國立交通大學

電機與控制工程研究所

博　士　論　文

提昇型離散小波轉換之研究及其在影像壓縮中之應用

A STUDY OF LIFTING-BASED DISCRETE WAVELET TRANSFORM AND ITS APPLICATIONS IN IMAGE COMPRESSION

研 究 生：蘇建焜

指導教授：林昇甫 教授

中華民國九十七年七月

提昇型離散小波轉換之研究及其在影像壓縮中之應用

研究生：蘇建焜　　　　　　指導教授：林昇甫 教授

國立交通大學電機與控制工程研究所

摘　要

繼離散型餘弦轉換 (discrete cosine transform; DCT) 被廣泛應用於影像壓縮領域之後，離散型小波轉換 (discrete wavelet transform; DWT) 是目前學術界和工業界研究、發展的重點，並且已經被新一代的靜態影像壓縮標準 (如 JPEG2000) 和動態影像壓縮標準 (如 MPEG-4，H.264/AVC) 所採用。早期的離散型小波轉換其實現方式是基於迴旋計算 (convolution-based) 方式，而新一代的離散型小波轉換其實現方式是所謂提昇型(lifting-based) 。「提昇型離散小波轉換」 (lifting-based DWT)具有計算效率高、節省記憶空間、可執行整數至整數小波轉換、適合平行處理，以及在某些場合無邊界延伸的問題等優點。又因其使用上非常具有彈性，所以目前仍然是大家研究的重點。

本論文首先討論方向適應型離散小波轉換 (direction-adaptive DWT) ，然後提出應用於任意形狀物體影像的提昇式形狀適應型離散小波轉換 (lifting shape-adaptive DWT)，以及可以應用於任意形狀影像、且具方向適應性的形狀方向適應型離散小波轉換 (shape-direction-adaptive DWT；SDA-DWT)。由實驗結果得知本論文提出的新方法除了保有提昇型離散小波轉換的優點之外，因為具有形狀及方向上的適應性，故可以在有限的代價上獲得影像壓縮效能的顯著提昇。除此之外我們也討論結合提昇型離散小波轉換 (lifting-based DWT) 和向量量化 (vector quantization; VQ) 來改善影像壓縮效率的一些方法。

A STUDY OF LIFTING-BASED DISCRETE WAVELET TRANSFORM AND ITS APPLICATIONS IN IMAGE COMPRESSION

Student: Chien-Kun Su Advisor: Prof. Sheng-Fuu Lin

Department of Electrical and Control Engineering
National Chiao Tung University

Abstract

After the discrete cosine transform (DCT) was widely used in image compression, discrete wavelet transform (DWT) was the new dominated transform for research and industrial applications, and it has been adopted in the new still-image compression standards (e.g. JPEG2000) and video compression standards (e.g. MPEG-4 and H.264/AVC). In literature, the realization of DWT was convolution-based in early days, and for better computation efficiency, saving memory space, integer-to-integer transform, parallel processing, and boundary extension problem the lifting-based DWT was proposed later. Lifting-based DWT is still an important research topic, since it is more flexible than the convolution-based DWT to adopt new functionalities.

In this dissertation, the lifting-based DWT was studied, and a lifting-based direction-adaptive DWT was discussed. Then, the lifting-based shape-adaptive DWT (LSA-DWT) and the lifting-based shape-direction-adaptive DWT, which was adaptive in shape and direction, were proposed. Because SDA-DWT has the adaptabilities of shape and direction, by paying affordable cost, we can achieve superior improvement in image compression. Beside lifting-based SA-DWT and SDA-DWT, the combination of lifting-based DWT and vector quantization (VQ) was discussed in the dissertation.

