
| $20_2$ | $20_3$ | $21_2$ | $21_3$ | $22_2$ | $22_3$ | $23_2$ | $23_3$ |
| $30_0$ | $30_1$ | $31_0$ | $31_1$ | $32_0$ | $32_1$ | $33_0$ | $33_1$ |
| $30_2$ | $30_3$ | $31_2$ | $31_3$ | $32_2$ | $32_3$ | $33_2$ | $33_3$ |

Figure 3.6: Blocks to wavelet tree rearrangement for N=4 on 8×8 image

```
Block2Tree( Blocks,Tree,rows,columns,tree_levels)
{
BlkSize = 2^(tree_leveles);
rblks = rows/BlkSize;
cblks = columns/BlkSize;

for (i=0:rblks-1)
  for (j=0:cblks-1)
    for (p=0:BlkSize-1)
      for (q=0:BlkSize-1)
        Tree[i+rblks*p][j+cblks*q]=Blocks[i*BlkSize+p][j*BlkSize+q];
}
```

Figure 3.7: The pseudo code for blocks to wavelet tree rearrangement.

**The zero-order entropy values**

The zero-order entropy values, calculated using the weighted entropy equation 3.9, for the wavelet packet sub band representation of the IWHT applied on the test image set for different block sizes are as in Table 3.2.

| N | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| Gold Hill | 5.625 | 5.122 | 4.995 | 4.949 | 4.870 |
| Barbara1 | 5.959 | 5.497 | 5.304 | 5.265 | 5.172 |
| Barbara2 | 5.928 | 5.523 | 5.404 | 5.350 | 5.253 |
| Boats | 5.191 | 4.720 | 4.650 | 4.683 | 4.663 |
| Black Board | 4.794 | 4.270 | 4.182 | 4.238 | 4.247 |
| Average | 5.499 | 5.026 | 4.907 | 4.897 | 4.841 |

Table 3.2: Weighted zero-order entropy values in bpp for IWHT

The weighted entropy values in Table 3.2 show that the greater the block size, N, the lower the weighted entropy values. All images, except the black board image, have recorded their lowest entropy values for block sizes of 32, which is also analogous with a 5 scale ($\log_2 32$) wavelet packet transform. However, it is evident that the net benefit obtained from larger N decreases with the increasing N. For example, the net gain achieved by increasing N from 8 to 16 is lower than the gain achieved by increasing N from 4 to 8. Therefore, in the case of a low complexity requirement N=8 is a better option.

## 3.3 The Integer Discrete Cosine Transform (IDCT)

The DCT-II, defined in equation 1.5 on page 6, uses cosine functions as basis vectors. Therefore, the transform matrix consists of values between 1 and -1, resulting non integer coefficients in the transform domain. In this section, current methods on designing the IDCT-II and a new method for an N-point IDCT-II where N is any integer power of two are presented.

The design of the IDCT algorithms can be influenced from the design concepts used in designing the fast DCT algorithms, as they involve factorising the DCT matrix into sparse matrices. For example, the fast algorithms based on Culey-Tukey type (decimation-in-time) and Sande-Tukey (decimation-in-frequency) type algorithms [81] using butterfly representation, which are also similar to lattice structures, can be converted into ladder structures [6], so that the lifting techniques can be employed. Already published fast algorithms for computing the DCT can be categorised into three groups based on their methods of approach as below.

1. Direct methods.
2. Indirect methods.
3. Recursive methods.

The direct methods use sparse matrix factorisation of the DCT matrix. The DCT matrix is an unitary matrix, which can easily be factorised into products of sparse matrices [82, 83, 84, 85]. The indirect methods include using the fast Fourier transform (FFT) [86, 87, 88], the Walsh-Hadamard transform (WHT) techniques [9, 89] and the Hartley transform [90] to compute the DCT coefficients. In the recursive algorithms the higher order DCT is computed from the lower order DCT coefficients [91, 92, 93].

The already published IDCT algorithms mainly use the direct factorisation techniques [53] and the indirect deriving methods [94].

The technique presented in [53] (LDCT), uses direct factorisation to decompose the un-normalised 8-point DCT-II matrix into the product of matrices $\mathbf{DP_1LUP_2}$, where $\mathbf{P_1}$ and $\mathbf{P_2}$ are permutation matrices, $\mathbf{D}$ is a diagonal matrix representing the scaling operations and $\mathbf{L}$ and $\mathbf{U}$ are lower and upper triangular matrices with $\mathbf{L}_{i,i} = \mathbf{U}_{i,i} = 1$, $i = 0, \ldots, 7$. The integer coefficients $c$ for an input vector x are obtained by $c = \mathbf{P_1}[\mathbf{L}[\mathbf{UP_2x}]]$, where [.] denotes rounding to the nearest integer. The above decompositions are not unique. The authors have derived the particular decomposition heuristically. With this method, separate decompositions have to be performed for the other DCT sizes.

In [94] (IntDCT), the authors have used an indirect factorisation based on the Walsh Hadamard transform (*WHT*) to decompose the N-point DCT-II matrix into the product of matrices $\sqrt{1/N}\mathbf{BTBH_W}$, where $\mathbf{B}$ is the bit reversal operation, $\mathbf{H_W}$ is the un-normalised *WHT* and $\mathbf{T}$ is a block diagonal matrix. The blocks in the $\mathbf{T}$ matrix were implemented using the lifting factorisation of Givens rotations as introduced in the perfect reconstruction network techniques in [80].

In both the above methods, the authors have either ignored the scaling factors or have incorporated them into the input data. The IntDCT has been designed for an 8-point DCT. It can be extended for N values higher than 8, by computing the angles and their Givens rotations for the diagonal block matrices that require further sparse factorisation operations when their dimensions are higher than two.

A novel technique for N-point IDCT-II with normalised coefficients using recursive methods and lifting techniques is discussed in the following section. This derivation can be used for any N-point IDCT-II, where N is a power of 2.

### 3.3.1 The IDCT using recursive methods and lifting

The DCT-II, shown in equation 1.5, can be rewritten as in equation 3.36.

$$\mathbf{X} = \alpha_{\mathbf{N}}\, \mathbf{D_N}\, \mathbf{x} \tag{3.36}$$

where, $\mathbf{X}$ and $\mathbf{x}$ are column vectors of size $N \times 1$ denoting the DCT output in increasing frequency order and the input data sequences respectively. $\mathbf{D_N}$ is the N-point DCT matrix of size $N \times N$ and $\alpha_{\mathbf{N}}$ is a diagonal matrix of size $N \times N$ denoting the normalising constants.

In the recursive fast DCT algorithm presented in [91], the matrix $\mathbf{D}_N$ was rearranged by permuting its columns and rows, so that it can be partitioned into four quadrants. The row permutation causes the even indexed rows grouped in the upper half of $\mathbf{D}_N$ and the odd indexed rows grouped in the lower half of $\mathbf{D}_N$, which in turn causes the same ordering in $\mathbf{X}$. The column permutation arranges the columns in a way such that, for the even indexed rows, the left and the right quadrants are the same and for the odd indexed rows, the left and the right quadrants are opposite in sign. The permutated coefficient matrices for N=2, 4 and 8 can be found in Appendix B. Since the column permutation operation directly relates to the input data, in order to compensate the column permutation, the input vector **x** is rearranged as in equation 3.37.

$$\left.\begin{array}{lcl}\tilde{\mathbf{x}}(n) & = & \mathbf{x}(2n)\\ \tilde{\mathbf{x}}(N+\frac{n}{2}) & = & \mathbf{x}(N-(2n+1))\end{array}\right\} \; n=0,\ldots,N/2-1. \tag{3.37}$$

The effect of the above permutation operations can be mathematically depicted as below (equations 3.38–3.43).

The un-normalised N-point DCT-II coefficients $X(k)$, where k=0,. . . ,N-1 for the input signal $x(n)$, where n=0,. . . ,N-1 are as in equation 3.38.

$$X(k)=\sum_{n=0}^{N-1}x(n)\cos\left((2n+1)\frac{\pi k}{2N}\right) \tag{3.38}$$

Using equation 3.38, the even indexed coefficients can be written as in equation 3.39.

$$X(2k)=\sum_{n=0}^{N-1}x(n)\cos\left((2n+1)\frac{\pi 2k}{2N}\right) \tag{3.39}$$

Using the rearranged input sequence $\tilde{x}$ as in equation 3.37, the above can be rewritten as in equation 3.40.

$$\begin{aligned} X(2k) &= \sum_{n=0}^{\frac{N}{2}-1}\tilde{x}(n)\cos\left((2\,(2n){+}1)\frac{\pi k}{2\frac{N}{2}}\right)+\sum_{n=0}^{\frac{N}{2}-1}\tilde{x}(\tfrac{N}{2}+n)\cos\left((2\,(N{-}\,(2n{+}1)){+}1)\frac{\pi k}{2\frac{N}{2}}\right)\\ &= \sum_{n=0}^{\frac{N}{2}-1}\tilde{x}(n)\cos\left((4n{+}1)\frac{\pi k}{2\frac{N}{2}}\right)+\sum_{n=0}^{\frac{N}{2}-1}\tilde{x}(\tfrac{N}{2}+n)\cos\left(2\pi k{-}(4n{+}1)\frac{\pi k}{2\frac{N}{2}}\right)\\ &= \sum_{n=0}^{\frac{N}{2}-1}\left[\tilde{x}(n)+\,\tilde{x}(\tfrac{N}{2}+n)\right]\cos\left((4n{+}1)\frac{\pi k}{2\frac{N}{2}}\right)\\ &\quad (for\;\; k=0,\ldots,\frac{N}{2}-1). \end{aligned} \tag{3.40}$$

Further, as can be verified from the transform matrices in Appendix B, it can be shown that,

$$\left.\begin{array}{c}\cos\left((4n{+}1)\frac{\pi k}{2\frac{N}{2}}\right)\\ for\;\; n=0,\ldots,N/2\;-1\end{array}\right\}=\left\{\begin{array}{c}D_M=\cos\left((2n{+}1)\frac{\pi k}{2M}\right)\\ for\;\; n=0,2,\ldots,M{-}2,M{-}1,M{-}3,\ldots,1\\ where,\;\; M=\frac{N}{2}\end{array}\right. \tag{3.41}$$

Thereby, the equation 3.40 becomes,

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} \left[\tilde{x}(n) + \tilde{x}(\tfrac{N}{2}+n)\right] \underbrace{\cos\left((4n+1)\frac{\pi k}{2\frac{N}{2}}\right)}_{\frac{N}{2}\ point\ DCT} \tag{3.42}$$

The equation 3.42 shows that the even indexed coefficient calculations reduce to an $\frac{N}{2}$-point DCT-II coefficient calculation for the sum of the upper and the lower halves of the input signal $\tilde{x}$.

The odd indexed coefficients can be rewritten as in equation 3.43 using a similar approach used for the even indexed coefficients.

$$\begin{aligned}
X(2k+1) &= \sum_{n=0}^{N-1} x(n)\cos\left((2n+1)\frac{\pi(2k+1)}{2N}\right) \\
&= \sum_{n=0}^{\frac{N}{2}-1} \tilde{x}(n)\cos\left((4n+1)\frac{\pi(2k+1)}{2N}\right) \\
&\quad + \sum_{n=0}^{\frac{N}{2}-1} \tilde{x}(\tfrac{N}{2}+n)\cos\left((2\,(N-(2n+1))-1)\frac{\pi(2k+1)}{2\frac{N}{2}}\right) \\
&= \sum_{n=0}^{\frac{N}{2}-1} \tilde{x}(n)\cos\left((4n+1)\frac{\pi(2k+1)}{2N}\right) \\
&\quad + \sum_{n=0}^{\frac{N}{2}-1} \tilde{x}(\tfrac{N}{2}+n)\cos\left(\pi(2k+1) - (4n+1)\frac{\pi(2k+1)}{2N}\right) \\
&= \sum_{n=0}^{\frac{N}{2}-1} \left[\tilde{x}(n)-\tilde{x}(\tfrac{N}{2}+n)\right] \underbrace{\cos\left((4n+1)\frac{\pi(2k+1)}{2N}\right)}_{E_{\frac{N}{2}}} \\
&\quad (for\ \ k = 0, \ldots, \frac{N}{2}-1).
\end{aligned} \tag{3.43}$$

The equation 3.43 shows that the odd indexed coefficient calculations reduce to a matrix multiplication of the difference of the upper and the lower halves of the input signal vector $\tilde{x}$ with the matrix $E(\frac{N}{2})$, the derivation of which for lossless implementation is discussed in section 3.3.1.3.

#### 3.3.1.1 Incorporating the normalising factors

The normalising factor $\alpha_N$ for each coefficient is as in equation 3.44.

$$\begin{aligned} \alpha_N(k) &= \sqrt{\frac{2}{N}}\,\epsilon_k \qquad (3.44) \\ where,\ \epsilon_k &= \begin{cases} \frac{1}{\sqrt{2}} & if\ k = 0\,, \\ 1 & else. \end{cases} \end{aligned}$$

The normalisation constants for different values of N, where N is an integer power of 2 are shown in table 3.3.

| N | $k = 0$ | $k > 0$ |
|---|---|---|
| 2 | $\frac{1}{\sqrt{2}}$ | 1 |
| 4 | $\frac{1}{\sqrt{4}}$ | $\frac{1}{\sqrt{2}}$ |
| 8 | $\frac{1}{\sqrt{8}}$ | $\frac{1}{\sqrt{4}}$ |
| 16 | $\frac{1}{\sqrt{16}}$ | $\frac{1}{\sqrt{8}}$ |
| 32 | $\frac{1}{\sqrt{32}}$ | $\frac{1}{\sqrt{16}}$ |

Table 3.3: The normalisation constants for DCT-II

It is evident from the table 3.3 that the normalising constants for the N-point DCT coefficients can be obtained by multiplying the normalising constants of the $\frac{N}{2}$-point DCT coefficients by a factor $\frac{1}{\sqrt{2}}$. This can be incorporated into equations 3.42 and 3.43 respectively as below.

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} \underbrace{\cos\left((4n+1)\frac{\pi k}{2\frac{N}{2}}\right)}_{\frac{N}{2}\ point\ DCT} \frac{1}{\sqrt{2}} \left[\tilde{x}(n) + \tilde{x}(\tfrac{N}{2}+n)\right] \qquad (3.45)$$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} \underbrace{\cos\left((4n+1)\frac{\pi(2k+1)}{2N}\right)}_{E_{\frac{N}{2}}} \frac{1}{\sqrt{2}} \left[\tilde{x}(n) - \tilde{x}(\tfrac{N}{2}+n)\right] \qquad (3.46)$$

This can be summarised into matrix format as below.

$$\mathbf{X} = \mathbf{D_N}\,\tilde{\mathbf{x}}$$

$$\left[\begin{array}{c}\mathbf{X_{2k}}\\\hline\mathbf{X_{2k+1}}\end{array}\right] = \left[\begin{array}{c}\mathbf{D_{\frac{N}{2}}\,U_1}\\\hline\mathbf{E_{\frac{N}{2}}\,U_2}\end{array}\right] \tag{3.47}$$

$$\textit{where,}\quad \left[\begin{array}{c}\mathbf{U_1}\\\hline\mathbf{U_2}\end{array}\right] = \underbrace{\frac{1}{\sqrt{2}}\left[\begin{array}{c|c}\mathbf{I_{N/2}} & \mathbf{I_{N/2}}\\\hline\mathbf{I_{N/2}} & -\mathbf{I_{N/2}}\end{array}\right]}_{\mathbf{A}}\left[\begin{array}{c}\tilde{\mathbf{x}}(0,\cdots,\frac{N}{2}-1)\\\hline\tilde{\mathbf{x}}(\frac{N}{2}+1,\cdots,N-1)\end{array}\right] \tag{3.48}$$

$$\textit{where, } \mathbf{I_{N/2}} \textit{ is the Identity matrix of size } \tfrac{N}{2}$$
$$\textit{and } \mathbf{A} \textit{ corresponds to the } \mathbf{WH_2}.$$

#### 3.3.1.2 Lossless realisation of the even-indexed coefficients

The transformation of $\tilde{\mathbf{x}}$ into $\mathbf{U_1}$ and $\mathbf{U_2}$, as in equation 3.48, is the same as transforming $\tilde{\mathbf{x}}$ with the 2-point Walsh Hadamard transform, $\mathbf{WH_2}$. The $\mathbf{WH_2}$ operations can be factorised into lifting steps as in equations 3.32 - 3.35 in section 3.2.2.3, thus they can be realised with integer coefficients easily.

As shown in equations 3.42 and 3.47, the even-indexed rows of the DCT matrix reduce to the $\frac{N}{2}$-point DCT matrix. Therefore, the even-indexed coefficients can be realised by recursively computing $\mathbf{D_M}$, where $\mathbf{M} = (\frac{N}{2}, \frac{N}{4}, \ldots, 2)$. At the recursion termination point, where N=2, the $\mathbf{D_2}$ matrix is the same as the $\mathbf{WH_2}$ matrix, the lossless realisation of which has already been discussed in section 3.2.2.3.

#### 3.3.1.3 Lossless realisation of the odd-indexed coefficients

The realisation of the odd-indexed coefficients at a given stage is achieved by multiplying the vector $\mathbf{U_2}$ with the corresponding cosine matrix, $\mathbf{E_{\frac{N}{2}}}$ (equation 3.47). Some intrinsic properties of $\mathbf{E_{\frac{N}{2}}}$ are discussed below. They can be used for further factorisation of $\mathbf{E_{\frac{N}{2}}}$, so that the lifting techniques can be used to obtain integer coefficients.

Let the elements of $\mathbf{E_{\frac{N}{2}}}$ be $C_N^{kn}$, where

$$\begin{aligned} C_N^{kn} &= \cos\left((4n+1)\frac{\pi(2k+1)}{2N}\right) \\ &= \cos\left(\frac{(4n+1)\pi}{2N}(2k+1)\right) \\ &= \cos\left(\phi_N^n(2k+1)\right) \\ &= \cos\left(\theta_{kn}\right) \\ &\quad \textit{where,}\quad k,n = 0,\ldots,\tfrac{N}{2}-1. \end{aligned} \tag{3.49}$$

The basic angle, $\phi_N^n = \frac{(4n+1)\pi}{2N}$, corresponds to $\theta_{kn}$ for $k = 0$.

For a given $k$ (row),

$$\begin{aligned} for\ n_0 = 0, \ldots, \tfrac{N}{4} - 1 & \\ C_N^{kn_0} &= \cos\left((4n_0 + 1)\frac{\pi(2k+1)}{2N}\right) \\ &= \cos(\theta_{kn_0}) \end{aligned} \tag{3.50}$$

$$\begin{aligned} for\ n_1 = n_0 + \tfrac{N}{4} & \\ C_N^{kn_1} &= \cos\left((4(n_0 + \tfrac{N}{4}) + 1)\frac{\pi(2k+1)}{2N}\right) \\ &= \cos\left(\theta_{kn_0} + \tfrac{\pi}{2}(2k+1)\right) \\ &= \begin{cases} \cos\left(\theta_{kn_0} + \frac{\pi}{2}\right) & for\ even\ k \\ \cos\left(\theta_{kn_0} + \frac{3\pi}{2}\right) & for\ odd\ k \end{cases} \end{aligned} \tag{3.51}$$

The equations 3.50 and 3.51 show that the elements of $C_N^{kn}$ that are separated from an $\frac{N}{4}$ distance are separated from an angle $\frac{\pi}{2}(2k+1)$. Likewise, it can be shown for any distance $m$, where $m \in \{\frac{N}{4}, \frac{N}{8}, \frac{N}{16}, \ldots, 1\}$, that

$$C_N^{k(n+m)} = \cos\left(\theta_{kn} + \tfrac{2m\pi}{N}(2k+1)\right) \tag{3.52}$$

Similarly, for a given $n$ (column),

$$\begin{aligned} for\ k_0 = 0, \ldots, \tfrac{N}{4} - 1 & \\ C_N^{k_0 n} &= \cos\left((4n + 1)\frac{\pi(2k_0+1)}{2N}\right) \\ &= \cos(\theta_{k_0 n}) \end{aligned} \tag{3.53}$$

$$\begin{aligned} for\ k_1 = \tfrac{N}{2} - 1 - k_0 & \\ C_N^{k_1 n} &= \cos\left((4n + 1)\frac{\pi(2\,(\frac{N}{2} - 1 - k_0) + 1)}{2N}\right) \\ &= \cos\left(\tfrac{\pi}{2}(4n+1) - \theta_{k_0 n}\right) \\ &= \cos\left(\tfrac{\pi}{2} - \theta_{k_0 n}\right),\ \forall\ n \end{aligned} \tag{3.54}$$

The equation 3.53 shows that the angles in successive rows are separated by an angle of $2\phi_n$, where $\phi_n$ is the basic angle for a given column as defined in equation 3.49. The matrix $C_N^{nk}$ can be rewritten as below using the expressions (equations 3.50 - 3.54) for the angles in four quadrants.

$$\begin{aligned} C_N^{nk} &= \begin{bmatrix} \cos(\theta_{k_0 n_0}) & \cos(\theta_{k_0 n_1}) \\ \cos(\theta_{k_1 n_0}) & \cos(\theta_{k_1 n_1}) \end{bmatrix} \\ &= \begin{bmatrix} \cos(\theta_{k_0 n_0}) & \cos\left(\theta_{k_0 n_0} \pm \frac{\pi}{2}\right) \\ \cos\left(\frac{\pi}{2} - \theta_{k_0 n_0}\right) & \cos\left(\frac{\pi}{2} - \theta_{k_0 n_0} \mp \frac{\pi}{2}\right) \end{bmatrix} \\ &= \begin{bmatrix} \cos(\theta_{k_0 n_0}) & \mp\sin(\theta_{k_0 n_0}) \\ \sin(\theta_{k_0 n_0}) & \pm\cos(\theta_{k_0 n_0}) \end{bmatrix} \qquad (3.55) \end{aligned}$$

Equation 3.55 shows that the elements in $C_N^{nk}$ can be rearranged using the indices $k_0$, $k_1$, $n_0$ and $n_1$ into 2×2 partial matrices, which can be either of the two forms as above with an unit determinant. The matrices with a unit determinant and a non zero $(0,1)^{th}$ element can be factorised into lifting steps, thus they can be realised losslessly [80]. This concept can be used to realise odd-indexed coefficients losslessly.

The DCT matrix arranged in increasing frequency order and the corresponding WHT matrix arranged in increasing sequency order share the same signs. For the same reason, some authors had considered the DCT matrix as an amplitude modulated WHT matrix [9, 89]. This relationship is used for further analysis of the odd indexed coefficient rows in the DCT matrix as below.

$$\begin{aligned} C_N^{nk} &= \left(C_N^{nk} \times \mathbf{WH}_{\frac{\mathbf{N}}{\mathbf{2}}}\right) \times \mathbf{WH}_{\frac{\mathbf{N}}{\mathbf{2}}} \qquad (3.56) \\ & \quad (Because\ \mathbf{WH_N} \times \mathbf{WH_N} = \mathbf{I_N}) \end{aligned}$$

So far in this derivation only a $\frac{1}{\sqrt{2}}$ factor has been used as the normalising factor for the odd-indexed coefficients (equation 3.48). The rest of the normalising is incorporated into the elements of $\mathbf{E}_{\frac{\mathbf{N}}{\mathbf{2}}}$ i.e. $C_N^{nk}$.

$$C_{N\ new}^{nk} = \underbrace{\left(\sqrt{2}\,\alpha_N \times C_N^{nk} \times \mathbf{WH}_{\frac{\mathbf{N}}{\mathbf{2}}}\right)}_{O_N^{nk}} \times \mathbf{WH}_{\frac{\mathbf{N}}{\mathbf{2}}} \qquad (3.57)$$

As seen earlier, the $\mathbf{WH}_{\frac{\mathbf{N}}{\mathbf{2}}}$ can be realised losslessly. Therefore, factoring $O_N^{nk}$ into lifting steps leads to the lossless realisation of the odd-indexed coefficients of the N-point DCT. The columns and the rows of the matrix $O_N^{nk}$ can be permuted using the permutation matrices $\mathbf{P1_N}$ and $\mathbf{P2_N}$ and the signs of the columns are set using the sign matrix $\mathcal{S}_\mathbf{N}$, so that the elements of $O_N^{nk}$ are arranged as in equation 3.58.

$$O_N^{nk} = \ \mathbf{P2_N} \times$$

$$\left(\begin{array}{ccc}
c_{\frac{\pi}{8}} \times & \underbrace{\left[\begin{array}{ccc}
c_{\frac{\pi}{16}} \times \cdots & \underbrace{\left[\begin{array}{cc}
c_{\frac{\pi}{N}} R_{\frac{\pi}{2N}} & -s_{\frac{\pi}{N}} R_{\frac{\pi}{2N}} \\
s_{\frac{\pi}{N}} R_{\frac{\pi}{2N}+\frac{pi}{4}} & c_{\frac{\pi}{N}} R_{\frac{\pi}{2N}+\frac{\pi}{4}} \\
 & \\
c_{\alpha} R_{\alpha 1} & -s_{\alpha} R_{\alpha 1} \\
s_{\alpha} R_{\alpha 2} & c_{\alpha} R_{\alpha 2}
\end{array}\right]}_{A0} & -s_{\frac{\pi}{16}} \times [A0] \\
s_{\frac{\pi}{16}} \times \cdots & \underbrace{\left[\begin{array}{cc}
c_{\frac{9\pi}{N}} R_{\frac{9\pi}{2N}} & -s_{\frac{9\pi}{N}} R_{\frac{9\pi}{2N}} \\
s_{\frac{9\pi}{N}} R_{\frac{9\pi}{2N}+\frac{pi}{4}} & c_{\frac{9\pi}{N}} R_{\frac{9\pi}{2N}+\frac{pi}{4}} \\
 & \\
c_{\beta} R_{\beta 1} & -s_{\beta} R_{\beta 1} \\
s_{\beta} R_{\beta 2} & c_{\beta} R_{\beta 2}
\end{array}\right]}_{A1} & c_{\frac{\pi}{16}} \times [A1]
\end{array}\right]}_{A} & -s_{\frac{\pi}{8}} \times [A] \\
s_{\frac{\pi}{8}} & \underbrace{\left[\begin{array}{ccc}
c_{\frac{5\pi}{16}} \times \cdots & \underbrace{\left[\begin{array}{cc}
c_{\frac{5\pi}{N}} R_{\frac{5\pi}{2N}} & -s_{\frac{5\pi}{N}} R_{\frac{5\pi}{2N}} \\
s_{\frac{5\pi}{N}} R_{\frac{5\pi}{2N}+\frac{\pi}{4}} & c_{\frac{5\pi}{N}} R_{\frac{5\pi}{2N}+\frac{\pi}{4}} \\
 & \\
c_{\gamma} R_{\gamma 1} & -s_{\gamma} R_{\gamma 1} \\
s_{\gamma} R_{\gamma 2} & c_{\gamma} R_{\gamma 2}
\end{array}\right]}_{B0} & -s_{\frac{5\pi}{16}} \times [B0] \\
s_{\frac{5\pi}{16}} \times \cdots & \underbrace{\left[\begin{array}{cc}
c_{\frac{13\pi}{N}} R_{\frac{13\pi}{2N}} & -s_{\frac{\pi}{N}} R_{\frac{13\pi}{2N}} \\
s_{\frac{13\pi}{N}} R_{\frac{13\pi}{2N}+\frac{\pi}{4}} & c_{\frac{13\pi}{N}} R_{\frac{13\pi}{2N}+\frac{\pi}{4}} \\
 & \\
c_{\delta} R_{\delta 1} & -s_{\delta} R_{\delta 1} \\
s_{\delta} R_{\delta 2} & c_{\delta} R_{\delta 2}
\end{array}\right]}_{B1} & c_{\frac{5\pi}{16}} \times [B1]
\end{array}\right]}_{B} & c_{\frac{\pi}{8}} \times [B]
\end{array}\right)$$

$$\times\ \mathcal{S}_{\mathbf{N}} \times\ \mathbf{P1_N} \tag{3.58}$$

$$\begin{aligned}
&\textit{where } c_\theta = \cos(\theta), \quad s_\theta = \sin(\theta), \\
&\alpha = \frac{\pi(4(\frac{N}{8}-4)+1)}{N}, \quad \alpha 1 = \frac{\alpha}{2}, \quad \alpha 2 = \alpha 1 + \frac{\pi}{4}, \\
&\beta = \frac{\pi(4(\frac{N}{8}-2)+1)}{N}, \quad \beta 1 = \frac{\beta}{2}, \quad \beta 2 = \beta 1 + \frac{\pi}{4}, \\
&\gamma = \frac{\pi(4(\frac{N}{8}-3)+1)}{N}, \quad \gamma 1 = \frac{\gamma}{2}, \quad \gamma 2 = \gamma 1 + \frac{\pi}{4},
\end{aligned}$$

$$\delta = \frac{\pi(4(\frac{N}{8}-1)+1)}{N},\ \ \delta 1 = \frac{\delta}{2},\ \ \delta 2 = \delta 1 + \frac{\pi}{4}\ and$$

$$R_\theta = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{Givens\ Rotation\ Matrix} \tag{3.59}$$

The factorised $O_N^{nk}$ in equation 3.58 can be re-written using the Kroneckor products of rotation matrices and the Identity matrices as below.

$$\begin{aligned} O_N^{nk} &= \mathbf{P2_N} \times \begin{bmatrix} R_{\frac{\pi}{2N}} & & & & & \\ & R_{\frac{\pi}{2N}+\frac{\pi}{4}} & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & R_{\frac{\pi(4(\frac{N}{4}-1)+1)}{2N}-\frac{\pi}{4}} & \\ & & & & & R_{\frac{\pi(\frac{N}{4}-1)+1)}{2N}} \end{bmatrix} \times \cdots \\ &\cdots \times \begin{bmatrix} R_{\frac{\pi}{32}} \otimes \mathbf{I_{\frac{N}{16}}} & & & \\ & R_{\frac{9\pi}{32}} \otimes \mathbf{I_{\frac{N}{16}}} & & \\ & & R_{\frac{5\pi}{32}} \otimes \mathbf{I_{\frac{N}{16}}} & \\ & & & R_{\frac{13\pi}{32}} \otimes \mathbf{I_{\frac{N}{16}}} \end{bmatrix} \\ &\times \begin{bmatrix} R_{\frac{\pi}{16}} \otimes \mathbf{I_{\frac{N}{8}}} & \\ & R_{\frac{5\pi}{16}} \otimes \mathbf{I_{\frac{N}{8}}} \end{bmatrix} \times \begin{bmatrix} R_{\frac{\pi}{8}} \otimes \mathbf{I_{\frac{N}{4}}} \end{bmatrix} \times \mathcal{S}_\mathbf{N} \times \mathbf{P1_N} \end{aligned} \tag{3.60}$$

The Givens rotation matrix (equation 3.59) $R_\theta$ possesses a unit determinant and a non-zero $(0,1)^{th}$ element. The $(0,1)^{th}$ element is $-\sin(\theta)$, which becomes zero only when $\theta = k\pi$, where $k \in \mathcal{N}$. Such values of $\theta$ do not occur in the above derivation. Therefore, they can be factorised into a product of upper and lower triangular matrices as shown in [80] and [55].

$$R_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \frac{\cos(\theta)-1}{\sin(\theta)} \\ 0 & 1 \end{bmatrix}}_{R2} \underbrace{\begin{bmatrix} 1 & 0 \\ \sin(\theta) & 1 \end{bmatrix}}_{R1} \underbrace{\begin{bmatrix} 1 & \frac{\cos(\theta)-1}{\sin(\theta)} \\ 0 & 1 \end{bmatrix}}_{R0} \tag{3.61}$$

The corresponding lifting steps with rounding for the integer realisation are as below.

$$From\ \mathbf{R0}: \quad x_1 \leftarrow \quad x_1 + \left\lfloor \frac{\cos(\theta)-1}{\sin(\theta)} x_2 \right\rfloor \tag{3.62}$$

$$\textit{From } \mathbf{R1}: \quad x_2 \leftarrow \quad x_2 + \lfloor \sin(\theta)\, x_1 \rfloor \tag{3.63}$$

$$\textit{From } \mathbf{R2}: \quad x_1 \leftarrow \quad x_1 + \left\lfloor \frac{\cos(\theta)-1}{\sin(\theta)}\, x_2 \right\rfloor \tag{3.64}$$

All the sub matrices in equation 3.60 can be realised losslessly with the above lifting steps.

The rotation angle values in sub matrices in equations 3.58 and 3.60 are the same as the basic angle $\phi_N^n$ for $n = 0, \ldots, \frac{N}{4}-1$ where $N = 4, 8, \ldots, M$ for $\mathbf{IDCT_M}$. Equation 3.60 can be written in generic form as below.

$$\begin{aligned} O_N^{nk} \;=\; & \mathbf{P2_N} \times \left[\; diag(R_{\phi_N^n} \otimes \mathbf{I_{\frac{N}{N}}})_{n=0,\ldots,\frac{N}{4}-1} \;\right] \times \cdots \\ & \cdots \times \left[\; diag(R_{\phi_{16}^n} \otimes \mathbf{I_{\frac{N}{16}}})_{n=0,\ldots,3} \;\right] \\ & \times \left[\; diag(R_{\phi_8^n} \otimes \mathbf{I_{\frac{N}{8}}})_{n=0,\ldots,1} \;\right] \times \left[ R_{\phi_4^0} \otimes \mathbf{I_{\frac{N}{4}}} \right] \times \mathcal{S}_\mathbf{N} \times \mathbf{P1_N} \end{aligned} \tag{3.65}$$

The values of basic angle $\phi_N^n$ at each level is computed as in Figure 3.8.

```
function compute_basic_angle (N, n, Phi)
{
% This computes the basic angles for odd indexed rows of N-point DCT.
   if (N>2)
   {
      compute_basic_angle (N/2, 2*n, Phi/2);
      compute_basic_angle (N/2, 2*n+1, (Phi/2)+(PI/4));
   }
}

%Call the function as below
%PI=3.14159
compute_basic_angle (N/2, 0, PI/8);    % for N-point DCT
```

Figure 3.8: The pseudo code for computing the basic angle $\phi_N^n$.

### 3.3.1.4 Permutation ($\mathbf{P1_N}$ & $\mathbf{P2_N}$) and Sign matrices ($\mathcal{S}_N$)

The $O_N^{nk}$ matrix, the odd-indexed rows of the N-point DCT factorised into sub matrices with the Kroneckor products of the rotation matrices of the basic angles at each stage with the corresponding identity matrices, needs to be row and column wise permutated in order to achieve the correct ordering of the DCT basis vectors as in equation 3.43. This is achieved by incorporating $\mathbf{P2_N}$ and $\mathbf{P1_N}$ operations corresponding to row and column permutation as in equation 3.60. Computing indexes for row and column permutation for $O_N^{nk}$ using recursive functions is as in the pseudo codes shown in Figure 3.9 and 3.10.

```
function OddDCT_Row_PermIndex(index_array, N, NN)
{
     if (N>2)
     {
         OddDCT_Row_PermIndex(index_array,N/2,NN);
         for (i = N/4 : N/2-1)
              index_array[i]=N-2-index_array[i-N/4];
         for (i= NN/2+N/4 : NN/2+N/2-1)
              index_array[i]=N-index_array[i-N/4];
     }
     else
     {
         index_array[0]=0;
         index_array[NN/2]=1;
     }
}
  %Call the function as below
  OddDCT_Row_PermIndex(index_array,N/2,N/2);  % for N-point DCT
  New_odd_rows=old_odd_rows[index_array];
```

Figure 3.9: The pseudo code for index computation for row permutation of $O_N^{nk}$.

Although some authors have considered the exact sign similarity between the WHT (in Hadamard order) and the rearranged DCT [9, 89], this is only true for 2-point and 4-point transforms. There are a few sign mismatches which appear in transforms with dimensions higher than four, as can be seen in the transform matrices listed in Appendix B. To alleviate this sign mismatch, a sign compensation ($\mathcal{S}_N$) is performed in $O_N^{nk}$. The computation of sign compensation up to N=32, N-point DCT is shown as in figure 3.11. The upper half of the sign matrix for N-point DCT is the sign matrix for the $\frac{N}{2}$-point DCT.

```
function OddDCT_Col_PermIndex(index_array, N, NN)
{
     if (N>2)
     {
         OddDCT_Col_PermIndex(index_array,N/2,NN);
         for (i = N/4 : N/2-1)
             index_array[i]=(i-N/4)*4+N/2+1-index_array[i-N/4];
         for (i= NN/2+N/4 : NN/2+N/2-1)
             index_array[i]=(i-(NN/2+N/4))*4+N/2+1-index_array[i-N/4];
     }
     else
     {
         index_array[0]=0;
         index_array[NN/2]=1;
     }
}
  %Call the function as below
  OddDCT_col_PermIndex(index_array,N/2,N/2);  % for N-point DCT
  New_odd_cols[index_array]=old_odd_cols;
```

Figure 3.10: The pseudo code for index computation for column permutation of $O_N^{nk}$.

```
function Get_New_Sign(sign_array,N)
{
   switch N
      case {2},
          sign_array=[1 1];,
      case {4},
         sign_array=[Get_New_Sign(sign_array,2) 1 -1];,
      case {8},
          sign_array=[Get_New_Sign(sign_array,4) 1 -1 -1 -1 ];,
      case {16},
          sign_array=[Get_New_Sign(sign_array,8) 1 -1 -1 -1 -1 -1 -1  1];,
}
  %Call the function as below
  Get_New_Sign(sign_array,N/2);  % for N-point DCT
  New_odd_cols=old_odd_cols*sign_array; % for each element
```

Figure 3.11: The sign compensation for odd indexed rows in N-point DCT

#### 3.3.1.5 A Block diagram for the IDCT

The above discussed integer implementation of the $\mathbf{IDCT_N}$ can be summarised in a block diagram as in Figure 3.12. The column permutation stage at the beginning corresponds to $x(n) \rightarrow \tilde{x}(n)$ conversion shown in equation 3.37, whereas the row permutation stage at the end corresponds to rearranging the upper and lower halves of the DCT matrix into increasing frequency order. The pseudo code for the middle stage, integer DCT operations, is as in Figure 3.13. The inverse transform is implemented by reversing the recursive operations and the lifting steps by changing the sign and the order of operation of lifting equations. In both forward and inverse transforms, all the operations can be performed as in-place computations, which result in low resource requirements in software / hardware implementations.

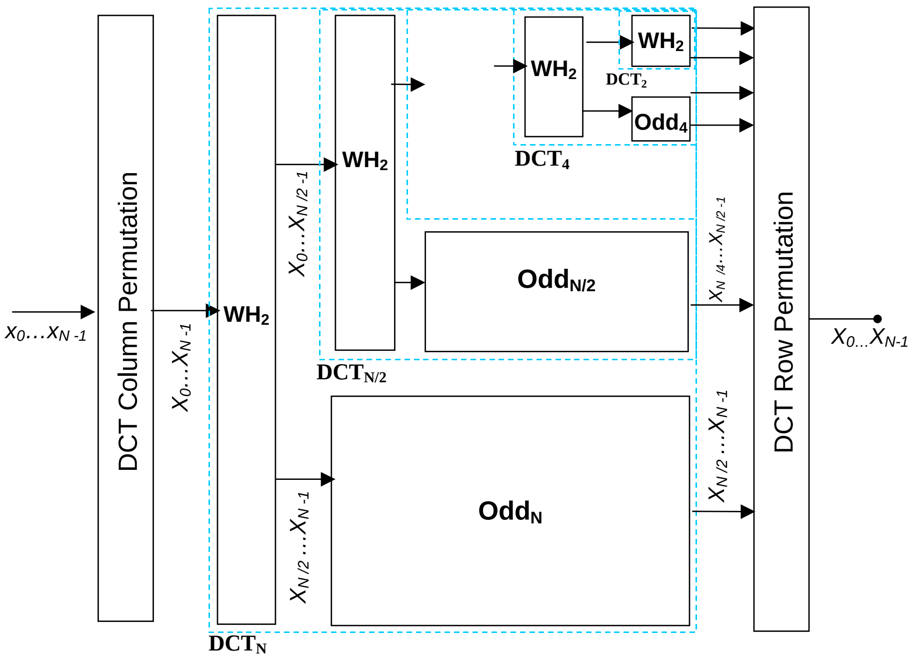


Figure 3.12: Block diagram for integer $\mathbf{IDCT_N}$.

The signal flow diagram for the forward $\mathbf{IDCT_8}$ is as in Figure 3.14. The inverse is computed by following the inverse signal flow with the reversed operations including lifting, row / column permutations and sign changes.

```
function LiftDCT(input, N, position)
{
     kron(Walsh2, I(N/2))  % Walsh2 is 2X2 Walsh Hadamard Matrix
                           % I is the Identity Matrix
                           % kron is the Kroneckor product operation
     if (N>2)
     {
         % Even rows (upper half) operations
             LiftDCT(input, N/2, position);

         % Odd rows (lower half) operations
         % Do WHT(N/2) for position+N/2 to position+N
             DoWalsh(input ,N/2, position+N/2);
             Odd_Perm_Cols(input, N/2, position+N/2);
             Odd_ChangeSign(input, N/2, position+N/2);
             Do_Basic_Angle_Rotations(input, N/2, position+N/2, PI/8);
             Odd_Perm_Rows(input, N/2, position+N/2);

         % PI/8 is the starting basic angle for any dimension
     }
}
  %Call the function as below for N-point DCT
  Do_Column_order(input, N)  % Equation 3.37
  LiftDCT(input, N, 0)
  Do_Row_order(input, N)     % to increasing frequency order
```

Figure 3.13: The pseudo code for integer DCT operations in $\mathbf{IDCT_N}$.

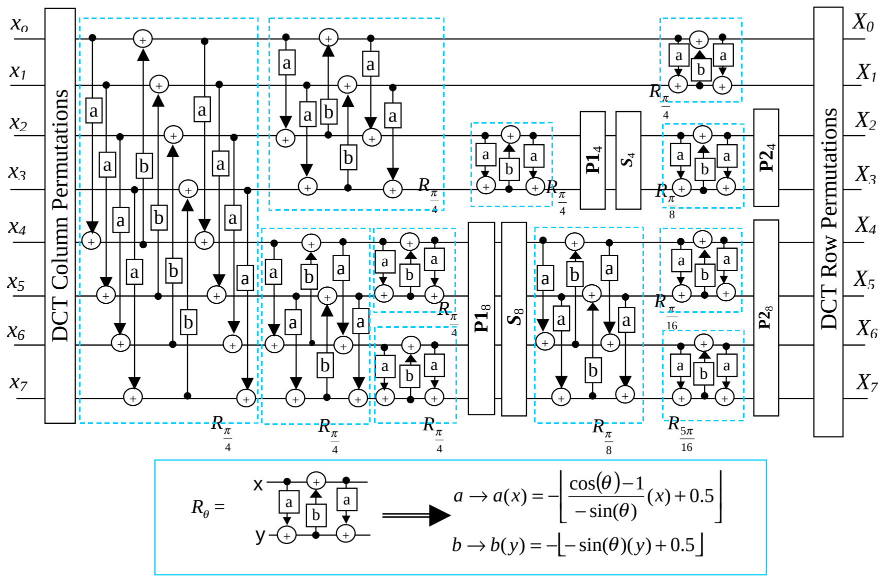


Figure 3.14: Signal flow diagram for $\mathbf{IDCT_8}$.

### 3.3.2 The zero-order entropy values

The performance of the IDCT on lossless image coding with different block sizes is presented here. For each block, the transform is applied separately on rows and columns and then the block structure is converted to the corresponding wavelet packet tree structure introduced on page 41 using the same algorithm. The weighted entropy values, computed based on the packet sub bands for the image set using the IDCT for different N values, are as in Table 3.4.

| N | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| Gold Hill | 5.631 | 5.031 | 4.825 | 4.727 | 4.626 |
| Barbara1 | 5.965 | 5.222 | 4.850 | 4.656 | 4.514 |
| Barbara2 | 5.934 | 5.362 | 5.108 | 4.963 | 4.824 |
| Boats | 5.203 | 4.556 | 4.347 | 4.281 | 4.214 |
| Black Board | 4.809 | 4.162 | 3.982 | 3.955 | 3.925 |
| Average | 5.508 | 4.867 | 4.622 | 4.516 | 4.421 |

Table 3.4: Weighted zero-order entropy values in bpp for IDCT

These results show that as the block size, N, increases, the weighted entropy values decrease, thus giving better results. In this case, the 32-point IDCT provides the best performance for all the images in the test image set. However, it is noted that the net entropy savings gained by opting for greater N, decrease with the increasing N. Further, greater N causes higher computational complexity.

## 3.4 The Integer Discrete Sine Transform (IDST)

Although the DCT has been widely used in transform based coding of natural images, the discrete sine transform (DST), first introduced in [95], has not been used commonly in image coding due to its poor data decorrelation capability and poor energy compaction. Consequently, the DST coefficients of highly correlated source data are also highly correlated [2]. Since the DST does not compute a zero frequency or a bias component, the DC component of the source data is spread over other frequencies in the transform domain. This effect can be reduced, if not eliminated entirely, by subtracting the global mean from the source data, even though the local mean of a given block is not completely removed by this method.

However, the DST has been used in modelling of random processes, in order to make their KLT fast [95, 96]. Further, it has been used in image reconstruction [97]. The authors of [2] and [98] have suggested the possibility of using DST for non-intra frame transform coding. An integer version of the DST is implemented in this section, so that the performance of DST on lossless image coding and lossless non-intra frame coding can be evaluated.

The DST for a one dimensional (1-D) data sequence $x(n+1)$, where $n = 0, \ldots, N-1$, and its inverse are defined as in equations 3.66 and 3.67 respectively [96].

$$X(k) = \epsilon_k \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n+1)\,\sin(\,(2n+1)\frac{\pi k}{2N}\,) \qquad for\ k = 1, \ldots, N \tag{3.66}$$

$$x(n+1) = \sqrt{\frac{2}{N}} \sum_{k=1}^{N} \epsilon_k\, X(k)\,\sin(\,(2n+1)\frac{\pi k}{2N}\,) \qquad for\ n = 0, \ldots, N-1 \tag{3.67}$$

$$where,\ \epsilon_k = \begin{cases} \frac{1}{\sqrt{2}} & if\ k = N\,, \\ 1 & else. \end{cases}$$

### 3.4.1 Derivation of the IDST

There has not been any report of integer implementation of the DST in published literature. It was shown in section 3.3, how the IDCT can be derived from the fast algorithm factorisations. The same approach has been followed in this research for the derivation of the IDST.

All the fast DST algorithms in literature have used recursive methods. The earliest example of the fast DST used a sparse matrix factorisation which leads to a recursive structure and consequently leading to an efficient algorithm for implementing the DST [96]. Following Hou's implementation of the fast DCT using recursive methods [91], two similar approaches [92, 99] have been presented for the fast implementation of the DST using recursive methods. In the fast recursive DCT algorithms, the even-indexed rows in the DCT matrix were used to compute the odd-indexed rows using trigonometric identities [91], whereas in the fast recursive DST algorithms, the odd-indexed rows in the DST matrix were used to compute the even-indexed rows using trigonometric identities.

#### 3.4.1.1 The IDST using recursive methods and lifting

The DST, shown in equation 3.66, can be rewritten as in equation 3.66.

$$\mathbf{X} = \alpha_{\mathbf{N}}\, \mathbf{T}_{\mathbf{N}}\, \mathbf{x} \tag{3.68}$$

where, $\mathbf{X}$ and $\mathbf{x}$ are column vectors of size $N \times 1$ denoting the DST output in increasing frequency order and the input data sequences respectively. $\mathbf{T}_{\mathbf{N}}$ is the N-point DST matrix of size $N \times N$ and $\alpha_{\mathbf{N}}$ is a diagonal matrix of size $N \times N$ denoting the normalising constants.

This derivation is also started with the un-normalised N-point DST coefficients $X(k)$, where k=1,...,N for the input signal $x(n+1)$, where n=0,...,N-1 as shown in equation 3.69.

$$X(k) = \sum_{n=0}^{N-1} x(n+1) \sin\left((2n+1)\frac{\pi k}{2N}\right) \tag{3.69}$$

Since the Cosines and the Sines are similar functions with a $\frac{\pi}{2}$ phase difference, the column permutation used for the DCT (equation 3.37), adjusted for the DST index notations, is used in this derivation.

$$\left.\begin{array}{lcl} \tilde{\mathbf{x}}(n) & = & \mathbf{x}(2n+2) \\ \tilde{\mathbf{x}}(N+\frac{n}{2}) & = & \mathbf{x}(N-(2n+1)) \end{array}\right\} \; n = 0, \ldots, N/2-1. \tag{3.70}$$

Using equation 3.69, the even indexed coefficients can be written as in equation 3.71.

$$\begin{aligned}
X(2k) &= \sum_{n=0}^{N-1} x(n+1)\sin\left((2n{+}1)\frac{\pi 2k}{2N}\right)\\
&= \sum_{n=0}^{\frac{N}{2}-1} \tilde{x}(n)\sin\left((2\,(2n){+}1)\frac{\pi k}{2\frac{N}{2}}\right) + \sum_{n=0}^{\frac{N}{2}-1} \tilde{x}(\tfrac{N}{2}+n)\sin\left((2\,(N-\,(2n{+}1)){+}1)\frac{\pi k}{2\frac{N}{2}}\right)\\
&= \sum_{n=0}^{\frac{N}{2}-1} \tilde{x}(n)\sin\left((4n{+}1)\frac{\pi k}{2\frac{N}{2}}\right) + \sum_{n=0}^{\frac{N}{2}-1} \tilde{x}(\tfrac{N}{2}+n)\sin\left(2\pi k-(4n{+}1)\frac{\pi k}{2\frac{N}{2}}\right)\\
&= \sum_{n=0}^{\frac{N}{2}-1} \left[\tilde{x}(n)-\tilde{x}(\tfrac{N}{2}+n)\right] \underbrace{\sin\left((4n{+}1)\frac{\pi k}{2\frac{N}{2}}\right)}_{\frac{N}{2}\ point\ DST}\\
&\quad (for\quad k=1,\ldots,\frac{N}{2}).
\end{aligned} \tag{3.71}$$

Similarly for the odd indexed coefficients,

$$\begin{aligned}
X(2k{-}1) &= \sum_{n=0}^{N-1} x(n+1)\sin\left((2n{+}1)\frac{\pi(2k{-}1)}{2N}\right)\\
&= \sum_{n=0}^{\frac{N}{2}-1} \tilde{x}(n)\sin\left((4n{+}1)\frac{\pi(2k{-}1)}{2N}\right)\\
&\quad + \sum_{n=0}^{\frac{N}{2}-1} \tilde{x}(\tfrac{N}{2}+n)\sin\left((2\,(N-\,(2n{+}1)){-}1)\frac{\pi(2k{-}1)}{2\frac{N}{2}}\right)\\
&= \sum_{n=0}^{\frac{N}{2}-1} \tilde{x}(n)\sin\left((4n{+}1)\frac{\pi(2k{-}1)}{2N}\right)\\
&\quad + \sum_{n=0}^{\frac{N}{2}-1} \tilde{x}(\tfrac{N}{2}+n)\sin\left(\pi(2k{-}1)-(4n{+}1)\frac{\pi(2k{-}1)}{2N}\right)\\
&= \sum_{n=0}^{\frac{N}{2}-1} \left[\tilde{x}(n)+\ \tilde{x}(\tfrac{N}{2}+n)\right] \underbrace{\sin\left((4n{+}1)\frac{\pi(2k{-}1)}{2N}\right)}_{F_{\frac{N}{2}}}\\
&\quad (for\quad k=1,\ldots,\frac{N}{2}).
\end{aligned} \tag{3.72}$$

### 3.4.1.2 Incorporating the normalising factors

The normalising factor, $\alpha_N$, for each coefficient is as in equation 3.73.

$$\alpha_N(k) = \sqrt{\frac{2}{N}}\,\epsilon_k \tag{3.73}$$

$$where,\ \epsilon_k = \begin{cases} \frac{1}{\sqrt{2}} & if\ k=N\,, \\ 1 & else. \end{cases}$$

This is similar to the normalisation factors of the DCT as shown in table 3.3. Therefore, the same relationship of the $\frac{1}{\sqrt{2}}$ factor as in the DCTs can be considered. This leads to the equations below for the normalised $X(2k)$ and $X(2k-1)$.

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} \underbrace{\sin\left((4n+1)\frac{\pi k}{2\frac{N}{2}}\right)}_{\frac{N}{2}\ point\ DST} \frac{1}{\sqrt{2}}\left[\tilde{x}(n)-\tilde{x}(\tfrac{N}{2}+n)\right] \tag{3.74}$$

$$X(2k-1) = \sum_{n=0}^{\frac{N}{2}-1} \underbrace{\sin\left((4n+1)\frac{\pi(2k-1)}{2N}\right)}_{F_{\frac{N}{2}}} \frac{1}{\sqrt{2}}\left[\tilde{x}(n)+\tilde{x}(\tfrac{N}{2}+n)\right] \tag{3.75}$$

The above equations can be summarised into matrix form as below.

$$\begin{aligned} \mathbf{X} &= \mathbf{T_N}\,\tilde{\mathbf{x}} \\ \left[\begin{array}{c} \mathbf{X_{2k}} \\ \hline \mathbf{X_{2k\text{–}1}} \end{array}\right] &= \left[\begin{array}{c} \mathbf{T_{\frac{N}{2}}\ U_1} \\ \hline \mathbf{F_{\frac{N}{2}}\ U_2} \end{array}\right] \end{aligned} \tag{3.76}$$

$$where,\ \left[\begin{array}{c} \mathbf{U_1} \\ \hline \mathbf{U_2} \end{array}\right] = \underbrace{\frac{1}{\sqrt{2}}\left[\begin{array}{c|c} \mathbf{I_{N/2}} & -\mathbf{I_{N/2}} \\ \hline \mathbf{I_{N/2}} & \mathbf{I_{N/2}} \end{array}\right]}_{\mathbf{A}} \left[\begin{array}{c} \tilde{\mathbf{x}}(0,\cdots,\frac{N}{2}-1) \\ \hline \tilde{\mathbf{x}}(\frac{N}{2}+1,\cdots,N-1) \end{array}\right] \tag{3.77}$$

*where,* $\mathbf{I_{N/2}}$ *is the Identity matrix of size* $\frac{N}{2}$
*and* $\mathbf{A}$ *corresponds to* $(\mathbf{R_{\frac{\pi}{4}}} \otimes \mathbf{I_{\frac{N}{2}}})$.

#### 3.4.1.3 The relationship between the rows of DCT and DST matrices

The left half of the DST matrix, $\mathbf{T_N}$, for any row, $k_s$, can be written as below using equations 3.74 and 3.75.

$$\begin{aligned} S_N^{nk_s} &= \sin\left((4n+1)\frac{\pi k_s}{2N}\right) \\ &= \cos\left(\frac{\pi}{2}-(4n+1)\frac{\pi k_s}{2N}\right) \\ & \quad for\ n=0,\ldots,N/2-1 \\ & \quad k_s = 1,\ldots,N \end{aligned} \tag{3.78}$$

Similarly, the left half of the DCT matrix, $\mathbf{D_N}$, for any row, $k_s$, can be written as below using equations 3.45 and 3.46.

$$\begin{aligned} C_N^{nk_c} &= \cos\left((4n{+}1)\frac{\pi k_c}{2N}\right) \\ &\quad for\ n = 0,\ldots,N/2{-}1 \\ &\quad\quad k_c = 0,\ldots,N{-}1 \end{aligned} \tag{3.79}$$

Then, the condition for $S_N^{nk_s} = C_N^{nk_c}$ is

$$\begin{aligned} \left((4n{+}1)\frac{\pi k_c}{2N}\right) &= 2l\pi \pm \left(\frac{\pi}{2} - (4n{+}1)\frac{\pi k_s}{2N}\right) \\ &\quad where,\ l \in \mathcal{N} \\ \left((4n{+}1)\frac{\pi(k_c \pm k_s)}{2N}\right) &= 2l\pi \pm \frac{\pi}{2} \qquad (3.80) \\ for\ n = l, \quad k_c \pm k_s &= N \\ feasible\ solution & \\ k_c &= N - k_s \qquad (3.81) \end{aligned}$$

Further, the normalisation constants also share the same relationship, according to their definitions as in equations 1.5-1.6 and 3.66-3.67.

$$\alpha_{k_c} = \alpha_{(N-k_s)} \tag{3.82}$$

The right halves of the DCT and the DST matrices correspond to the second half, i.e. $\tilde{x}(\frac{N}{2}+n)$ of the rearranged input, $\tilde{x}(n)$. It is seen that the signs of $\tilde{x}(\frac{N}{2}+n)$ in equations 3.45-3.46 for the DCT are opposite to those in equations 3.74-3.75 for the DST. With this observation, the right half of the DCT and the DST matrices can be regarded as the same by negating the input $\tilde{x}(\frac{N}{2}+n)$. This observation, coupled with the row permutation resulting from the relationship in equations 3.81 and 3.82 leads to computing the DST coefficients using the DCT processes.

$$\begin{aligned} \mathbf{X} &= \mathbf{DST_N} \times \mathbf{x} \\ &= \mathbf{P_{rows}} \times \mathbf{DCT_N} \times \tilde{\mathbf{x}} \end{aligned} \tag{3.83}$$

where, $\tilde{\mathbf{x}}$ is defined as below.

$$\tilde{\mathbf{x}} = \left\{ \begin{array}{lcl} \tilde{\mathbf{x}}(n) & = & \mathbf{x}(2n+2) \\ \tilde{\mathbf{x}}(N+\frac{n}{2}) & = & -\mathbf{x}(N-(2n+1)) \end{array} \right\} \; n = 0,\ldots,N/2-1. \tag{3.84}$$

The $\mathbf{P_{rows}}$ consists of operations to rearrange DCT rows into DST rows in increasing frequency order using equation 3.81.

### 3.4.1.4 The block diagram for the IDST

With the above derivation, the IDST is computed using the previously designed IDCT algorithm. The block diagram and the pseudo code for the IDST are as in Figure 3.15 and Figure 3.16. The column permutation stage at the beginning corresponds to $x(n) \rightarrow \tilde{x}(n)$ conversion shown in equation 3.70, whereas the row permutation stage at the end corresponds to the combined action of ordering rows to achieve the DST from the DCT (as in equation 3.81) and rearranging the upper and lower halves of the DST matrix into increasing frequency order. The pseudo code for the middle stage, integer DCT operations, are as in Figure 3.13. The inverse transform is implemented by reversing the recursive operations and the lifting steps by changing the sign and the order of operation of lifting equations.

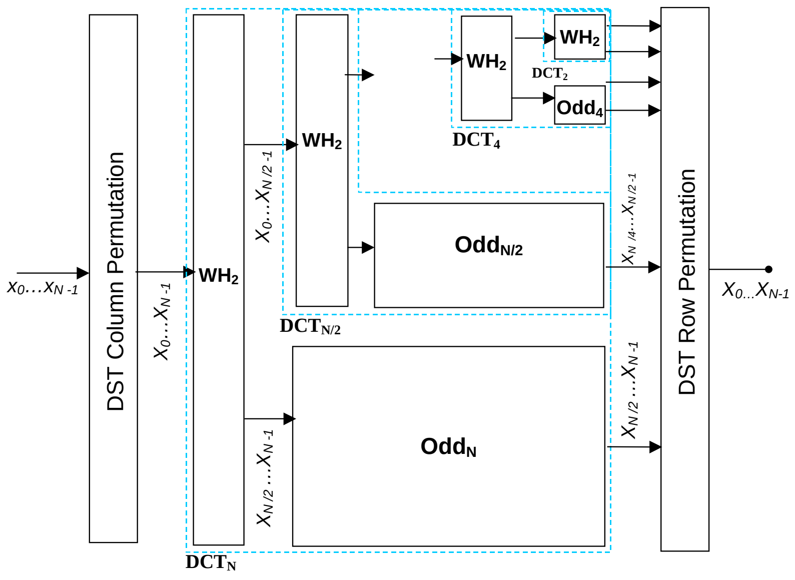


Figure 3.15: Block diagram for integer $\mathbf{IDST_N}$.

```
% Call the function as below for N-point IDST
    Do_Column_order_for_DST(input, N)   % Equation 3.70
    LiftDCT(input, N, 0)                % Figure 3.13
    Row_order_frm_DCT_to_DST(input, N)  % Equation 3.81
    Do_Row_order_for_DST(input, N)      % to arrange in frequency order
```

Figure 3.16: The Pseudo code for integer $\mathbf{IDST_N}$.

As an example, the signal flow diagram for the forward $\mathbf{IDST_8}$ is shown in Figure 3.17. The inverse is computed by following the inverse signal flow with the reversed operations including lifting, row / column permutations and sign changes.

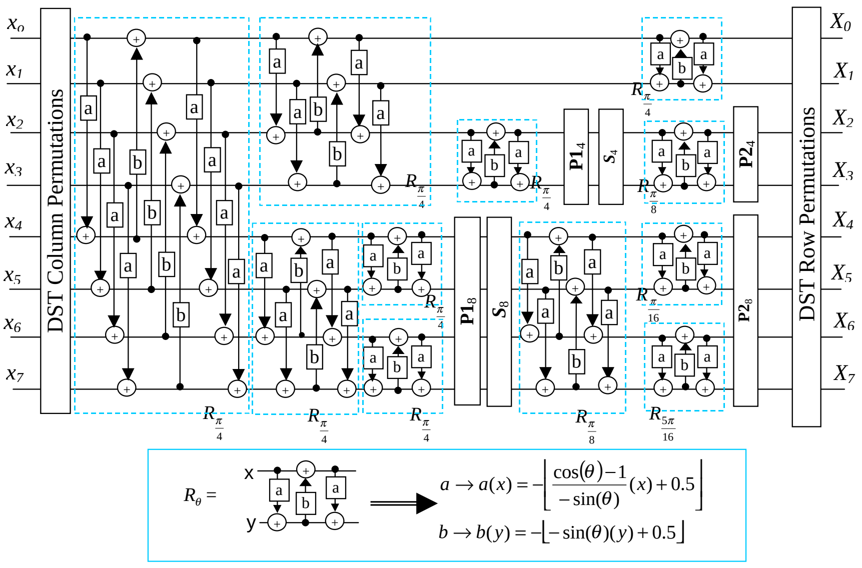

DST Column Permutations
DST Row Permutations
x0 x1 x2 x3 x4 x5 x6 x7
X0 X1 X2 X3 X4 X5 X6 X7
P1_8 S_8 P1_4 S_4 P2_4 P2_8
R_{π/4} R_{π/8} R_{π/16} R_{5π/16}
R_θ =
a → a(x) = −⌊(cos(θ)−1)/(−sin(θ)) (x) + 0.5⌋
b → b(y) = −⌊−sin(θ)(y) + 0.5⌋

Figure 3.17: Signal flow diagram for **IDST₈**.

### 3.4.2 The zero-order entropy values

The performance of the IDST on lossless image coding with different block sizes is presented here. The total image mean is removed from the image in order to minimise the effect of dc frequency leakage into other frequencies, which is intrinsic to the DST. For each block the transform is applied separately on rows and columns and then the block structure is converted to the corresponding wavelet packet tree structure introduced on page 41 using the same algorithm. The weighted entropy values calculated based on the packet sub bands for the image set using the IDST for different block sizes (N) are as in Table 3.4.

| N | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| Gold Hill | 5.631 | 5.710 | 5.605 | 5.382 | 5.053 |
| Barbara1 | 5.965 | 6.010 | 5.875 | 5.633 | 5.232 |
| Barbara2 | 5.934 | 6.030 | 5.907 | 5.656 | 5.313 |
| Boats | 5.203 | 5.354 | 5.321 | 5.157 | 4.866 |
| Black Board | 4.809 | 4.968 | 4.975 | 4.930 | 4.727 |
| Average | 5.508 | 5.614 | 5.537 | 5.352 | 5.038 |

Table 3.5: Weighted zero-order entropy values in bpp for IDST

On average, the 32-point IDST has recorded the best entropy performance. As seen in previous block transforms, it is also evident that the performance improved as the block size, N, of the IDST increased. However, the 4-point IDST has shown the worst performance on average and for most of the test images.

## 3.5 Integer Non-Linear Transforms (INLT)

In section 2.1.1, the existing work on sub band based predictive techniques [41, 42, 44, 45, 46] were introduced. In those examples, the authors reported separate use of non-linear predictions [45] and multiscale decompositions [46]. In this section, the use of median based non-linear prediction guided sub band splitting as an integer non-linear transform (INLT) is investigated. The support neighbourhood for median prediction in each sub band, the update method for the LL sub band and the number of decomposition scales for the INLTs are also analysed.

### 3.5.1 INLT Design

In an early example of designing perfect reconstructing non-linear filter banks, the splitting of the input signal into two channels and the use of a lattice structure based four or less non-linear functions, based on a new theoretical framework for non-linear filter banks, have been presented in [100].

A pyramidal coder using a non-linear filter bank based on quincunx sub band splitting was introduced in [101], in which the non-linear transform was obtained by hierarchical application of median filter predictor at the sub sampled versions of the original image.

In this section, three types of non-linear transforms, namely, INLT1, INLT2 and INLT3, which are based on quincunx splitting followed by non linear sub band coding, similar to that in [101] are considered. In this sub band splitting, it is assumed that the pixels in the input image represent one of the four polyphase components: 00, 01, 10 and 11, as shown in Figure 3.18. The input image is split into four sub bands which correspond to each polyphase component. For an input image, $x$, the notation $x_{00}, x_{01}, x_{10}$ and $x_{11}$ is used to represent the four sub bands.

| | | | | | | |
|---|---|---|---|---|---|---|
| 00 | 01 | 00 | 01 | 00 | 01 | 00 |
| 10 | 11 | 10 | 11 | 10 | 11 | 10 |
| 00 | 01 | 00 | 01 | 00 | 01 | 00 |
| 10 | 11 | 10 | 11 | 10 | 11 | 10 |
| 00 | 01 | 00 | 01 | 00 | 01 | 00 |
| 10 | 11 | 10 | 11 | 10 | 11 | 10 |

Figure 3.18: Pixel labelling for quincunx splitting

#### 3.5.1.1 INLT1

The INLT1 is the same as the non-linear filter used in [101]. This is used as a benchmark for the comparison of performance of the INLT2 and the INLT3.

The prediction functions used for $x_{01}, x_{10}$ and $x_{11}$ are as in equations 3.85-3.87.

$$\begin{aligned} x_{01} &= x_{01} - \lfloor \mathcal{F}(x_{00}, x_{11}) \rfloor & (3.85) \\ x_{10} &= x_{10} - \lfloor \mathcal{F}(x_{00}, x_{11}) \rfloor & (3.86) \\ x_{11} &= x_{11} - \lfloor \mathcal{F}(x_{00}) \rfloor & (3.87) \end{aligned}$$

The median interpolator is used as the prediction function $\mathcal{F}$ due to its low complexity, its good interpolation performance at edges, whenever the edge is horizontal, vertical or diagonal and its capability of discarding impulse noise components. The prediction masks used are shown in Figure 3.19. The polyphase component $x_{00}$ is regarded as the LL output and used as the input to the next level of decompositions using the same process as above.

| | | |
|---|---|---|
| | 11 | |
| 00 | **(01)** | 00 |
| | 11 | |

| | | |
|---|---|---|
| | 00 | |
| 11 | **(10)** | 11 |
| | 00 | |

| | | |
|---|---|---|
| 00 | | 00 |
| | **(11)** | |
| 00 | | 00 |

Figure 3.19: The prediction masks for $x_{01}, x_{10}$ and $x_{11}$ for the INLT1.

#### 3.5.1.2 INLT2

The INLT2 uses the same sub band decompositions as in the INLT1. An additional step is included to update the $x_{00}$ component before using it as the input of the next level. In a non-linear sub band decomposition example for lossy image coding [102], it was concluded that employing an updating filter, which is 0.5 of the median of the update mask, had improved the PSNR results. Since the main object of using integer transforms in lossless coding is to decode the lossless bit streams at low bit rates, the INLT2 is designed to investigate the effect of updating in multilevel non-linear sub band decompositions. The update function is as in equation 3.88 and the function $\mathcal{F}$ is the median of the update mask shown in figure 3.20. The prediction functions are the same as equations 3.85-3.87.

$$x_{00} = x_{00} + \left\lfloor \frac{1}{2} \mathcal{F}(x_{11}) \right\rfloor \tag{3.88}$$

| | | |
|---|---|---|
| 11 | | 11 |
| | **(00)** | |
| 11 | | 11 |

Figure 3.20: The update mask for $x_{00}$ for INLT2.

#### 3.5.1.3 INLT3

The INLT3 is designed to demonstrate a novel selection of the elements of the prediction and the update masks used in the INLT1 and the INLT2. The prediction and updating functions are as in equations 3.89-3.92 and the templates are as in figure 3.21.

$$x_{11} = x_{11} - \lfloor \mathcal{F}(x_{00}, x_{01}, x_{10}) \rfloor \tag{3.89}$$
$$x_{10} = x_{10} - \lfloor \mathcal{F}(x_{00}, x_{01}) \rfloor \tag{3.90}$$
$$x_{01} = x_{01} - \lfloor \mathcal{F}(x_{00}) \rfloor \tag{3.91}$$
$$x_{00} = x_{00} + \left\lfloor \frac{1}{2} \mathcal{F}(x_{01}, x_{10}, x_{11}) \right\rfloor \tag{3.92}$$

| | | |
|---|---|---|
| 00 | 01 | 00 |
| 10 | **(11)** | 10 |
| 00 | 01 | 00 |

| | | |
|---|---|---|
| 01 | 00 | 01 |
| | **(10)** | |
| 01 | 00 | 01 |

| | | |
|---|---|---|
| | | |
| 00 | **(01)** | 00 |
| | | |

| | | |
|---|---|---|
| 11 | 10 | 11 |
| 01 | **(00)** | 01 |
| 11 | 10 | 11 |

Figure 3.21: The prediction and update masks for $x_{11}, x_{10}, x_{01}$ and $x_{00}$ for INLT3.

### 3.5.2 The zero-order entropy values

These non-linear transforms, applied on the test images, result in a sub band structure similar to that of the wavelet transforms. The performance of the above transforms, measured using the weighted entropy values for up to five scales of decompositions, are shown in Table 3.6- Table 3.8. The average values for the test image set are summarised in Table 3.9.

It is evident from the tables 3.6-3.8 that as the number of scales increases, the weighted entropy values decrease, providing the best results for 5 scales. On average, a weighted entropy reduction of 0.5 bpp was experienced by using 5 decomposition levels compared to a single level decomposition. Out of three transforms considered, the INLT3 outperformed the other two by 0.06 bpp on average for the 5-scale decomposition case. The INLT2 has also shown a slight reduction of the weighted entropy values over the INLT1.

| Scales | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Gold Hill | 5.547 | 5.153 | 5.076 | 5.061 | 5.057 |
| Barbara1 | 5.947 | 5.653 | 5.599 | 5.589 | 5.587 |
| Barbara2 | 5.953 | 5.663 | 5.605 | 5.594 | 5.591 |
| Boats | 5.199 | 4.821 | 4.751 | 4.738 | 4.735 |
| Black Board | 4.864 | 4.473 | 4.407 | 4.395 | 4.393 |
| Average | 5.502 | 5.153 | 5.088 | 5.075 | 5.073 |

Table 3.6: Weighted zero-order entropy values in bpp for INLT-1

| Scales | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Gold Hill | 5.548 | 5.157 | 5.081 | 5.065 | 5.061 |
| Barbara1 | 5.946 | 5.651 | 5.596 | 5.585 | 5.583 |
| Barbara2 | 5.951 | 5.657 | 5.600 | 5.587 | 5.584 |
| Boats | 5.201 | 4.822 | 4.752 | 4.738 | 4.734 |
| Black Board | 4.852 | 4.465 | 4.396 | 4.383 | 4.380 |
| Average | 5.499 | 5.150 | 5.085 | 5.072 | 5.068 |

Table 3.7: Weighted zero-order entropy values in bpp for INLT-2

| Scales | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Gold Hill | 5.496 | 5.083 | 5.001 | 4.984 | 4.980 |
| Barbara1 | 5.897 | 5.589 | 5.528 | 5.517 | 5.514 |
| Barbara2 | 5.949 | 5.662 | 5.601 | 5.587 | 5.584 |
| Boats | 5.154 | 4.771 | 4.701 | 4.686 | 4.682 |
| Black Board | 4.781 | 4.377 | 4.306 | 4.293 | 4.290 |
| Average | 5.455 | 5.096 | 5.027 | 5.013 | 5.010 |

Table 3.8: Weighted zero-order entropy values in bpp for INLT-3

| Scales | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| INLT-1 | 5.502 | 5.153 | 5.088 | 5.075 | 5.073 |
| INLT-2 | 5.499 | 5.150 | 5.085 | 5.072 | 5.068 |
| INLT-3 | 5.455 | 5.096 | 5.027 | 5.013 | 5.010 |

Table 3.9: Summary of average weighted zero-order entropy values in bpp for INLTs

## 3.6 Discussion

In this chapter, the concepts of integer transforms using lifting were introduced with the use of lifting factorisation of the wavelet transforms. Novel integer versions of the WHT, the DCT, and the DST, influenced by the lifting concepts and the fast transform implementation techniques, were designed using the lifting techniques and exploiting the intrinsic properties of those transforms.

The IWHT was designed using factorisation of the WHT matrix (including the normalising factor) into sub matrices of the Kroneckor products of the $\mathrm{WHT}_2$ and the corresponding identity matrices, which leads to the conversion of the $\mathrm{WHT}_N$ into applying the $\mathrm{WHT}_2$ along a binary tree recursively to the lower and the upper halves of the signal. The integer version was designed by implementing the $\mathrm{WHT}_2$, which is also similar to the S transform, in integer form using lifting steps.

The IDCT-II was designed by considering its intrinsic properties that lead to partitioning the transform matrix into four quadrants. With some row and column permutation it was seen that the upper left quadrant of the N point DCT transform matrix is the transform matrix for the $\frac{N}{2}$-point DCT. Further, the left and right halves in the upper half share the same signs, whereas those in the lower half contain the opposite signs. Using those properties, at any given N-point, the integer $\mathrm{WHT}_2$ is applied first, so that the normalising constants are automatically incorporated into the upper halves at each stage. The upper half is recursively $\frac{N}{2}$-point DCT transformed until N=2. The lower half of the N-point DCT, which corresponds to the odd-indexed rows, is computed using the $\mathrm{IWHT}_{\frac{N}{2}}$ followed by the Kroneckor products of rotations by the basic angles and corresponding Identity matrices. The use of IWHT and the lifting factorisation of the rotation matrix enabled the integer implementation of the N-point DCT transform, where N is an integer power of two, including the normalising factors.

The IDST-II was designed by using the one-to-one relationship of the DCT and the DST coefficient matrices. The IDST coefficients were computed by incorporating column and row permutation, derived from their relationship, into either ends of the IDCT procedure. This relationship can be used to compute the DST coefficients using any DCT processors, especially in fast transform applications.

In all the above three designs, the transforms can be implemented as in-place operations, which is an added advantage in software/hardware implementations.

Finally, the non-linear transforms were devised in order to investigate their usability in lossless image coding. This was done by using a median based non-linear prediction function in predicting the pixels in sub bands obtained by quincunx splitting. A non-linear updating method was also employed in these transforms.

### 3.6.1 The comparison of the transform performances

The integer transforms, with scales of decompositions of 1, 2, 3, 4 and 5 for pyramidal sub band based transforms and corresponding block sizes of 2, 4, 8, 16 and 32 for block based orthogonal transforms, were tested on the test image set. It is generally known that the wavelet transforms provide the best performance when used with a greater number of scales. From the results already presented in this chapter, it can be concluded that all the transforms provide their best performance when applied with 5 scales for pyramidal transforms or with 32×32 block sizes for block based orthogonal transforms. Table 3.10 summarises the weighted entropy values for all the transforms with 5 scales or with 32×32 blocks applied on the test image set.

| | Gold Hill | Barbara1 | Barbara2 | Boats | Black Board | Average |
|---|---|---|---|---|---|---|
| IWT (5 scales) | | | | | | |
| (4,4) | 4.702 | 4.787 | 5.008 | 4.192 | 3.878 | 4.513 |
| (2,2) | 4.705 | 4.958 | 5.066 | 4.234 | 3.888 | 4.570 |
| (2+2,2) | 4.694 | 4.808 | 5.024 | 4.183 | 3.870 | 4.516 |
| (4,2) | 4.702 | 4.810 | 5.024 | 4.195 | 3.886 | 4.523 |
| S (1,1) | 5.038 | 5.487 | 5.453 | 4.643 | 4.172 | 4.959 |
| S+P | 4.759 | 4.876 | 5.041 | 4.269 | 3.974 | 4.584 |
| Block transforms (32×32) | | | | | | |
| $IWHT_{32}$ | 4.870 | 5.172 | 5.253 | 4.663 | 4.247 | 4.841 |
| $IDCT_{32}$ | 4.626 | 4.514 | 4.824 | 4.214 | 3.925 | 4.421 |
| $IDST_{32}$ | 5.053 | 5.232 | 5.313 | 4.866 | 4.727 | 5.038 |
| Non-linear transforms (5 Scales) | | | | | | |
| INLT-1 | 5.057 | 5.587 | 5.591 | 4.735 | 4.393 | 5.073 |
| INLT-2 | 5.061 | 5.583 | 5.584 | 4.734 | 4.380 | 5.068 |
| INLT-3 | 4.980 | 5.514 | 5.584 | 4.682 | 4.290 | 5.010 |

Table 3.10: Summary of weighted zero-order entropy values (bpp).

Out of all the integer wavelet transforms considered, the (4,4) IWT provides the best performance on all images due to its greater number of vanishing moments in the primal and dual lifting steps. The IWT results are in accordance with the number of vanishing moments involved in the lifting steps. As seen from the results, the greater the vanishing moments involved in lifting steps, the lower the weighted entropy values. Although the S+P transform shows better performance than the S transform, due to the additional prediction step introduced in the S+P transform, it does not outperform other IWTs.

The IDCT has the best performance out of all the block transforms and out of all the other transforms. As expected, the IDST performance on lossless image coding is the worst out of all the block based transforms. This may be due to the IDST's inapplicability to highly correlated images. The IWHT, which can also be considered as a wavelet packet decomposition of the S transform, performs better than the S transform; however, it does not outperform the S+P transform.

Overall, the non-linear transforms resulted in the highest weighted entropy values, thus providing the worst lossless performance. However, the INLT3, the best of the three non-linear transforms considered, outperforms the IDST on average and for most of the test images.

From the table it can be seen that the performance of the IDCT is the best for lossless still image coding. On average, the IDCT gains an advantage of 0.09 bpp over the second best transform, the (4,4) IWT.

In this section, only the zero-order entropy values were compared for different transforms. The entropy coding of the above integer transform coefficients in an embedded coding frame work is discussed in Chapter 4 and Chapter 5. In entropy coding, further reductions of bit rates can be gained by employing efficient scanning techniques and coding contexts. The lossless and rate distortion performance of each transform are compared for different block sizes and scale levels for both intra frames and non-intra frames in Chapter 5 and Chapter 6 respectively.

# Chapter 4

# Embedded Quantiser Design

## 4.1 Introduction

In the previous chapter, the transforms that map integers into integers were presented in terms of their design, implementation and the performance on lossless image coding, measured in zero-order entropy values for the test image set. The next step of lossless image coding is entropy coding, by which the integer coefficients are packed efficiently by exploiting coding redundancy. As mentioned earlier in section 2.4, an embedded coding framework has been used in this research. This chapter discusses the embedded coding techniques and their usage on embedded to lossless image coding. The rest of the chapter is organised as follows. Section 4.2 introduces embedded coding, including the coding steps and the integer coefficient weighting. Section 4.3 analyses the existing methods for scanning coefficients and presents a novel and more efficient scanning scheme, Adaptive Quadtree Splitting (AQS), while sections 4.4 and 4.5 present the coding of the sign and coefficient refinement in an embedded coding framework.

## 4.2 Embedded Coding

As defined in Definition 1.1 (on page 10), in embedded coding, bit streams for all other lower bit rates are embedded within any given bit rate of the coded image bit stream. This is achieved by grouping the bits in the coded bit stream according to their significance. The embedded coding algorithms use scalar quantisation in several passes, starting with a quantisation bin size corresponding to the largest quantisation step that gives at least one non-zero quantised coefficient and thereafter reducing the bin size progressively in successive passes up to the targeted bit rate. In lossless embedded coding, the above process is continued up to the unit quantisation bin size. Defining these quantisation bin sizes as $2^n$ with $n \in \{ msb, \ldots, 1, 0 \}$ ($msb$ is the most significant bit plane number) corresponds to bit plane-wise embedded coding of the coefficients in the sign magnitude binary representation. As in most of the published work [14, 13, 15], a bit plane based embedded coding technique was used in this research. The advantage of bit plane coding is that the output from each quantisation level is binary, so that it can be encoded using binary entropy coding.

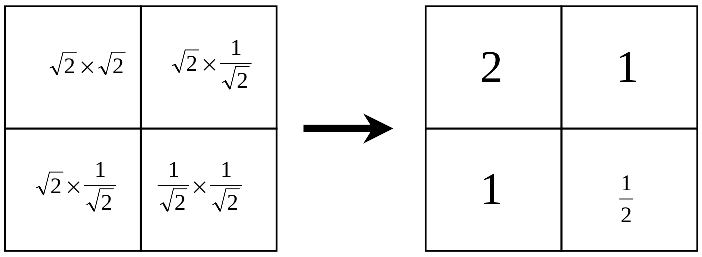
### 4.2.1 Weighted Bit Planes (WBP)

In order to achieve exact integer representation, the lifting steps in the integer wavelet and the Walsh Hadamard transforms presented in Chapter 3 were performed ignoring the normalising (scaling) lifting steps, which correspond to the **K** matrices. In such a coefficient domain, a bit plane across the whole coefficient set does not represent the same significance (according to the rms error) for all the sub bands, as the normalising factors depend on the transform scale and the sub band where the coefficients belong. Therefore, in order to adjust the relative significance of the coefficients in different sub bands, such transform coefficients have to be normalised. A method to normalise the transform coefficients by weighting the sub bands according to the corresponding normalising mask is presented in the following sub sections.