誌　謝

感謝指導教授及師母多年來的諄諄教誨，謝謝口試委員在盛夏溽暑百忙中撥冗為我口試。謝謝所有電控系、所和交大其他系、所、單位，直接或間接教導、幫助過我的師長和小姐、先生們，也要謝謝辛錫進教授（國立聯合大學）和高川原教授（中華大學）的協助，以及實驗室內歷屆學長和學弟的幫忙。同時感謝父母的照顧和太太的包容，還有小女、小犬對我精神上的激勵。引用陳之藩先生的話：「太多人要感謝，不如就謝天吧！」

CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOL LIST

$\mathbf{x}$	: input vector, original signal
$x(n)$	: discrete signal sample of $\mathbf{x}$
$\hat{\mathbf{x}}$	: reconstruction signal of $\mathbf{x}$
A^t	: transpose of matrix A
A^{-1}	: inverse of matrix A
$det(A)$	: determinant of matrix A
$\mathbf{R}$	: field of real numbers, vector space of real numbers
$\mathbf{Z}$	: field of integers, vector space of integers
$\mathrm{GL}(2, \mathbf{R}[z, z^{-1}])$	: mathematical ring whose elements are 2×2 matrices with Laurent polynomial entries
$\mathrm{SL}(2, \mathbf{R}[z, z^{-1}])$	: mathematical ring whose elements are 2×2 matrices with determinant 1 and Laurent polynomial entries
h, $\tilde{h}$	: low-pass FIR filters
g, $\tilde{g}$	: high-pass FIR filters
$q(z) = a(z)/b(z)$	: quotient of $a(z)$ divided by $b(z)$

$r(z) = a(z)\ mod\ b(z)$	: remainder of $a(z)$ divided by $b(z)$
$\lvert a(z) \rvert$	: degree of Laurent polynomial $a(z)$
bpp	: bit-per-pixel (bits used to represent one pixel)
$RMSE$	: root-mean-square error
$PSNR$ (dB) = $20log_{10}(255/RMSE)$	: signal to noise power ratio
$\downarrow 2$	: downsampling (subsampling) operation
$\uparrow 2$	: upsampling operation

CHAPTER 1

INTRODUCTION

Discrete wavelet transform (DWT) are widely and successfully used in many fields, especially for image compression. Since more and more new applications and requirements are emerging, some variants of discrete wavelet transform have been designed to support new functionalities of these new applications. Shape-adaptive functionality is required in object-based image compression (e.g. MPEG-4), and Li *et al.* [1] and Lu *et al.* [2] had proposed to use convention DWTs to solve this problem. Recently, the direction-adaptive functionality of DWT was discussed by Ding *et al.* [3] and Chang *et al.* [4], and they both use lifting-based structures to design DWTs which are directional adaptive and achieve very efficient results. Those instances inspired us to develop a novel method that is both shape and directional adaptive, and to well exploit the correlation of images to achieve better performance of image compression.

In this dissertation, a shape-direction-adaptive lifting-based discrete wavelet transform (SDA-DWT), which is direction adaptive and can be used for arbitrarily shaped segments, is proposed. The SDA-DWT contains three major techniques: the lifting-based DWT, the adaptive directional technique, and the concept of object-based compression in MPEG-4. The conventional separable 2-D DWT can be implemented by using 1-D DWT on the horizontal and vertical directions, respectively. Therefore, for images containing large amount of non-horizontal and non-vertical line textures, the conventional DWT is not efficient for image coding, and the direction-adaptive DWT (DA-DWT) can improve the performance for such cases. On the other hand, the traditional 2-D DWT also requires the images that are going to be transformed to be rectangular and their width and height are multiples of two. Such a requirement of the conventional DWT confines its applications for arbitrarily-shaped-region or object-image compression. Hence, some shape-adaptive DWTs have been proposed for solving this problem. The lifting-based DWT implements the DWT by factoring it into three lifting steps and the lifting technique simplifies hardware implementation and SDA-DWT realization. For supporting the shape-adaptive and direction-adaptive functionalities at the same time, we propose the new SDA-DWT which can handle arbitrarily shaped still images and are directional adaptive. SDA-DWT can improve energy compaction for any shaped region containing sharp edges or line-type textures, and improve the overall coding efficiency. For the application in object-based image compression, compared to the shape-adaptive DWTs proposed in [1] and [2], SDA-DWT has the advantages that it can well exploit the orientation correlation in images and it is easier to implement for using the lifting DWT. The disadvantages of SDA-DWT include that SDA-DWT needs more side information than the shape-adaptive DWTs need, and it needs extra computation for direction decision. For the application in normal size image compression, compared to direction-adaptive DWTs, SDA-DWT can handle arbitrarily shaped partition and have higher resolution to exploit the correlation hiding in shape and orientation, while the direction-adaptive DWTs only can process rectangular partition.