#### 4.2.1.1 For the IWT

In the integer wavelet transforms listed in Appendix A, the normalising operations with the factor $k = \sqrt{2}$ were excluded from the lifting steps. In a 1-D transform, the net effect of normalising is multiplying the coefficients in the low pass sub band ($s$) by $\sqrt{2}$ and dividing the coefficients in the high pass sub band ($d$) by $\sqrt{2}$. In a 2-D separable transform the net effect of normalising for a single scale is multiplying the coefficients in each sub band, namely, LL, LH, HL and HH by 2, 1, 1 and $\frac{1}{2}$ respectively as in Figure 4.1. The zeroth scale normalising factor ($\mathrm{NF}_0$) set is $\left\{ 2,\ 1,\ 1,\ \frac{1}{2} \right\}$. The normalising factors for higher transform sub bands ($\mathrm{NF}_i$) are found by multiplying the normalising factor for the LL sub band ($\mathrm{NF}_{(i-1)_{LL}}$) with the $\mathrm{NF}_0$ (Figure 4.2).

As all the normalising factors are powers of two, the normalising of coefficients is performed by shifting operations. All shifting operations are left-bound and performed



$\sqrt{2}\times\sqrt{2}$
$\sqrt{2}\times\frac{1}{\sqrt{2}}$
$\sqrt{2}\times\frac{1}{\sqrt{2}}$
$\frac{1}{\sqrt{2}}\times\frac{1}{\sqrt{2}}$
2
1
1
$\frac{1}{2}$

Figure 4.1: The net effect of scaling on sub bands.

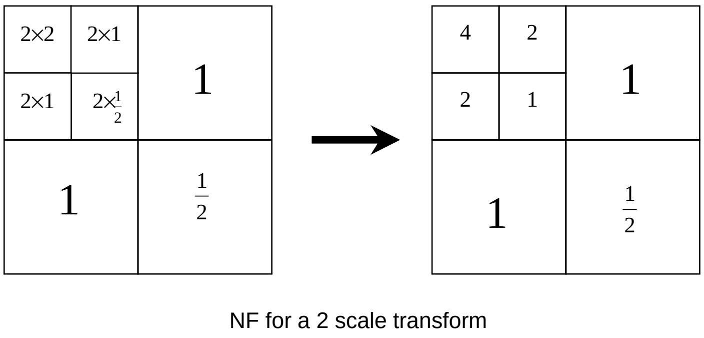


Figure 4.2: The normalising factors for higher scales.

relative to the smallest normalising factor. The number of bits ($b_{r_{XX}}$) to be leftward shifted in the sub band $r_{XX}$ is found as in equation 4.1.

$$b_{r_{XX}} = \log_2\left(\frac{NF_{r_{XX}}}{min\left\{NF_{(s-1)_{LL}}, \cdots, NF_{0_{HH}}\right\}}\right) \tag{4.1}$$

The smallest normalising factor for the IWT is ($\frac{1}{2}$), which occurs in the HH$_0$ sub band.

This leftward shifting operation can also be interpreted as vertically sliding of bit planes in the sub bands, when all coefficients are considered as a 3-D cube of ones and zeros. The weighted bit planes (WBP) are obtained by upward sliding of the coefficient bits in each sub band by the number of bit planes calculated from the equation 4.1 relative to the HH$_0$ sub band (Figure 4.3). However, in software implementation, no actual bit plane sliding is required. Instead, the lowest significant bit index after weighting, which is the also same as the number of bit planes to be slid, $b_{r_{XX}}$ (as in equation 4.1), is used as a threshold to stop refining bits for a given sub band. With this WBP arrangement, the original bit space, according to the dynamic range of the coefficients prior to normalising, is preserved. This prevents coding of bits that contain no information. No additional information regarding those thresholds needs to be coded, as they can be computed using equation 4.3 at the decoding end. Further, this method avoids any overhead bits that are incurred due to the expanded dynamic range resulting from the four extra lifting steps method for lossless normalising, as described in section 3.2.2.2.

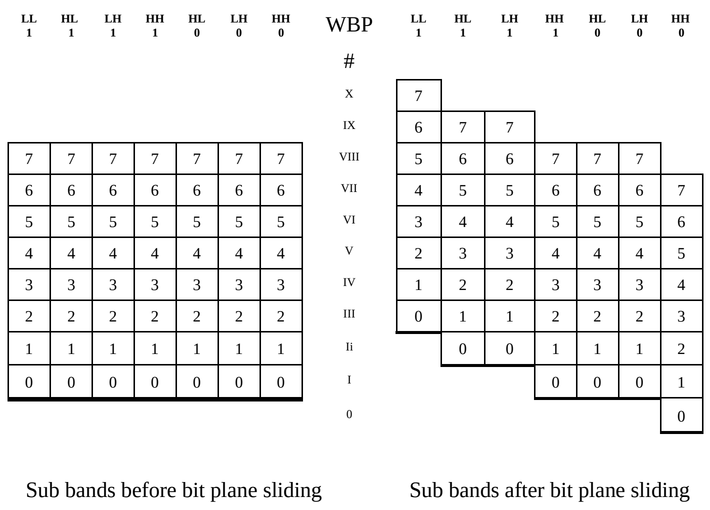

LL 1
HL 1
LH 1
HH 1
HL 0
LH 0
HH 0
WBP #
X
IX
VIII
VII
VI
V
IV
III
Ii
I
0
Sub bands before bit plane sliding
Sub bands after bit plane sliding

Figure 4.3: The weighted bit planes by bit plane sliding.

#### 4.2.1.2 For the IWHT

Since the IWHT is a block based transform, the transformed coefficients need to be rearranged in the corresponding tree order as in Figure 3.6 prior to employing any embedded quantising.

The scaling factors for a 2-D 2-point IWHT are the same as those for a single scale IWT as shown in Figure 4.1. The scaling factors for a 2-D N-point IWHT, where N>2, become a recursive multiplication of $\mathbf{KK_2}$, the scaling mask for a 2-point IWHT, along a quadtree in the coefficient domain.

$$\mathbf{KK_2} = \begin{array}{|c|c|} \hline \times 2 & \times 1 \\ \hline \times 1 & \times \frac{1}{2} \\ \hline \end{array}$$

An example for N=4 is shown in Figure 4.4.

$$\mathbf{KK_4} = \begin{array}{|c|c|}\hline \times 2 \begin{array}{|c|c|}\hline \times 2 & \times 1 \\ \hline \times 1 & \times \frac{1}{2} \\ \hline\end{array} & \times 1 \begin{array}{|c|c|}\hline \times 2 & \times 1 \\ \hline \times 1 & \times \frac{1}{2} \\ \hline\end{array} \\ \hline \times 1 \begin{array}{|c|c|}\hline \times 2 & \times 1 \\ \hline \times 1 & \times \frac{1}{2} \\ \hline\end{array} & \times \frac{1}{2} \begin{array}{|c|c|}\hline \times 2 & \times 1 \\ \hline \times 1 & \times \frac{1}{2} \\ \hline\end{array} \\ \hline\end{array} \rightarrow \begin{array}{|c|c|c|c|}\hline \times 4 & \times 2 & \times 2 & \times 1 \\ \hline \times 2 & \times 1 & \times 1 & \times \frac{1}{2} \\ \hline \times 2 & \times 1 & \times 1 & \times \frac{1}{2} \\ \hline \times 1 & \times \frac{1}{2} & \times \frac{1}{2} & \times \frac{1}{4} \\ \hline\end{array}$$

Figure 4.4: The scaling factors for 2-D IWHT$_4$ block.

Therefore, in a 2-D IWHT, separable implementation of the lifting steps for the matrix **K**, as in equations 3.27-3.31 is replaced with the bit plane weighting according to the mask $\mathbf{KK_N}$ for each $\mathbf{WH_N}$ block. The same virtual bit plane sliding can be performed as above, relative to the smallest scaling factor, which occurs in the (N-1,N-1)$^{th}$ element in the 2-D IWHT$_N$. With this method, any further increment in the dynamic range of the coefficients can be avoided.

#### 4.2.1.3 For the other transforms

The integer implementation of the IDCT and the IDST, introduced in sections 3.3 and 3.4 respectively, includes the normalisation constants into the factorisation of those matrices. Therefore, a scaling operation at this stage is not required.

The INLTs introduced in section 3.5 do not involve any scaling due to their non-linear prediction and updating steps, which do not obey the orthogonality property. However, in order to arrange the coefficients according to their contribution to the total energy, a scaling process similar to that of IWT is used with the INLTs.

### 4.2.2 Embedded coding steps

In bit plane oriented embedded coding, each weighted bit plane is coded from the most significant to the least significant bit plane. Within a bit plane, sub bands are coded from the lowest frequency to the highest and the highest scale to the lowest. For example, in a five scale wavelet transform based embedded coding, the sub bands are ordered as in $\{\, LL_4,\, LH_4,\, HL_4,\, HH_4,\, LH_3,\, \cdots,\, HH_0 \,\}$ according to the increasing frequency.

Each weighted bit plane or output from each quantisation step in general terms is coded in two different coding passes, namely

1. Switching pass and
2. Refining pass.

In the switching pass, coefficients which become significant in the current weighted bit plane are switched on followed by their signs. Refinement of already switched coefficients is done in the refinement pass. Switching in the switching pass is achieved in this research by defining a significance switching mask (SSM) for each weighted bit plane. The following three sections (4.3, 4.4 and 4.5) present the three coding steps of embedded coding, viz., significant switching mask coding, sign coding and refining data coding respectively.

## 4.3 Significance Switching Mask (SSM) Coding

An image, $I$, with $M \times N$ dimensions decomposed using a transform $T$ produces the coefficient set $I_T(x, y)$, where $x = 0, \ldots, M-1$ and $y = 0, \ldots, N-1$. The $r^{th}$ weighted bit plane $WBP_r$ represents a range $\left(2^r, 2^{r+1}\right]$ with a quantisation step $2^r$. Therefore, as seen on the $WBP_r$, the magnitude of any $I_T(x, y)$ can be categorised into three groups.

$$[I_T(x,y)]_{WBP_r} = \begin{cases} S & if \quad 2^r \leq |I_T(x,y)| < 2^{r+1} \\ N & if \quad |I_T(x,y)| < 2^r \\ X & if \quad 2^{r+1} \leq |I_T(x,y)| \end{cases} \tag{4.2}$$

The significance switching mask used in this research uses the same grouping of coefficients as above. In embedded coding terms, the coefficients identified as $S$ type become significant in the $WBP_r$. The $N$ type coefficients are the ones yet to become significant, whereas the $X$ type coefficients are the ones which have already become significant in previously scanned bit planes. The encoder needs to code only the $S$ and $N$ type coefficients in the significance switching mask, since the encoder and the decoder are synchronised according to the mask scanning order.

The cost of embedded coding is the bits used to code $N$ type coefficients in the SSM. As the $N$ type bits represent bits whose indexes are higher than their corresponding most significant bits, coding of $N$ does not transmit information regarding image energy to the decoder. This is illustrated with the rate distortion plots as in Figure 4.5 by comparing the bit rates for a given distortion for an embedded codec with those for a non embedded codec using Gold Hill image. The bit rates are in bpp, calculated using the zero-order entropy formula (equation 1.11) for the overall symbol stream in the embedded coding example and using the weighted entropy formula (equation 3.9) considering 5 level wavelet transform in the non embedded coding example. The distortion measure used is the rms error. The vertical distance between the plots is the cost of embedding for a given image distortion. The embedding cost varies from 30% to 80% with an average around 50% when measured as a percentage of corresponding non-embedded coding bit rates. The bit rates are plotted on the log scale in figure 4.6, so that the cost of embedding at low bit rates are shown clearly.

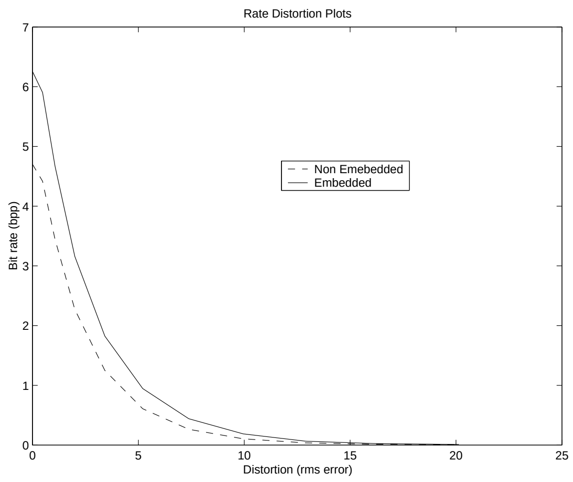


Figure 4.5: The comparison of embedded and non embedded coding.

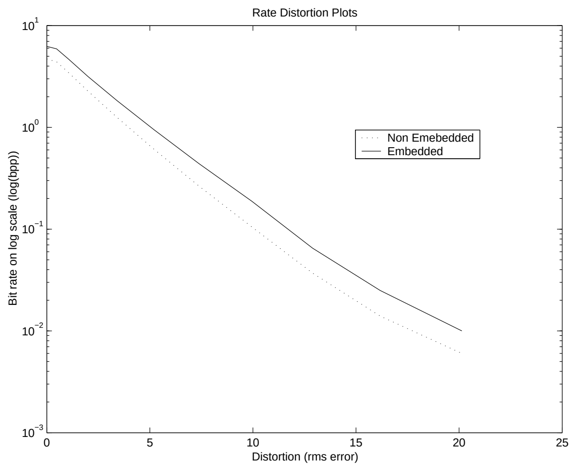
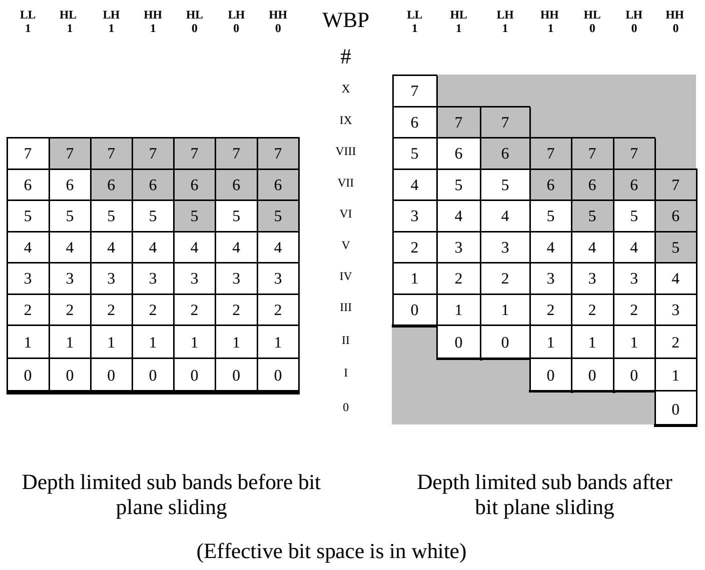


Figure 4.6: The comparison of embedded and non embedded coding (log scale representation).

On the other hand, coding of N bits provides the information regarding the position of the $S$ bits in the SSM for a given bit plane. As both the encoder and the decoder are synchronised with the scanning, careful selection of a scanning scheme which minimises the necessity for coding of $N$ bits would help to reduce the cost of embedding significantly.

Furthermore, the bit space to be coded is reduced by sending the highest msb index for each sub band as side information. This process, called depth limiting, will avoid unnecessary coding of $N$ bits above the highest msb bit for a given sub band. The msb for each sub band can be coded by using at most 4 bits. The bit space after bit plane sliding and depth limiting appears as in Figure 4.7



LL 1
HL 1
LH 1
HH 1
HL 0
LH 0
HH 0
WBP #
X
IX
VIII
VII
VI
V
IV
III
II
I
0
Depth limited sub bands before bit plane sliding
Depth limited sub bands after bit plane sliding
(Effective bit space is in white)

Figure 4.7: The final bit space for embedded coding.

### 4.3.1 Scanning schemes

**Definition 4.1 (Scan)** *A* scan *on a significance switching mask (SSM) with dimensions $M{\times}N$ to order the bits in the SSM is a one-to-one function $f_s$ defining an index in the closed interval $[0,\ldots,(M \times N)-1]$ from the original index pairs $\{(x,y) \ : \ 0 \leq x < M \, , \, 0 \leq y < N\}$.*

$$[0,\ldots,(M \times N)-1] \xleftrightarrow{f_s} \{(x,y) \ : \ 0 \leq x < M \, , \, 0 \leq y < N\}$$

In this section, a few scanning schemes that can be used in SSM coding will be presented and evaluated using the (4,4) integer wavelet coefficients from the test image set. The main object of this exercise is to cluster the $N$-type bits in a given SSM at the rear of the scanning sequence or into a separate group within the scanned sequence, so that coding of such clusters that wholly consist of $N$ bits can be avoided, thereby reducing the cost of embedding.

Scanning schemes discussed in this section can be grouped into three categories.

1. Intra sub band techniques
2. Inter sub band techniques
3. Whole SSM based techniques

#### 4.3.1.1 Intra sub band techniques

In intra sub band techniques, each sub band is scanned separately, without considering any inter sub band dependencies in coefficients. Scanning of coefficients in any single sub band is completed before scanning the following sub bands, which are ordered in the increasing frequency order. The sub band order also features decreasing normalised sub band energy. The intra sub band scanning can be either 1-D (when scanned into an array) or 2-D (when scanned into blocks).

The scanning schemes considered in this exercise are listed below.

**A) Raster :** Left to right, top to bottom scan (Figure 4.8). This scan is useful to exploit horizontal dependencies which are important in modelling coding contexts (Section 5.1.1). This is a 1-D scan, as the bits are ordered into an array.

**B) Zigzag :** This is the scanning method used in the JPEG baseline standard to scan the DCT coefficients in an 8×8 block. In this experiment, the zigzag scan is performed on the whole sub band (Figure 4.8). Since this scan runs from top to bottom along the diagonals, the diagonal directional dependencies can be exploited in the context modelling. This is also a 1-D scan.

**C) Z Scan :** This scan traverses 4 symbols in a square block of 2×2 symbols (min block) as in a letter Z shape and then four min blocks are traversed using the same pattern until the whole sub band is traversed and arranged into an array (Figure 4.8). In this method, horizontal vertical and diagonal dependencies can be exploited. This is also a 1-D scan.

**D) Quadtree :** The quadtree approach is a 2-D technique. A quadtree is a tree structure in which each non-terminating node has four children (Figure 4.9). In quadtree coding, a square block, which is depicted as a node in the tree terminology, is split into four quadrants if it contains at least one $S$ bit or terminates the splitting otherwise. This process is performed until a given minimum block (min block) size is reached, or all the nodes in the tree are terminated.

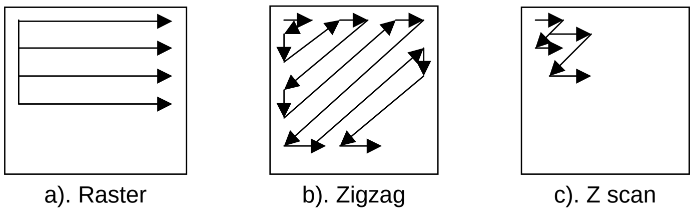


Figure 4.8: 1-D intra sub band scanning techniques.

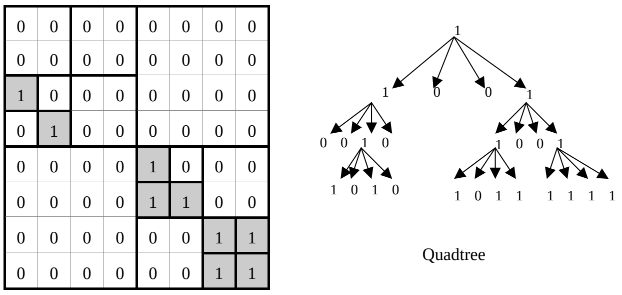


Figure 4.9: Quadtree scanning.

#### 4.3.1.2 Inter sub band techniques

These methods exploit inter sub band dependencies based on the hierarchical representation across the sub bands, as represented by wavelet trees across the scales as in Figure 4.10. Two inter band scanning techniques are to be considered.

**E) Zero Tree :** In this scanning technique, $S$ and $N$ bits on a switching plane are coded using the zero tree symbols introduced in [13]. The $S$ bits are coded with a single symbol irrespective of the sign. The $N$ bits are categorised into two groups, hence represented by either of two symbols. If none of the descending coefficients in the wavelet tree which originated from an $N$ bit is a type $S$, then that $N$ bit is classified as a Zero Tree Root (ZTR) symbol and the descendants from that node need not be coded. Otherwise, it is classified as an Isolated Zero (IZ) symbol and coding along the tree is continued.

**F) WTVHZ :** In this method, for each pixel in the LL sub band, a corresponding wavelet tree as a square is constructed and then each tree (square) is scanned (as in Figure 4.11 ) vertically in HL sub band, horizontally in LH sub bands and z scan in HH sub bands. These scanning patterns for separate sub bands were determined after considering the correlation orientation of the coefficients within a sub band.

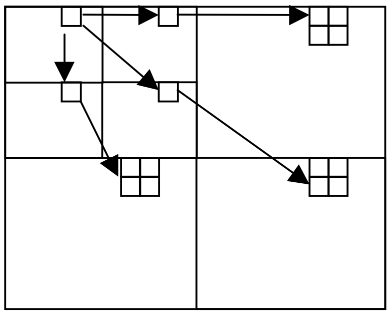


Figure 4.10: Wavelet tree organisation.

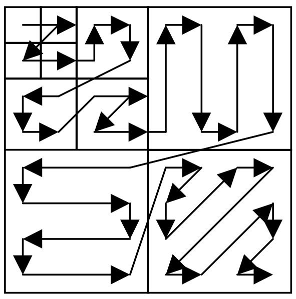


Figure 4.11: WTHVZ scan for a wavelet tree.

#### 4.3.1.3 The whole SSM based techniques

The intra sub band scanning techniques described above can be used on the entire SSM rather than from sub band to sub band. However, using 1-D scans like raster and zigzag in this manner does not scan the coefficients according to their energy contribution, which is based on the sub band they appear in. Quadtree and Z scans can traverse according to the energy ordering for a square shaped image, which is not common in video applications where the aspect ratios are normally 4:3 or 16:9. In this case, this type of scanning methods is excluded from this experiment.

#### 4.3.1.4 Scanning schemes : Analysis

**Raster, Zigzag and Z scans**

The 1-D intra band scanning techniques (A, B and C) presented above produce only two types of symbols (S and N types) as output. The entropy in bits per symbol for such a scan of total number of S and N type symbols, T, and the number of S type symbols, s, using a memoryless model is as below.

$$E1_{A,B,C} = p_s \log_2\left(\frac{1}{p_s}\right) + (1-p_s)\log_2\left(\frac{1}{1-p_s}\right) \qquad (4.3)$$
$$where,\ p_s = \frac{s}{T}$$

Further, coding of N type symbols beyond the last S symbol for a sub band can be avoided by introducing an extra symbol ( 'stop' symbol) to terminate scanning for a sub band as used in the JPEG baseline codec [5]. The entropy in bits per symbol for such a scan of total number of S and N type symbols, T, with the number of S type symbols, s, and the last symbol occurring at the y$^{th}$ position using a memoryless model is as below.

$$E2_{A,B,C} = \frac{1}{T}\left[s\log_2\left(\frac{1}{p_s}\right) + (y-s)\log_2\left(\frac{1}{p_n}\right) + \log_2\left(\frac{1}{p_y}\right)\right] \qquad (4.4)$$
$$where,\ p_s = \frac{s}{y+1},\ \ p_n = \frac{y-s}{y+1},\ \ p_y = \frac{1}{y+1}$$

The performance of $E2_{A,B,C}$ varies with T, s and y. A comparison of $E1_{A,B,C}$ and $E2_{A,B,C}$ for T=100 and different values of y (for y=30, 50, 70 and 90) against different s values ( $s = 1 \cdots y$ for each y value) is shown in Figure 4.12. It is evident that the E2 values provide plots with lower entropy for lower values of y. According to the figure, the E2 plots breakeven with the E1 plot at s=4, 7, 14 and 41 respectively for y=30, 50, 70 and 90. This leads to the conclusion that for a given T value, only a smaller s is required for E2 to outperform E1 when the y value is low. The converse of this is that for a given y value, a smaller s is required for E2 to outperform E1 when the T value is high. This suggests the use of E2 in higher bit planes of high frequency sub bands where T is larger and the N bit count is significantly higher than the S bit count, in order to achieve lower entropy values in those sub bands.

However, the coefficients in a wavelet transform sub band represent the components of a specific frequency band at a local position of a scaled original signal (image). Therefore, the coefficients in a sub band are not arranged according to increasing frequency as in a transform block in an orthogonal block transform, like the DCT. Instead, the coefficients represent the original spatial structure of the image. Therefore,

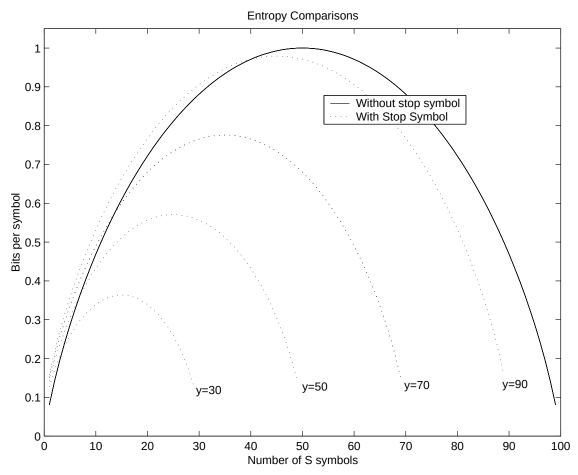


Figure 4.12: Comparison of 1-D scans with a 'stop' symbol and without.

the different 1-D methods, namely, raster, zigzag and Z scan produce the same results using E1. Further, the possibility of them producing significantly different results using E2 is also low due to the above mentioned coefficient arrangement. None of those scans guarantees that any will produce a bit stream with a low y value, which is a requirement for the better performance of E2. Due to these reasons and for its simplicity, only the raster scan without 'stop' symbol is considered in this scanning scheme experiment.

### Quadtrees

The quadtree structure and model for representing 2-D binary data were first introduced in [103] and further probabilistic models were reported in [104]. A traditional quadtree consists of three types of nodes, namely, all white, all black and grey (mixture of white and black). In order to keep the output symbol set binary, only all white and grey nodes are considered in this experiment. This is similar to the pointer quadtree structure, which has only two types of nodes, namely, non-leaf and leaf, introduced in [103]. In the SSM, where there are three types of symbols as opposed to two in binary images, the S type bits are regarded as black, while the N and X types are considered as white. A non-leaf node corresponds to a homogeneous block of white pixels, while a leaf node corresponds to a non-homogeneous block containing at least a single S bit.

A quadtree is normally performed on a square block (max block), the dimensions of which are a power of two, so that the quadtree terminates with the single pixel nodes, corresponding to the elements of the min block. Within a non homogeneous min block, only the N type bits are considered as a node since the X type bits are not coded in the SSM coding.

A max block with a side length $2^d$, generates a quadtree with a maximum depth of $d$ non-terminating (leaf) nodes. The number of leaf nodes and non-leaf nodes generated by the quadtree are determined by the position and the number of S type bits. Furthermore, when the amount of aggregation of the S type bits is minimal, the quadtree representation becomes inefficient, producing more symbols than in a corresponding 1-D scan. The quadtree methods are more efficient when homogeneous regions of N and X type bits are present in the SSM, which normally occurs in the most top bit planes for a given sub band.

In this experiment, max block sizes of 32, 16 and 8 were used for the sub bands in the wavelet scale number $s = 0$, $s = 1$ and $s \geq 2$ respectively and a min block size of 2 was used for all the sub bands to generate the quadtrees, whose cost of representation was computed using the total number of leaf and non-leaf nodes in all the quad trees for a given SSM.

### Wavelet trees

The sub bands are rearranged into a wavelet tree (square) following the inverse of the Block2tree procedure shown on page 42. For example, an $l$ scale wavelet transform applied on an M×N image produces $\frac{M \times N}{2^{2l}}$ wavelet squares of $2^l \times 2^l$ size. This generates a tree for each coefficient in the LL sub band (Figure 4.10 on page 83).

The main difference of the quadtree and the wavelet tree is the origin of the tree. The quadtrees are originated with reference to the centre point, while the wavelet trees are originated from the top left hand corner. The quadtrees contain a four way branching process, whereas the wavelet trees follow a three way branching for the highest level and four way branching thereafter.

Two techniques to code these wavelet trees are considered in this experiment. The first method is the zero tree technique as introduced in the section 4.3.1.2. The zero tree coding outputs three different symbols, the additional symbol being used to represent groups of non-S type bits present in the deeper levels of the wavelet tree. Although it introduces an extra symbol to the alphabet, it reduces the total number of symbols required to represent a tree.

The second method is the WTHVZ method (Figure 4.11 on page 83), introduced in the section 4.3.1.2. A wavelet tree represents the frequency components corresponding to a $2^l \times 2^l$ spatial location. This scanning method tries to use the increasing frequency arrangement within a tree following the coefficient traversal shown in Figure 4.11. Furthermore, the truncation of a tree traversal using a 'stop' symbol can be incorporated into the WTHVZ scanning technique.

In a coding method that uses either of the above methods, after completing the tree scans (zero tree or WTHVZ), which classify the tree elements and identify the elements to be coded, the wavelet squares are organised back to the original form (multiresolutional wavelet transform). This allows the scanning of a given SSM from sub band to sub band from the lowest frequency to the highest frequency.

#### 4.3.1.5 Scanning schemes : Results

From the above discussion, the following scanning techniques were considered in this experiment.

1. Raster scan (Representing 1-D intra band methods)
2. Quadtree scan (2-D intra band technique)
3. Wavelet tree - Zero tree
4. Wavelet tree - WTHVZ

The zero-order entropy values in bpp were computed for each weighted bit plane of the (4,4) IWT coefficients of each image in the test image set. The results for individual images are recorded in the Appendix C. The average entropy values (bpp) for the whole image set are shown in Table 4.1, where the best method for each bit plane is shown in bold font.

From Table 4.1 and the tables in Appendix C, it can be seen that the quadtree method outperforms the other methods, when the total entropy for all bit planes $(13 \cdots 0)$ is considered. The raster technique produces the second best method while the zero tree method records the worst performance, although there is only a 0.082 bpp difference between the best and the worst techniques. The inferior performance of the zero tree technique can be explained by its lack of effect on the WBP bit space resulting from the processes of bit plane sliding and depth limiting. Those processes have already removed the unnecessary $N$ bits, which otherwise would have been efficiently coded by the zero tree symbols.

| WBP | Raster | Quad Tree | WT-ZT | WTHVZ |
|---|---|---|---|---|
| 13 | 0.0009 | **0.0008** | 0.0009 | 0.0009 |
| 12 | 0.0007 | **0.0005** | 0.0006 | 0.0009 |
| 11 | 0.0013 | **0.0010** | 0.0012 | 0.0012 |
| 10 | 0.0033 | 0.0027 | **0.0025** | 0.0038 |
| 9 | 0.0119 | **0.0091** | 0.0094 | 0.0130 |
| 8 | 0.0421 | **0.0280** | 0.0319 | 0.0394 |
| 7 | 0.1046 | **0.0711** | 0.0829 | 0.1025 |
| 6 | 0.1791 | **0.1312** | 0.1494 | 0.1777 |
| 5 | 0.2785 | **0.2204** | 0.2420 | 0.2729 |
| 4 | 0.4030 | **0.3644** | 0.3786 | 0.3986 |
| 3 | **0.5752** | 0.6178 | 0.6290 | 0.5807 |
| 2 | **0.6222** | 0.7073 | 0.7309 | 0.6284 |
| 1 | **0.3620** | 0.4051 | 0.3910 | 0.3676 |
| 0 | **0.0700** | 0.0793 | **0.0700** | 0.0752 |
| $13\cdots0$ | 2.6547 | **2.6387** | 2.7205 | 2.6628 |

Table 4.1: Average zero-order entropy values (bpp) for the test image set

However, the tables show that no single technique has consistently produced the best performance for all the bit planes. In general, the raster scan performs best for the lower bit planes and so does the quadtree scan for higher bit planes. As can be seen from the table, using the quadtree scan from the bit plane 13 to the bit plane 4 and using the raster scan for the rest, provides a total bit rate of 2.4586 bpp, which is lower than those for any other individual scan.

However, the transition bit plane from quadtrees to raster depends on the sub band statistics in a given bit plane for a given image. More advantage can be achieved by adaptively deciding the transition of scans from quadtree to raster and vice versa, if necessary, rather than using a fixed transition point as above. A novel adaptive scan selection scheme, Adaptive Quadtree Splitting (AQS), is investigated in the next section.

### 4.3.2 The Adaptive Quadtree Splitting (AQS)

The quadtree scan and the z scan follow the same bit traversal, the only difference being in the output symbols, for which the quadtree scan uses two quadtree symbols, leaf and non-leaf, to group quadrants of non-S type bits. Therefore, by avoiding the use of leaf and non-leaf grouping, a quadtree scan can be transformed into a Z scan and vice versa. As stated earlier, the final entropy value for the Z scan, which is a 1-D technique, for a sub band is the same as that for the raster scan. Therefore, in this section, the term 'raster' is used to address all the 1-D scans.

#### AQS: The Components

The adaptive switching between the quadtree and the raster is achieved by designing two quadtree algorithms, namely, QT1 and QT2. The first quadtree algorithm, QT1, is same as the previous quadtree scanning, in which quadtree splitting is performed until a predefined min block size, which is 2×2, is met. In a min block, the S and N bits are the only bits output for entropy coding.

The second quadtree function, QT2, adaptively changes the scanning method from quadtree to raster depending on the occurrence of $X$ bits within a block. In this, if a block contains at least one X symbol, the block is split into four sub blocks without producing any quadtree symbols, i.e. leaf and non-leaf symbols, for the block. If the block does not contain any X bits, QT1 is used. This is repeated until the min block size is met. The S and N bits in a min block are entropy coded as those in the QT1 algorithm.

#### AQS: The Algorithm

Coding using the QT1 scan is expensive when the probabilities of $S$ and $N$ bits are similar and when they do not constitute homogeneous regions. This is common in the lower bit planes, due to the random nature of $S$ symbols and the higher number of $X$ symbols. As evident from the Table 4.1, the raster scan outperforms under these conditions. The QT2 algorithm can be used in such cases where the homogeneous regions of non-S type bits are not present. QT2 is the more important scan since it can adaptively change from quadtree scanning to raster scanning. If a block consists of no $S$ bits, but with $X$ and $N$ bits, then QT2 is costly. In this case QT1 outperforms QT2.

The adaptive quadtree splitting technique is designed to choose the better quadtree algorithm adaptively for each block. This is achieved by predicting the current block from the corresponding block in the parent sub band, which has already been scanned and the bits of which are already known to both coder and decoder. If a bit in the parent sub band is $N$ type, then there is a higher probability that all the corresponding bits in the immediate child sub band are $N$ type as well. The zero tree scan used here and in the EZW algorithm, makes this hypothesis to all the descendants of a node in the wavelet tree. The underlying assumption here is that there is a high probability for the normalised magnitude of the coefficients in a parent sub band being greater than that of the corresponding coefficients in the immediate descendant sub bands.

The AQS algorithm predicts four children of a parent as follows:

**N type :** If parent is $N$ type then child bits are $N$ type,

**A type :** If parent is $S$ type then child bits are either $S$ or $N$ type

**B type :** If parent is $X$ type then child bits are either $S$ or $N$ or $X$ type

A predicted block consists of N, A, B and a priori known $X$ symbols from the current block. If the predicted block consists of only N and $X$ symbols, QT1 is used and otherwise QT2 is used.

The above decision is made not only at the beginning of a max block but also at subsequent levels of splitting up to min blocks, where individual $S$ and $N$ bits are output for entropy coding. This type of decision making allows the AQS to adaptively switch between QT1 and QT2 according to the known sub band statistics (from $X$ bits) and the predicted sub band statistics. The flow charts summarising two quadtree algorithms are shown in Figure 4.13 and 4.14 respectively.

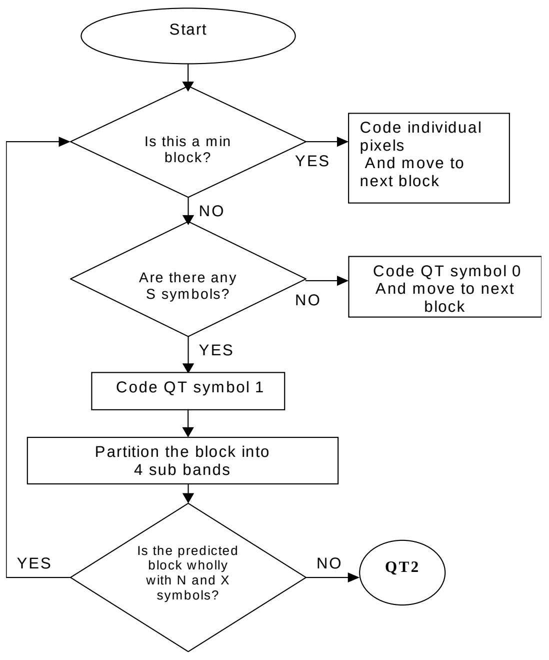

Start
Is this a min block?
YES
Code individual pixels And move to next block
NO
Are there any S symbols?
NO
Code QT symbol 0 And move to next block
YES
Code QT symbol 1
Partition the block into 4 sub bands
Is the predicted block wholly with N and X symbols?
YES
NO
QT2

Figure 4.13: The first quadtree scanning (QT1)

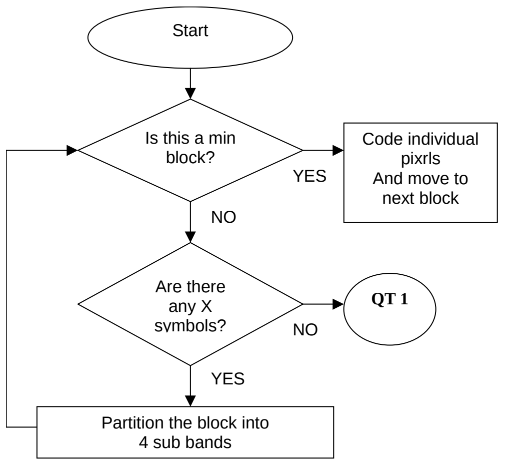

Start
Is this a min block?
YES
Code individual pixrls And move to next block
NO
Are there any X svmbols?
NO
QT 1
YES
Partition the block into 4 sub bands

Figure 4.14: The second quadtree scanning (QT2)

Four sub bands in the highest scale are scanned without any prediction from the parent sub band, as the parent sub band and the three child sub bands belong to the same scale in the wavelet pyramid. Therefore, the decision criteria on choosing QT1 and QT2 for these sub bands are based only on the presence of the $X$ bits in a given block.

**AQS: Performance comparison**

The average zero-order entropy values in bpp for the test image set using the AQS scanning method and initial four scanning schemes are summarised in Table 4.2 for comparison. The bit rates for individual images can be found in Appendix C. The best method for each bit plane is highlighted in bold font.

From Table 4.2 and the tables in Appendix C, it can be seen that the AQS scan, with a total bit rate of 2.3760 bpp, outperforms all other scans including the combined quadtree-raster method switching to raster at bit plane 3, which gives a bit rate of 2.4586 bpp. The AQS method produces the best bit rates for most of the bit planes, particularly for the middle range bit planes. On average it has an advantage of 0.2627 bpp (10%) over the next best single method and 0.0827 bpp (3%) over the combined quadtree-raster method.

| WBP | Raster | Quadtree | WT-ZT | WTHVZ | AQS |
|---|---|---|---|---|---|
| 13 | 0.0009 | **0.0008** | 0.0009 | 0.0009 | 0.0009 |
| 12 | 0.0007 | **0.0005** | 0.0006 | 0.0009 | 0.0007 |
| 11 | 0.0013 | **0.0010** | 0.0012 | 0.0012 | 0.0011 |
| 10 | 0.0033 | 0.0027 | 0.0025 | 0.0038 | **0.0024** |
| 9 | 0.0119 | 0.0091 | 0.0094 | 0.0130 | **0.0084** |
| 8 | 0.0421 | 0.0280 | 0.0319 | 0.0394 | **0.0263** |
| 7 | 0.1046 | 0.0711 | 0.0829 | 0.1025 | **0.0667** |
| 6 | 0.1791 | 0.1312 | 0.1494 | 0.1777 | **0.1208** |
| 5 | 0.2785 | 0.2204 | 0.2420 | 0.2729 | **0.1983** |
| 4 | 0.4030 | 0.3644 | 0.3786 | 0.3986 | **0.3247** |
| 3 | 0.5752 | 0.6178 | 0.6290 | 0.5807 | **0.5559** |
| 2 | **0.6222** | 0.7073 | 0.7309 | 0.6284 | 0.6373 |
| 1 | **0.3620** | 0.4051 | 0.3910 | 0.3676 | 0.3623 |
| 0 | **0.0700** | 0.0793 | **0.0700** | 0.0752 | **0.0700** |
| $13 \cdots 0$ | 2.6547 | 2.6387 | 2.7205 | 2.6628 | **2.3760** |

Table 4.2: Average zero-order entropy values (bpp) for all scans the image set

## 4.4 Coding The Signs

The sign of a coefficient needs to be coded only after the coefficient has become significant. The sign values, either positive or negative, can be represented in two symbols. The experiments on signs of the significant coefficients revealed that the two sign values are equiprobable. The zero centred symmetric Laplacian shape of the probability distribution of the wavelet coefficients proves the above observation. This leads to the conclusion that the sign information cost is equal to the total number of S symbols in the SSMs for all the bit planes.

Due to the above reasons, no special scanning schemes for sign are considered. Instead, in SSM, as and when an $S$ type bit is coded, its sign is coded subsequently.

## 4.5 Data Refinement

In embedded coding, switching of coefficients and refining of switched coefficients are done in two different passes for a given bit plane. In this research, the refining pass is carried on prior to the switching pass. This helps context modelling for the switching pass, which is discussed in section 5.1.1, as it provides more a priori known bits, in place of $X$ bits, in the neighbourhood of $S$ or $N$ bits.

The refining bits are either 1 or 0. Since these refining bits correspond to the bit planes that are lower than the msb for a given coefficient, they are highly decorrelated and possess a random nature. Due to these reasons, no special scanning order for data refinement for a given bit plane is considered. They are coded in the order of $X$ symbols occurring in the SSM, following a traditional raster scan.

## 4.6 Summary

The embedded quantiser presented in this chapter follows a simple bit plane based embedded coding system. The use of bit planes as the quantisation levels provides a quantisation scheme, whose quantisation bins are reduced by a factor of two at each bit plane traversal.

The normalisation of integer coefficients of the IWT, the IWHT and the INLT by the virtual bit plane sliding process scales the coefficients by their corresponding normalising constants while preserving the dynamic range. With this method, the unnecessary coding of zeros resulting from the traditional multiplication based normalising is avoided. Furthermore, coding the maximum coefficient height (msb) for each sub band as side information (depth limiting process), reduces coding of unnecessary zeros above the highest msb of a sub band. These two processes provide a compact effective bit space.

The embedded coding is costly compared to non-embedded coding mainly due to $N$ bits, which possess information regarding the position of $S$ bits, in the SSM. The methods to reduce the cost of embedding must concentrate on avoiding of coding such bits or clustering such bits into groups, so that each group can be represented by fewer bits. This was investigated by experimenting the efficient scanning schemes of the SSM. Out of the four scanning schemes considered, namely raster, quadtree, wavelet tree - zero tree and wavelet tree - HVZ, the quadtree based produced the lowest bit rates. However, when individual bit planes were considered, the quadtree scan was the best for the higher bit planes as was the raster scan for the lower bit planes.

The adaptive quadtree splitting (AQS) using two quadtree techniques QT1 and QT2 was designed to switch the raster and quadtree scanning adaptively according to the current block statistics. This was achieved by using the decision criteria based on the information predicted from the parent sub band and a priori known information from the current block. It is evident from the bit rate tables (Table 4.2 and the tables in Appendix C) that on average, the AQS has improved the results of the previous scans by 10%. Furthermore, the AQS has produced the lowest bit rates for most of the bit planes.

Finally, coding of signs and refining bits were briefly presented. No special scanning techniques for these bits, which were binary, were considered due to their high randomness in occurrence.

# Chapter 5

# Lossless Coding of Intra Frames

In the previous two chapters, the two main components of an embedded lossless coder, namely the integer transforms and embedded quantising were discussed in detail. In chapter 3, the integer forms of the wavelet, the Walsh Hadamard, the Discrete Cosine, the Discrete Sine and the non-linear transforms were introduced, designed and their entropy performances were evaluated. In chapter 4, the essence of embedded coding was introduced including the coefficient weighting and the steps, with all coding and modelling examples using the (4,4) integer wavelet. In this section the entropy coding of embeddedly quantised symbol streams for intra (I) frames of a video coder, which are also considered as still images, is discussed. The rest of the chapter is organised as below. In section 5.1, the Embedded to Lossless Image Coding (ELIC) algorithm which uses the AQS technique presented in 4.3.2, is discussed with the results using (4,4) and other integer transforms. Section 5.2 shows how to incorporate near-lossless features into ELIC, while section 5.3 presents the quasi lossless compression performance of ELIC using all integer transforms. Finally, section 5.4 discusses the compression performance of ELIC for intra frames.

## 5.1 The ELIC Algorithm

The Embedded to Lossless Image Coding (ELIC) algorithm uses the transforms that map integers to integers followed by embedded coding of the coefficients. The coding is based on bit significance criteria in which the coefficients that become significant in each bit plane are identified using a significant switch mask as shown in section 4.3. The scaled coefficients are coded from the most significant weighted bit plane to the lowest significant bit plane. Each weighted bit plane is visited from the lowest frequency ($LL$) sub band to the highest ($HH_0$) frequency for coding / decoding and each sub band in a given bit plane is coded in two passes for data refinement and significant switching, the latter being coded using the AQS scanning scheme. The symbols generated by the above process are entropy coded using a context based adaptive arithmetic coding as and when they are generated. The individual stages of the ELIC algorithm are shown in Figure 5.1.

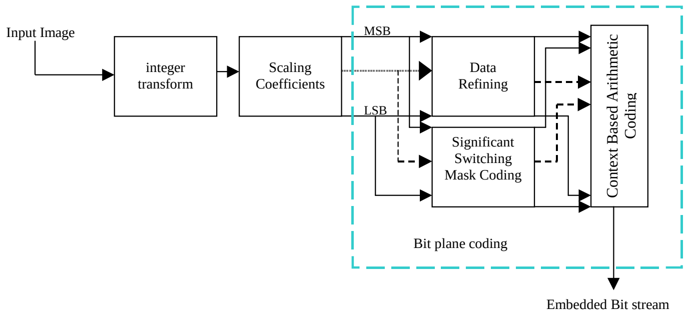


Figure 5.1: ELIC block diagram.

### 5.1.1 Context modelling for entropy coding

#### Context modelling : Basics

In context based entropy coding, the probability for each incoming symbol is calculated based on the probability distribution function of a coding context in which the symbol appears. From a universal source coding point of view, a source drawn from a smaller alphabet can be better modelled as a Markov process, and can thereby be coded efficiently compared to a source with a larger alphabet [105, 106]. This feature is satisfied with the embedded quantiser designed in the previous chapter, as all the symbols output at each stage are binary, which is the smallest alphabet possible. The main objective of context modelling is to remove the statistical redundancy in the output symbols. This is normally achieved by using a set of past observations ($X(C)$) on which the probability of the current symbol ($Y_i$) is conditioned to predict the conditional probability of the current symbol ($P(Y_i|X(C))$. The better the model fits the source, the smaller the bit rate an entropy coder can achieve.

Context modelling for entropy coding has successfully been used in both lossless image coding [32, 33, 34] and lossy image coding [106, 107, 108, 109]. In recent years, context based entropy coding of wavelet coefficients have become an important component of wavelet based image coding. In, one such example, ECECOW (Embedded Conditional Entropy Coding of Wavelet Coefficients) [107], an extensive use of coding contexts modelled using the already coded neighbouring bits in the same sub band and corresponding bits in the parent sub band has produced improved results among the wavelet based image coding. However, this has resulted in high modelling costs. In [108], the authors have improved the efficiency of context modelling by eliminating repetitive arithmetic, logic and memory operations, removing the parent sub bands in the context modelling and sharing contexts with signs and texture (switching and refining processes). The use of context merging to improve model costs have been demonstrated in [109].

### Context modelling in ELIC: For switching

The simplest coding context is to use the most previously coded symbol ($Y_{i-1}$) to condition the probability of the current symbol ($Y_i$) (Markov-I model). The statistical redundancy can be exploited more effectively by using contexts which involve the neighbours and corresponding parent pixels.

The dependencies of the current bit on the parent bits and other neighbouring bits in an embedded coding framework are briefly illustrated in this section using the (4,4) IWT coefficients of the test image set.

The neighbouring bits for a given bit can be either $S$, $N$ or $X$ type. For an $X$ type neighbouring bit, the actual refining bit is known, as a bit plane is refined before the switching pass. Since in AQS scan, a bit plane is traversed in Z scan form, one or more of its neighbours at positions Np, W, NW, SW and NE are already known at a given current bit position, $Y_i$, as in Figure 5.2. These bits and the corresponding parent (P) are used as the members of the context template.

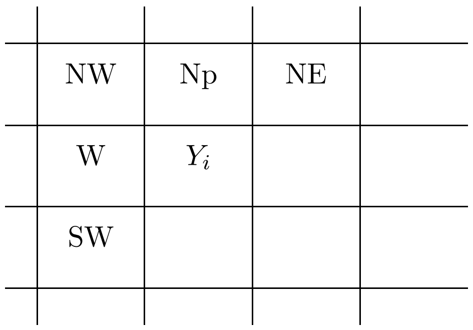


Figure 5.2: ELIC neighbourhood template for coding context.

The average probability values of current bit, $Y_i$, being $N$ type and being equal to the predictor bit,$C_i$, for all possible types (1, 0 and 2 for $S$, $N$ and $X$ respectively) of each predictor bit in the above template for the test image set are tabulated in Table 5.1.

| $C_i$ | P | Np | W | NW | NE | SW |
|---|---|---|---|---|---|---|
| $P(Y_i = 0 \mid C_i = 0)$ | 0.783 | 0.753 | 0.753 | 0.748 | 0.763 | 0.793 |
| $P(Y_i = 0 \mid C_i = 1)$ | 0.734 | 0.597 | 0.597 | 0.608 | 0.620 | 0.660 |
| $P(Y_i = 0 \mid C_i = 2)$ | 0.436 | 0.545 | 0.544 | 0.532 | 0.532 | 0.531 |
| $P(Y_i = C_i \mid C_i \neq 2)$ | 0.552 | 0.652 | 0.652 | 0.645 | 0.659 | 0.688 |

Table 5.1: The probability values of $Y_i$ being $N$ type for different types of $C_i$

From these observations in the table, the following conclusions can be arrived at. The insignificance of neighbours and parent has about over 74% influence on the insignificance of the current bit. When P=1, there still is a 74% probability of the current bit being insignificant. Similarly, when other neighbours are significant there is about a 60% possibility for $Y_i$ to be insignificant. When $C_i$ is $X$ type, then the significance and the insignificance occurrences of $Y_i$ are equi-probable. The bit plane wise observation suggests that these dependencies are stronger in the higher bit planes than in the lower bit planes.

Two types of predictions can be performed based on the above context mask. The first prediction method is to use the Maximum Likelihood (ML) criteria based on the context template. It was found by experiment that when the ML is $N$ type, $Y_i$ is also insignificant with average probabilities of more than 60% for the higher (un-normalised) bit planes and 41% for the lowest bit plane. Similarly, when the ML is $S$ or $X$ types, $Y_i$ is significant with average probabilities of more than 60% for the lowest and the highest bit planes and 35%-50% for the rest.

The second method is to use a gradient oriented significance prediction (gosp), adapted from GAP [32, 31], using only the already coded neighbouring bits in the context template. Provided all neighbouring bits are already coded gosp predicts the current bit as follows considering the values 0, 1 and 2 for $N$, $S$ and $X$ types respectively.

$$\begin{aligned} D_h &= |NW - Np| + |Np - NE| \\ D_v &= |NW - W| + |W - SW| \end{aligned}$$

$$\begin{aligned} if\ (D_h > D_v) &: \{\hat{X}_i = Np\} \\ if\ (D_h < D_v) &: \{\hat{X}_i = W\} \end{aligned}$$

```
$if\ (D_h = D_v)$ : {
        $if\ (NW \geq Max\,(Np, W)$ : $\{\hat{X}_i = Min\,(Np, W)\}$
        $if\ (NW \leq Min\,(Np, W)$ : $\{\hat{X}_i = Max\,(Np, W)\}$
        $else$ : $\{\hat{X}_i = NW\}$
        }
```

This type of prediction is used only if at least all of Np, W and NW neighbours are known to the coder and decoder. If either of SW or NE is not already coded before coding $Y_i$, then the respective $D_v$ or $D_h$ is weighted accordingly prior to the subsequent decision making. When both SW and NE are not already coded or not available at the image boundaries, the $(D_h = D_v)$ case is used.

The context model used for arithmetic coding of individual bits in min blocks in AQS coding for switching passes of ELIC, includes both the above methods, ML and gosp, and the strong influence of an insignificant parent bit. In context based arithmetic coding used in ELIC, the probability of the symbol to be coded is found from a collection of probability distributions identified by the context index computed as below.

The prediction method is grouped into three types, according to the availability of the neighbouring bits. The type 1, where Np, W, NW and either or both of NE and SW of the template are available, is predicted using gosp, whereas the type 2, where only Np, W and NW are present, is predicted as in the case for $(D_h = D_v)$. Type 3, where one or more of Np, W and NW is not available, is predicted using the ML criteria conditioned with insignificancy of the parent pixel P (i.e. when P is insignificant $Y_i$ is also predicted as insignificant irrespective of the ML prediction). A prediction value of 2, which corresponds to an $X$ type prediction, is considered as a prediction of significance.

The context index is determined by the prediction type (1, 2 and 3), the predicted value (0 and 1) and the sub band orientation (LH, HL and HH) when the prediction type is 1 or 2. The probability distributions corresponding to these context indexes are initialised at the beginning of each bit plane, so that each distribution can represent the statistics related to that bit plane.

Two separate global probability distributions are used for quadtree symbols (leaf and non-leaf) output in QT1 and QT2 respectively.

**Context modelling in ELIC: For signs**

In an embedded quantising framework, the sign of a coefficient is coded only after that coefficient has become significant. The authors of ECECOW algorithm have claimed that part of the superior compression performance of it is due to its sign context modelling, which includes three online estimated probabilities, conditioned on the sign of neighbours, depending on the sub band orientation [107]. Later, in an extension of ECECOW, they have shown that the same performance can be gained by combining the sign and texture (switching and refining) contexts together [108].

In most cases, it has generally been assumed that no advantage is to be gained by entropy coding of signs. Recently, it was shown that the wavelet coefficient signs, resulting from 9/7 wavelet, are strongly negatively correlated across edges [110]. However, due to bit plane by bit plane switching of the coefficients in embedded coding, the signs of only some of the neighbours in a causal / non causal template are already known for all the significant bits on a bit plane. This handicaps the chances of capturing any sign correlation mentioned above. Therefore, in ELIC, the sign bits for significant coefficients are coded using a Markov-I context modelling, which considered the previously coded sign to condition the probability of the current sign.

**Context modelling in ELIC: For refining**

In a data refinement pass in the embedded coding, the neighbouring bits can be of three types: Already refined in the current bit plane, not yet refined in the current bit plane and yet to become significant. Out of these three types only the first type carry information relevant to bit plane statistics. Therefore, the neighbouring bits in a bit plane are not considered in designing the context for data refining in ELIC.

Instead, data refinement bit contexts are defined using the already coded bits from the current coefficient with reference to the original bit positioning in the wavelet domain before normalising. The context $C_i$ for the $\text{bit}_i$ of a coefficient is selected using the two higher bits, $\text{bit}_{i-1}$ and $\text{bit}_{i-2}$ as in Table 5.2.

| $C_i$ | $\text{bit}_{i-1}$ | $\text{bit}_{i-2}$ |
|---|---|---|
| 1 | 0 | 0 |
| 2 | 0 | 1 |
| 3 | 1 | 1 or 0 |

Table 5.2: Context Selection for data refinement bits

The actual coding context is selected by referring to a look up table with two input parameters, namely initial context, $C_i$, and the original bit plane number for the $\text{bit}_i$

prior to normalising. This makes 3 $\times$(msb index) data contexts for each WBP. The memory used for these distributions in one WBP are reused for another WBP by resetting the distributions at the beginning of each WBP. This also helps the probability distributions to represent the statistics related to a specific bit plane.

### Context modelling in ELIC: Performance

The performance of the above presented context modelling for ELIC is demonstrated as in Table 5.3, by comparing its lossless bit rates with those of the no context modelling case and a Markov-1 context case for the lossless coding case. the bit rates shown in the table are the average lossless bit rates for the test image set.

| Model | Lossless Bit rate |
|---|---|
| No Contexts | 4.549 |
| Markov-I | 4.517 |
| ELIC | 4.341 |

Table 5.3: Context modelling comparisons

According to the above table, entropy coding without any contexts compresses the test image set to 4.549 bpp on average and use of contexts in entropy coding reduces the bit rates to 4.517 bpp and 4.341 bpp for Markov-I context model and the contexts model used in ELIC respectively. The context model in ELIC gains an advantage of 0.21 bpp (4.6%) and 0.18 bpp (3.9%) over the first two methods respectively.

## 5.1.2 ELIC results with the IWT

Bit rates (bpp) at lossless level for ELIC, using the (4,4) IWT with 5 decomposition scales are compared with JPEG-LS, which is a prediction based method, and SPIHT lossless, which is based on S+P with embedded coding up to a certain bit rate and then progressive coding (ProgCode). The results for the test image set are as in Table 5.4.

JPEG-LS gives the best compression for most of the images. But the disadvantage of JPEG-LS is the coded image can only be decoded to the lossless level since it is not an embedded coding technique. On the other hand both SPIHT and ELIC can be decoded to other compression levels. On average at the lossless level, ELIC performs within 0.1% of JPEG-LS and 0.7% better than SPIHT lossless.

|  | JPEG-LS | SPIHT | ELIC |
|---|---|---|---|
| Gold Hill | 4.557 | 4.630 | 4.602 |
| Barbara1 | 4.720 | 4.580 | 4.524 |
| Barbara2 | 4.777 | 4.792 | 4.792 |
| Boats | 3.946 | 4.053 | 4.026 |
| Black board | 3.684 | 3.809 | 3.765 |
| Average | 4.337 | 4.373 | 4.341 |

Table 5.4: Lossless performance (in bpp)

### 5.1.3 ELIC results with the other transforms

The lossless performance of other integer transforms, the IWHT, the IDCT, the IDST and the INLT, designed in Chapter 3 are also compared using ELIC. According to tables 3.2, 3.4 and 3.5, the block transforms using a block size of 32×32 result in the lowest weighted entropy values, which are computed using the packet sub bands arrangement of the block transforms.

However, ELIC was designed in the previous and the current chapters considering the usual wavelet sub band decomposition. Earlier it was shown that the block transforms coefficients can be reorganised into a packet transform sub band structure. In the literature, zero tree coding has been used in wavelet packet coding by rearranging them back into traditional wavelet sub band structure so that the correct positions of the parents and their corresponding children coefficients can be considered in wavelet tree based coding [111]. A similar approach has been followed in ELIC when used with the block transforms.

The performance of IWHT, IDCT and IDST with ELIC using different block sizes are as in tables 5.5, 5.6 and 5.7 respectively.

| N | 4 | 8 | 16 | 32 |
|---|---|---|---|---|
| Gold Hill | 4.992 | **4.973** | 5.028 | 5.130 |
| Barbara1 | 5.196 | **5.195** | 5.335 | 5.519 |
| Barbara2 | **5.286** | 5.294 | 5.402 | 5.5640 |
| Boats | **4.533** | 4.587 | 4.737 | 4.9180 |
| Black board | **4.127** | 4.168 | 4.335 | 4.519 |
| Average | **4.827** | 4.843 | 4.967 | 5.130 |

Table 5.5: Lossless performance (in bpp) for $\mathbf{IWHT_N}$ using ELIC

| N | 4 | 8 | 16 | 32 |
|---|---|---|---|---|
| Gold Hill | 4.897 | 4.778 | **4.745** | 4.752 |
| Barbara1 | 5.002 | 4.760 | 4.648 | **4.640** |
| Barbara2 | 5.139 | 4.981 | **4.915** | 4.925 |
| Boats | 4.386 | 4.259 | **4.249** | 4.289 |
| Black board | 4.040 | **3.943** | 3.961 | 4.024 |
| Average | 4.693 | 4.544 | **4.504** | 4.526 |

Table 5.6: Lossless performance (in bpp) for $\mathbf{IDCT_N}$ using ELIC

| N | 4 | 8 | 16 | 32 |
|---|---|---|---|---|
| Gold Hill | 5.408 | 5.474 | 5.426 | **5.312** |
| Barbara1 | 5.649 | 5.747 | 5.707 | **5.551** |
| Barbara2 | 5.744 | 5.791 | 5.733 | **5.626** |
| Boats | **5.052** | 5.198 | 5.201 | 5.118 |
| Black board | **4.688** | 4.872 | 4.984 | 4.990 |
| Average | **5.308** | 5.416 | 5.410 | 5.319 |

Table 5.7: Lossless performance (in bpp) for $\mathbf{IDST_N}$ using ELIC

From the above tables it is evident, that the transforms with smaller block sizes produce lower lossless bit rates. On average, the 16-point IDCT, the 4-point IWHT and the 4-point IDST, performed better than any other block sizes for respective transforms. This discrepancy in the entropy values and the actual bit rates is due to the wavelet packet sub band nature of those transforms. In these transforms, the packet sub band organisation has to be rearranged in order to use the corresponding parent-child orientation as in a dyadic wavelet tree, as used in ELIC.

Since the INLT-3 follows the usual wavelet sub band decomposition, a 5 scale INLT-3 is used to demonstrate the INLT with ELIC for lossless image coding. Bit rates at the lossless level for all above integer transforms including (4,4) IWT are summarised in Table 5.8 for comparison.

As can be seen from the above table, ELIC using the IWT produces the best lossless performance on average. The 16-point IDCT produces the next best results. The INLT-3 and the 4-point IWHT produce comparable results. As expected, the IDST provides the worst lossless performance for intra frame type images.

|  | IWT-(4,4) | IWHT$_4$ | IDCT$_{16}$ | IDST$_4$ | INLT-3 |
|---|---|---|---|---|---|
| Gold Hill | **4.602** | 4.992 | 4.745 | 5.408 | 4.895 |
| Barbara1 | **4.524** | 5.196 | 4.648 | 5.649 | 5.211 |
| Barbara2 | **4.792** | 5.286 | 4.915 | 5.744 | 5.298 |
| Boats | **4.026** | 4.533 | 4.249 | 5.052 | 4.471 |
| Black board | **3.765** | 4.127 | 3.961 | 4.688 | 4.124 |
| Average | **4.341** | 4.827 | 4.504 | 5.308 | 4.800 |

Table 5.8: Lossless performance (in bpp) for ELIC using integer transforms

## 5.2 Near-Lossless Compression

As stated in Definition 1.3 (page 13), in near-lossless coding, each reconstructed pixel of the image / video sequence output from the decoder differs from the corresponding one in the input to the encoder by not more than a pre specified value $\delta$. Near-lossless coding can easily be incorporated into predictive lossless coding techniques, like JPEG-LS, as it is normally achieved by quantising the prediction error according to $\delta$. However, this is not possible in transforms based methods, since coding is performed in the transform coefficients domain. Therefore, maintaining a maximum error value of $\pm\delta$ in the transform domain does not guarantee the same error level in the image pixel domain.

Usually, near-lossless coding using the IWT is performed by a pre-quantisation process, in which the input to the wavelet transforms is quantised using the near-lossless quantisation [70], as in equation 2.3 (on page 24). In the following sub sections, performance of this technique for near-lossless image coding is compared with two novel techniques where the quantisation process is incorporated into lifting steps.

### 5.2.1 Pre-quantisation

The input to the integer transform is quantised using the maximum allowed error, $\delta$, as in **Quant**( ), Q$_\delta$( ), process and the output from the inverse transform is dequantised to achieve the decoded value as in **Dequant**( ), D$_\delta$( ), process as shown in below equations.

$$\textbf{Quant}(x) \quad : \qquad Q_\delta(x) = sign(x) \times \left\lfloor \frac{|x| + \delta}{2\delta + 1} \right\rfloor \tag{5.1}$$

$$\textbf{Dequant}(x) \quad : \qquad D_\delta(x) = x \times (2\delta + 1) \tag{5.2}$$

The main advantage of this method is that it can be used with any integer transform. The disadvantages are that the pre-quantisation reduces the dynamic range of the input to the transform and that the coding is not optimised according to $\delta$ as in predictive near-lossless techniques. The reduction of dynamic range through pre quantisation represents an image with statistical characteristics different from those of the image prior to quantisation.

### 5.2.2 Incorporating near-lossless quantisation into lifting steps

As mentioned earlier in section 3.1, the input signal is split into two channels using the lazy wavelet transform and then P and U lifting steps are performed on those two channels to obtain the wavelet coefficients. In this section, the possibility of incorporating above near-lossless quantisation functions, **Quant**( ) and **Dequant**( ), into lifting steps is considered. In addition to these two processes, a new process, **Requant**( ), R$_\delta$( ), is introduced to re-quantise a quantised and dequantised input signal value. The **Requant**( ) process, which is the inverse of **Dequant**( ) process, is performed as in equation 5.3.

$$\textbf{Requant}(x) \quad : \qquad R_\delta(x) = \left\lfloor \frac{x}{2\delta+1} \right\rfloor \tag{5.3}$$

Two techniques based on 1-D transform and 2-D transform are designed and evaluated as follows.

#### 5.2.2.1 1-D online (in-transform) near-lossless quantised lifting

In this method, near-lossless quantisation is incorporated considering the 1-D lifting transform. Let the output from the lazy wavelet be $s$ and $d$. The Q$_\delta$, D$_\delta$ and R$_\delta$ processes are used on $s$ and $d$ channels in line with P and U lifting steps. The online (in-transform) near-lossless lifting steps are as follows,

$$s \leftarrow Q_\delta(s) \tag{5.4}$$
$$s \leftarrow D_\delta(s) \tag{5.5}$$
$$d \leftarrow d - \left\lfloor P(s) + \frac{1}{2} \right\rfloor \tag{5.6}$$
$$d \leftarrow Q_\delta(d) \tag{5.7}$$
$$s \leftarrow R_\delta(s) \tag{5.8}$$
$$s \leftarrow s + \left\lfloor U(s) + \frac{1}{2} \right\rfloor \tag{5.9}$$

Steps 5.4-5.5 represent incorporation of the effect of near-lossless quantising of $s$ channel prior to using it to predict $d$ channel. Step 5.7 represents near-lossless quantising of the prediction error as performed in predictive coding techniques. In step 5.8, $s$ channel coefficients are quantised back to their near-lossless quantised dynamic range. The inverse transform, as shown in steps 5.10-5.13, reverses the order and the operation of corresponding steps in the forward transform (steps 5.4-5.9).

$$s \leftarrow s - \left\lfloor U(s) + \frac{1}{2} \right\rfloor \tag{5.10}$$

$$s \leftarrow D_\delta(s) \tag{5.11}$$

$$d \leftarrow D_\delta(d) \tag{5.12}$$

$$d \leftarrow d + \left\lfloor P(s) + \frac{1}{2} \right\rfloor \tag{5.13}$$

In general, the stand alone $Q_\delta$ and $R_\delta$ processes in the forward transform are replaced with the $D_\delta$ process in the inverse transform. The sequential $Q_\delta$ and $D_\delta$ processes, as in steps 5.4 and 5.5, in the forward transform do not need to be inversed, as their operations are off set against each other in the forward transform.

In a 1-D transform the above steps are performed only for the first decomposition level. The higher decompositions are performed as in a normal wavelet transform, using steps 5.6 and 5.9. In a 2-D transform, where the 1-D transform is applied separately for rows and columns, the above steps are performed only in the 1-D transform performed in the first dimension i.e. usually for rows.

#### 5.2.2.2 2-D online (in-transform) near-lossless quantised lifting

This method incorporates online near-lossless quantised lifting into a 2-D wavelet transform. As evident from the above section, due to the inability of applying the 1-D technique separably into the 2-D case, a non separable method using quincunx sub band splitting is introduced. In the following design the usual four sub bands namely, 00, 01, 10 and 11, obtained from quincunx splitting (section 3.5), are named as LL, HL, LH and HH respectively, in accordance with the traditional wavelet transform sub band notation.

| | |
|---|---|
| $LL$ | $HL$ |
| $LH$ | $HH$ |

The online near-lossless lifting steps for the forward transform are as follows,

$$
\begin{aligned}
LL &\leftarrow Q_\delta(LL) && (5.14)\\
LL &\leftarrow D_\delta(LL) && (5.15)\\
LH &\leftarrow LH - \left\lfloor P(LL) + \frac{1}{2} \right\rfloor && (5.16)\\
LH &\leftarrow Q_\delta(LH) && (5.17)\\
LL &\leftarrow LL + \left\lfloor U(LH) + \frac{1}{2} \right\rfloor && (5.18)
\end{aligned}
$$

$$
\begin{aligned}
HL &\leftarrow HL - \left\lfloor P(LL) + \frac{1}{2} \right\rfloor && (5.19)\\
HL &\leftarrow Q_\delta(HL) && (5.20)
\end{aligned}
$$

$$
\begin{aligned}
LL &\leftarrow LL - \left\lfloor U(LH) + \frac{1}{2} \right\rfloor && (5.21)\\
LH &\leftarrow D_\delta(LH) && (5.22)\\
LH &\leftarrow LH + \left\lfloor P(LL) + \frac{1}{2} \right\rfloor && (5.23)
\end{aligned}
$$

$$
\begin{aligned}
HH &\leftarrow HH - \left\lfloor P(LH) + \frac{1}{2} \right\rfloor && (5.24)\\
HH &\leftarrow Q_\delta(HH) && (5.25)
\end{aligned}
$$

$$
\begin{aligned}
HH &\leftarrow HH - \left\lfloor P(HL) + \frac{1}{2} \right\rfloor && (5.26)\\
HL &\leftarrow HL + \left\lfloor U(HH) + \frac{1}{2} \right\rfloor && (5.27)
\end{aligned}
$$

$$
\begin{aligned}
LH &\leftarrow LH - \left\lfloor P(LL) + \frac{1}{2} \right\rfloor && (5.28)\\
LH &\leftarrow R_\delta(LH) && (5.29)\\
LL &\leftarrow R_\delta(LL) && (5.30)\\
LL &\leftarrow LL + \left\lfloor U(LH) + \frac{1}{2} \right\rfloor && (5.31)
\end{aligned}
$$

$$
\begin{aligned}
LL &\leftarrow LL + \left\lfloor U(LH) + \frac{1}{2} \right\rfloor && (5.32)\\
LH &\leftarrow LH + \left\lfloor U(HH) + \frac{1}{2} \right\rfloor && (5.33)
\end{aligned}
$$

The inverse transform is obtained by reversing the operating order and the sign of the forward transform steps. As in the 1-D case, this is done by replacing stand alone

$Q_\delta$ and $R_\delta$ processes in the forward transform with the $D_\delta$ process and ignoring the sequential $Q_\delta$ and $D_\delta$ processes in the forward transform.

### 5.2.3 Near-lossless results

The performance of the above three methods using ELIC quantiser is compared with the near-lossless performance of the JPEG-LS (JPEG-NLS) for different $\delta$ values, $\delta$=1, 3 and 5. The near-lossless bit rates for those $\delta$ values for the test image set are shown in Tables 5.9 -5.11 and the average bit rates for different $\delta$ values are summarised as in Table 5.12.

| | JPEG-NLS | Pre-quant | Online 1-D | Online 2-D |
|---|---|---|---|---|
| Gold Hill | 3.039 | 3.136 | 3.111 | 3.099 |
| Barbara1 | 3.174 | 3.129 | 3.095 | 3.068 |
| Barbara2 | 3.222 | 3.361 | 3.337 | 3.316 |
| Boats | 2.487 | 2.712 | 2.664 | 2.643 |
| Black board | 2.209 | 2.488 | 2.430 | 2.406 |
| Avergage | 2.826 | 2.965 | 2.927 | 2.906 |

Table 5.9: Near-lossless performance (in bpp) for $\delta$=1.

| | JPEG-NLS | Pre-quant | Online 1-D | Online 2-D |
|---|---|---|---|---|
| Gold Hill | 1.970 | 2.200 | 2.140 | 2.106 |
| Barbara1 | 2.178 | 2.276 | 2.205 | 2.139 |
| Barbara2 | 2.172 | 2.420 | 2.377 | 2.334 |
| Boats | 1.514 | 1.882 | 1.823 | 1.760 |
| Black board | 1.276 | 1.664 | 1.603 | 1.560 |
| Average | 1.822 | 2.088 | 2.030 | 1.980 |

Table 5.10: Near-lossless performance (in bpp) for $\delta$=3.

| | JPEG-NLS | Pre-quant | Online 1-D | Online 2-D |
|---|---|---|---|---|
| Gold Hill | 1.525 | 1.794 | 1.713 | 1.655 |
| Barbara1 | 1.717 | 1.884 | 1.798 | 1.702 |
| Barbara2 | 1.674 | 1.959 | 1.908 | 1.845 |
| Boats | 1.130 | 1.501 | 1.435 | 1.357 |
| Black board | 0.861 | 1.328 | 1.271 | 1.189 |
| Average | 1.381 | 1.693 | 1.625 | 1.550 |

Table 5.11: Near-lossless performance (in bpp) for $\delta$=5.

| $\delta$ | JPEG-NLS | Pre-quant | Online 1-D | Online 2-D |
|---|---|---|---|---|
| 1 | 2.826 | 2.965 | 2.927 | 2.906 |
| 3 | 1.822 | 2.088 | 2.030 | 1.980 |
| 5 | 1.381 | 1.693 | 1.625 | 1.550 |
| 7 | 1.147 | 1.428 | 1.343 | 1.259 |

Table 5.12: Summarised Near-lossless performance (in bpp) for the image set.

The summarised near lossless bit rates in Table 5.12 show that JPEG-NLS, which is a predictive technique, performs better than any other method. However, the newly designed 1-D and 2-D online quantisation methods outperform the pre-quantisation method, which has been commonly used in integer transforms based near-lossless coding. It is also evident from the table that the 2-D online method produces better results than those of the 1-D online quantiser. The superiority of in-transform quantising techniques over pre-quantising is evident not only in the bit rates, but also in corresponding rms error values of the decoded image. The rms error - bit rate plots for Gold Hill, Barbara1 and for the whole image set is as in figures 5.3-5.5. The 2-D online (in-transform) quantisation method achieves better rms error / bit rate performance compared to the pre-quantisation method.

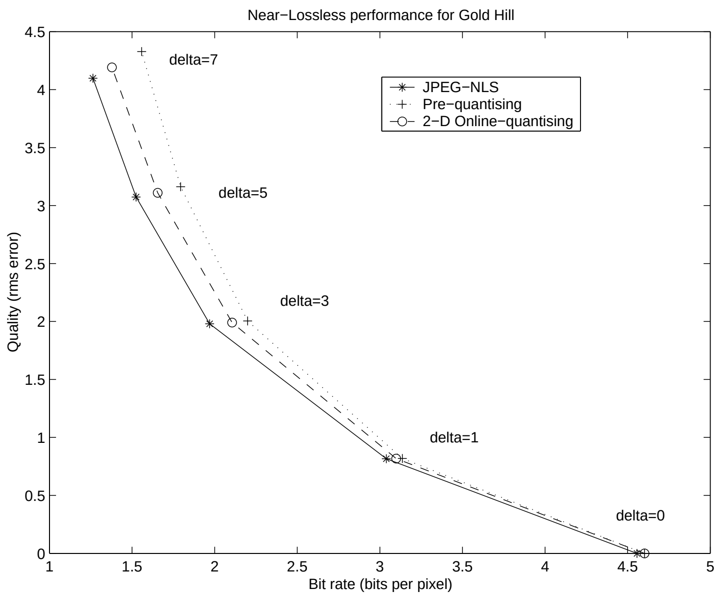


Figure 5.3: Near-lossless performance for Gold Hill.

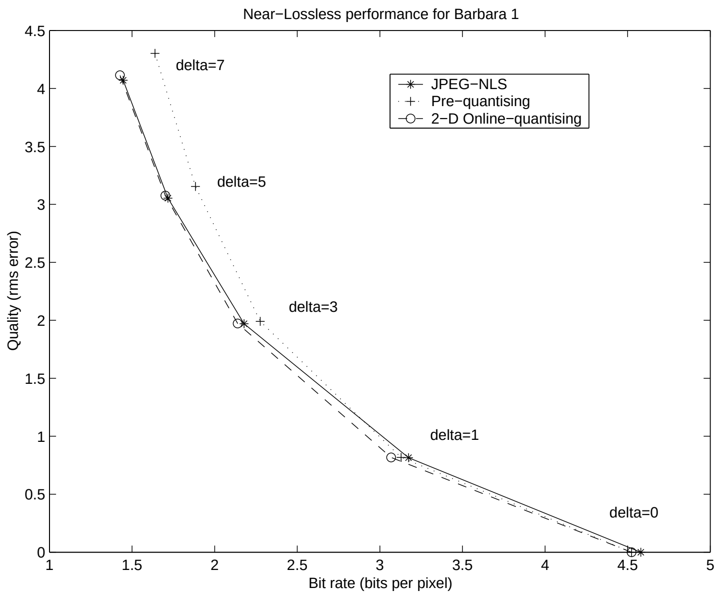


Figure 5.4: Near-lossless performance for Barbara1.

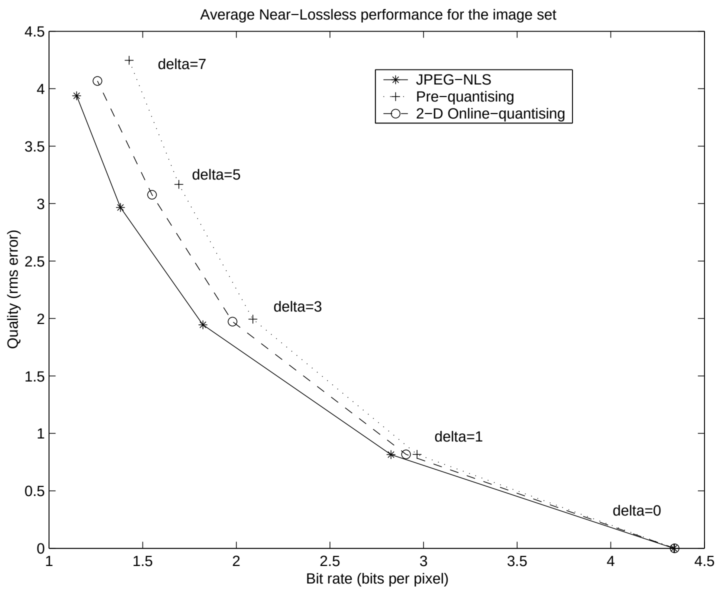


Figure 5.5: Average near-lossless performance for the image set.

## 5.3 Quasi Lossless Compression Performance

The ELIC algorithm can code / decode to other bit rates lower than the lossless bit rate. The quasi lossless performance of ELIC using the (4,4) IWT and the IDCT is compared with those of SPIHT algorithm for bit rates up to 1 bpp starting from the lossless bit rates. The rms error vs. bit rate plots for Gold Hill and Barbara 1 are in Figure 5.6 and 5.7 respectively.

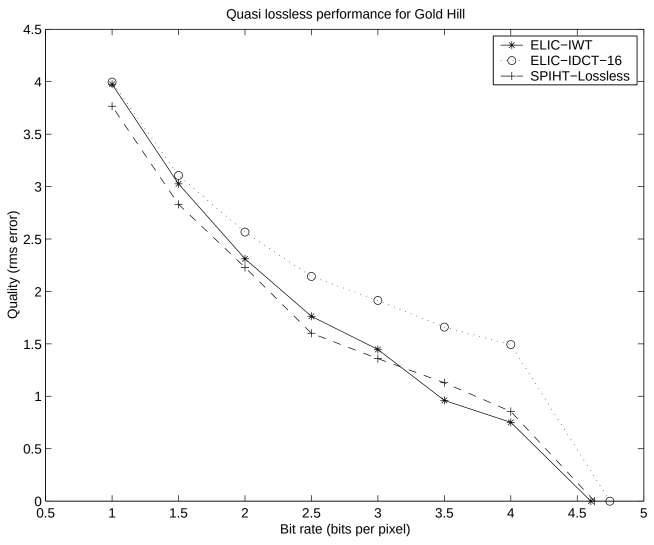


Figure 5.6: Quasi lossless performance for Gold Hill.