1.1 Discrete Wavelet Transform

Wavelet transform [5]-[8] is well known as a multiresolution analysis that provides many advantages: joint space-spatial frequency localization, clustered wavelet coefficients of significance with strong correlations between subbands, and exact reconstruction, which are truly beneficial to image compression. Discrete wavelet transform (DWT) decomposes a signal: $S_\ell(n)$ at resolution ℓ into two components:

$$S_{\ell+1}(n) = \sum_k S_\ell(k)\,\tilde{h}(2n-k)\,, \tag{1.1a}$$

$$D_{\ell+1}(n) = \sum_k S_\ell(k)\,\tilde{g}(2n-k), \tag{1.1b}$$

where $S_{\ell+1}(n)$ is its approximation (lowpass signal) at the next coarser resolution $\ell+1$, $D_{\ell+1}(n)$ is the detail information (highpass signal) between the two successive resolutions: ℓ and $\ell+1$, $\tilde{h}(n) = <\phi, \quad \phi_{-1,-n}>$, $\tilde{g}(n) = <\psi, \quad \phi_{-1,-n}>$, $<\cdot, \quad \cdot>$ is an inner product operator, ψ is a valid (mother) wavelet, ϕ is the scaling function that is an aggregation of wavelets at scales larger than 1, and $\phi_{-j,-n}(x) = 2^{-j/2}\phi(2^{-j}x-n)$. The original signal $S_{\ell}(n)$ can be exactly reconstructed from $S_{\ell+1}(n)$ and $D_{\ell+1}(n)$ by using the following inverse DWT (IDWT):

$$S_{\ell}(n) = \sum_{k} S_{\ell+1}(k)\, h(n-2k) + \sum_{k} D_{\ell+1}(k)\, g(n-2k), \tag{1.2}$$

where $h(n) = \tilde{h}(-n)$ and $g(n) = \tilde{g}(-n)$.

For image applications, the two-dimensional DWT can be obtained by using the tensor product of two one-dimensional DWT, i.e. the row processing followed by column processing, or vice versa. Figure 1.1 shows a 3-level, 2-D DWT in a pyramid structure. Where, HL_{ℓ}, LH_{ℓ} and HH_{ℓ} are the wavelet subbands composed of the wavelet coefficients $D_{\ell}^{1}(m,n)$, $D_{\ell}^{2}(m,n)$ and $D_{\ell}^{3}(m,n)$, representing the detail information at resolution ℓ in the horizontal, vertical and diagonal directions, respectively, LL_3 is composed of the scaling coefficients $S_3(m,n)$ representing the approximation at the coarsest resolution 3, and the original image is usually considered the scaling coefficients $S_0(m,n)$ at the finest resolution 0. $S_{\ell}(m,n)$ can be decomposed into $S_{\ell+1}(m,n)$, $D_{\ell+1}^{1}(m,n)$, $D_{\ell+1}^{2}(m,n)$ and $D_{\ell+1}^{3}(m,n)$ by using the 2-D DWT. And, the 2-D IDWT obtained by using the tensor product of two 1-D IDWT exactly reconstructs $S_{\ell}(m,n)$ from $S_{\ell+1}(m,n)$, $D_{\ell+1}^{1}(m,n)$, $D_{\ell+1}^{2}(m,n)$ and $D_{\ell+1}^{3}(m,n)$.