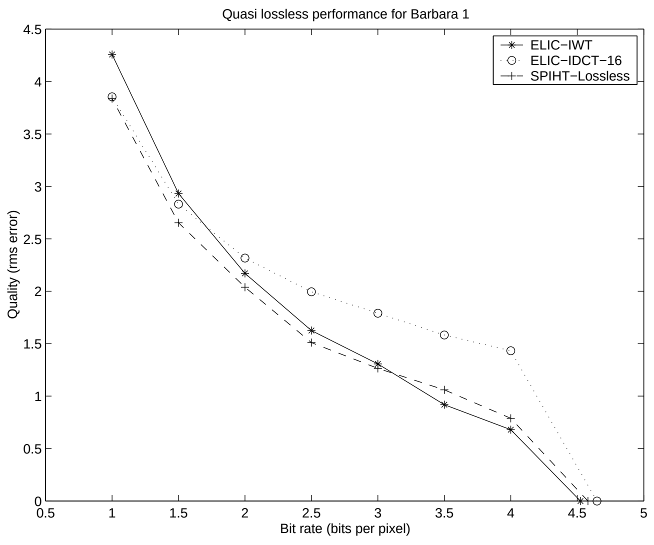


Figure 5.7: Quasi lossless performance for Barbara1.

As seen from the above plots, ELIC algorithm with (4,4) IWT produces the best quasi lossless results at bit rates higher than 3 bpp. Furthermore, it is evident that although the 16-point IDCT based ELIC performs less well at lossless and high bit rates, the performance of it becomes comparable with ELIC-IWT at the lower bit rates.

## 5.4 Discussion

In this section, the use of the integer transforms and embedded quantising on embedded lossless image coding was investigated. First, it was shown that further reduction of bit rates can be gained by using context based entropy coding as used in ELIC. The extensively designed context model for ELIC reduces its lossless bit rates by 4.6% on average for the test image set. Lossless bit rates for ELIC-IWT outperform those of SPIHT by 0.7% on average. Moreover, on average, ELIC-IWT performs within 0.1% of JPEG-LS, which is a predictive lossless coding method where the lossless bit stream can only be decoded at the lossless bit rate.

The experiments on performance of other transforms using ELIC show a discrepancy on the optimum block sizes for the block based orthogonal transforms, with those found in the initial entropy computations. This is mainly due to their wavelet packet type sub band arrangement being reorganised into such a way that the correct parent-child relationship can be used in ELIC. A block size of 16 for the IDCT and a block size of 4 for the IWHT and the IDST produce the lowest lossless bit rates for those transforms. However, when all the transforms are compared, the best results were achieved by the IWT-(4,4), followed by the $\text{IDCT}_{16}$, INLT-3, the $\text{IWHT}_4$ and the $\text{IDST}_4$.

The near-lossless coding, in which each reconstructed pixel in the output from decoder differs from the input to the coder by not more than a value $\delta$ specified at the time of coding, is more commonly used with the prediction based lossless coding methods. Normally near-lossless coding in integer transforms based lossless coding is achieved by quantising the input using $\delta$ prior to the forward transform. The near-lossless results, both bit rates and rms error, achieved through pre-quantisation are always inferior to those obtained from predictive techniques. It was shown in this chapter, that such performance can be improved by incorporating near-lossless quantisation into lifting steps in the transform either by considering 1-D or 2-D transforms. The latter has shown lower bit rates and rms error values at the tested near-lossless levels for the image set.

Finally, the quasi lossless performance of ELIC using the (4,4) IWT and the 16-point IDCT, was compared with those achieved from the lossless mode of SPIHT. ELIC-IWT provides the best performance at lossless level and the bit rates higher than 3 bpp for most of the images in the image set. ELIC with the 16-point IDCT produces results comparable with ELIC-IWT at lower bit rates.

# Chapter 6

# Lossless Coding of Non-intra Frames

## 6.1 Introduction

In previous chapters the main components of lossless embedded coding were discussed in detail. The performance of the integer transforms, presented in Chapter 3, for embedded lossless coding of intra frames using the ELIC quantiser was evaluated for both lossless and quasi-lossless bit rates in Chapter 5. In this chapter, the performance of those integer transforms for embedded lossless coding of non-intra frames in a lossless video coder, using the MPEG-2 video motion compensated prediction framework is presented. The non-intra frames of such codec contain the prediction residuals, which need to be coded to correct the predicted frames. The rest of this chapter is organised as follows. Section 6.2 looks at the characteristics of the motion compensated prediction residuals, the knowledge of which is vital for transform and quantiser selection. The use of wavelet transforms on non-intra frames is discussed in section 6.3. Likewise, the performance of the other integer transforms on non-intra frames are investigated in section 6.5. Section 6.6 presents the entropy coding of integer transform coefficients using ELIC and finally section 6.7 discusses the findings of these experiments.

## 6.2 Characteristics of Non-intra Frames

The statistical characteristics of intra and non-intra frames are significantly different. Therefore, careful investigation of the statistical properties is vital for efficient encoding. The main properties, including the magnitude histograms, the normalised auto correlation coefficients, the magnitude spectrums and the energy in the DCT coefficients are compared for intra and non-intra frame. These comparisons are illustrated below using a non-intra frame and its corresponding intra frame from Mobile sequence.

### Magnitude histograms

The magnitude histogram plots of a non-intra frame and its corresponding intra frame are as in Figure 6.1. It can be seen from the figure, that the magnitude histogram of the intra frame is spread over a range of 0-255, with short peaks according to the local luminance values of the image. Likewise, the magnitude histogram of the residuals of the non-intra frame constitute a zero centred double sided geometrical distribution with a high narrow peak at zero and mean value close to zero. It spans a range of -255 to 255. The width of the peak of the geometrical distribution is dependent on the motion content of the frame. Frames with low motion content give a high and narrow peak at zero and short tails, while the frames with high motion content produce longer tails. This shape of distribution suggests that the values in the residuals are already decorrelated to a certain extent depending on the level of motion present in the frame.

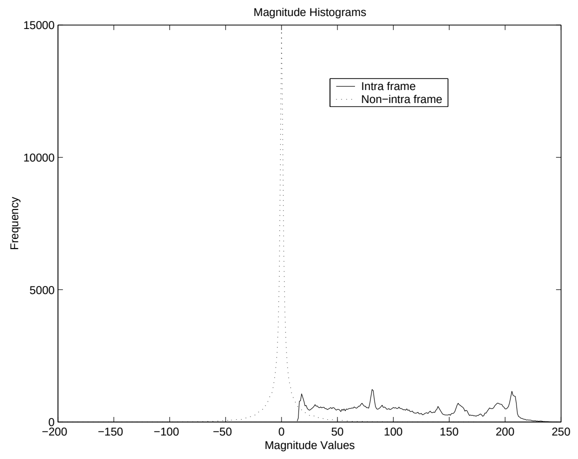
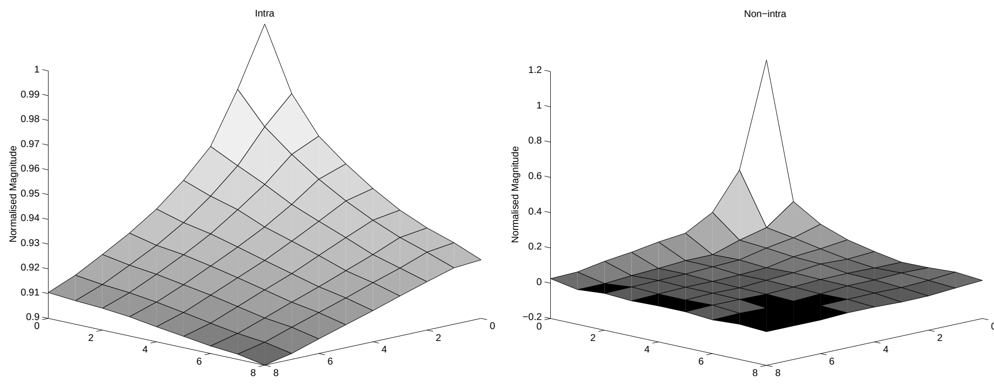


Figure 6.1: Magnitude histograms of Intra and Non-intra type frames.

### Auto correlation coefficients

The normalised auto correlation plots for a pixel neighbourhood of -4· · ·4 in both x and y directions for intra and non-intra frame are as in Figure 6.2. Intra frame pixels contain a high positive inter pixel correlation, whereas, non-intra frame pixels contain a low inter pixel correlation, where most of the values are close to zero. This also suggests that the residuals from motion compensated prediction in a lossless video are already highly decorrelated by the prediction processes.



Intra Non-Intra

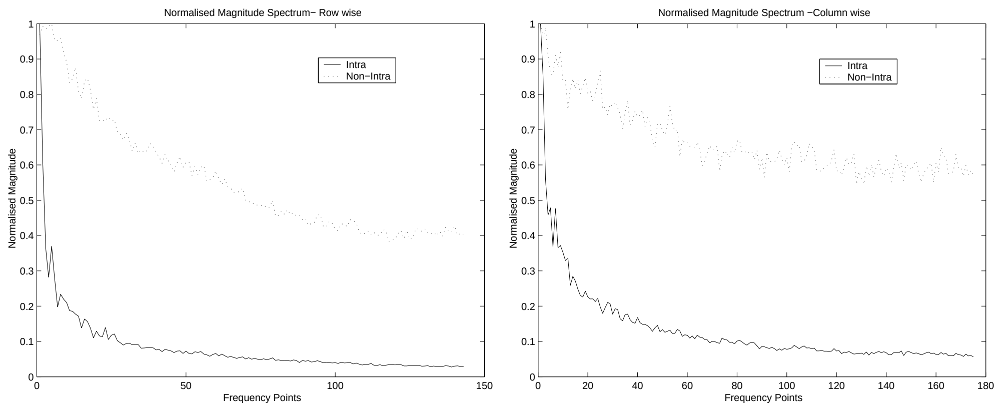
Figure 6.2: Auto correlation coefficients of Intra and Non-intra type frames.

### Magnitude Spectrum

The average normalised magnitudes of the coefficients from the FFT performed both row and column wise for both types of frames are shown in Figure 6.3. As can be seen from the figure, intra frames possess an exponentially decreasing power spectrum in both directions. The normalised magnitudes of the high frequency components in intra frames are smaller compared to those of non-intra frames. The high frequency components with comparatively large magnitudes present in non-intra frame residuals can be costly in compression of such frames.

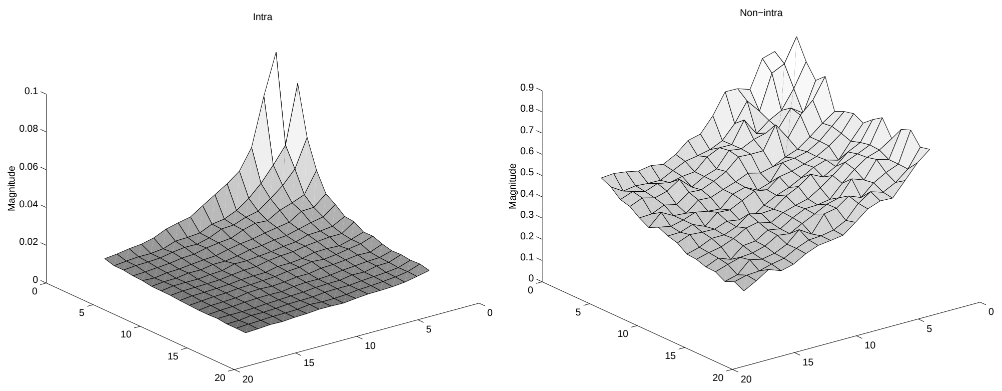
### The DCT coefficient magnitudes

The DCT is used as the transform for both intra and non-intra type frames in MPEG-2 and most other video coding standards. The magnitudes of the a.c. components of the 2-D 16-point DCT coefficients normalised with respect to the d.c. component for both types of frames are as in Figure 6.4. The 16-point DCT was used in this analysis as it is the same as the size of a macro block used in motion compensation. The 2-D DCT



For Rows For Columns

Figure 6.3: Magnitude spectrum of Intra and Non-intra type frames.



Intra Non-Intra

Figure 6.4: Normalised magnitudes of a.c. components of the DCT coefficients for Intra and Non-intra type frames.

coefficients are usually arranged in increasing frequencies along the diagonals, starting from the top left corner, which corresponds to the d.c. value. As can be seen from the figure, the presence of comparatively large amplitude high frequency components in non-intra frame residuals is further made evident in the DCT coefficients.

#### Residuals in lossy coding Vs Residuals in lossless coding

Since the reference frames used for forward and backward motion compensated prediction in a lossless video coding framework are coded losslessly, the residuals in lossless video coding do not contain quantisation errors resulting from lossy coding of reference frames as in lossy video coding. These residuals mainly consist of the noise due to the motion compensated prediction process and the natural noise in individual frames, whose level is increased by the subtraction used to find the residuals.

By the above statistical properties, it can be concluded that the residuals in lossless video coding are fairly decorrelated frames with large magnitude high frequency components. This amount of decorrelation may be adequate for efficient lossless coding without using a transform. However, in addition to decorrelation of data, a transform is required for efficient compaction of input energy, which is vital for embedded coding. Therefore, coding of residuals has to use a transform, which should be selected by considering the statistical characteristics of the residuals.

## 6.3 Integer Wavelet Transforms on Non-intra Frames

Although the DCT has been the preferred transform method for residuals in current video coding standards including MPEG-2, with the success of using wavelet transforms in still image coding, the wavelet transforms also have been used for coding non-intra frames [112]. In this section, the use of integer wavelet transforms is considered for lossless coding of such frames.

In lossless image coding, it is well known that wavelet transforms with more vanishing moments, such as (4,4), (4,2) and (2+2,2) outperform wavelet transforms with fewer vanishing moments, since still images mainly consist of highly smooth regions and a few edges. Such wavelets use a larger neighbourhood of pixels for predicting and updating lifting steps. This was again evident from the entropy values for still images listed in Table 3.1. Since the residuals contain a higher proportion of high frequency information, wavelets with fewer vanishing moments, such as (0,0) (lazy wavelet), (1,1) (S transform) and (2,2) transform were chosen for initial experiments on residuals. Residuals from Y components of the test sequence set were used in the following experiments.

### 6.3.1 Sub band entropy and energy distributions

Wavelet transforms are applied separately in horizontal and vertical directions producing four sub bands. When still images or intra frames are wavelet transformed, most of the image energy is concentrated in the LL sub band, which normally contains a sub sampled original image with statistics similar to those of the original image. Further, the LL sub band contains higher entropy compared to that of the other three sub bands. These characteristics lead to iteration of the wavelet transforms on LL sub band for four or five levels of decomposition in still image coding.

On the other hand, the LL sub band in the wavelet transformed residuals does not possess the highest entropy or the highest energy compared to the other sub bands. This is illustrated for 40 non-intra frames obtained from the Mobile sequence using the (2,2) transform as in Figure 6.5-6.6. The average sub band entropy and energy values for the Mobile sequence using wavelets (1,1), (2,2) and (4,4) are as in Figure 6.7.

It can be seen from Figure 6.5 and 6.6, that there is no single sub band consistently containing the highest entropy or energy percentages for all the frames. LL sub band has shown the highest energy and entropy only for a few frames, mostly for P types. The entropy and energy average % values using different wavelets, as shown in Figure 6.7, also reveal that all four sub bands contain comparable energy and entropy values. This is due to the larger proportion of high frequency information present in the residuals. Because of this, further decomposition of LL sub band as in still image coding does not improve the total weighted entropy. Therefore, in these experiments, the wavelet transforms were applied only up to one scale, resulting in only four sub bands.

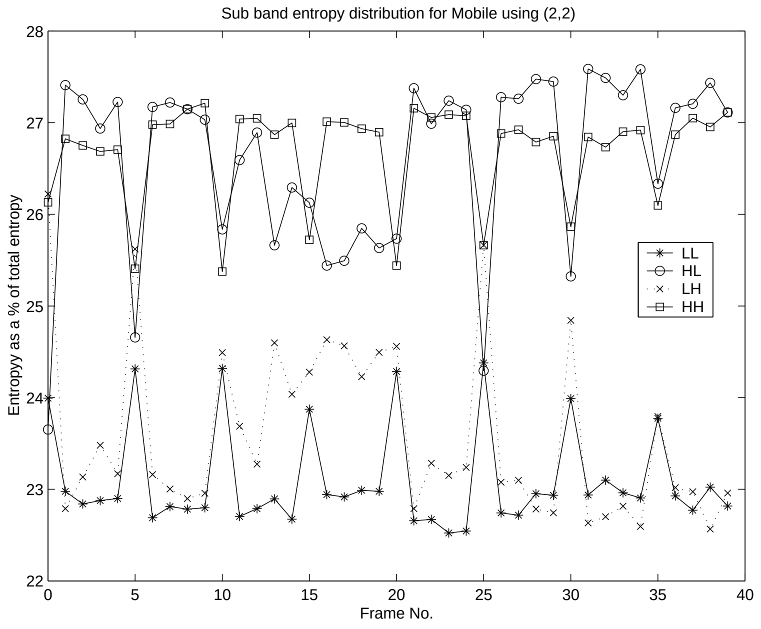


Figure 6.5: The entropy of each sub band as a % of the entropy of the frame for Mobile residuals.

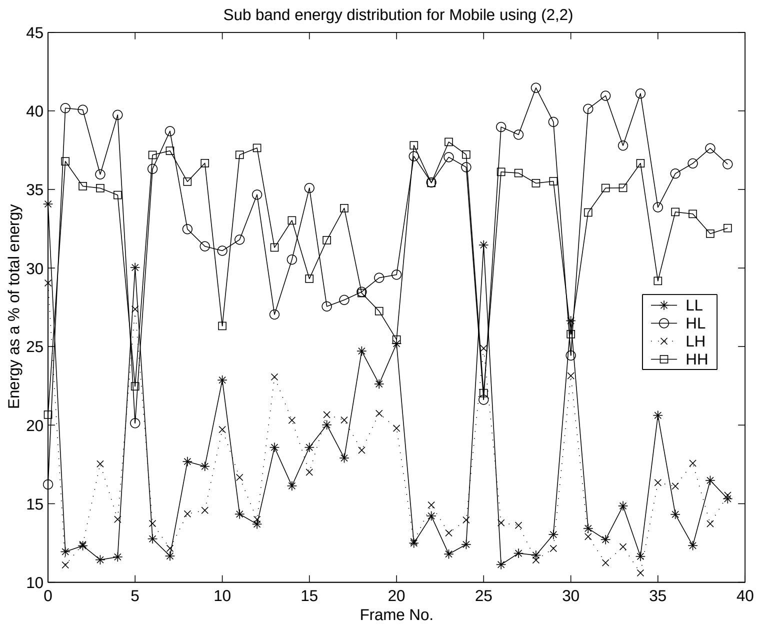
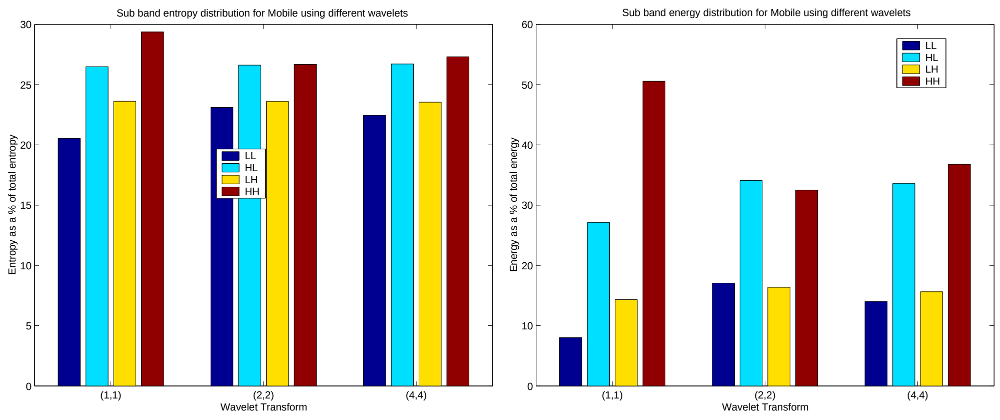


Figure 6.6: The energy of each sub band as a % of the energy of the frame for Mobile residuals.



Entropy Energy

Figure 6.7: Sub band entropy and energy average distributions as a % of total entropy and energy using different wavelets

### 6.3.2 The best wavelet basis

The zero-order weighted entropy values in bits per pixel (bpp) for each non-intra frame in the test sequences using the transforms applied up to a single level of iteration were computed. As an example those values for non-intra frames from Mobile and Kiel sequences are shown in Figure 6.8 and 6.9. The average entropy values for all non-intra frames are recorded in Table 6.1.

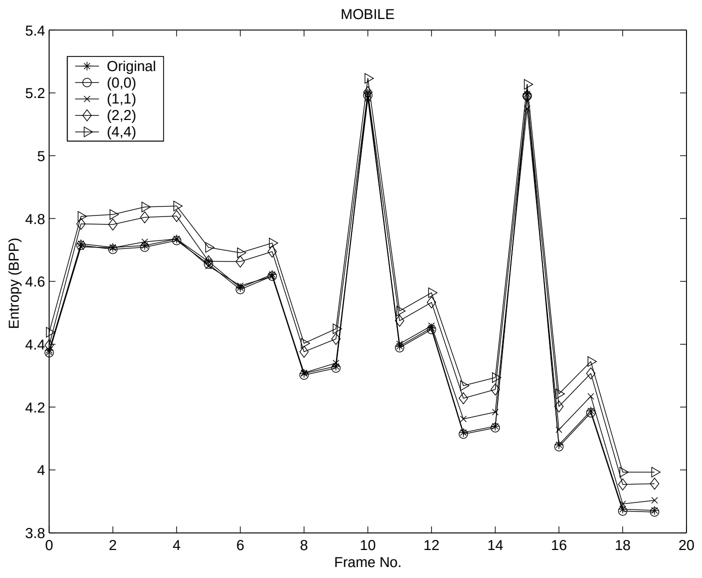


Figure 6.8: Total entropy (in bpp) using different wavelets for Mobile residuals

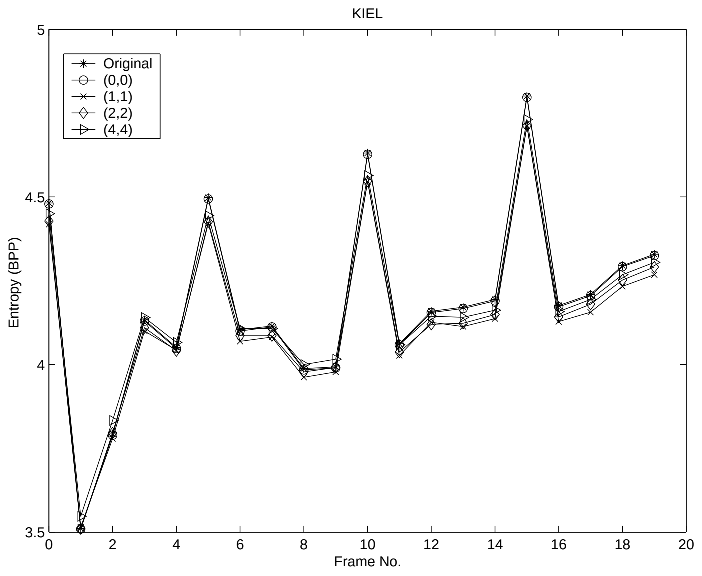


Figure 6.9: Total entropy (in bpp) using different wavelets for Kiel residuals

| | original entropy | (0,0) | (1,1) | (2,2) | (4,4) |
|---|---|---|---|---|---|
| Claire | 2.176 | 2.174 | 2.204 | **2.160** | 2.208 |
| Mobile | 4.529 | **4.524** | 4.531 | 4.594 | 4.625 |
| Kiel | 4.502 | 4.499 | **4.428** | 4.439 | 4.452 |
| Unicycle | 4.499 | 4.494 | 4.383 | **4.326** | 4.338 |
| Avergage | 3.927 | 3.923 | 3.886 | **3.880** | 3.906 |

Table 6.1: Average entropy (in bpp) using different IWT

These entropy values show that applying the wavelet transforms on residuals achieves only a slight advantage over not using a transform. The (1,1) transform and the (2,2) transform produce the lowest entropy values for the sequences. None of the transforms performed as the best option for all the sequences. Generally, any one of the transforms can perform best for a given sequence according to the extent to which they are initially decorrelated. It is further apparent from Figure 6.8 and 6.9, that the performance of each transform varies for different frames of the same sequence. However, wavelets with fewer vanishing moments performed better compared to those with more vanishing moments, as expected.

## 6.4 Spatially Adaptive Lifting

As stated above, the four wavelet transforms considered perform differently for each of the sequences. In this section, a novel method to choose P and U functions in the lifting process adaptively depending on the local statistics of the residuals is presented. A similar approach has been used in adaptive switching between different predictors based on the local edginess of the image in still image coding [113, 114, 115].

In any given frame, there are regions with different amounts of motion. Regions with low motion content produce smooth regions of low valued residuals due to accurate predictions. As a result, a high decorrelation can be seen in such regions. The regions with high motion cause high valued residuals, decorrelated to a certain extent, due to inaccurate predictions. Therefore, the amount of local motion present in a frame causes regions of high and low decorrelated regions in a residual field. A priori knowledge of such regions can be used to choose a suitable wavelet to transform the residual field. Spatially adaptive selection of wavelet basis, resulting in non-linear wavelets can easily be achieved with the lifting process. As the templates used to predict or update the members in one sub sample belong to the other sub sample, the members of that sub sample, which are a priori known at both coding and decoding ends, can be used for adaptive selection criteria. Because of this, the spatially adaptive lifting presented here can be used without coding any extra information regarding the basis function being used for each pixel. The problem of adaptively choosing vanishing / preserving moments is formulated based on two approaches, namely, an optimal prediction (AL-1) and an adaptive interpolation (AL-2), as discussed below.

### 6.4.1 An optimal prediction approach for adaptive lifting (AL-1)

In this approach the P lifting step is considered as a prediction process. The conditions for minimising the prediction error, which is represented as the $d$ channel, are derived in this approach. The same approach is used to minimise the $s$ channel coefficients in the U lifting step. These processes are described in the following two sub sections.

#### 6.4.1.1 Choosing the Predictor (P)

With the initial results, the four predictors, namely (0,$\tilde{N}$), (1,$\tilde{N}$), (2,$\tilde{N}$) and (4,$\tilde{N}$), where 0, 1, 2 and 4 are the numbers of vanishing moments to be introduced in the P lifting step and $\tilde{N}$ is the number of preserving moments in U lifting, which is independent of the number of vanishing moments in P lifting, are considered in this analysis. The mathematical formulation of the problem is as follows.

The prediction lifting step for $d_i$ using the neighbouring samples from $s$ channel :

$$\begin{aligned} d_i \;\leftarrow\; d_i \;-\; P(s_{\,A}) \quad (6.1) \\ where,\; A \;=\; (\,i - \lceil N/2 \rceil + 1\,, \ldots, i + \lfloor N/2 \rfloor\,) \\ N\, is\, the\, number\, of\, dual\, vanishing\, moments\, in\, d. \end{aligned}$$

This can be rewritten as below.

$$d_i \;\leftarrow\; d_i - \sum_{n=-\lceil N/2 \rceil+1}^{\lfloor N/2 \rfloor} a_{N_n} s_{i+n} \quad (6.2)$$

$a_{N_n}$ represents the prediction weights corresponding to the number of vanishing moments (N) of the predictor function as shown in Table 6.2.

| n | -1 | 0 | 1 | 2 |
|---|---|---|---|---|
| Predictors | | | | |
| N=0 | 0 | 0 | 0 | 0 |
| N=1 | 0 | 1 | 0 | 0 |
| N=2 | 0 | $\frac{1}{2}$ | $\frac{1}{2}$ | 0 |
| N=4 | $-\frac{1}{16}$ | $\frac{9}{16}$ | $\frac{9}{16}$ | $-\frac{1}{16}$ |

Table 6.2: The Predictor weights

It is known that the lower the new value for $d_i$, which is the prediction error, the higher the compression achieved. Therefore, the objective is to minimise the summed squared values of $d_i$ over the signal length. The cost function $E_P$, which is the summed squared error for the channel length, $\frac{L}{2}$ is as below.

$$E_P = \sum_{i=0}^{\frac{L}{2}-1} \left( d_i - \sum_{n=-\lceil N/2 \rceil+1}^{\lfloor N/2 \rfloor} a_{N_n} s_{i+n} \right)^2 \tag{6.3}$$

The minimum point of the above function occurs when its first differential with respect to a given predictor weight, $a_{N_k}$, is zero, as demonstrated below.

$$\begin{aligned} 0 &= \frac{\partial E_P}{\partial a_{N_k}} && (6.4) \\ &= \frac{\partial}{\partial a_{N_k}} \left( \sum_{i=0}^{\frac{L}{2}-1} \left( d_i - \sum_{n=-\lceil N/2 \rceil+1}^{\lfloor N/2 \rfloor} a_{N_n} s_{i+n} \right)^2 \right) \\ &= -2 \sum_{i=0}^{\frac{L}{2}-1} \left( \left( d_i - \sum_{n=-\lceil N/2 \rceil+1}^{\lfloor N/2 \rfloor} a_{N_n} s_{i+n} \right) s_{i+k} \right) && (6.5) \\ &= -2 \sum_{i=0}^{\frac{L}{2}-1} d_i s_{i+k} \; + \; 2 \sum_{n=-\lceil N/2 \rceil+1}^{\lfloor N/2 \rfloor} a_{N_n} \sum_{i=0}^{\frac{L}{2}-1} s_{i+n} s_{i+k} && (6.6) \end{aligned}$$

Equation 6.6 leads to the condition for the minimum error as below.

$$\underbrace{\sum_{i=0}^{\frac{L}{2}-1} d_i s_{i+k}}_{r_{ds}(k)} = \sum_{n=-\lceil N/2 \rceil+1}^{\lfloor N/2 \rfloor} a_{N_n} \underbrace{\sum_{i=0}^{\frac{L}{2}-1} s_{i+n} s_{i+k}}_{r_{ss}(k-n)} \tag{6.7}$$

where, $r_{ds}$ is the cross correlation between $d$ and $s$ channels and $r_{ss}$ is the auto correlation of the $s$ channel. Although the whole channel length, L/2, was considered in the preceding mathematical modelling of this problem, from this point onwards a finite window of length, N, which is the same as the predictor length, spanning from $-\lceil N/2 \rceil + 1$ to $\lfloor N/2 \rfloor$ is used in correlation computations. The points outside the window are considered as zero.

$$\sum_{i=0}^{\frac{L}{2}-1} s_{i+n}s_{i+k} \Rightarrow \quad \textstyle\sum_{i=-\lceil N/2 \rceil+1}^{\lfloor N/2 \rfloor - k} s_i s_{i+(k-n)} \quad = r_{ss}(k-n)_N \tag{6.8}$$

$$\sum_{i=0}^{\frac{L}{2}-1} d_i s_{i+k} \Rightarrow \quad \textstyle\sum_{i=-\lceil N/2 \rceil+1}^{\lfloor N/2 \rfloor - k} d_i s_{i+k} \quad = r_{ds}(k)_N \tag{6.9}$$

Further, by considering the initial spatial positions of the members of the $s$ and $d$ channels in the original data vector, $x$,

$$\begin{aligned} r_{ds}(k)_N &= \sum_{i=-\lceil N/2 \rceil+1}^{\lfloor N/2 \rfloor - k} d_i s_{i+k} \\ &= \sum_{i=-\lceil N/2 \rceil+1}^{\lfloor N/2 \rfloor - k} x_{2i+1}x_{2i+2k} \\ &= r_{xx}(2k-1)_N \end{aligned} \tag{6.10}$$

Hence, equation 6.7 becomes,

$$r_{xx}(2k-1)_N = \sum_{n=-\lceil N/2 \rceil+1}^{\lfloor N/2 \rfloor} a_{N_n} r_{ss}(k-n)_N \tag{6.11}$$

which in turn can be written in matrix form as in equation 6.11 by considering the predictors in Table 6.2.

$$\begin{bmatrix} r_{xx}(-3)_N \\ r_{xx}(-1)_N \\ r_{xx}(1)_N \\ r_{xx}(3)_N \end{bmatrix} = \begin{bmatrix} r_{ss}(0)_N & r_{ss}(-1)_N & r_{ss}(-2)_N & r_{ss}(-3)_N \\ r_{ss}(1)_N & r_{ss}(0)_N & r_{ss}(-1)_N & r_{ss}(-2)_N \\ r_{ss}(2)_N & r_{ss}(1)_N & r_{ss}(0)_N & r_{ss}(-1)_N \\ r_{ss}(3)_N & r_{ss}(2)_N & r_{ss}(1)_N & r_{ss}(0)_N \end{bmatrix} \begin{bmatrix} a_{N_{-1}} \\ a_{N_0} \\ a_{N_1} \\ a_{N_2} \end{bmatrix} \tag{6.12}$$

It should be noted that $r(-l) = r(l)$. If all $r_{ss}(l)_N$ and $r_{xx}(l)_N$ values are known, the equation 6.11 can be solved for the adaptive predictor weights, $a_{N_n}$. Equations 6.11 and 6.12 are similar to Wiener-Hopf equations for adaptive prediction based on minimum mean square error criteria [116, 117].

In this case, from a decoding viewpoint, although all members of $s$ channel are a priori known, not all members of $d$ channel are known a priori. Therefore, only the causal $d$ channel values are used to compute $r_{xx}(l)_N$. For this reason and due to the fact that the predictor weights are already defined in Table 6.2, the matrix equation 6.12 is used to determine the adaptive predictor selection criteria.

The maximum predictor length of the predictors in 6.2 is four and therefore, four equations can be written as in matrix equation 6.12. For each equation an error value $e(k)$ is computed as below.

$$e(k) \quad = \quad r_{xx}(2k-1)_N \; - \sum_{n=-\lceil N/2 \rceil+1}^{\lfloor N/2 \rfloor} a_{N_n}\, r_{ss}(k-n)_N \tag{6.13}$$

Hence, a mean squared error $E_N$ for each predictor is computed.

$$E_N \;= \tfrac{1}{4} \sum_{k=-1}^{k=2} e(k)^2 \tag{6.14}$$

The objective here is to select the predictor, so that the mean squared error for the constraint equation set is minimum. Therefore, the order of the predictor for a given signal member is determined as the predictor that corresponds to the minimum $E(N)$.

$$\begin{aligned} P_N() \quad &= \quad \{a_{N_{-1}}\; a_{N_0}\; a_{N_1}\; a_{N_2}\} \\ & \qquad where,\; N = E_N^{-1}\left(\min_{N\in\{0,1,2,4\}} E_N\right) \end{aligned} \tag{6.15}$$

#### 6.4.1.2 Choosing the Updator (U)

The main objective of the updating process is to preserve the running average of the low passed sub samples the same as that of the original signal and to avoid aliasing due to poor frequency separation in sub sampling operations. Updating is more important when the transform is iterated on the LL sub band and the coefficients are quantised in lossy coding applications. However, in adaptive updator selection criteria design, it is not feasible to incorporate either of the above objectives. In that case, the cost function to be minimised in this process is chosen as the summed squares of the updated value in $s$ channel. This leads to minimising the upper limit of the zero-order entropy of the $s$ channel.

As were considered in prediction, four corresponding updators, namely ($N$,0), ($N$,1), ($N$,2) and ($N$,4), where 0, 1, 2 and 4 are the number of moments preserved in $s$ channel in the U lifting step and $N$ is the number of vanishing moments in the P lifting step, which is independent of the number of preserving moments in U lifting, are considered. The problem of adaptive selection of an updator is mathematically formulated as follows.

The updating lifting step for $s_i$ using neighbouring samples from $d$ channel :

$$\begin{aligned} s_i \;\leftarrow\;& s_i + U(d_B) \qquad (6.16)\\ where,\; B \;=\;& (\, i - \lfloor \tilde{N}/2 \rfloor , \ldots , i + \lceil \tilde{N}/2 \rceil - 1\,) \\ & \tilde{N}\; is\; the\; number\; of\; preserving\; moments\; in\; s. \end{aligned}$$

This can be rewritten as below.

$$s_i \;\leftarrow\; s_i + \sum_{n=-\lfloor \tilde{N}/2 \rfloor}^{\lceil \tilde{N}/2 \rceil - 1} b_{\tilde{N}_n} d_{i+n} \qquad (6.17)$$

$b_{\tilde{N}_n}$ represents the updating weights corresponding to the number of preserving moments ($\tilde{N}$) of the updator function as shown in Table 6.3.

| n | -2 | -1 | 0 | 1 |
|---|---|---|---|---|
| Predictors | | | | |
| N=0 | 0 | 0 | 0 | 0 |
| N=1 | 0 | 0 | $\frac{1}{2}$ | 0 |
| N=2 | 0 | $\frac{1}{4}$ | $\frac{1}{4}$ | 0 |
| N=4 | $-\frac{1}{32}$ | $\frac{9}{32}$ | $\frac{9}{32}$ | $-\frac{1}{32}$ |

Table 6.3: The Predictor weights

As discussed earlier, the summed squared value of updates $s$ channel coefficients for the channel length, $\frac{L}{2}$, is considered as the cost function $E_U$, shown in equation 6.18.

$$E_U = \sum_{i=0}^{\frac{L}{2}-1} \left( s_i + \sum_{n=-\lfloor \tilde{N}/2 \rfloor}^{\lceil \tilde{N}/2 \rceil - 1} b_{\tilde{N}_n} d_{i+n} \right)^2 \qquad (6.18)$$

The minimum value of the above cost function occurs when its first differential with respect to a given updator weight is zero. This is demonstrated below with respect to $b_{\tilde{N}_k}$ as follows.

$$
\begin{aligned}
0 &= \frac{\partial E_U}{\partial b_{N_k}} && (6.19)\\
&= \frac{\partial}{\partial b_{\tilde{N}_k}} \left( \sum_{i=0}^{\frac{L}{2}-1} \left( s_i + \sum_{n=-\lfloor \tilde{N}/2 \rfloor}^{\lceil \tilde{N}/2 \rceil -1} b_{\tilde{N}_n} d_{i+n} \right)^2 \right) \\
&= 2 \sum_{i=0}^{\frac{L}{2}-1} \left( \left( s_i + \sum_{n=-\lfloor \tilde{N}/2 \rfloor}^{\lceil \tilde{N}/2 \rceil -1} b_{\tilde{N}_n} d_{i+n} \right) d_{i+k} \right) && (6.20)\\
&= 2 \sum_{i=0}^{\frac{L}{2}-1} s_i d_{i+k} + 2 \sum_{n=-\lfloor \tilde{N}/2 \rfloor}^{\lceil \tilde{N}/2 \rceil -1} b_{\tilde{N}_n} \sum_{i=0}^{\frac{L}{2}-1} d_{i+n} d_{i+k} && (6.21)
\end{aligned}
$$

Equation 6.21 leads to the condition for the minimum cost as below.

$$
\underbrace{-\sum_{i=0}^{\frac{L}{2}-1} s_i d_{i+k}}_{r_{sd}(k)} = \sum_{n=-\lfloor \tilde{N}/2 \rfloor}^{\lceil \tilde{N}/2 \rceil -1} b_{\tilde{N}_n} \underbrace{\sum_{i=0}^{\frac{L}{2}-1} d_{i+n} d_{i+k}}_{r_{dd}(k-n)} \qquad (6.22)
$$

where, $r_{sd}$ is the cross correlation between $s$ and $d$ channels and $r_{dd}$ is the auto correlation of the $d$ channel. Also in this case, a finite window of length, N, which is the same as the updator length, spanning from $-\lfloor \tilde{N}/2 \rfloor$ to $\lceil \tilde{N}/2 \rceil - 1$ is used in correlation computations. The points outside the window are considered as zero.

$$
\begin{aligned}
\sum_{i=0}^{\frac{L}{2}-1} s_i d_{i+k} \Rightarrow \quad & \textstyle\sum_{n=-\lfloor \tilde{N}/2 \rfloor}^{\lceil \tilde{N}/2 \rceil -1-k} s_i d_{i+k} && = \sum_{n=-\lfloor \tilde{N}/2 \rfloor}^{\lceil \tilde{N}/2 \rceil -1-k} x_{2i} x_{2(i+k)+1} \\
& && = r_{xx}(2k+1)_{\tilde{N}} && (6.23)\\
\sum_{i=0}^{\frac{L}{2}-1} d_{i+n} d_{i+k} \Rightarrow \quad & \textstyle\sum_{n=-\lfloor \tilde{N}/2 \rfloor}^{\lceil \tilde{N}/2 \rceil -1-k} d_i d_{i+(k-n)} && = r_{dd}(k-n)_{\tilde{N}} && (6.24)
\end{aligned}
$$

Hence equation 6.22 becomes,

$$
-r_{xx}(2k+1)_{\tilde{N}} = \sum_{n=-\lfloor \tilde{N}/2 \rfloor}^{\lceil \tilde{N}/2 \rceil -1} b_{\tilde{N}_n} r_{dd}(k-n)_{\tilde{N}} \qquad (6.25)
$$

The following matrix equation can be written from equation 6.25.

$$\begin{bmatrix} r_{xx}(-3)_{\tilde{N}} \\ r_{xx}(-1)_{\tilde{N}} \\ r_{xx}(1)_{\tilde{N}} \\ r_{xx}(3)_{\tilde{N}} \end{bmatrix} = - \begin{bmatrix} r_{dd}(0)_{\tilde{N}} & r_{dd}(-1)_{\tilde{N}} & r_{dd}(-2)_{\tilde{N}} & r_{dd}(-3)_{\tilde{N}} \\ r_{dd}(1)_{\tilde{N}} & r_{dd}(0)_{\tilde{N}} & r_{dd}(-1)_{\tilde{N}} & r_{dd}(-2)_{\tilde{N}} \\ r_{dd}(2)_{\tilde{N}} & r_{dd}(1)_{\tilde{N}} & r_{dd}(0)_{\tilde{N}} & r_{dd}(-1)_{\tilde{N}} \\ r_{dd}(3) & r_{dd}(2)_{\tilde{N}} & r_{dd}(1)_{\tilde{N}} & r_{dd}(0)_{\tilde{N}} \end{bmatrix} \begin{bmatrix} b_{\tilde{N}_{-2}} \\ b_{\tilde{N}_{-1}} \\ b_{\tilde{N}_0} \\ b_{\tilde{N}_1} \end{bmatrix} \tag{6.26}$$

It should be noted that $r(-l) = r(l)$. If all $r_{dd}(l)_N$ and $r_{xx}(l)_N$ values are known, the equation 6.25 can be solved for the adaptive updator weights, $b_{\tilde{N}_n}$. In this case, from a decoding viewpoint, although all members of $d$ channel are a priori known, not all members of $s$ channel are known a priori. Therefore, only the causal $s$ channel values are used to compute $r_{xx}(l)_{\tilde{N}}$. For this reason and due to the fact that the updator weights are already defined in Table 6.3, the matrix equation 6.26 is used to determine the adaptive updator selection criteria.

The maximum updator length of the updators in 6.3 is four and therefore, four equations can be written as in matrix equation 6.26. For each equation a final value $f(k)$ is computed as below.

$$f(k) \quad = \quad r_{xx}(2k+1)_{\tilde{N}} + \sum_{n=-\lfloor \tilde{N}/2 \rfloor}^{\lceil \tilde{N}/2 \rceil - 1} b_{\tilde{N}_n} r_{dd}(k-n)_{\tilde{N}} \tag{6.27}$$

Hence, a mean squared final value $F_N$ for each updator is computed.

$$F_{\tilde{N}} \quad = \tfrac{1}{4} \sum_{k=-2}^{k=1} f(k)^2 \tag{6.28}$$

The objective here is to select the updator, so that the mean squared final value for the constraint equation set is minimum. The order of the updator for a given signal member is the updator that corresponds to the minimum $F_{\tilde{N}}$.

$$\begin{aligned} U_{\tilde{N}}() \quad &= \quad \{b_{\tilde{N}_{-1}}\ b_{\tilde{N}_0}\ b_{\tilde{N}_1}\ b_{\tilde{N}_2}\} \\ &\qquad where,\ \tilde{N} = F_{\tilde{N}}^{-1}\left(\min_{\tilde{N}\in\{0,1,2,4\}} (F_{\tilde{N}})\right) \end{aligned} \tag{6.29}$$

### 6.4.2 An interpolation based approach for adaptive lifting (AL-2)

The first step in lifting is sub sampling the input data vector into two sub channels, $s$ and $d$. The P lifting step can be regarded as interpolating the $s$ channel to obtain the missing points due to previous sub sampling and the $d$ channel recording the error between the interpolated value and the corresponding original value. In the U lifting step, these interpolation errors in the $d$ channel are interpolated and a half of the interpolated value is added to the corresponding element in the $s$ channel, thereby maintaining the running average of the original input.

Traditionally, the process of interpolation is mostly associated with image re-sampling applications, where the interpolation of the discrete image to a continuous image and then sampling the interpolated image are involved [118]. The interpolation is mainly concerned with fitting a continuous function to discrete points in a digital signal. The most common interpolation functions are the nearest neighbour, linear and cubic functions, which are also analogous to the first, second and fourth moments of the polynomials used in the lifting steps respectively.

It is well known that a signal can be reconstructed from samples if the signal is band limited and the sampling is done at a frequency higher than the Nyquist rate. However, the residuals, and thereby their sub sampled channels, cannot be considered as band limited signals, as can be seen from the magnitude spectrum plots in Figure 6.3. The down sampling process can also be considered as replicating the frequency spectrum at the multiples of $2\omega_s$, where $\omega_s$ is the original sampling frequency. The interpolation process removes those replicates of the spectrum. According to the Wiener-Khintchine theorem, the spectrum of a finite energy signal can be obtained by the Fourier transform of the auto correlation sequence of the input signal. This suggests that the auto correlation sequence can be used to determine the interpolation criteria in the space domain. However, in this case, due to the time variant property of the down sampling processes, the famous relationship among input-output cross and auto correlation sequences and the filter impulse response cannot be used for the above purpose.

The nearest neighbour and the linear interpolation functions use two successive points in the down sampled stream to determine the point equidistant to those two points. The use of two successive points corresponds to the unit lag auto correlation of the down sampled signal. In this approach, the interpolator that maintains the unit lag normalised auto correlation of the down sampled signal at the local point of interest after the interpolation process is used to interpolate the local point. The derivation of the selection criteria are as below.

Let X and Y be the values of the two points to be interpolated. It is also noted that |Y|≥|X| and X and Y are not necessarily at the left and the right sides respectively. Since the normalised auto correlation is considered in this analysis, the two values 1 and $\frac{X}{Y}$, where $-1 \leq \frac{X}{Y} \leq 1$ and Y≠0, are used as the two values to be interpolated. It is expected to interpolate these two points into $\frac{1}{2}(X+Y)$ (or$\frac{1}{2}(\frac{X}{Y}+1)$ ), using the linear interpolation and into X (or $\frac{X}{Y}$), which is the lower absolute value, or Y (or 1), which is the higher absolute value, using the nearest neighbour interpolation. The latter interpolation is different from the prediction method used in the S transform, where the left side value in the P lifting and the right side value in the U lifting are always chosen irrespective of the value of the other point. The unit lag normalised auto correlation computed using the two points with values 1 and $\frac{X}{Y}$ for the down sampled signal and the corresponding unit lag normalised auto correlation using three points (original two points + interpolated value) for the interpolated signal for the above three scenarios are shown in the Figure 6.10.

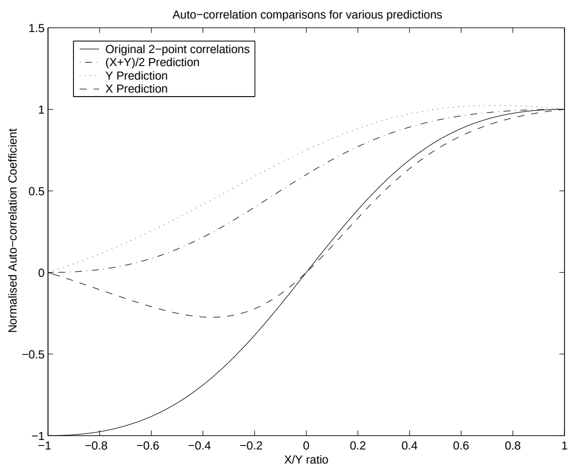
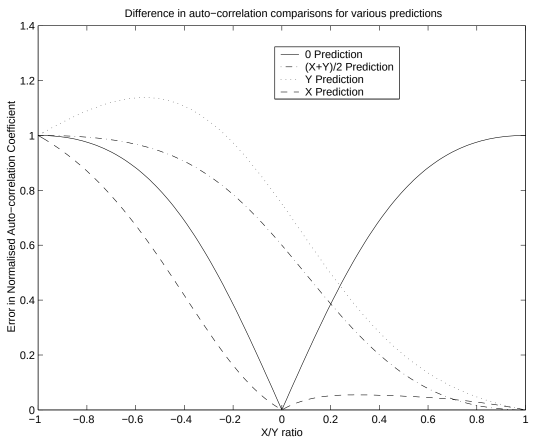
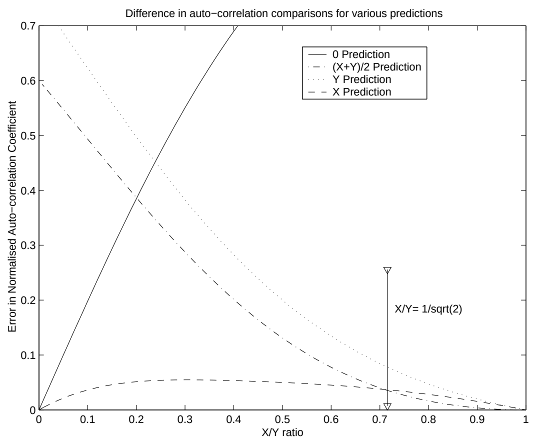


Figure 6.10: Resulting auto-correlation values for different interpolators

The plots of absolute difference of the unit lag normalised auto correlation for the three interpolated signals and that of the two points to be interpolated are shown in Figure 6.11. The resulting unit lag normalised auto correlation difference due to zero padded interpolation, which corresponds to the lazy wavelet, is also shown in the figure. It can be seen from the plots, that for $0 < \frac{X}{Y} < \frac{1}{\sqrt{2}}$ the nearest lower neighbour interpolation provides the closest auto correlation match and for $\frac{1}{\sqrt{2}} \leq \frac{X}{Y} \leq 1$ the linear interpolator provides the closest match. The values $\frac{X}{Y} < 0$ correspond to X and Y with different signs and to negative correlation. It is clear from Figure 6.11.a. that the nearest lower neighbour interpolation produces the closest match in this region. However, due to the sign difference of the neighbours, and thereby the negative auto correlation, a zero padded interpolation is considered in this region. The same treatment is given when X=Y=0.



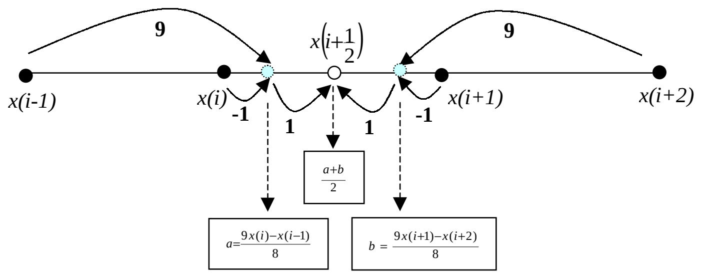


6.11.a) $-1 \le \frac{X}{Y} \le 1$ 6.11.b) $0 \le \frac{X}{Y} \le 1$

Figure 6.11: Resulting auto-correlation difference for different interpolators.

#### 6.4.2.1 Extension to cubic interpolation

The cubic interpolation uses four points to find the value at the mid point (using the two most immediate neighbours from either sides), whereas the above analysis for the nearest neighbour and the linear interpolation considered only two points. The weights in cubic interpolation for points at i-1, i, i+1 and i+2 positions are $\{-\frac{1}{16}, \frac{9}{16}, \frac{9}{16}, -\frac{1}{16}\}$ respectively . This can be interpreted as a linear interpolation of two points at i+$\frac{1}{8}$ and i+$\frac{7}{8}$, the values of which are computed by extrapolating the values at i-1 and i by 9:-1 ratio and extrapolating the values at i+1 and i+2 by -1:9 ratio, respectively as shown in Figure 6.12. The interpolated values a and b (Figure 6.12) at positions i+$\frac{1}{8}$ and i+$\frac{7}{8}$ can be considered as X and Y in the previous analysis for linear / nearest neighbour interpolation. The use of cubic interpolation can be determined by considering the ratio $\frac{X}{Y}$.



9
9
$x\left(i+\frac{1}{2}\right)$
x(i-1)
x(i)
x(i+1)
x(i+2)
-1
1
1
-1
$\frac{a+b}{2}$
$a=\frac{9x(i)-x(i-1)}{8}$
$b=\frac{9x(i+1)-x(i+2)}{8}$

Figure 6.12: Two point interpretation of the cubic interpolation

#### 6.4.2.2 The algorithm

The adaptive interpolation function selection can be summarised as in Figure 6.13. The same algorithm can be used in both P and U lifting steps. In P lifting the interpolated value is used to predict the members of the $d$ channel, whereas, in U lifting a half of the interpolated value is added to the corresponding members in the $s$ channel to update them with the running average.

```
X=x(i);  Y=x(i+1);
if ((X==0) AND (Y==0))  {"Use No Interpolation"};
else {
     if (sign(X) == sign(Y)) {
          if (abs(X) >= abs(Y))  {ratio = abs(Y)/abs(X)};
          else                   {ratio = abs(X)/abs(Y)};
          if (ratio < 1/sqrt(2)) {
              if (abs(X) >= abs(Y)) {"Nearest Neighbour Int. with Y"};
              else                  {"Nearest Neighbour Int. with X"};
          }
          else {
              a=(9*x(i)-x(i-1))/8;  b=(9*x(i+1)-x(i+2))/8;
              if (abs(a) >= abs(b)) {ratio_c = abs(b)/abs(a)};
              else                  {ratio_c = abs(a)/abs(b)};

              if (ratio > ratio_c)  {"Linear Interpolation"};
              else                  {"Cubic Interpolation"};
          }
     }
     else    {"Use No Interpolation"}
}
```

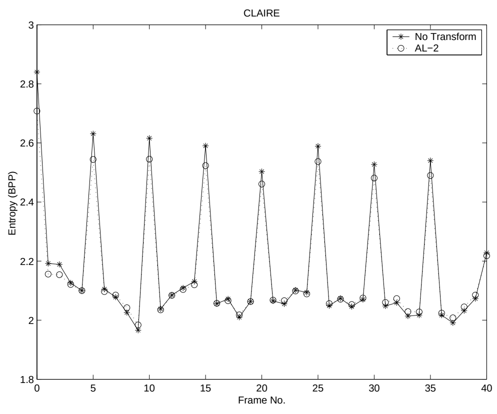
Figure 6.13: AL-2 Summary.

### 6.4.3 The zero-order entropy values

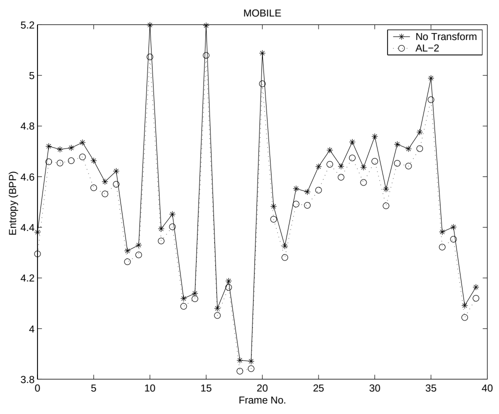
According to the above derivations, the adaptive lifting schemes possess the ability to choose the best wavelet basis from the basis set $(N, \tilde{N})$, where $N, \tilde{N} \in \{0, 1, 2, 4\}$, for a point by considering the statistics of its neighbours. The performance of two adaptive lifting approaches, namely optimal prediction based (AL-1) and interpolation based (AL-2), is compared against the usual wavelet filters, considered in section 6.3, by considering 40 consecutive non-intra frames from each sequence in the test sequence set. The average entropy values in bpp are as in Table 6.4.

| | Original entropy | (0,0) | (1,1) | (2,2) | (4,4) | AL-1 | AL-2 |
|---|---|---|---|---|---|---|---|
| Claire | 2.176 | 2.174 | 2.204 | 2.160 | 2.208 | 2.236 | 2.164 |
| Mobile | 4.529 | 4.524 | 4.531 | 4.594 | 4.625 | 4.525 | 4.469 |
| Kiel | 4.502 | 4.499 | 4.428 | 4.439 | 4.452 | 4.465 | 4.439 |
| Unicycle | 4.499 | 4.494 | 4.383 | 4.326 | 4.338 | 4.454 | 4.359 |
| Avergage | 3.927 | 3.923 | 3.886 | 3.880 | 3.906 | 3.920 | 3.858 |

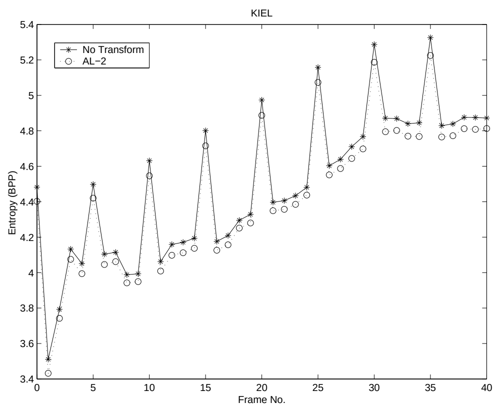
Table 6.4: Average entropy (in bpp) comparison for adaptive lifting



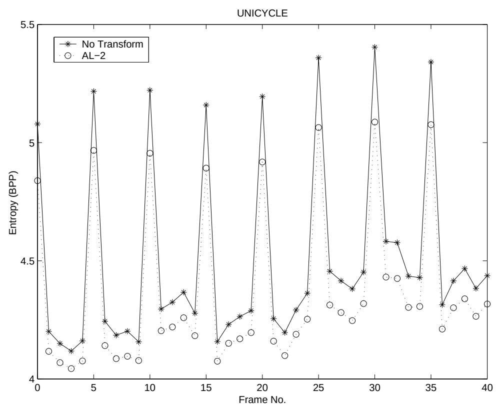
6.14.a. Claire



6.14.b. Mobile



6.14.c. Kiel



6.14.d. Unicycle

Figure 6.14: Entropy comparison in bpp for non-intra frames using AL-2.

As can be seen from Table 6.4, the adaptive lifting algorithm, AL-2, provides the overall lowest entropy for the non-intra frame set. The entropy values using AL-2 and the original entropy values without a transform for individual frames of the sequence tests are shown in Figure 6.14.

On the other hand, the AL-1 entropy values are only lower than the entropy values of the lazy wavelet transform and the original residuals without a transform. The inferior performance of the AL-1 may be due to the a priori non availability of points at the right hand side of the point to be predicted or updated for the $r_{xx}(l)$ computations. The approach in AL-1 is more suitable for determining the adaptive weights ($a_{N_n}$ and $b_{\tilde{N}_n}$) by solving the equations 6.12 and 6.26 rather than for adaptive selection of the order, i.e. the number of vanishing / preserving moments of the predictor or the updator by approximating the left and the right hand sides of those equations.

## 6.5 The Other Integer Transforms on Residuals

In this section the performance of the other integer transforms, presented and designed in Chapter 3 is compared in terms of zero-order entropy values. The IDCT, for its non-integer version's usual use in video coding, the IWHT, for its analogy with the S transform and the IDST, due to the highly decorrelated nature of the residuals were found by experiment as the transform option in non-intra frame coding. The best block size for the above transforms that gives the lowest entropy values for the residuals in non-intra frames were investigated and the average entropy values for such frames in the test sequence set are presented in Table 6.5 - Table 6.7. In the tables, the lowest entropy values for each sequence in each transform method are depicted in bold font.

| N | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| Claire | **2.172** | 2.270 | 2.369 | 2.469 | 2.523 |
| Mobile | **4.530** | 4.685 | 4.840 | 4.841 | 4.569 |
| Kiel | 4.421 | 4.441 | 4.487 | 4.477 | **4.258** |
| Unicycle | **4.382** | 4.438 | 4.530 | 4.582 | 4.418 |
| Avergage | **3.876** | 3.958 | 4.056 | 4.092 | 3.942 |

Table 6.5: Average entropy (in bpp) using IWHT$_N$

| N | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| Claire | **2.213** | 2.354 | 2.486 | 2.641 | 2.720 |
| Mobile | **4.539** | 4.669 | 4.806 | 4.822 | 4.644 |
| Kiel | 4.432 | 4.447 | 4.493 | 4.498 | **4.349** |
| Unicycle | 4.388 | 4.395 | 4.443 | 4.471 | **4.389** |
| Avergage | **3.893** | 3.966 | 4.057 | 4.108 | 4.025 |

Table 6.6: Average entropy (in bpp) using IDCT$_N$

| N | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| Claire | **2.213** | 2.377 | 2.516 | 2.661 | 2.733 |
| Mobile | **4.539** | 4.707 | 4.834 | 4.836 | 4.650 |
| Kiel | 4.432 | 4.491 | 4.533 | 4.522 | **4.361** |
| Unicycle | **4.388** | 4.489 | 4.559 | 4.582 | 4.438 |
| Avergage | **3.893** | 4.016 | 4.110 | 4.150 | 4.045 |

Table 6.7: Average entropy (in bpp) using IDST$_N$

The above results show that the smaller the block size, N, the lower the average weighted entropy of the residual frames. For most of the instances, the lowest entropy values were achieved for N=2. Unlike with still images, all three transforms performed comparably on residuals. Based on the overall averages, the 2-point IWHT has recorded the best results, followed by the 2-point IDST and the 2-point IDCT, which are the same. It can also be noted that a block size of 2 corresponds to a single scale wavelet transform, which is also the desired number of scales for the wavelet transforms based residual coding as shown earlier. The entropy values for the non-linear transform, INLT-3 are shown in Table 6.8 for different numbers of scales. In this case also it is evident that no entropy gains can be achieved by applying the transform in higher scales for the non-intra frames, as already seen for the other transforms.

| Scales | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Claire | **2.209** | 2.249 | 2.261 | 2.264 | 2.265 |
| Mobile | **4.686** | 4.742 | 4.757 | 4.760 | 4.760 |
| Kiel | **4.620** | 4.655 | 4.665 | 4.667 | 4.667 |
| Unicycle | **4.439** | 4.471 | 4.483 | 4.486 | 4.486 |
| Avergage | **3.988** | 4.029 | 4.041 | 4.044 | 4.044 |

Table 6.8: Average entropy (in bpp) using INLT-3

### 6.5.1 Transforms on residuals : Summary

The entropy values using the integer transforms on non-intra frames can be summarised as in Table 6.9. The average entropy values for the (2,2)-IWT, AL-1, Al-2 and the INLT-3 (all on a single scale) and 2-point block transforms (IWHT, IDST and IDCT) are compared with the original entropy values in the table below.

| | Original entropy | IWT (2,2) | AL-1 | AL-2 | $IWHT_2$ | $IDCT_2$ | $IDST_2$ | INLT-3 |
|---|---|---|---|---|---|---|---|---|
| Claire | 2.176 | 2.160 | 2.236 | **2.164** | 2.172 | 2.213 | 2.213 | 2.209 |
| Mobile | 4.529 | 4.594 | 4.525 | **4.469** | 4.530 | 4.539 | 4.539 | 4.686 |
| Kiel | 4.502 | 4.439 | 4.465 | 4.439 | **4.421** | 4.432 | 4.432 | 4.620 |
| Unicycle | 4.499 | **4.326** | 4.454 | 4.359 | 4.382 | 4.388 | 4.388 | 4.439 |
| Avergage | 3.927 | 3.880 | 3.920 | **3.858** | 3.876 | 3.893 | 3.893 | 3.988 |

Table 6.9: Average entropy (in bpp) comparison for integer transforms

As can be seen from the above table, the overall lowest entropy value is recorded for the spatially adaptive lifting algorithm, AL-2. However, unlike in still image coding, the overall differences in entropy values among different transforms are small. This is due to the decorrelation caused by the motion compensated prediction process. This is also made evident by comparing the average original entropy, which has only been decreased by 0.069 bpp when the best transform option, AL-2 is used. Therefore, the use of transforms on lossless coding of non-intra frames may sometimes not be useful when the high computational costs associated with the transforms are taken into account. However, the use of transforms may become necessary when there presents a high motion content in frames, thereby producing higher residual energy due to inaccurate motion compensated predictions. In this case, the use of an adaptive algorithm like, Al-1 or AL-2 is beneficial as they can adapt the transformation according to the spatial statistics of the residuals.

## 6.6 ELIC on Residuals

In this section, the possible further bit rate reductions by using an embedded quantiser are investigated. The embedded quantiser, ELIC, based on the adaptive quadtree splitting, designed in section 4.3.2 and the context model designed in section 5.1.1 is used with the above experimented integer transform options. Although the ELIC has been designed for still images, it can be used for non-intra frames, since the probability distributions of the sub bands after transformations in both types of frames can easily be fitted to a zero centred double sided geometrical distribution.

### 6.6.1 Lossless results for the sub band based transforms

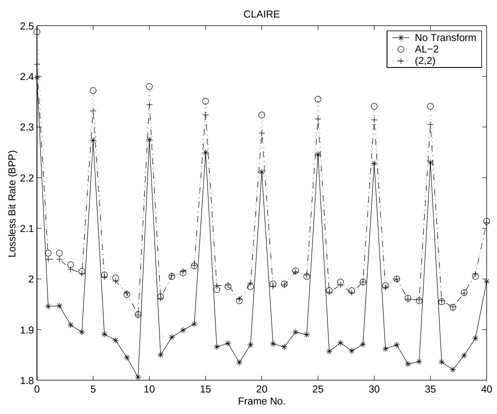
The lossless bit rates for non-intra frames, coded using ELIC with sub band based transforms are compared with the bit rates coded without using a transform as in Table 6.10. The table summarised the average bit rates for the non-intra frames using the (2,2)-IWT, AL-1, AL-2 and INLT-3 transforms with ELIC.

| | No Transform | (2,2) (1 Scale) | AL-1 (1 Scale) | AL-2 (1 Scale) | INLT-3 (1 Scale) |
|---|---|---|---|---|---|
| Claire | 1.951 | 2.059 | 2.122 | 2.067 | 2.107 |
| Mobile | 4.221 | 4.451 | 4.421 | 4.378 | 4.563 |
| Kiel | 4.124 | 4.186 | 4.228 | 4.213 | 4.357 |
| Unicycle | 4.208 | 4.233 | 4.372 | 4.284 | 4.368 |
| Average | 3.626 | 3.732 | 3.786 | 3.735 | 3.849 |

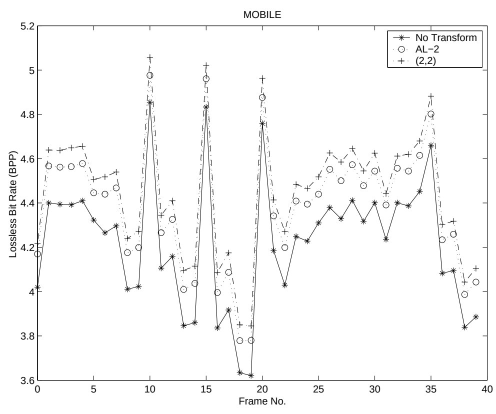
Table 6.10: Average lossless bit rates (in bpp) using integer wavelet transforms

As can be seen from the above table, the lowest lossless bit rates are achieved by not employing a transform. The lossless coding performance of the next best options, the (2,2)-IWT and the AL-2 transforms, are comparable to each other. However, on average 0.106 bpp advantage can be gained by not using a transform when compared with the (2,2) transform. The lossless bit rates for individual non-intra frames of the test sequence set using the non-transform case, the AL-2 transform and the (2,2) transform are compared in Figure 6.15.

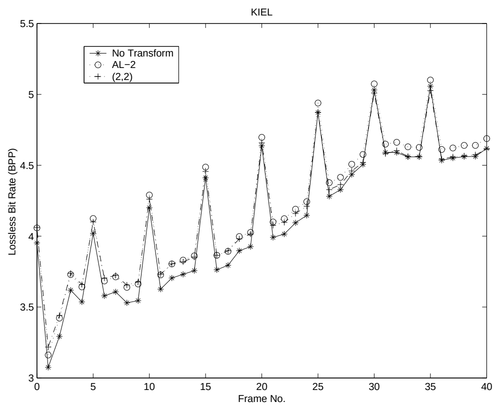
It can be seen from the plots in Figure 6.15, that all the frames not using a transform give the lowest bit rates at the lossless levels. Although the AL-2 transform gives the second best rates for all the non-intra frames in the Mobile sequence, it is comparable or slightly worse than the (2,2) lossless bit rates for the frames in other sequences.



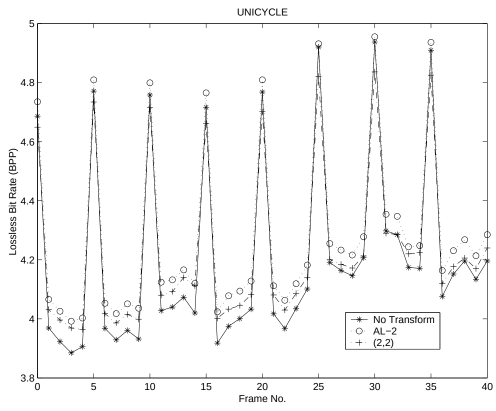
6.15.a. Claire



6.15.b. Mobile



6.15.c. Kiel



6.15.d. Unicycle

Figure 6.15: Lossless bit rate comparison in bpp for non-intra frames.

### 6.6.2 Lossless results for the integer block transforms

The integer block transforms are applied on the residual frames and they are reorganised back to the sub band structure by considering their analogy to the complete tree wavelet packet transform structure, as shown in Figure 3.6, so that the ELIC algorithm can be performed on those coefficients. Although it is evident from section 6.5, that the 2-point transforms provide the lowest entropy values, the performance of ELIC is investigated for all the block sizes considered in section 6.5. The bit rates in bpp at lossless levels for IWHT$_N$, IDCT$_N$, IDST$_N$, are as in tables 6.11, 6.12 and 6.13 respectively.

| N | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| Claire | 2.060 | 2.310 | 2.520 | 2.756 | 2.983 |
| Mobile | 4.384 | 4.704 | 4.985 | 5.186 | 5.371 |
| Kiel | 4.163 | 4.355 | 4.550 | 4.759 | 4.955 |
| Unicycle | 4.254 | 4.454 | 4.666 | 4.876 | 5.083 |
| Avergage | 3.715 | 3.956 | 4.180 | 4.394 | 4.598 |

Table 6.11: Average lossless bit rates (in bpp) using IWHT$_N$

| N | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| Claire | 2.081 | 2.283 | 2.408 | 2.616 | 2.825 |
| Mobile | 4.391 | 4.552 | 4.691 | 4.852 | 5.037 |
| Kiel | 4.175 | 4.274 | 4.351 | 4.456 | 4.599 |
| Unicycle | 4.253 | 4.316 | 4.390 | 4.510 | 4.725 |
| Avergage | 3.725 | 3.856 | 3.960 | 4.108 | 4.297 |

Table 6.12: Average lossless bit rates (in bpp) using IDCT$_N$

| N | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| Claire | 2.081 | 2.281 | 2.420 | 2.631 | 2.838 |
| Mobile | 4.391 | 4.591 | 4.719 | 4.869 | 5.045 |
| Kiel | 4.175 | 4.321 | 4.389 | 4.484 | 4.615 |
| Unicycle | 4.253 | 4.391 | 4.471 | 4.596 | 4.772 |
| Avergage | 3.725 | 3.896 | 3.999 | 4.145 | 4.318 |

Table 6.13: Average lossless bit rates (in bpp) using IDST$_N$

It is evident from the above tables that the lowest lossless bit rates are achieved with the smallest block sizes for each of the block transforms. Following the same pattern as entropy computations in section 6.5, the higher the block size, N, the higher the lossless bit rates. Although the lossless bit rates for 2-point transforms provide the lowest entropy for the respective transform, none of those rates are lower than the lossless bit rates obtained by not using a transform. However, according to the tables, the IWHT$_2$, IDCT$_2$ and IDST$_2$ have achieved lossless bit rates lower than those of the (2,2) transform and the AL-2 transform in a few instances.

## 6.7 Discussion

In this chapter, the main components of embedded lossless coding of non-intra frames were discussed. The main consideration was given to choosing an appropriate transform that maps integers into integers by considering the statistical properties of such frames.

The non-intra frames, which contain motion compensated residuals, are already decorrelated to a certain extent by the motion compensated prediction process. This was made evident by the comparisons of the properties such as the magnitude histogram distribution, the auto correlation coefficients, the magnitude spectrum and the DCT coefficients of the non-intra frames and the corresponding intra frame as shown in Figures 6.1 - 6.4. Since the reference frames used in the motion compensated prediction in a lossless video coding framework are also losslessly coded, the energy of the residuals in non-intra frames contain only the errors due to the inaccuracy in the motion prediction process. Unlike in lossy coding, the non-intra frames in lossless video coding do not propagate the quantisation errors in the reference frames.

It was evident that the integer wavelet transforms with a fewer number of vanishing / preserving moments produced lower lossless bit rates compared to those achieved form the wavelet transforms with a greater number of vanishing / preserving moments, due to the above mentioned intrinsic properties of the non-intra frames. However, it has been understood that the motion content of a sequence changes for each frame in different extents and so does the decorrelation gained by the motion compensated prediction process for each frame. This has suggested that no single transform would produce the best lossless rates for all the non-intra frames in a given sequence. On average, the (2,2) transform has produced the best lossless bit rates. Spatially adaptive selection of the number of vanishing moments in the prediction and updating steps in the lifting was considered as a solution for the above problem.

Two approaches, namely an optimal prediction approach (AL-1) and an interpolation based approach (AL-2), were derived and investigated. The optimal prediction approach for spatially adaptive lifting considers minimising the local mean square error and derives an equation set to match as in Wiener-Hopf normal equations for different lifting predictors and the predictor that minimises the error in the equation set is chosen as the best predictor. The same approach is followed in the updating steps. In the interpolation based approach the unit lag normalised auto correlation values for each point, using the three main interpolators, namely, the nearest neighbour, linear and cubic, which correspond to 1, 2 and 4 vanishing moments in the lifting steps, are considered. It was discovered that the AL-2 algorithm produced the best lossless entropy values on average. A further advantage of the above adaptive lifting algorithms are that they are spatially adaptive and no additional information needs to be sent to the decoder regarding the adaptive selection of the number of vanishing / preserving moment criteria. However, the AL-1 has not improved lossless entropy results from the normal wavelet transforms with the fixed numbers of vanishing / preserving moments. This may be due to the a priori non availability of points at the right hand side of the point to be predicted or updated in the $r_{xx}(l)$ computations. It is further understood that the approach in AL-1 is more suitable for determining the adaptive weights ($a_{N_n}$ and $b_{\tilde{N}_n}$) by solving the constraint equation sets 6.12 and 6.26 rather than for adaptive selection of the number of vanishing / preserving moments of the predictor or the updator by approximating the left and the right hand sides of those equations.

Further, it was discovered for the above mentioned sub band based transforms and the integer non-linear transform (INLT-3) that the improvement in lossless entropy bit rates achieved by further splitting of LL sub band repetitively into higher wavelet scales, as performed in still image coding, was not significant. This was due to the decorrelation caused by motion compensated prediction. It was evident from the sub band energy and entropy distributions after a single scale transform, that they were equally distributed among the four sub bands. Moreover, the amount of low frequency components were comparable with those of high frequency components, so that the further decomposition of LL sub band was not logical. Therefore, in these experiments only single scale transforms were considered.

The other integer transforms, the IWHT, the IDCT and the IDST also showed a similar pattern to the above by producing the lowest lossless entropy values with the 2-point transforms, which are analogous to a single scale sub band splitting. The IWHT produced the best lossless entropy rate. However, it was also evident by comparing with the average original entropy values that the entropy reductions gained by using a transform is small, for example, the AL-2, which achieved the lowest entropy rates, has managed to decrease the original bit rates only by 0.069 bpp. Therefore, the use of a transform on lossless coding of non-intra frames may be justifiable only when the data needs to be organised in the order of their significance as in embedded coding. In this case, the use of an adaptive algorithm like, Al-1 or AL-2 is beneficial as they can adapt according to the spatial statistics of the residuals.

The similar result patterns were experienced when the transform coefficients were entropy code with the embedded coder, ELIC. It was discovered that the motion compensated prediction residuals coded using ELIC but without a transform produced the best lossless bit rates, which was on average 0.106 bpp lower than the next best transform methods the (2,2) IWT and the AL-2 adaptive lifting transform.

# Chapter 7

# Embedded Lossless Video Performance

## 7.1 Introduction

In the previous chapters, the main components of an embedded lossless video coding based on a motion compensated prediction framework were discussed. In Chapter 3, the lifting concept based integer wavelet transforms were introduced and the integer versions of the other orthogonal transforms: the WHT, the DCT and the DST were derived. In Chapter 4, the concepts of embedded coding that can be used for a single video frame were discussed. The use of integer transforms and frame-wise embedded coding techniques into lossless / nearly-lossless coding of intra frames and lossless embedded coding of non-intra frames were discussed in Chapters 5 and 6 respectively. In this chapter, all the components discussed individually in the previous chapters are integrated together to form an embedded lossless video coder, so that the lossless and nearly-lossless coding and decoding performance of the codec can be compared and analysed.

The rest of the chapter is organised as below. Section 7.2 summarises the lossless video codec system. In section 7.3, the lossless coding performance of the coder is analysed by comparison with different transform options based on the results of the previous chapters. Further, the use of motion compensation in such a video framework is also analysed by comparing the results with those from Motion-ELIC and Motion-JPEG-LS, where ELIC and JPEG-LS are used on individual frames without motion compensation respectively. The quasi-lossless results are compared in a similar manner as above, with both the coding and decoding point of view in section 7.4. Finally, a discussion of the findings in this chapter can be found in 7.5.

## 7.2 The Embedded Lossless Video Codec (ELViC)

This research on the lossless video coding was based on a motion compensated prediction based framework, similar to that of MPEG-2. The motion compensated prediction and motion vector coding stage of the codec was based on the motion compensation stage of MPEG-2. The same "Group Of Pictures" (GOP) structure, with I, P and B frames, as in MPEG was followed with the GOP parameters of M=6 and N=3. The intra (I) frames of the codec were encoded using the (4,4) IWT followed by ELIC quantiser at lossless or quasi-lossless rates. Likewise, the performance of three types of transform options, viz., not using a transform, the (2,2) IWT and the AL-2 transform, followed by ELIC was researched with the non-intra frames. As ELIC is an embedded quantiser, the coding / decoding of any frame can be stopped at any given bit rate lower than the targeted bit rate or the lossless bit rate. A block diagram for the encoding part of the lossless video codec is shown in Figure 7.1

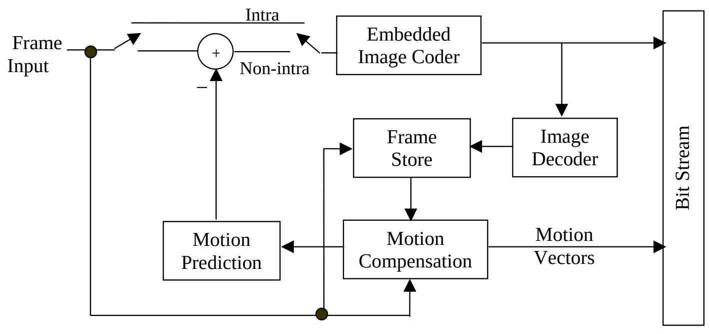


Figure 7.1: ELViC Block Diagram.

## 7.3 Lossless Coding Results

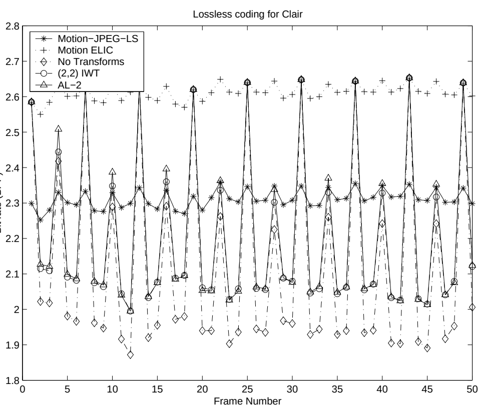
The lossless coding performance of ELViC, using three different transform options for non-intra frames is compared with those of Motion-JPEG-LS and Motion-ELIC. In Motion-JPEG-LS, the individual frames of a video sequence are coded losslessy using the current still image lossless coding standard, JPEG-LS. In Motion-JPEG-LS, the frames can be coded / decoded only at the lossless bit rate. In Motion-ELIC, the same concept as in Motion-JPEG-LS, but using the embedded quantiser ELIC is followed. With this method, each frame can be independently decoded at other bit rates lower than the lossless or the targeted bit rate. In both these methods, the temporal redundancy of video sequences have not been taken into account. On the other hand, the third algorithm, ELViC, employs a motion compensated prediction framework and a frame wise embedded coding strategy. Further, the three transform options, namely a zero-scale transform, i.e. not using a transform, the (2,2)-IWT and the AL-2 adaptive transform, considered for non-intra frames in ELViC are also analysed for its lossless performances. The lossless bit rates (in bpp) using fifty luminance (Y) frames of each test sequence are shown in Table 7.1. The lossless bit rates for individual frames are as in Figure 7.2.

| | Motion | Motion | ELViC | | |
|---|---|---|---|---|---|
| | JPEG-LS | ELIC | (0 scales) | (2,2) | (AL-2) |
| Claire | 2.312 | 2.612 | 2.121 | 2.210 | 2.217 |
| Mobile | 5.312 | 5.836 | 4.569 | 4.757 | 4.698 |
| Kiel | 4.973 | 5.159 | 4.367 | 4.417 | 4.440 |
| UniCycle | 5.039 | 5.246 | 4.449 | 4.469 | 4.511 |
| Avergage | 4.409 | 4.713 | 3.877 | 3.963 | 3.966 |

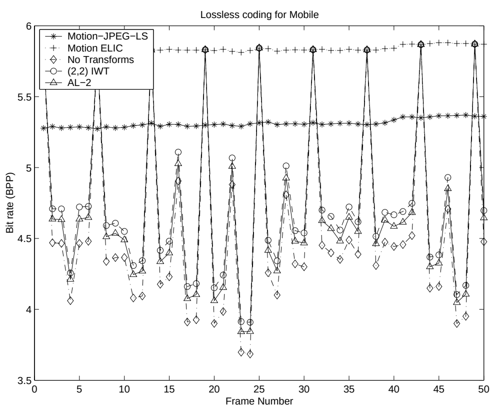
Table 7.1: Lossless coding performance (in bpp) of the lossless video codecs

As can be seen from Table 7.1, it is evident that significant reductions in bit rates can be achieved by using the motion compensation process. The three methods involving motion compensation have outperformed the other two methods for all four test sequences. The method that does not use a transform on residual frames has resulted in the lowest lossless bit rates, whereas the (2,2)-IWT and the AL-2 based techniques for residuals have produced rather inferior, but mutually comparable lossless bit rates.