In wavelet domain, an image is decomposed into subbands with orientation selectivity. Wavelet coefficients taken from all the subbands of the same orientation are rearranged to form the wavelet trees. The tree hierarchy is based on the resolution level. The wavelet coefficients at coarse resolution are called parent nodes, each of which has four children nodes at the next finer resolution. Tree roots are at the coarsest resolution, and tree leaves are at the finest resolution. Figure 1.2 shows a wavelet tree in the diagonal direction. Many natural images are composed of large portions of homogeneous regions, textures, together with a small portion of edges, which are typically the low, middle and high frequency components, respectively. The significant wavelet coefficients of the homogeneous regions are usually at the coarser resolutions, i.e. in the lower frequency subbands, while those near the noticeable edges are usually clustered in the higher frequency subbands with strong similarities across subbands. If a non-leave node is insignificant, then all the descendants at the finer resolutions are likely to be insignificant. This cross-subband dependency of wavelet coefficients can be exploited to improve the image compression performance.

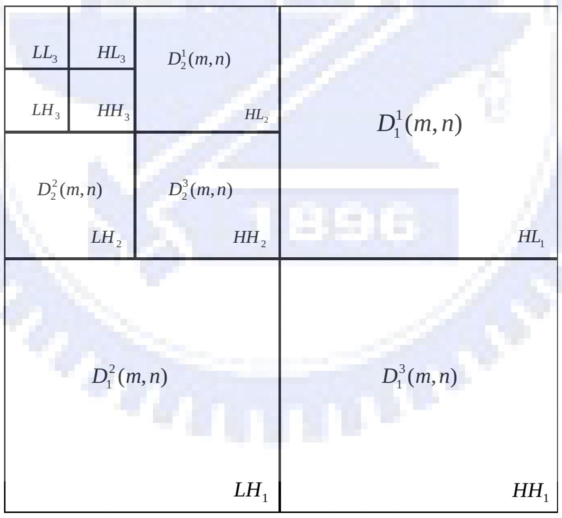


Figure 1.1 Example of 3-level 2-D DWT with subbands delimited by thick lines.

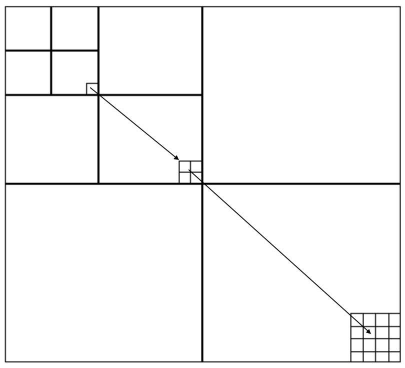


Figure 1.2 Example of 3-level 2-D DWT and a wavelet tree in the diagonal direction.

1.2 The Applications and Limitations of Conventional DWT

The concept of wavelet transform was discovered by mathematicians more than one hundred years ago [9][10], and the applications of wavelets were developed independently in many fields such as mathematics, quantum physics, seismic geology, and electrical engineering. Exchanging ideas among these fields, during the past two decades, have led to many novel wavelet applications, for example, molecular dynamics, ab initio calculations, density-matrix localization, seismic geophysics, optics, quantum and turbulence mechanics, image processing, speech recognition, general signal processing, multifractal analysis, DNA analysis, protein analysis, blood-pressure, ECG and heart-rate analyses. The wavelet transform is usually compared with the Fourier transform [11], since most persons are more familiar with the Fourier transform than the wavelet transform. Hence, in most of the applications of wavelet transforms, people directly replaced the conventional Fourier transform with wavelet transforms in a large amount of applications which were originally Fourier-transform-based. In the dissertation, we focus on image compression applications.

The discrete wavelet transforms implemented by using Eq. (1.1) are called convolution-based because of involving convolution computation. The conventional method for realizing DWT is to use the convolution-based or finite impulse response (FIR) filter bank structures. Compared to the block-based implementation of discrete cosine transform (DCT), DWT is essentially a frame-based realization. Generally speaking, a frame-based realization costs more computations and memory spaces than a block-based realization, and these two disadvantages limit the DWT for either high-speed or low-power image and video processing applications. Besides complexity and large storage space requirement, the convolution-based DWT is difficult for hardware implementation. Daubechies and Sweldens had proposed a new approach, called lifting-based DWT [12]-[14], for implementing DWT. The lifting-based scheme is to decompose a discrete wavelet transform into a finite sequence of simple filtering steps, which are called lifting steps. Using the language of algebraists, the decomposition of lifting-based DWT corresponds to a factorization of the polyphase matrix of the wavelet into elementary matrices. The lifting-based approach can provide advantages such as in-place implementation of the fast DWT, capability of integer-to-integer transform, ease for hardware implementation, less storage space requirement, and flexibility for some adaptations on DWT. For the lifting structure, each finite impulse response (FIR) wavelet filter is factored into several pairs of lifting steps. One pair of lifting steps includes a prediction step followed by one update step.

In Chapter 2 we will discuss the lifting-based discrete wavelet transform which is the second generation discrete wavelet transform and also the foundation of the proposed method. The mathematical theories and realizations of the lifting-based DWT are discussed in detail. Two potential applications which are originally two wavelet-based hybrid coders are given in Chapter 3. The proposed methods of lifting-based DWTs are discussed in Chapter 4, in which lifting shape-adaptive DWT and the shape-direction adaptive DWT are introduced. Chapter 5 includes the experimental results of the proposed SDA-DWT in object image compression and regular still image compression. Finally, conclusions are given in Chapter 6.

CHAPTER 2

LIFTING-BASED DISCRETE WAVELET TRANSFORM

Lifting-based DWT is a very flexible method for implementing DWTs, and it makes the proposed method, that will be discussed in Chapter 4, easy to adopt functionalities such as shape-adaptive and directional-adaptive abilities. In this chapter, the mathematical theories of lifting-based DWTs will be thoroughly discussed in Section 2.1, and some examples, including well-known Harr, 5/3, and 9/7 wavelet transforms, are discussed and implemented in Section 2.2. Lifting direction-adaptive DWT and shape-adaptive conventional DWT are discussed in Sections 2.3 and 2.4, respectively.

A discrete wavelet transform, whose high frequency and low frequency filters are complementary FIR filters, can be represented as factorization of lower triangular and upper triangular matrices. Each elementary matrix (upper triangular or lower triangular) is related to a lifting step. Since the operations in a lifting step can be executed parallel, the lifting DWTs are more efficient than the traditional convolution-based DWTs. Generally speaking, using lifting scheme to implement DWT can reduce about 50% computation time of the corresponding convolution DWT [12].

2.1 Mathematical Theories of Lifting-Based DWT

In the mid-eighties Mallat and Meyer proposed the multiresolution analysis [15]-[17] and the fast wavelet transform which connected subband filters and wavelets, and the connection led to new constructions, for example, the smooth orthogonal and compactly supported wavelets. Soon after that, a lot of generalizations to the biorthogonal or semiorthogonal (pre-wavelet) case were proposed, and, then, symmetric wavelets and linear phase filters were able to be constructed.

There are some different techniques to construct wavelet bases, or to factor existing wavelet filters into basic building blocks. Lifting is one of these. The original motivation to develop lifting was for building second generation wavelets. Wavelets are usually classified into two generations. First generation wavelets are all translates and dilates of one or some basic waveforms. On the other hand, second generation wavelet are able to be adapted to situations that translation and dilation are not allowed (e.g. non-Euclidean spaces). Using lifting to construct wavelet bases is entirely spatial, so it is suited for building second generation wavelets. The lifting becomes famous ladder type structures and certain factoring algorithms when it is restricted to the translation and dilation invariant case. Some discussions on lifting-based DWTs from the important papers by Daubechies and Sweldens [12]-[13] are adopted in this dissertation.