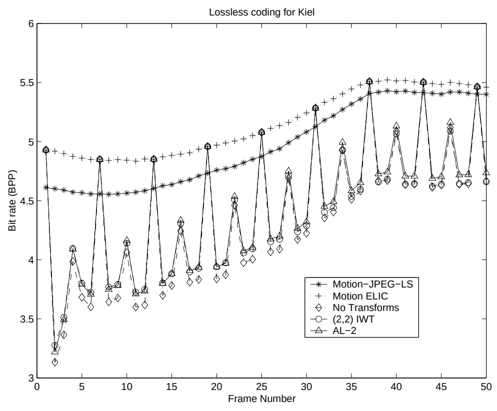
The plots in Figure 7.2 have further confirmed the superiority of not using a transform option over the other transform options, as it has produced the lowest lossless bit rate



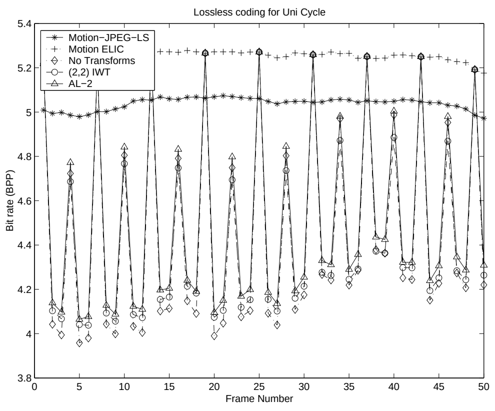
7.2.a. Claire



7.2.b. Mobile



7.2.c. Kiel



7.2.d. Unicycle

Figure 7.2: Lossless bit rate comparison in bpp for lossless video codecs.

for almost all the frames in the test sequences. Finally, it can be seen that Motion-ELIC has resulted in comparatively higher lossless bit rates compared to the Motion-JPEG-LS algorithm.

## 7.4 Quasi-Lossless Results

The main advantage of ELViC is the capability of lossless coding / decoding at other bit rates lower than the targeted bit rates. The quasi-lossless performance of ELViC using the three different transform options are investigated in this section. In a video codec, the reference frames are coded and decoded according to the bit rates, prior to using them for prediction of the current frame at the encoding end. This produces the reference frames as seen at the decoding end, provided that they are encoded and decoded to the same bit rate. However, in an embedded codec, the decoding bit rate is always equal or lower than the coding bit rate. This mismatch of bit rates at the decoding end for I and P frames, which are used as reference frames, could cause an accumulation of errors within a GOP, and thereby resulting in higher rms errors compared to the corresponding case where they are coded and decoded to the same bit rate. The susceptibility of ELViC at quasi-lossless bit rates are analysed using Mobile sequence as in Table 7.2.

| | BitRates | | | Average Bit Rate (bpp) | | | Case 1 rms error | | | Case 2 rms error | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | I | P | B | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| i | L | L | L | 4.569 | 4.757 | 4.698 | 0 | 0 | 0 | 0 | 0 | 0 |
| ii | L | L | 3 | 3.835 | 3.872 | 3.859 | 0.579 | 0.965 | 1.093 | 0.579 | 0.965 | 1.093 |
| iii | L | L | 2 | 3.195 | 3.232 | 3.219 | 1.155 | 1.614 | 1.851 | 1.155 | 1.614 | 1.851 |
| iv | L | 3 | 3 | 3.562 | 3.562 | 3.562 | 0.731 | 1.186 | 1.371 | 0.851 | 1.324 | 1.530 |
| v | L | 3 | 2 | 2.922 | 2.922 | 2.922 | 1.256 | 1.779 | 2.058 | 1.304 | 1.847 | 2.135 |
| vi | L | 2 | 2 | 2.742 | 2.742 | 2.742 | 1.478 | 2.042 | 2.361 | 1.658 | 2.246 | 2.592 |
| vii | 3 | 3 | 3 | 3.051 | 3.051 | 3.051 | 1.670 | 1.931 | 2.152 | 2.807 | 2.976 | 3.075 |
| viii | 3 | 3 | 2 | 2.411 | 2.411 | 2.411 | 2.058 | 2.461 | 2.785 | 3.002 | 3.266 | 3.445 |
| ix | 3 | 2 | 2 | 2.232 | 2.232 | 2.232 | 2.331 | 2.777 | 3.168 | 3.161 | 3.486 | 3.731 |
| x | 2 | 2 | 2 | 2.051 | 2.051 | 2.051 | 3.392 | 3.690 | 4.125 | 5.228 | 5.417 | 5.588 |

Table 7.2: Quasi-lossless coding performances of ELViC

In Table 7.2, the bit rate values for I, P and B type frames used are as combinations of the lossless bit rate L, and the quasi-lossless bit rates, 2 bpp and 3 bpp. In all combinations, the bit rate for an I frame has been chosen as equal to or higher than the bit rates for the other two frames and the bit rate for a B frame has been chosen as equal to or lower than the bit rates for the other two frames. The average bit rate in the third column is the final bit rate achieved by coding, using corresponding bit rates for I, P and B and coding of the motion vectors. The case 1 rms error refers to coding and decoding to the same bit rates, whereas the case 2 rms error corresponds to the lossless coding using the L-L-L bit rates and decoding to the bit rates up to the values in column 2 of the table. The sub columns, numbered 1, 2 and 3 correspond to the three transform techniques for non-intra frames, viz., zero-scale transforms, the (2,2) and the AL-2 respectively.

### 7.4.1 The embedded coding performances (Case 1)

The performance of coding and decoding a sequence using the same bit rates are analysed under this category. From the rms error values for the three methods in case 1, it is evident that not using a transform has resulted in the lowest rms errors for all bit rates. From these results, it can be inferred that even at quasi-lossless bit rate levels, the residuals are sufficiently decorrelated, so that they can be encoded without using a transform. The rate-distortion plots for all three methods, using the results available from Table 7.2 can be found in Figure 7.3. The higher rms error at bit rate 3.051 bpp, which corresponds to the scenario number vii, with ralative to the other neighbouring bit rates demonstrates the accumulated error caused by quasi-lossless coding of intra frames. This is also evident in Case 2, where quasi-lossless decoding is performed.

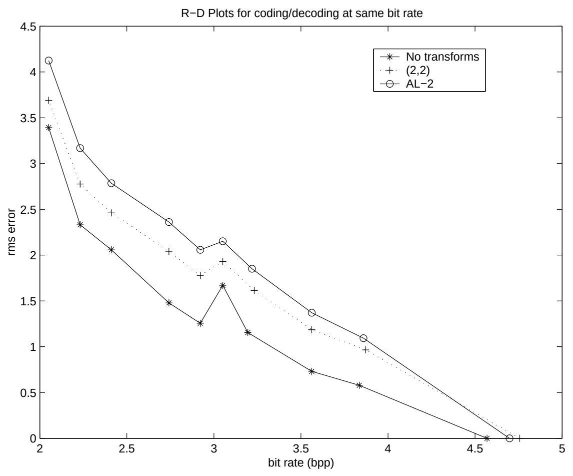
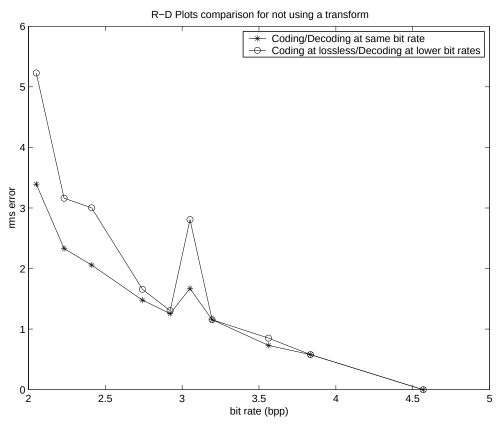


Figure 7.3: Rate-Distortion plots for Mobile coding and decoding to the same bit rates.

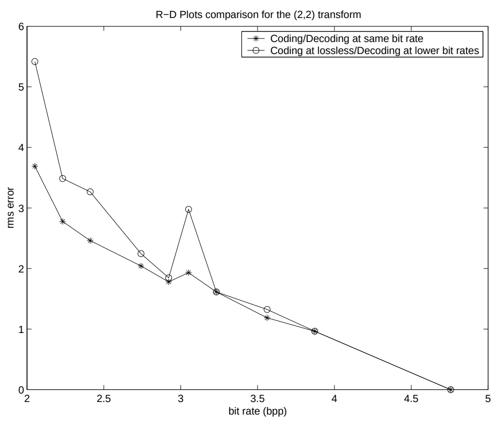
### 7.4.2 The embedded decoding performances (Case 2)

The performance of decoding a losslessly coded sequence into the lower bit rates shown in Table 7.2 is discussed under this category. The corresponding rms error values are reported under the case 2 category in Table 7.2. In this case also the non transform option has produced the lowest rms error values. The rate-distortion plots for the case 1, where coding and decoding are at the same bit rates, and the case 2, which corresponds to the quasi-lossless decoding of a losslessly a coded bit stream, for the three transform options are as in Figure 7.4.a-7.4.c.

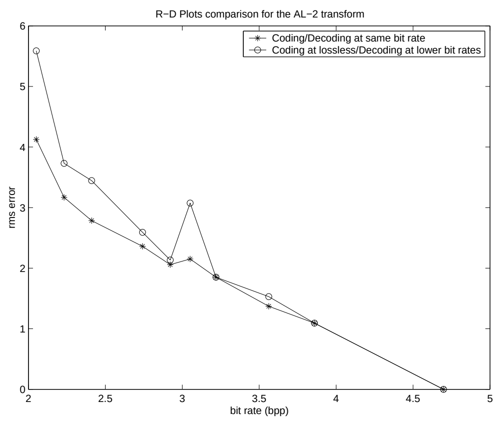
It can be seen from the plots, that an additional error is incurred in the embedded de-



7.4.a. Zero-scale transforms



7.4.b. (2,2)-IWT



7.4.c. AL-2

Figure 7.4: Embedded decoding R-D plots

coding process due to the inaccurate reference frames resulting from the above process. However, it is also evident that no additional errors due to the motion compensated prediction are accrued when all reference frames, i.e. I and P types, are coded and decoded losslessly and the non-reference frames, i.e. B type are decoded at quasi-lossless levels, as in bit rates ii and iii in Table 7.2. The extra rms error values produced by embedded decoding of the losslessly coded bit stream for the three transform methods are shown in Figure 7.5. As can be seen from the figure, although the zero scale transform option has produced the lowest increase in rms error at high bit rates, the AL-2 transform option has resulted in the lowest increase in rms error due to embedded decoding at lower bit rates. This also suggests the benefit of using a transform when decoding at lower bit rates.

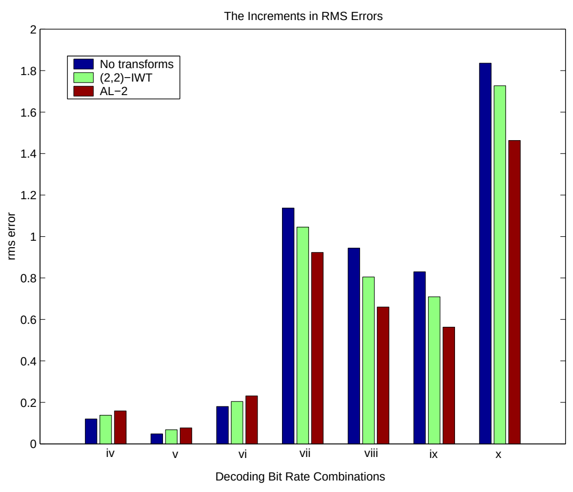


Figure 7.5: The rms error increment due to embedded decoding

## 7.5 Discussion

In this chapter, the lossless and the quasi-lossless performance of the embedded lossless video codec was analysed. As seen from the previous chapter, the superiority of not using a transform for non-intra frames was further confirmed in this section when compared with other lossless coding methods. Further, additional bit rate reductions by employing motion compensation processes were evident when the lossless bit rates for Motion-JPEG-LS and Motion-ELIC were compared with those of ELViC, which employs motion compensation. Although Motion-ELIC resulted in bit rates higher than those of the Motion-JPEG-LS, Motion-ELIC possesses the added advantage of embedded coding / decoding. With Motion-ELIC, the lossless bit stream can be decoded into other lower bit rates by considering frame by frame. Moreover, the embedded coding bit rate in Motion-ELIC can be more easily controlled compared to the rate control in ELViC, as all frames are equally important in Motion-ELIC. Embedded decoding of Motion-ELIC bit streams does not introduce additions to rms error values, whereas embedded decoding of ELViC bit streams at bit rates lower than the coded bit rate increases the rms errors.

The quasi-lossless decoding performance of ELViC, which is a motion compensation based video coder, was analysed from both a coding and decoding point of view in terms of the transform options used for non-intra frames. From the coding point of view, the lowest rms error values were achieved when a transform was not used in non-intra frame coding. The similar results were experienced from the decoding point of view. However, increased rms errors due to the inaccurate reconstruction of the reference frames were seen. This increment is lower when a transform is used for the lower quasi-lossless bit rates and AL-2 produced the smallest increments for low bit rates.

Finally, it can be concluded that although the motion compensation prediction process results in significant reductions in the lossless bit rates, the use of it can cause increased rms errors when decoded to lower bit rates in an embedded coding framework. However, this increase in rms error can be reduced by choosing an appropriate transform for coding non-intra frames.

# Chapter 8

# Conclusions

In lossless coding, compression and decompression of source data result in the exact recovery of the values of the individual elements in the original data source. Lossless video coding is useful in applications in which no error in pixel values is accepted after coding decoding processes. Lossless coding is vital in image / video archives, in temporary studio recordings in order to prevent accumulation of the quantisation artefacts caused by repeated coding decoding in programme editing, in inter studio transmissions and in coding of medical and remote sensing images. Although the lossless image coding has been given due consideration in published research, a little consideration has been given to research of lossless video coding. In this thesis, research on lossless and nearly-lossless video / image coding was presented. The majority of the lossless image coding techniques have used predictive coding techniques as the data correlation technique. Most of the lossless video coding research has been focussed on extending the 2-D techniques into lossless coding of 3-D signals. This thesis investigated the integer transforms based embedded lossless coding of video sequences using a motion compensated prediction based framework. Adding the embedded features into lossless coding has enabled decoding of a lossless bit stream at lower bit rates and has made the codec versatile, so that it can be used in both lossless / lossy coding / decoding.

The research presented in this thesis was mainly focussed on the integer transform options for intra frames (still images) and non-intra frames, an embedded quantiser that can be employed on intra and non-intra frames and the performance of frame-wise embedded coding / decoding in a motion compensated prediction based framework. The following remarks, grouped under the chapter names in which they appeared in, can be concluded from this research. Finally in section 8.6, the work that can be continued from this research is listed.

## 8.1 Integer Transforms

In Chapter 3, the concepts of integer transforms using lifting were introduced with the use of lifting factorisation of the wavelet transforms. The novel integer versions of the WHT, the DCT, and the DST, influenced by the lifting concepts and the fast transform implementation techniques, were designed using lifting and exploiting their intrinsic properties.

The IWHT was designed using factorisation of the WHT matrix (including the normalising factor) into sub matrices of Kroneckor products of $\text{WHT}_2$ and the corresponding Identity matrices. With this derivation, the $\text{WHT}_N$ can be regarded as applying the $\text{WHT}_2$ along a binary tree recursively to the lower and the upper halves of the signal. The integer version was designed by implementing the integer $\text{WHT}_2$, which is also similar to the S transform, using lifting steps.

The IDCT-II was designed by considering its intrinsic properties which lead to partitioning the transform matrix into four quadrants. With row and column permutations, it was seen that the upper left quadrant of the N-point DCT transform matrix is the same as the transform matrix for the $\frac{N}{2}$-point DCT. Further, the left and the right halves in the upper half share the same signs, whereas those in the lower half are with opposite signs. Further, it was shown that the upper and the lower half of the DCT matrix can be achieved by operating the $\text{IWHT}_2$ on the input data vector. The DCTs of smaller sizes, thereby the upper half of the N-point DCT matrix, can be obtained by recursively repeating this process until N=2. The lower half of the N-point DCT was computed using the $\text{IWHT}_{\frac{N}{2}}$ followed by the Kroneckor products of rotations by the basic angles and corresponding Identity matrices. The use of IWHT and the lifting factorisation of the rotation matrix enabled the integer implementation of the N-point DCT transform, where N is an integer power of 2, with the normalising factors being incorporated into the derivation.

The IDST-II was designed by using the relationship of the DCT and the DST coefficient matrices. The IDST coefficients were achieved by incorporating column and row permutation, derived from their relationship, into either ends of the IDCT algorithm. This relationship can be used to compute DST coefficients using any DCT processors, and can be used in other applications, such as fast transform design.

The non-linear transforms were devised in order to investigate their usability in lossless video coding. This was done by using a median based non-linear prediction function in predicting the pixels in sub bands obtained by quincunx splitting. A non-linear updating method was also introduced.

All these designs can be implemented as in-place operations, which lead to low resource demands. It was further shown that the block transform coefficients can be rearranged into a corresponding wavelet packet transform like sub band structure.

The zero-order weighted entropy values obtained from the integer wavelet transforms and the other integer transforms designed in chapter 3 for intra frames suggested the sub band based transforms with 5 scales and the block based transforms with $32\times32$ block sizes providing the best lossless bit rates.

The (4,4)-IWT resulted in the best lossless entropy values on all images, when compared with the other integer wavelet transforms, due its greater number of vanishing moments in primal and dual lifting steps. Further, it can be concluded that the IWT results are in accordance with the number of vanishing moments involved in lifting steps. As seen from the results, the greater the vanishing moments involved in lifting steps, the lower the weighted entropy values. Although the S+P transform shows better performance than the S transform, due to the additional prediction step, it has not outperformed the other IWTs. A similar performance can be seen with the (2+2,2) transform, which also includes an extra prediction step.

The IDCT has achieved the best zero-order entropy results, when compared with all the block transforms and the other transforms.

As expected, the IDST performance on lossless image coding was the worst out of all the block based transforms. It was assumed that this was due to its inapplicability to highly correlated data. The IWHT, which can also be considered as a wavelet packet decomposition of the S transform, performed better than the S transform; however, did not outperform the S+P transform.

As a whole, the non-linear transforms resulted in the highest weighted entropy values, thereby providing the worst lossless performance. However, the INLT3, the best of the three non-linear transforms considered, has outperformed the IDST on average and for most of the test images.

From the summarised lossless entropy values, it is evident that the performance of the IDCT is the best for lossless still image coding. On average, the $\text{IDCT}_{32}$ has achieved an advantage of 0.09 bpp over the second best transform, the (4,4) IWT.

## 8.2 Embedded Quantiser Design

The embedded quantiser presented in Chapter 4 used a simple bit plane based embedded coding system. The use of bit planes as the quantisation levels provided a quantisation scheme, of which the quantisation bins are reduced by a factor of two at each level.

The normalisation of the integer coefficients of the IWT, IWHT and INLT by virtual bit plane sliding process normalises the coefficient by their corresponding normalising constants while preserving the dynamic range. With this method, the unnecessary coding of zeros resulting from traditional multiplication based normalising was avoided. Furthermore, coding the maximum coefficient height (msb) for each sub band as side information (depth limiting process), avoided the coding of unnecessary zeros above the highest msb of a sub band. These two processes provided a compact effective bit space.

It was found that the cost of embedded coding can be reduced, if not eliminated, by devising SSM scanning schemes that can group insignificant bits, $N$, which possess information regarding the position of $S$ bits together so that they can be coded using fewer bits. This was investigated by experimenting with the efficient scanning schemes for the SSM. Out of the four scanning schemes considered: raster, quadtree, wavelet tree - zero tree and wavelet tree - HVZ, the quadtree based scanner produced the lowest bit rates. However, when individual bit planes were considered, the quadtree scan was the best for the higher bit planes as was the raster scan for the lower bit planes.

These observations were used to design an intelligent scanning scheme that can adaptively switch between the raster and the quadtree scans. This was achieved by the novel scanning scheme, adaptive quadtree splitting (AQS), which uses two quadtree techniques, QT1 and QT2, which are capable of switching between scans adaptively according to the current block statistics. A decision criteria based on the information predicted from the parent sub band and a priori known information from the current block was designed. It is evident from the bit rate tables (Table 4.2 and the tables in Appendix C) that the AQS has improved the results from the earlier scans by 10% on average. Furthermore, AQS has produced the lowest bit rates for most of the individual bit planes.

No special scanning techniques for coding the coefficient signs and refining data bits, which also constitute a binary symbol alphabet, were considered due to high randomness present in those bits.

## 8.3 Lossless Coding of Intra Frames

The use of integer transforms and embedded quantising on embedded to lossless image coding (ELIC) that can be used for intra frames was presented in Chapter 5.

It was evident from the experiments that further reduction of bit rates can be gained by using context based entropy coding in ELIC. The context model, designed for ELIC, reduces the lossless bit rates by 4.6% on average for the test image set. Only a simple context model using the Markov-1 prediction was used in modelling contexts for coding sign, due to the non availability of knowledge of most of the neighbouring sign information in an embedded coding framework.

The lossless bit rates for ELIC-IWT, using the (4,4) IWT, have outperformed those of SPIHT by 0.7% on average. Moreover, on average, ELIC-IWT has produced bit rates within 0.1% of JPEG-LS, which is a predictive lossless coding method, where the lossless bit stream can only be decoded at the lossless bit rate.

The optimum block sizes that give the lowest lossless bit rates for the other integer transforms using ELIC were found to be different from those discovered using initial entropy computations in Chapter 3. This is mainly due to their wavelet packet type sub band arrangement being reorganised into a such way that the correct parent-child relationship is considered in ELIC. A block size of 16 for the IDCT and a block size of 4 for the IWHT and the IDST produced the lowest lossless bit rates for those transforms. However, when all the transforms were considered, the lowest lossless bit rates for intra frame type images were achieved by the IWT-(4,4), followed by the $\mathrm{IDCT}_{16}$, INLT-3, the $\mathrm{IWHT}_4$ and the $\mathrm{IDST}_4$.

The near-lossless coding, in which each reconstructed pixel in the output from decoder differs from the input to the encoder by not more than a value, $\delta$, specified at the time of coding, is more commonly used with the prediction based lossless coding methods. Usually, near-lossless coding in integer transforms based lossless coding is achieved by quantising the input using $\delta$ prior to the forward transform. The near-lossless results, both bit rates and rms error, obtained using pre-quantisation are always inferior to those obtained from predictive techniques. Two novel near-lossless coding methods, based on incorporating the near-lossless quantisation into lifting steps in the first transform scale, by considering separable (1-D online) and non-separable (2-D online) transforms, was introduced. It was evident from the bit rate and rms error results, that the online (in-transform) quantisation methods have improved the transforms based near lossless coding performances. The 2-D online method produced lower bit rates and rms error values compared to those from the 1-D online method and the pre-quantisation method at the tested near-lossless levels for the image set.

The quasi lossless performance of ELIC using the (4,4) IWT and the 8-point IDCT, was compared with that of the lossless mode of SPIHT. ELIC-IWT produced the best performance at the lossless level and at the bit rates higher than 3 bpp for most of the images in the image set. Although ELIC with the 8-point IDCT produced inferior rate distortion performance at high bit rates, it has shown a trend of comparable performance with ELIC-IWT at lower bit rates.

## 8.4 Lossless coding of Non-Intra Frames

The non-intra frames, which contain motion compensated residuals, are already decorrelated to a certain extent by the motion compensated prediction process. This was evident from the comparisons of the properties such as the magnitude histogram distribution, the auto correlation coefficients, the magnitude spectrum and the DCT coefficients of such frames and the corresponding intra frame. Since the reference frames used in the motion compensated prediction in a lossless coding frame work are also coded losslessly, the energy of the residuals in non-intra frames contain only the errors due to inaccuracy in the motion prediction process. Unlike lossy coding, the non-intra frames do not propagate the quantisation errors in the reference frames.

It was evident that the integer wavelet transforms with a fewer number of vanishing / preserving moments produced lower lossless bit rates compared to those achieved form the wavelet transforms with a greater number of vanishing / preserving moments due to the above mentioned intrinsic properties of the non-intra frames. On average the (2,2) transform produced the best lossless bit rates. However, it has been understood that the motion content of a sequence changes for each frame in different extents and so does the decorrelation gained by the motion compensated prediction process for each frame. This has suggested that no single transform would produce the best lossless rates for all non-intra frames in a given sequence. A spatially adaptive selection of a number of vanishing moments in the prediction and the update steps in lifting was considered as a solution for the above problem.

Two spatially adaptive lifting algorithms, based on an optimal prediction approach (AL-1) and an adaptive interpolation based approach (AL-2), were designed and experimented with for non-intra frames. In the optimal prediction approach, an equation set as in Wiener-Hopf normal equations that minimise the local mean square error for different lifting predictors was derived and it was used to choose the best predictor. The same approach was used for the updator. In the interpolation based approach, the unit lag normalised auto correlation values for each point, using the three main interpolators, namely, the nearest neighbour, linear and cubic, which correspond to 1, 2 and 4 vanishing moments in the lifting steps, were considered to decide the best interpolator in the lifting steps.

On average, the AL-2 algorithm produced the best lossless entropy values. A further advantage of the above adaptive lifting algorithms is that they are spatially adaptive and no additional information needs to be sent to the decoder regarding the adaptive selection of the number of vanishing / preserving moment criteria. However, the AL-1 did not improve lossless entropy results from the normal wavelet transforms with a fixed number of vanishing / preserving moments. This may be due to the a priori non availability of points at the right hand side of the point to be predicted or updated in $r_{xx}(l)$ computations. It is further understood that the approach in AL-1 is more suitable for determining the adaptive weights ($a_{N_n}$ and $b_{\tilde{N}_n}$) by solving the normal equation sets rather than for adaptive selection of the number of vanishing / preserving moments of the predictor or the updator by approximating the left and the right hand sides of those equations.

Further, it was discovered for the above mentioned sub band based transforms and the integer non-linear transform (INLT-3) that the improvement in lossless entropy bit rates achieved by further splitting of LL sub band repetitively into higher wavelet scales, as performed in still image coding, was not significant. This was due to the decorrelation caused by motion compensated prediction. It was evident from the sub band energy and entropy distributions after a single scale transform that the total energy and the entropy were equally distributed among the four sub bands. Moreover, the amount of low frequency components in residuals were comparable with those of high frequency components, so that the further decomposition of LL sub band was not logical. Therefore, in these experiments only single scale transforms were considered.

The other integer transforms, the IWHT, the IDCT and the IDST also showed a similar pattern to the above by producing the lowest lossless entropy values with the 2-point transforms, which are analogous to a single scale sub band splitting. The IWHT produced the best lossless entropy rate out of above three. However, it was also evident by comparing with the average original entropy values that the entropy reductions gained by using a transform is small; for example, the AL-2, which achieved the lowest entropy rates, has managed to decrease the original bit rates only by 0.069

bpp. Therefore, the use of a transform on lossless coding of non-intra frames may be justifiable only when data needs to be organised in the order of its significance as in embedded coding. In this case, the use of an adaptive algorithm like, Al-1 or AL-2 is beneficial as they can adapt according to the spatial statistics of the residuals.

The similar result patterns were experienced when the transform coefficients were entropy coded with the embedded coder, ELIC. It was discovered that the motion compensated prediction residuals coded using ELIC but without a transform produced the best lossless bit rates, which was on average 0.106 bpp lower than the next best transform methods the (2,2) IWT and the AL-2, adaptive lifting transform.

## 8.5 Embedded Lossless Video Coding

The lossless and the quasi-lossless performance of the embedded lossless video codec (ELViC) were compared with the non-motion compensated methods, Motion-JPEG-LS and Motion-ELIC in chapter 7. From the lossless bit rate results, it can be concluded that incorporating the motion compensation achieves reasonable bit rate reductions in lossless coding. On average, it has achieved about 0.537 bpp bit rate reduction.

Three transforms techniques, namely a zero-scale transform, the (2,2)-IWT and the AL-2 were used as the transform option for the non-intra frames. The superior performance achieved by not using a transform for non-intra frames was further confirmed in this section when compared with other transforms methods. The other two transforms produced comparable results.

Out of non-motion compensated methods, although Motion-ELIC resulted in bit rates higher than those of the Motion-JPEG-LS, the lossless bit streams from Motion-ELIC can be decoded into other lower bit rates due to its embedded property, with which the bit rates can be more easily controlled. Since a motion compensation is not involved, embedded decoding can be performed without incurring extra rms error.

The quasi-lossless performance of ELViC, which is a motion compensation based video coder, was analysed from both a coding and decoding point of view in terms of the transform options used for non-intra frames. From the coding point of view, where the sequences are coded and decoded at the same bit rate, the lowest rms error values were achieved when a transform was not used in non-intra frame coding. The similar results were experienced from the decoding point of view, where a sequence is coded losslessly and decoded to a lower bit rate. However, increased rms errors due to the inaccurate reconstruction of the reference frames were seen in this case. This increment is lower when a transform is used for lower quasi-lossless bit rates and AL-2 produced the smallest increments for low bit rates.

It can be concluded that although the motion compensation prediction process results in significant reductions in the lossless bit rates, the use of it can cause increased rms errors when decoded to lower bit rates in an embedded coding frame work. However, this increase in rms error can be reduced by choosing an appropriate transform for coding non-intra frames. The results at lower bit rates suggest the use of the AL-2 for lower increments of rms errors.

## 8.6 Future Work

As a solution to the above problem experienced with the motion compensated prediction based embedded video coding, the use of 3 D transforms and 3-D embedded quantisers can be designed. With a 3-D transforms based embedded quantiser, the coding and decoding can be synchronised following the same pattern, as no motion compensation is involved. Due to this, it will result in the same rms error values in case 1 and case 2 type coding and decodings.

Further, the above experiments were performed only on luminance frames. The normal practice within the research community is to allocate the bit budget into three spectral bands according to a predetermined rate and to perform embedded coding individually on each spectral band. However, this can be made fully embedded by extending the dimensions of the transform and the embedded quantiser by one step, so that all three spectral bands can be taken into account in the embedding process. These types of transforms are applicable in the RGB domain rather than in the YUV domain, since the latter is the output of a principal component analysis process, that can also be considered as a form of transform.

In this research, the coefficients in embedded coding have been organised according to the bit significance criteria, which in turn is related to the image energy. By employing a visually embedding criterion, the lossless stream can be made visually embedded and thereby it can be decoded at a visually lossless bit rate, which is more useful in entertainment video applications.

# Appendix A

# Lifting Steps

The analysis (forward) lifting steps for the wavelet transforms tested in section 3.1.3 are as below. The channels, $s$ and $d$ represent low and high pass bands respectively. All the lifting steps are for $i = 0, \ldots, (L/2 - 1)$, where $L$ is the length of the input signal.

**The lazy wavelet - also known as (0,0) wavelet** [56]

$$d_i \leftarrow x_{2i+1} \tag{A.1}$$

$$s_i \leftarrow x_{2i} \tag{A.2}$$

**The S transform - also known as (1,1) wavelet** [56]

$$d_i \leftarrow d_i - s_i \tag{A.3}$$

$$s_i \leftarrow s_i + \left\lfloor \frac{1}{2} d_i + \frac{1}{2} \right\rfloor \tag{A.4}$$

**The (2,2) wavelet** [56]

$$d_i \leftarrow d_i - \left\lfloor \frac{1}{2} (s_i + s_{i+1}) + \frac{1}{2} \right\rfloor \tag{A.5}$$

$$s_i \leftarrow s_i + \left\lfloor \frac{1}{4} (d_{i-1} + d_i) + \frac{1}{2} \right\rfloor \tag{A.6}$$

**The (4,2) wavelet** [56]

$$\begin{aligned} d_i &\leftarrow d_i - \left\lfloor \frac{9}{16}(s_i + s_{i+1}) - \frac{1}{16}(s_{i-1} + s_{i+2}) + \frac{1}{2} \right\rfloor && \text{(A.7)} \\ s_i &\leftarrow s_i + \left\lfloor \frac{1}{4}(d_{i-1} + d_i) + \frac{1}{2} \right\rfloor && \text{(A.8)} \end{aligned}$$

**The (4,4) wavelet** [56]

$$\begin{aligned} d_i &\leftarrow d_i - \left\lfloor \frac{9}{16}(s_i + s_{i+1}) - \frac{1}{16}(s_{i-1} + s_{i+2}) + \frac{1}{2} \right\rfloor && \text{(A.9)} \\ s_i &\leftarrow s_i + \left\lfloor \frac{9}{32}(d_{i-1} + d_i) - \frac{1}{32}(d_{i-2} + d_{i+1}) + \frac{1}{2} \right\rfloor && \text{(A.10)} \end{aligned}$$

**The (2+2,2) wavelet** [56]

$$\begin{aligned} d_i &\leftarrow d_i - \left\lfloor \frac{9}{16}(s_i + s_{i+1}) - \frac{1}{16}(s_{i-1} + s_{i+2}) + \frac{1}{2} \right\rfloor && \text{(A.11)} \\ s_i &\leftarrow s_i + \left\lfloor \frac{9}{32}(d_{i-1} + d_i) - \frac{1}{32}(d_{i-2} + d_{i+1}) + \frac{1}{2} \right\rfloor && \text{(A.12)} \\ d_i &\leftarrow d_i - \Big\lfloor \alpha\,(-\frac{1}{2}\,s_{i-1} + s_i - \frac{1}{2}\,s_{i+1}) \\ &\qquad +\beta\,(-\frac{1}{2}\,s_i + s_{i+1} - \frac{1}{2}\,s_{i+2}) + \gamma\,d_{i+1} + \frac{1}{2} \Big\rfloor && \text{(A.13)} \\ &\qquad \textit{where, } \alpha = \beta = \frac{1}{8} \textit{ and } \gamma = 0. \end{aligned}$$

**The (2,4) wavelet** [56]

$$\begin{aligned} d_i &\leftarrow d_i - \left\lfloor \frac{1}{2}(s_i + s_{i+1}) + \frac{1}{2} \right\rfloor && \text{(A.14)} \\ s_i &\leftarrow s_i + \left\lfloor \frac{9}{32}(d_{i-1} + d_i) - \frac{1}{32}(d_{i-2} + d_{i+1}) + \frac{1}{2} \right\rfloor && \text{(A.15)} \end{aligned}$$

**The (6,2) wavelet** [56]

$$\begin{aligned} d_i &\leftarrow d_i - \Big\lfloor \frac{75}{128}(s_i + s_{i+1}) - \frac{25}{256}(s_{i-1} + s_{i+2}) \\ &\qquad + \frac{3}{256}(s_{i-2} + s_{i+3}) + \frac{1}{2} \Big\rfloor && \text{(A.16)} \\ s_i &\leftarrow s_i + \left\lfloor \frac{1}{4}(d_{i-1} + d_i) + \frac{1}{2} \right\rfloor && \text{(A.17)} \end{aligned}$$

**The Scalar and Prediction transform (S+P wavelet)** [63]

$$
\begin{aligned}
d_i &\leftarrow d_i - s_i & \text{(A.18)}\\
s_i &\leftarrow s_i + \left\lfloor \frac{1}{2} d_i + \frac{1}{2} \right\rfloor & \text{(A.19)}\\
d_i &\leftarrow d_i - \left\lfloor \frac{1}{8} (2(\triangle s_i + \triangle s_{i+1} - d_{i+1}) + \triangle s_{i+1}) + \frac{1}{2} \right\rfloor & \text{(A.20)}\\
& \quad where, \\
& \quad \triangle s_i = s_{i-1} - s_i
\end{aligned}
$$

The synthesis (inverse) lifting steps are obtained by reversing the operating order and the sign of the lifting steps.

# Appendix B

# Block Transform Matrices

The 2-, 4- and 8-point WHT, DCT and DST are tabulated below in their increasing sequency / frequency formats and permuted row-column formats that correspond to the Hadamard orders.

## B.1 Increasing sequency / frequency order

The basis functions for the 2-, 4- and 8-point WHT, DCT and DST, organised in increasing sequency / frequency are as below. This order is useful for arranging the transformed coefficients according to their significance.

### B.1.1 The Walsh Hadamard Transform (WHT)

The number of sign changes in a basis vector corresponds to the sequency number.

#### 2-point WHT

```
 0.7071   0.7071
 0.7071  -0.7071
```

### 4-point WHT

|  |  |  |  |
|---|---|---|---|
| 0.5000 | 0.5000 | 0.5000 | 0.5000 |
| 0.5000 | 0.5000 | -0.5000 | -0.5000 |
| 0.5000 | -0.5000 | -0.5000 | 0.5000 |
| 0.5000 | -0.5000 | 0.5000 | -0.5000 |

### 8-point WHT

|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 |
| 0.3536 | 0.3536 | 0.3536 | 0.3536 | -0.3536 | -0.3536 | -0.3536 | -0.3536 |
| 0.3536 | 0.3536 | -0.3536 | -0.3536 | -0.3536 | -0.3536 | 0.3536 | 0.3536 |
| 0.3536 | 0.3536 | -0.3536 | -0.3536 | 0.3536 | 0.3536 | -0.3536 | -0.3536 |
| 0.3536 | -0.3536 | -0.3536 | 0.3536 | 0.3536 | -0.3536 | -0.3536 | 0.3536 |
| 0.3536 | -0.3536 | -0.3536 | 0.3536 | -0.3536 | 0.3536 | 0.3536 | -0.3536 |
| 0.3536 | -0.3536 | 0.3536 | -0.3536 | -0.3536 | 0.3536 | -0.3536 | 0.3536 |
| 0.3536 | -0.3536 | 0.3536 | -0.3536 | 0.3536 | -0.3536 | 0.3536 | -0.3536 |

### B.1.2 The Discrete Cosine Transform (DCT)

The basis vectors are organised according to the increasing frequency order. Each row corresponds to the frequency index k=0,. . .,N-1.

### 2-point DCT

|  |  |
|---|---|
| 0.7071 | 0.7071 |
| 0.7071 | -0.7071 |

### 4-point DCT

|  |  |  |  |
|---|---|---|---|
| 0.5000 | 0.5000 | 0.5000 | 0.5000 |
| 0.6533 | 0.2706 | -0.2706 | -0.6533 |
| 0.5000 | -0.5000 | -0.5000 | 0.5000 |
| 0.2706 | -0.6533 | 0.6533 | -0.2706 |

**8-point DCT**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 |
| 0.4904 | 0.4157 | 0.2778 | 0.0975 | -0.0975 | -0.2778 | -0.4157 | -0.4904 |
| 0.4619 | 0.1913 | -0.1913 | -0.4619 | -0.4619 | -0.1913 | 0.1913 | 0.4619 |
| 0.4157 | -0.0975 | -0.4904 | -0.2778 | 0.2778 | 0.4904 | 0.0975 | -0.4157 |
| 0.3536 | -0.3536 | -0.3536 | 0.3536 | 0.3536 | -0.3536 | -0.3536 | 0.3536 |
| 0.2778 | -0.4904 | 0.0975 | 0.4157 | -0.4157 | -0.0975 | 0.4904 | -0.2778 |
| 0.1913 | -0.4619 | 0.4619 | -0.1913 | -0.1913 | 0.4619 | -0.4619 | 0.1913 |
| 0.0975 | -0.2778 | 0.4157 | -0.4904 | 0.4904 | -0.4157 | 0.2778 | -0.0975 |

### B.1.3 The Discrete Sine Transform (DST)

The basis vectors are organised according to the increasing frequency order. Each row corresponds to the frequency index k=1,. . .,N.