Exploiting the correlation structure in signals and building sparse approximations are the basic concepts of wavelet transform. Since adjacent samples and frequencies are more correlated than those far apart, the correlation structure is usually local in space (time) and frequency. Fourier transform was used to build the space-frequency localization of conventional wavelet constructions. It can be shown that building the space-frequency localization can be achieved by the following simple example.

Assume that $\mathbf{x}$ is a one-dimensional signal which is defined as $\mathbf{x} = \{x_k \mid x_k \in \mathbf{R}, \quad k \in \mathbf{Z}\}$. Then, we split $\mathbf{x}$ into two disjoint subsets which are called polyphase components. One subset $\mathbf{x_o}$ contains all the odd samples of $\mathbf{x}$, and the other subset $\mathbf{x_e}$ includes all the even samples of $\mathbf{x}$. Since $\mathbf{x_o}$ and $\mathbf{x_e}$ are usually closely correlated, given anyone of them (say $\mathbf{x_e}$), we can build a good predictor P for the other set ($\mathbf{x_o}$). Using the idea of DPCM (differential pulse code modulation), we record the difference or detail **d**:

$$\mathbf{d} = \mathbf{x_o} - P(\mathbf{x_e}) \,. \tag{2.1}$$

This is because that we expect **d** is a sparse set, and the first entropy of **d** is smaller than $\mathbf{x_o}$'s. The operation that calculates a prediction and records the detail is called a lifting step. Some of the spatial correlation is exploited by the prediction steps, but we also need to get some separation in the frequency domain. Frequency separation could be done by applying another lifting step which uses an update operator U on the details to determine a smoothed values **s** and use it to replace $\mathbf{x_e}$:

$$\mathbf{s} = \mathbf{x_e} + U(\mathbf{d}). \tag{2.2}$$

It is trivial that $\mathbf{x_e}$ and $\mathbf{x_o}$ can be reconstructed by applying Eqs. (2.3) and (2.4):

$$\hat{\mathbf{x}}_\mathbf{e} = \mathbf{s} - U(\mathbf{d}), \tag{2.3}$$

$$\hat{\mathbf{x}}_\mathbf{o} = \mathbf{d} + P(\mathbf{x_e}), \tag{2.4}$$

where $\hat{\mathbf{x}}_\mathbf{e}$ and $\hat{\mathbf{x}}_\mathbf{o}$ are the reconstruction versions of $\mathbf{x_e}$ and $\mathbf{x_o}$, respectively**.** Hence, the scheme is always invertible and leads to critically sampled perfect reconstruction filter banks. Figure 2.1 shows the block diagram of a pair of lifting steps which contain a prediction and an update lifting steps. According to Figure 2.1, the following Eqs.:

$$x_{predict}(2k+1) = P(x(2k), x(2k+2)) = \tfrac{1}{2}[x(2k) + x(2k+2)], \tag{2.5a}$$

$$d(k) = x(2k+1) - x_{predict}(2k+1), \tag{2.5b}$$

$$U(d(k-1), d(k)) = \tfrac{1}{4}[d(k-1) + d(k)], \tag{2.6a}$$

$$s(k) = x(2k) + U(d(k), d(k-1)), \qquad (2.6b)$$

show a simple example of 5/3 lifting DWT, where Eqs. (2.5a) and (2.6a) are the prediction and update functions, respectively. The $x_{predict}(2k+1)$ is the prediction value of the odd sample $x(2k+1)$, and it is the average of two nearest even neighbors $x(2k)$ and $x(2k+1)$. The update-function output for an even sample (say $x(2k)$ is a quarter of the detail signal sum of its two odd neighbors (i.e. $d(k-1) + d(k+1)$). Because of downsampling, $d(k)$ and $d(k-1)$ are corresponded to the original $x(2k+1)$ and $x(2k-1)$, respectively. The reason to use Eq. (2.6a) as the update function is to keep the running average the same as the original **x**. In wavelet transform Eqs. (2.3)-(2.6) are corresponded to the biorthogonal (2, 2) wavelet transform [18]. Thus, this example implements the biorthogonal (2, 2) wavelet transform by using a pair of lifting steps. In other words, lifting structure provides a frame work which can construct certain biorthogonal wavelets which are able to be generalized to the second generation setting.

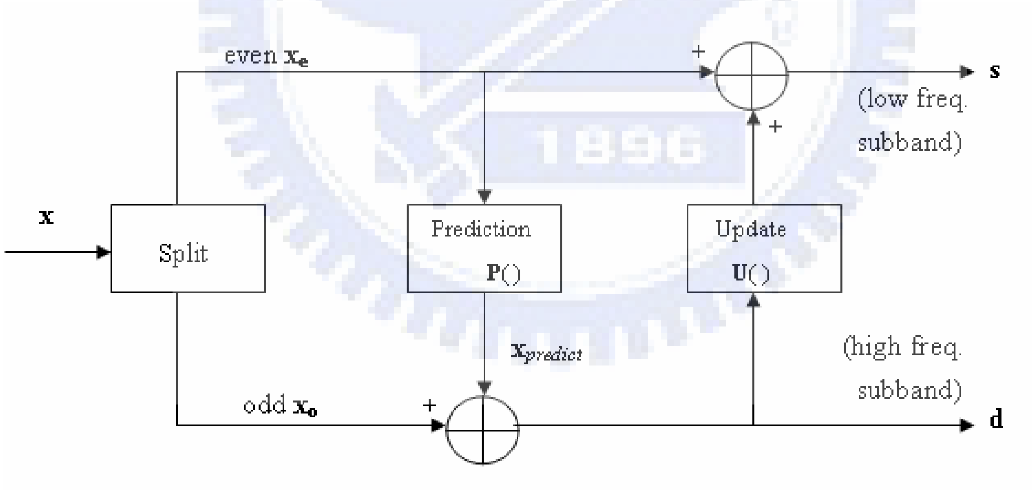


Figure 2.1 Block diagram of a pair of lifting steps (prediction and update steps). The **x** means the input signal vector, and **s** and **d** are the output subsampled smooth (low-pass) and detail (high-pass) signal vectors, respectively.

Representing the wavelet transform in the polyphase form, statements concerning perfect reconstruction can be made by using matrices with polynomial or Laurent polynomial entries. A lifting step is equivalent to an elementary matrix that is a triangular matrix with all diagonal entries equal to one, in matrix algebra. By the theorem in Matrix algebra, any matrix with polynomial entries and determinant one is able to be factored into elementary matrices. From above discussion, we can conclude that every FIR (finite impulse response) wavelet or filter bank can be decomposed into lifting steps.

2.1.1 Filters and Laurent polynomials

Strictly speaking, a (digital) filter [19] suggests a system which passes certain frequency components and totally rejects all others, but in a broader context any system which modifies certain frequencies corresponding to others is also called a filter. A filter is also a linear time-invariant operator which can be completely determined by its impulse response: $\{h(k) \mid h(k) \in \mathbf{R}, k \in \mathbf{Z}\}$. Based on the lengths of digital filters, digital filters are commonly classified into two categories which are finite-impulse-response (FIR) and infinite-impulse-response (IIR). The former (FIR) has finite number of non-zero filter coefficients, and the later (IIR) has infinite number of non-zero coefficients. Only FIR filters are discussed in the dissertation. For a linear time-invariant system, the relation among input **x**, system impulse response **h**, and the output **y** is Eq. (2.7):

$$y(n) = \sum_{k=-\infty}^{\infty} x(k) \cdot h(n-k) . \tag{2.7}$$

The most important advantage of IIR filters is that a variety of frequency-selective filters are able to designed using closed-form design formulas. Thus, once the design problem has been specified in terms appropriate for a given approximation method, then the order of the filter which will meet the specifications can be obtained by substitution into some design equations straightforwardly. This advantage makes it feasible to design an IIR filter by manual computation if necessary and it leads to straightforward non-iterative computer programs for IIR filter design. On the contrarily, FIR filters can have precisely linear phase, although the closed-form design equations do not exist for FIR filters.