**2-point DST**

| | |
|---|---|
| 0.7071 | 0.7071 |
| 0.7071 | -0.7071 |

**4-point DST**

| | | | |
|---|---|---|---|
| 0.2706 | 0.6533 | 0.6533 | 0.2706 |
| 0.5000 | 0.5000 | -0.5000 | -0.5000 |
| 0.6533 | -0.2706 | -0.2706 | 0.6533 |
| 0.5000 | -0.5000 | 0.5000 | -0.5000 |

**8-point DST**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.0975 | 0.2778 | 0.4157 | 0.4904 | 0.4904 | 0.4157 | 0.2778 | 0.0975 |
| 0.1913 | 0.4619 | 0.4619 | 0.1913 | -0.1913 | -0.4619 | -0.4619 | -0.1913 |
| 0.2778 | 0.4904 | 0.0975 | -0.4157 | -0.4157 | 0.0975 | 0.4904 | 0.2778 |
| 0.3536 | 0.3536 | -0.3536 | -0.3536 | 0.3536 | 0.3536 | -0.3536 | -0.3536 |
| 0.4157 | 0.0975 | -0.4904 | 0.2778 | 0.2778 | -0.4904 | 0.0975 | 0.4157 |
| 0.4619 | -0.1913 | -0.1913 | 0.4619 | -0.4619 | 0.1913 | 0.1913 | -0.4619 |
| 0.4904 | -0.4157 | 0.2778 | -0.0975 | -0.0975 | 0.2778 | -0.4157 | 0.4904 |
| 0.3536 | -0.3536 | 0.3536 | -0.3536 | 0.3536 | -0.3536 | 0.3536 | -0.3536 |

## B.2 The Rearranged Transform Matrices

### B.2.1 Walsh Hadamard Transform (WHT)

#### 2-point WHT

| | |
|---|---|
| 0.7071 | 0.7071 |
| 0.7071 | -0.7071 |

#### 4-point WHT

| | | | |
|---|---|---|---|
| 0.5000 | 0.5000 | 0.5000 | 0.5000 |
| 0.5000 | -0.5000 | 0.5000 | -0.5000 |
| 0.5000 | 0.5000 | -0.5000 | -0.5000 |
| 0.5000 | -0.5000 | -0.5000 | 0.5000 |

#### 8-point WHT

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 |
| 0.3536 | -0.3536 | 0.3536 | -0.3536 | 0.3536 | -0.3536 | 0.3536 | -0.3536 |
| 0.3536 | 0.3536 | -0.3536 | -0.3536 | 0.3536 | 0.3536 | -0.3536 | -0.3536 |
| 0.3536 | -0.3536 | -0.3536 | 0.3536 | 0.3536 | -0.3536 | -0.3536 | 0.3536 |
| 0.3536 | 0.3536 | 0.3536 | 0.3536 | -0.3536 | -0.3536 | -0.3536 | -0.3536 |
| 0.3536 | -0.3536 | 0.3536 | -0.3536 | -0.3536 | 0.3536 | -0.3536 | 0.3536 |
| 0.3536 | 0.3536 | -0.3536 | -0.3536 | -0.3536 | -0.3536 | 0.3536 | 0.3536 |
| 0.3536 | -0.3536 | -0.3536 | 0.3536 | -0.3536 | 0.3536 | 0.3536 | -0.3536 |

#### The relationship

$$W_N = W_{\frac{N}{2}} \otimes W_2 \tag{B.1}$$

The basis vectors are organised to the Hadamard order.

### B.2.2 Discrete Cosine Transform (DCT)

The following tables show rearranged DCT matrices where the columns are arranged according to equation 3.37. The rows are arranged by grouping the odd indexed vectors to the lower half and the even indexed vectors to the upper half and recursively performing this on the upper half until N is 2.

#### 2-point DCT

| | |
|---|---|
| 0.7071 | 0.7071 |
| 0.7071 | -0.7071 |

#### 4-point DCT

| | | | |
|---|---|---|---|
| 0.5000 | 0.5000 | 0.5000 | 0.5000 |
| 0.5000 | -0.5000 | 0.5000 | -0.5000 |
| 0.2706 | 0.6533 | -0.2706 | -0.6533 |
| 0.6533 | -0.2706 | -0.6533 | 0.2706 |

#### 8-point DCT

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 | 0.3536 |
| 0.3536 | -0.3536 | 0.3536 | -0.3536 | 0.3536 | -0.3536 | 0.3536 | -0.3536 |
| 0.1913 | 0.4619 | -0.1913 | -0.4619 | 0.1913 | 0.4619 | -0.1913 | -0.4619 |
| 0.4619 | -0.1913 | -0.4619 | 0.1913 | 0.4619 | -0.1913 | -0.4619 | 0.1913 |
| 0.0975 | 0.4157 | 0.4904 | 0.2778 | -0.0975 | -0.4157 | -0.4904 | -0.2778 |
| 0.4157 | -0.4904 | 0.2778 | 0.0975 | -0.4157 | 0.4904 | -0.2778 | -0.0975 |
| 0.4904 | 0.2778 | -0.0975 | -0.4157 | -0.4904 | -0.2778 | 0.0975 | 0.4157 |
| 0.2778 | 0.0975 | -0.4157 | 0.4904 | -0.2778 | -0.0975 | 0.4157 | -0.4904 |

### B.2.3 Discrete Sine Transform (DST)

The following tables show rearranged DCT matrices where the columns are arranged according to equation 3.70. The rows are arranged by grouping the odd indexed vectors to the lower half and the even indexed vectors to the upper half and recursively performing this on the upper half until N is 2.

#### 2-point DST

| | |
|---|---|
| 0.7071 | -0.7071 |
| 0.7071 | 0.7071 |

#### 4-point DST

| | | | |
|---|---|---|---|
| 0.5000 | 0.5000 | -0.5000 | -0.5000 |
| 0.5000 | -0.5000 | -0.5000 | 0.5000 |
| 0.2706 | 0.6533 | 0.2706 | 0.6533 |
| 0.6533 | -0.2706 | 0.6533 | -0.2706 |

#### 8-point DST

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.3536 | 0.3536 | 0.3536 | 0.3536 | -0.3536 | -0.3536 | -0.3536 | -0.3536 |
| 0.3536 | -0.3536 | 0.3536 | -0.3536 | -0.3536 | 0.3536 | -0.3536 | 0.3536 |
| 0.1913 | 0.4619 | -0.1913 | -0.4619 | -0.1913 | -0.4619 | 0.1913 | 0.4619 |
| 0.4619 | -0.1913 | -0.4619 | 0.1913 | -0.4619 | 0.1913 | 0.4619 | -0.1913 |
| 0.0975 | 0.4157 | 0.4904 | 0.2778 | 0.0975 | 0.4157 | 0.4904 | 0.2778 |
| 0.4157 | -0.4904 | 0.2778 | 0.0975 | 0.4157 | -0.4904 | 0.2778 | 0.0975 |
| 0.4904 | 0.2778 | -0.0975 | -0.4157 | 0.4904 | 0.2778 | -0.0975 | -0.4157 |
| 0.2778 | 0.0975 | -0.4157 | 0.4904 | 0.2778 | 0.0975 | -0.4157 | 0.4904 |

# Appendix C

# Scanning Schemes Results

The weighted entropy values (in bpp) for the five scanning schemes, namely, Raster, Quadtree, Wavelet Tree - Zero tree(WT-ZT), wavelet tree - HVZ scan (WTHVZ) and Adpative Quadtree Splitting (AQS) using the (4,4) IWT coefficients of the test image set are shown in tables C.1-C.6. The best method for each bitplane is shown in bold font.

| WBP | Raster | Quadtree | WT-ZT | WTHVZ | AQS |
|---|---|---|---|---|---|
| 13 | 0.0008 | **0.0007** | 0.0008 | 0.0008 | 0.0008 |
| 12 | **0.0006** | **0.0006** | **0.0006** | 0.0010 | **0.0006** |
| 11 | 0.0010 | **0.0008** | 0.0009 | 0.0009 | 0.0010 |
| 10 | 0.0024 | 0.0022 | **0.0021** | 0.0032 | **0.0021** |
| 9 | 0.0077 | 0.0061 | 0.0062 | 0.0068 | **0.0055** |
| 8 | 0.0251 | 0.0196 | 0.0202 | 0.0235 | **0.0182** |
| 7 | 0.0797 | 0.0553 | 0.0593 | 0.0686 | **0.0512** |
| 6 | 0.1608 | 0.1272 | 0.1375 | 0.1621 | **0.1166** |
| 5 | 0.3140 | 0.2620 | 0.2855 | 0.2948 | **0.2384** |
| 4 | 0.4983 | 0.4899 | 0.5101 | 0.4910 | **0.4416** |
| 3 | 0.6553 | 0.7275 | 0.7388 | 0.6599 | **0.6519** |
| 2 | **0.5676** | 0.6429 | 0.6518 | 0.5730 | 0.5713 |
| 1 | **0.2990** | 0.3395 | 0.3199 | 0.3049 | **0.2990** |
| 0 | **0.0610** | 0.0698 | **0.0610** | 0.0665 | **0.0610** |
| $13 \cdots 0$ | 2.6734 | 2.7440 | 2.7948 | 2.6572 | **2.4592** |

Table C.1: zero-order entropy values (bpp) for different scans on each WBP for Gold Hill

| WBP | Raster | Quadtree | WT-ZT | WTHVZ | AQS |
|---|---|---|---|---|---|
| 13 | **0.0009** | 0.0010 | **0.0009** | **0.0009** | **0.0009** |
| 12 | **0.0005** | 0.0006 | **0.0005** | 0.0010 | **0.0005** |
| 11 | 0.0017 | 0.0013 | **0.0012** | 0.0014 | **0.0012** |
| 10 | 0.0045 | 0.0036 | 0.0032 | 0.0048 | **0.0031** |
| 9 | 0.0156 | 0.0115 | 0.0124 | 0.0172 | **0.0105** |
| 8 | 0.0520 | 0.0334 | 0.0404 | 0.0511 | **0.0312** |
| 7 | 0.1461 | 0.0985 | 0.1288 | 0.1561 | **0.0936** |
| 6 | 0.2447 | 0.1644 | 0.2135 | 0.2362 | **0.1512** |
| 5 | 0.3205 | 0.2424 | 0.2907 | 0.3233 | **0.2187** |
| 4 | 0.4238 | 0.3669 | 0.4031 | 0.4321 | **0.3267** |
| 3 | 0.5633 | 0.5906 | 0.6077 | 0.5702 | **0.5291** |
| 2 | **0.5914** | 0.6751 | 0.6841 | 0.5976 | 0.6048 |
| 1 | **0.3538** | 0.3966 | 0.3795 | 0.3593 | 0.3542 |
| 0 | **0.0714** | 0.0808 | **0.0714** | 0.0765 | **0.0714** |
| $13 \cdots 0$ | 2.7902 | 2.6667 | 2.8376 | 2.8275 | **2.3972** |

Table C.2: zero-order entropy values (bpp) for different scans on each WBP for Barbara1

| WBP | Raster | Quadtree | WT-ZT | WTHVZ | AQS |
|---|---|---|---|---|---|
| 13 | 0.0010 | **0.0009** | 0.0010 | 0.0010 | 0.0010 |
| 12 | 0.0004 | **0.0003** | 0.0004 | 0.0009 | 0.0004 |
| 11 | 0.0008 | **0.0007** | 0.0008 | 0.0008 | 0.0008 |
| 10 | 0.0023 | 0.0021 | 0.0019 | 0.0030 | **0.0018** |
| 9 | 0.0142 | 0.0105 | 0.0116 | 0.0162 | **0.0099** |
| 8 | 0.0574 | 0.0354 | 0.0446 | 0.0529 | **0.0334** |
| 7 | 0.1380 | 0.0944 | 0.1163 | 0.1447 | **0.0890** |
| 6 | 0.2528 | 0.1892 | 0.2126 | 0.2414 | **0.1743** |
| 5 | 0.3639 | 0.2937 | 0.3183 | 0.3524 | **0.2627** |
| 4 | 0.4694 | 0.4435 | 0.4464 | 0.4709 | **0.3929** |
| 3 | 0.5980 | 0.6666 | 0.6699 | 0.6023 | **0.5942** |
| 2 | **0.5537** | 0.6304 | 0.6537 | 0.5602 | 0.5616 |
| 1 | **0.2874** | 0.3254 | 0.3088 | 0.2923 | 0.2875 |
| 0 | **0.0517** | 0.0597 | **0.0517** | 0.0564 | **0.0517** |
| $13 \cdots 0$ | 2.7909 | 2.7529 | 2.8382 | 2.7953 | **2.4612** |

Table C.3: zero-order entropy values (bpp) for different scans on each WBP for Barbara2

| WBP | Raster | Quadtree | WT-ZT | WTHVZ | AQS |
|---|---|---|---|---|---|
| 13 | **0.0009** | **0.0009** | **0.0009** | **0.0009** | **0.0009** |
| 12 | 0.0009 | **0.0006** | 0.0008 | 0.0008 | 0.0009 |
| 11 | 0.0012 | **0.0007** | 0.0009 | 0.0010 | 0.0009 |
| 10 | 0.0033 | 0.0027 | 0.0028 | 0.0040 | **0.0026** |
| 9 | 0.0101 | 0.0090 | 0.0086 | 0.0125 | **0.0083** |
| 8 | 0.0407 | 0.0283 | 0.0295 | 0.0368 | **0.0264** |
| 7 | 0.0857 | 0.0598 | 0.0628 | 0.0762 | **0.0555** |
| 6 | 0.1420 | 0.1068 | 0.1150 | 0.1483 | **0.0974** |
| 5 | 0.2394 | 0.1819 | 0.1943 | 0.2303 | **0.1625** |
| 4 | 0.3532 | 0.2974 | 0.3062 | 0.3366 | **0.2624** |
| 3 | 0.5324 | 0.5461 | 0.5495 | 0.5385 | **0.4922** |
| 2 | **0.6703** | 0.7644 | 0.7897 | 0.6767 | 0.6947 |
| 1 | **0.4243** | 0.4714 | 0.4603 | 0.4305 | 0.4249 |
| 0 | **0.0831** | 0.0936 | **0.0831** | 0.0887 | **0.0831** |
| $13 \cdots 0$ | 2.5872 | 2.5637 | 2.6045 | 2.5819 | **2.3127** |

Table C.4: zero-order entropy values (bpp) for different scans on each WBP for Boats

| WBP | Raster | Quadtree | WT-ZT | WTHVZ | AQS |
|---|---|---|---|---|---|
| 13 | 0.0009 | **0.0007** | 0.0009 | 0.0009 | 0.0009 |
| 12 | 0.0009 | **0.0005** | 0.0008 | 0.0007 | 0.0009 |
| 11 | 0.0018 | **0.0013** | 0.0020 | 0.0018 | 0.0018 |
| 10 | 0.0038 | 0.0028 | 0.0027 | 0.0041 | **0.0026** |
| 9 | 0.0119 | 0.0084 | 0.0083 | 0.0123 | **0.0077** |
| 8 | 0.0354 | 0.0235 | 0.0249 | 0.0328 | **0.0222** |
| 7 | 0.0735 | 0.0474 | 0.0472 | 0.0670 | **0.0444** |
| 6 | 0.0953 | 0.0684 | 0.0684 | 0.1003 | **0.0645** |
| 5 | 0.1548 | 0.1220 | 0.1214 | 0.1637 | **0.1093** |
| 4 | 0.2704 | 0.2243 | 0.2274 | 0.2622 | **0.2001** |
| 3 | 0.5272 | 0.5580 | 0.5791 | 0.5327 | **0.5123** |
| 2 | **0.7278** | 0.8236 | 0.8752 | 0.7345 | 0.7542 |
| 1 | **0.4454** | 0.4926 | 0.4867 | 0.4511 | 0.4458 |
| 0 | **0.0827** | 0.0928 | **0.0827** | 0.0879 | **0.0827** |
| $13 \cdots 0$ | 2.4319 | 2.4663 | 2.5276 | 2.4521 | **2.2495** |

Table C.5: zero-order entropy values (bpp) for different scans on each WBP for Black board

| WBP | Raster | Quadtree | WT-ZT | WTHVZ | AQS |
|---|---|---|---|---|---|
| 13 | 0.0009 | **0.0008** | 0.0009 | 0.0009 | 0.0009 |
| 12 | 0.0007 | **0.0005** | 0.0006 | 0.0009 | 0.0007 |
| 11 | 0.0013 | 0.0010 | 0.0012 | 0.0012 | **0.0011** |
| 10 | 0.0033 | 0.0027 | 0.0025 | 0.0038 | **0.0024** |
| 9 | 0.0119 | 0.0091 | 0.0094 | 0.0130 | **0.0084** |
| 8 | 0.0421 | 0.0280 | 0.0319 | 0.0394 | **0.0263** |
| 7 | 0.1046 | 0.0711 | 0.0829 | 0.1025 | **0.0667** |
| 6 | 0.1791 | 0.1312 | 0.1494 | 0.1777 | **0.1208** |
| 5 | 0.2785 | 0.2204 | 0.2420 | 0.2729 | **0.1983** |
| 4 | 0.4030 | 0.3644 | 0.3786 | 0.3986 | **0.3247** |
| 3 | 0.5752 | 0.6178 | 0.6290 | 0.5807 | **0.5559** |
| 2 | **0.6222** | 0.7073 | 0.7309 | 0.6284 | 0.6373 |
| 1 | **0.3620** | 0.4051 | 0.3910 | 0.3676 | 0.3623 |
| 0 | **0.0700** | 0.0793 | **0.0700** | 0.0752 | **0.0700** |
| $13 \cdots 0$ | 2.6547 | 2.6387 | 2.7205 | 2.6628 | **2.3760** |

Table C.6: Average zero-order entropy values (bpp) for different scans on each WBP for the image set

# Appendix D

# Author's Publications

## In Refereed Conference Proccedings

1. "Incorporating near-lossless quantisation into lifting steps for near-lossless image coding", G.C.K. Abhayaratne and D.M. Monro, *Submitted to ACM Multimedia 2002 - will be held in December 2002.*

2. "Embedded to Lossless Image Coding (ELIC)", G.C.K. Abhayaratne and D.M. Monro, *Proc. IEEE Nordic Signal Processing Symposium (NORSIG 2000), pp. 255-258, Kolmården, Sweden, 13-15 June 2000.*

3. "Embedded to lossless coding of motion compensated prediction residuals in lossless video coding", G.C.K. Abhayaratne and D.M. Monro, *Visual Communications and Image Processing 2001, Proc. SPIE vol. 4310, pp. 175-185, San Jose, CA, 21-26 January 2001.*

## In Progress

1. An Integer version of the Walsh Hadamard Transform using lifting.

2. An Integer version of the Discrete Cosine Transform using lifting.

3. An Integer version of the Discrete Sine Transform using lifting.

4. Incorporating Near-lossless quantisation into lifting steps for near-lossless image coding.

# References

[1] M. Ghanbari, *Video coding : an introduction to standard codecs*, Number 42 in IEE telecommunications series. Institution of Electrical Engineers, London, 1999.

[2] J.L. Mitchell, W.B. Pennebaker, C.E. Fogg, and D.J. LeGall, *MPEG: Video Compression Standard*, Digital multimedia standard series. Chapman and Hall, New York and London, 1996.

[3] J. M. Martinez, "Overview of the mpeg-7 standard (version 6.0)," Tech. Rep. N4509, ISO/IEC JTC1/SC29/WG11, 2001.

[4] J. Bormans and K. Hill, "Mpeg-21 overview (version 3)," Tech. Rep. N4511, ISO/IEC JTC1/SC29/WG11, 2001.

[5] G.K. Wallace, "Jpeg: Still image compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, April 1991.

[6] G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley, MA, 2 edition, 1997.

[7] N. Ahmad, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, pp. 90–93, 1974.

[8] W.K. Pratt, *Digital Image Processing*, John Wiley and Sons, New York anf London, 1991.

[9] K.G. Beauchamp, *Transforms for Engineers: A Guide to Signal Processing*, Oxford Science Publications. Oxford University Press, Oxford, 1987.

[10] Y. Meyer and R. Ryan, *Wavelets: Algorithms and Applications*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991.

[11] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," in *Wavelet Applications in Signal and Image Processing III*, A. F. Laine and M. Unser, Eds. 1995, pp. 68–79, Proc. SPIE 2569.

[12] M. Antonini, M.Barlaud, P. Mathieu, and I. Dauberchies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 205–220, April 1992.

[13] J. Shapiro, "Embedded image coding using zero trees of wavelet coeffiecients," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.

[14] A. Said and W. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, June 1996.

[15] A. Zandi, J. D. Allan, E. L. Schwartz, and M. Boliek, "Crew: Compression with reversible embedded wavelets," in *Data Compression Conference*, Piscataway, NJ, 1995, pp. 212–221.

[16] R.W. Buccigrossi E.P.Simoncelli, "Image compression via joint statistical characterization in the wavelet domain.," Technical Report 414, Grasp Laboratory, University of Pennsylvania, 1998.

[17] I. Witten, R. Neal, and J. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, June 1987.

[18] M. Nelson and J. Gailly, *The Data Compression Book*, M & T Books, New York, NY, 2 edition, 1996.

[19] R. Clark, "Fcd 14495, lossless and near-lossless coding of continuous tone still images (jpeg-ls)," Public Draft FCD 14495, ISO / IEC JTC1 SC29 WG1 (JPEG / JBIG), 1998.

[20] M. J. Gormish, E. L. Schwartz, A. Keith, M. Boliek, and A.Zandi, "Lossless and nearly lossless compression for high quality images," in *very High resolution and Quality Imaging II*, 1997, vol. Proc. SPIE 3025, pp. 62–70.

[21] N. Jayant, J. Johnston, and R. Safranek, "Signal compression based on models of human perception," *Proceedings of the IEEE*, vol. 81, no. 10, pp. 1385–1421, Oct. 1993.

[22] W.E. Glenn, "Digital image compression based on visual perception," in *Digital Images and Human Vision*, A.B. Watson, Ed., pp. 63–72. The MIT Press, Cambridge, MA, 1993.

[23] A.P. Bradley, "A wavelet visible difference predictor," *IEEE Transactions on Image Processing*, vol. 8, no. 5, pp. 717–730, May 1999.

[24] P.J. Heartly, "Achieving and confirming optimum image quality," in *Digital Images and Human Vision*, A.B. Watson, Ed., pp. 149–162. The MIT Press, Cambridge, MA, 1993.

[25] S. Daly, "The visible differences predictor: An algorithm for the assessment of image quality," in *Digital Images and Human Vision*, A.B. Watson, Ed., pp. 179–206. The MIT Press, Cambridge, MA, 1993.

[26] CCIR, "Method for the subjective assessment of the quality of television pictures," Recommendation 500-4, CCIR, 1990.

[27] M. Miyahara, K. Kotani, and V.R. Algazi, "Objective picture quality scale (pqs) for image coding," *IEEE Transactions on Communications*, vol. 6, no. 9, pp. 1215–1232, Sept. 1998.

[28] A.B.Watson, "The cortex transform: Rapid computation of simulated neural images," *Computer Vision Graphics and Image Processing*, vol. 39, no. 3, pp. 311–327, Sept. 1987.

[29] A.B.Watson, "Efficiency of a model human image code," *Journal of Optical Soceity of America A*, vol. 4, no. 12, pp. 2401–2417, Dec. 1987.

[30] X. Wu, "Lossless compression of continuous-tone images via context selection, quantization, and modeling," *IEEE Transactions on Image Processing*, vol. 6, no. 5, pp. 656–664, May 1997.

[31] N. Memon and X. Wu, "Recent developments in context-based predictive techniques for lossless image compression.," *The Computer Journal*, vol. 40, no. 2/3, pp. 127–136, 1997.

[32] X. Wu and N. Memon, "Context based, adaptive, lossless image coding," *IEEE Transactions on Communications*, vol. 45, no. 4, pp. 437–444, April 1997.

[33] X. Wu and N. Memon, "Calic- a context based adaptive lossless image codec," in *International Conference on Acoustics, Speech and Signal Processing*, Piscataway, NJ, 1996, vol. 4, pp. 1890–1893.

[34] M.J. Weinberger, G. Seroussi, and G. Sapiro, "Loco-i: A low complexity, context based, lossless image compression algorithm," in *Data Compression Conference*, Snow Bird UT, 1996, vol. 1, pp. 141–150.

[35] N. Memon, "Adaptive coding of dct coefficients by golomb-rice codes," in *International Conference on Image Processing*, Los Alamitos, CA, 1998, vol. 1, pp. 516–520.

[36] M.J. Weinberger, G. Seroussi, and G. Sapiro, “Loco-i lossless image compression algorithm: Principles and standardization into jpeg-ls,” *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1309–1324, Aug. 2000.

[37] I. Tăbuş, J. Rissanen, and J. Astola, “Adaptive l-predictors based on finite state machine context selection,” in *International Conference on Image Processing*, Los Alamitos, CA, 1997, vol. 1, pp. 401–404.

[38] X. Xue and W. Gao, “Prediction based on backward adaptive recognition of local texture orientation and poisson statistical model for lossless/near-lossless image compression,” in *International Conference on Acoustics, Speech and Signal Processing*, Piscataway, NJ, 1999, vol. 6, pp. 3137–3140.

[39] B. Aiazzi, P. S. Alba, L. Alparone, and S. Baronti, “Lossless image compression based on a fuzzy linear prediction with context based entropy coding,” in *International Conference on Image Processing*, Los Alamitos, CA, 1998, vol. 3, pp. 905–909.

[40] G. Deng, “Symbol mapping and context filtering for lossless image compression,” in *International Conference on Image Processing*, Los Alamitos, CA, 1998, vol. 1, pp. 526–529.

[41] P. Roos, M.A. VierGever, M.C.A. Van Dijke, and J.H. Peters, “Reversible intraframe compression of medical images,” *IEEE Transactions on Medical Imaging*, vol. 7, no. 4, pp. 328–336, Dec. 1988.

[42] T. Endoh and Y. Yamazaki, “Progressive coding for multi level images,” in *Picture Coding Symposium*, Tokyo, 1986, pp. 21–22.

[43] M. Iwahashi, S. Fukuma, and N. Kambayashi, “Lossless coding of still images with four channel prediction,” in *International Conference on Image Processing*, Los Alamitos, CA, 1997, vol. 2, pp. 266–269.

[44] A. Abrardo, L. Alparone, and F. Batolini, “Encoding-interleaved hierarchical interpolation for lossless image compreession,” *Signal Processing*, vol. 56, no. 3, pp. 321–328, Feb. 1997.

[45] G. Deng, “A new interpolative sub band coding algorithm for lossless image compression,” in *International Conference on Image Processing*, Los Alamitos, CA, 1996, vol. 1, pp. 93–96.

[46] G. Deng, “An interpolative sub band coding algorithm for lossless image compression,” *Signal Processing : Image Communication*, vol. 14, no. 9, pp. 721–736, 1999.

[47] F. Golchin and K.K. Paliwal, "Minimum entropy clustering and its application to lossless image coding," in *International Conference on Image Processing*, Los Alamitos, CA, 1997, vol. 2, pp. 262–265.

[48] V. N. Ramaswamy, K. R. Namuduri, and N. Ranganathan, "Lossless image compression using wavelets and variable block size segmentation," in *IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis*, Piscataway, NJ, 1996, pp. 113–116.

[49] V. N. Ramaswamy, K. R. Namuduri, and N. Ranganathan, "Context modelling of wavelet coefficients in ezw-based lossless image coding," in *International Conference on Acoustics, Speech and Signal Processing*, Piscataway, NJ, 1999, vol. 6, pp. 3165–3168.

[50] V. N. Ramaswamy, K. R. Namuduri, and N. Ranganathan, "Performance analysis of wavelets in embedded zerotree-based lossless image coding schemes," in *International Conference on Image Processing*, Los Alamitos, CA, 1997, vol. 2, pp. 278–281.

[51] W. Philips and K. Denecker, "New embedded lossless/quasi-lossless image coder based on the hadamard transform," in *International Conference on Image Processing*, Los Alamitos, CA, 1997, vol. 1, pp. 667–670.

[52] W. Philips, K. Denecker, P. De Neve, and S. Van Assche, "Lossless quantization of hadamard transform coefficients," *IEEE Transactions on Image Processing*, vol. 9, no. 11, pp. 1995–1999, Sep. 2000.

[53] W. Philips, "The lossless dct for combined lossy / lossless image coding," in *International Conference on Image Processing*, Los Alamitos, CA, 1998, vol. 3, pp. 871–875.

[54] P.P. Vaidyanathan and P.-Q. Hoang, "Lattice structures for optimal design and robust implementation of two-band perfect reconstruction qmf banks," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 1, pp. 81–94, Jan. 1988.

[55] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, no. 3, pp. 245–267, 1998.

[56] 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. 332–369, 1998.

[57] W. Sweldens, "Wavelets and the lifting scheme: A 5 minute tour," *Z. Angew. Math. Mech.*, vol. 76 (Suppl. 2), pp. 41–44, 1996.

[58] R.E. Blahut, *Fast Algorithms for Digital signal processing*, Addison Wesley, Reading MA, 1984.

[59] G. Uytterhoeven, F. Van Wulpen, M. Jansen, D. Roose, and A. Bultheel, "Waili: Wavelets with integer lifting," Report TW262, Katholieke Universiteit Leuven, July 1997.

[60] G. Uytterhoeven, D. Roose, and A. Bultheel, "Wavelet transforms using lifting scheme," Report ITA Wavelets- WP 1.1, Katholieke Universiteit Leuven, Apr. 1997.

[61] A. Said and W. A. Pearlman, "Reversible image compression via multiresolution representation and predictive coding," in *Visual Communications and Image Processing*, 1993, vol. Proc. SPIE. 2094, pp. 664–674.

[62] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Losless image compression using integer to integer wavelet transforms," in *International Conference on Image Processing (ICIP), Vol. I.* 1997, pp. 596–599, IEEE Press.

[63] A. Said and W. A. Pearlman, "An image multiresolution representation for lossless and lossy compression," *IEEE Transactions on Image Processing*, vol. 5, no. 9, pp. 1303–1310, Sep. 1996.

[64] N. Memeon, X. Wu, and B. L. Yeo, "Improved techniques for lossless image compression with reversible integer wavelet transforms," in *International Conference on Image Processing*, Los Alamitos, CA, 1998, vol. 3, pp. 891–895.

[65] G.A. Triantafyllidis and M.G. Strintzis, "A context based adaptive arithmetic coding technique for lossless image compression," *IEEE Signal Processing Letters*, vol. 6, no. 7, pp. 168–170, July 1999.

[66] J Jiang and C. Grecos, "Adding rate control to jpeg-ls," *Journal of Electronic Imaging*, vol. 10, no. 3, pp. 727–734, July 2001.

[67] L. Ke and M.W. Marcellin, "Near-lossless image compression: Minimum-entropy contrained-error dpcm," *IEEE Transactions on Image Processing*, vol. 7, no. 2, pp. 225–228, Feb. 1998.

[68] N. Moayeri, "A near-lossless trellis-searched predictive image compression," in *International Conference on Image Processing*, Los Alamitos, CA, 1996, vol. 2, pp. 93–96.

[69] R. Iordache, I. Tăbuş, and J. Astola, "Fixed-slope near-lossless context-based image compression," in *International Conference on Image Processing*, Los Alamitos, CA, 1998, vol. 1, pp. 512–515.

[70] R. Ansari, N. Memon, and E. Ceran, "Near-lossless image compression techniques," *Journal of Electronic Imaging*, vol. 7, no. 3, pp. 486–494, July 1998.

[71] I. Avcibaş and N. Memon, "Lossless and near-lossless image compression with successive refinement," in *Visual Communications and Image Processing.* 20001, pp. 41–52, Proc. SPIE 4310.

[72] N.D. Memon and K. Sayood, "Lossless compression of video sequences," *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1340–1345, Oct. 1996.

[73] X. Wu, W. Choi, and N. Memon, "Lossless interframe image compression via context modeling," in *Data Compression Conference*, Piscataway, NJ, 1998, pp. 378–387.

[74] B. Aiazzi, P.S. Alba, S. Baronti, and L. Alparone, "Three dimensional lossless compression based on a separable generalised recursive interpolation," in *International Conference on Image Processing*, Los Alamitos, CA, 1996, vol. 1, pp. 85–88.

[75] R. Oami and M. Ohta, "Efficient lossless video coding compatible with mpeg-2," in *IEEE International Conference on Communications, ICC.*, Piscataway, NJ, 1998, vol. 2, pp. 901–905.

[76] G. Fernández, S. Periaswamy, and Wim Sweldens, "LIFTPACK: A software package for wavelet transforms using lifting," in *Wavelet Applications in Signal and Image Processing IV*, M. Unser, A. Aldroubi, and A. F. Laine, Eds. 1996, pp. 396–408, Proc. SPIE 2825.

[77] C.M. Brislawn, "Classification of non-expansive symmetric extension transforms for multirate filter banks," *Applied and Computational Harmonic Analysis*, vol. 3, pp. 337–357, 1996.

[78] K.G. Beauchamp, *Walsh functions and their applications*, Academic Press, London, 1975.

[79] K. G. Beauchamp, *Applications of Walsh and related functions: with an introduction to sequency theory*, Microelectronics and signal processing. London : Academic Press, London, 1984.

[80] Fons A. M. Bruekers and Ad W. M. van den Enden, "New networks for perfect inversion and perfect reconstruction," *IEEE Journal on Selected areas in Communications*, vol. 10, no. 1, pp. 130–137, Jan. 1992.

[81] P. Yip, *Sine and Cosine Transforms*, Number 1 in Electrical engineering handbook series. IEEE Press and CRC Press, Florida, 1995.

[82] W.H.Chen, C.H.Smith, and S.C. Fralick, "A fast computational algorithm for the dct," *IEEE Transactions on Communications*, vol. COM-25, pp. 1004–1008, Sep. 1977.

[83] B.G.Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-32, pp. 1243–1245, Dec. 1984.

[84] M. Ghanbari and D. E. pearson, "Fast cosine transform implementation for television signals," *IEE Proceedings-F*, vol. 129, no. 1, pp. 59–68, Feb. 1982.

[85] B.G. Sherlock and D.M. Monro, "Algorithm 749: Fast discrete cosine transform," *ACM Transactions on Mathematical Software*, vol. 21, no. 4, pp. 372–378, December 1995.

[86] M. Vetterli and H. Nussbaumer, "Simple fft and dct algorithms with reduced number of operations," *Signal Processing*, vol. 6, no. 4, pp. 267–278, Aug. 1984.

[87] B. D. Tseng and W. C. Miller, "On computing the discrete cosine transform," *IEEE Transactions on Computers*, vol. C-27, pp. 966–968, Oct. 1978.

[88] S-C. Chan and K-L. Ho, "Fast algorithms for computing the discrete cosine transform," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 39, no. 3, pp. 185–190, March 1992.

[89] D. Hein and N. Ahmed, "On a real time walsh hadamard cosine transform image processor," *IEEE Transactions on Electromagnetic Compatability*, vol. EMC-20, pp. 453–457, Aug. 1978.

[90] H. Malvar, "Fast computation of discrete cosine transform through fast hartley transform," *Electronic Letters*, vol. 22, no. 7, pp. 352–353, March 1986.

[91] H. S. Hou, "A fast recursive agorithm for computing the discrete cosine transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-35, no. 10, pp. 1455–1461, Oct. 1987.

[92] Z. Cvetković and M. V. Popović, "New fast recursive agorithms for the computation of discrete cosine and sine transforms," *IEEE Transactions on Signal Processing*, vol. 40, no. 8, pp. 2083–2086, Aug. 1992.

[93] W. Li, "A new agorithm to compute the dct and its inverse," *IEEE Transactions on Signal Processing*, vol. 39, no. 6, pp. 1305–1313, June 1991.

[94] Y-J. Chen, S. Oraintara, and T. Nguyen, "Video compression using integer dct," in *International Conference on Image Processing*, Vancouver, BC, Canada, 2000, vol. 2, pp. 844–847.

[95] A.K. Jain, “Fast karhunen-loève transform for a class of stochastic processes,” *IEEE Transactions on Communications*, vol. COM-24, pp. 1023–1029, Sept. 1976.

[96] P. Yip and K.R. Rao, “A fast computational algorithm for the discrete sine transform,” *IEEE Transactions on Communications*, vol. COM-28, no. 2, pp. 304–307, Feb. 1980.

[97] S. Cheng, “Application of the sine transform method in time-of-flight positron emmission image reconstruction algorithm,” *IEEE Transactions on Biomedical Engineering*, vol. BME-32, pp. 185–192, Mar. 1985.

[98] N.S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*, Prentice-Hall Signal Processing Series. Prentice-Hall, 1984.

[99] A. Gupta and K.R. Rao, “A fast recursive algorithm for the discrete sine transform,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 38, no. 3, pp. 553–557, March 1990.

[100] D.A.F. Florêncio and R.W. Schafer, “Perfect reconstructing non-linear filter banks,” in *International Conference on Acoustics, Speech and Signal Processing*, Piscataway, NJ, 1996, vol. 3, pp. 1814–1817.

[101] R. L. de Quiroz and D.A.F. Florêncio, “A pyramidal coder using a non-linear filter bank,” in *International Conference on Acoustics, Speech and Signal Processing*, Piscataway, NJ, 1996, vol. 3, pp. 1570–1573.

[102] F.J. Hampson and J.-C Pesqet, “A nonlinear subband decomposition with perfect reconstruction,” in *International Conference on Acoustics, Speech and Signal Processing*, Piscataway, NJ, 1996, vol. 3, pp. 1523–1526.

[103] H. Samet, *The design and analysis of spatial data structures*, Addison-Wesley, reading, MA, 1990.

[104] E. Nardelli and G. Proietti, “Probabilistic models for images and quadtrees: differences and equivalences,” *Image and vision computing*, vol. 17, no. 9, pp. 659–665, 1999.

[105] M.J. Weinberger, J.J. Rissanan, and R.B. Arps, “Applications of universal context modeling to lossless compression of gray-scale images,” *IEEE Transactions on Image Processing*, vol. 5, no. 4, pp. 575–586, Apr. 1996.

[106] X. Wu and J-H. Chen, “Context modeling and entropy coding of wavelet coefficients for image compression,” in *International Conference on Acoustics, Speech and Signal Processing*, Piscataway, NJ, 1997, vol. 4, pp. 3097–3100.

[107] X. Wu, “High-order context modelling and embedded conditional entropy coding of wavelet coefficients for image compression,” in *31st Asilomar Conference on Signals Systems and Computers*, IEEE Comp.Soc., Los Alamitos, CA, 1997, pp. 1378–1382.

[108] X. Wu, “Low complexity high-order context modeling of embedded wavelet bit streams,” in *Data Compression Conference*, Piscataway, NJ, 1999, pp. 112–120.

[109] X. Wu, “Context quantization with fisher discriminant for adaptive embedded wavelet image coding,” in *Data Compression Conference*, Piscataway, NJ, 1999, pp. 102–111.

[110] A. Deever and S.S. Hemami, “What is your sign?: Efficient sign coding for embedded wavelet image coding,” in *Data Compression Conference*, Piscataway, NJ, 2000, pp. 273–282.

[111] N.M. Rajpoot, F.G. Meyer, R.G. Wilson, and R.R. Coifman, “On zerotree quantization for embedded wavelet packet image coding,” in *International Conference on Image Processing*, Los Alamitos, CA, 1999, vol. 2, pp. 283–287.

[112] D. Blasiak and W-Y. Chan, “Efficient wavelet coding of motion compensated prediction residuals,” in *International Conference on Image Processing*, Chicago, IL, 1998, vol. 2, pp. 287–290.

[113] R. Claypoole, G. Davies, W. Sweldens, and R. Baraniuk, “Non-linear wavelet transforms for image coding using lifting,” in *31st Asilomar Conference on Signals, Systems and Computers*, IEEE Comp.Soc., Los Alamitos, CA, 1998, vol. 1, pp. 662–667.

[114] R. Claypoole, R. Baraniuk, and R Novak, “Adaptive wavelet transform via lifting,” in *International Conference on Acoustics, Speech and Signal Processing*, Piscataway, NJ, 1998, vol. 3, pp. 1513–1516.

[115] R. Claypoole, G. Davies, W. Sweldens, and R. Baraniuk, “Lifting for non-linear image processing,” in *Wavelet Applications in Signal and Image Processing VII*, 1999, vol. Proc. SPIE 3813, pp. 372–383.

[116] M. H. Hayes, *Statistical digital signal processing and modeling*, John Wiley and Sons, New York, 1996.

[117] S. Haykin, *Adaptive Filter Theory*, Prentice Hall Imformation Systems Science. Prentice Hall, Englewood Cliffs, NJ, 1986.

[118] J.A. Parker, R.V. Kenyon, and D.E. Troxel, “Comparison of interpolating methods for image resampling,” *IEEE Transactions on Medical Imaging*, vol. MI-2, no. 1, pp. 31–39, March 1983.