A Laurent polynomial [20] with coefficients in the field **F** is an algebraic object typically expressed as that in Eq. (2.8),

$$\cdots + a_{-n}z^{-n} + a_{-(n-1)}z^{-(n-1)} + \cdots + a_0 + a_1 z + a_2 z^2 + \cdots + a_n z^t + \cdots , \tag{2.8}$$

where the coefficients a_i's are elements of **F** and the number of nonzero terms is finite. For example, the collection of Laurent polynomials with coefficients in a field **R** form a ring, denoted **R**[z, z^{-1}], with ring operations given by componentwise addition and multiplication according to Eqs. (2.9) and (2.10), respectively:

$$\left(\sum_k a(k)\cdot z^k\right) + \left(\sum_k b(k)\cdot z^k\right) = \sum_k [(a(k)+b(k)]\cdot z^k , \tag{2.9}$$

$$\left(\sum_k a(k)\cdot z^k\right)\cdot\left(\sum_k b(k)\cdot z^k\right) = \sum_k \left(\sum_{i,j:i+j=k} a(i)\cdot b(j)\right)\cdot z^k . \tag{2.10}$$

The equation (2.11):

$$h(z) = \sum_{k=k_l}^{k_u} h(k)\cdot z^{-k} , \tag{2.11}$$

shows the z-transform of a FIR filter h(z), where k_l and k_u are the smallest and largest k, respectively, for which $h(k)$ is not zero. Thus, Eq. (2.11) is a Laurent polynomial.

The degree of a Laurent polynomial $h(z)$ in Eq. (2.11) is defined as Eq. (2.12):

$$|h(z)| = k_u - k_l . \tag{2.12}$$

Hence, the length of a FIR filter is the degree of its corresponding Laurent polynomial plus one. By the above definition, z^n has degree zero when it is seen as a Laurent polynomial, but it has degree n when it is seen as a regular polynomial. Assume that $a(z)$ and $b(z)$ are two Laurent polynomials with $b(z) \neq 0$ and $|a(z)| \geqq |b(z)|$. Thus, there always exists two Laurent polynomials $q(z)$ and $r(z)$ with $|q(z)| = |a(z)| - |b(z)|$, such that $a(z) = b(z)q(z) + r(z)$. The Laurent polynomials $q(z)$ and $r(z)$ are the quotient and remainder respectively of the result that $a(z)$ is divided by $b(z)$, and they are denoted as :

$$q(z) = a(z)/b(z),$$

and

$$r(z) = a(z) \; mod \; b(z).$$

If $b(z)$ is a monomial, then $|b(z)| = 0$, $r(z) = 0$, and the division is exact. Any Laurent polynomial is invertible if and only if it is a monomial. Note that for regular polynomials, only constant polynomials are invertible. Another important property is that the long division of Laurent polynomials is not unique.

Example 2.1 Assuming $a(z) = z^{-1} + 6 + z$, $b(z) = 3 + 3z$, we want to determine the quotients $q(z)$ and remainders $r(z)$ of that $a(z)$ is divided by $b(z)$. Since the degrees of $a(z)$ and $b(z)$ are 2 and 1 respectively, the quotient $q(z)$ is a Laurent polynomial of degree one. The corresponding remainder can be determined by the relation: $r(z) = a(z) – b(z)q(z)$, and $b(z)q(z)$ has to match $a(z)$ in two terms.

If we choose $b(z)$ to match $a(z)$ with $z^{-1} + 6$, then $q(z) = \frac{1}{3}(z^{-1} + 5)$ and $r(z) = -4z$. The degree of the Laurent polynomial $r(z)$ is zero. However, if we let the two match terms be $z^{-1} + z$, the new answer is $q(z) = \frac{1}{3}(z^{-1} + 1)$ and $r(z) = 4$. Finally, if we select to match $6 + z$ in $a(z)$, then the third answer is $q(z) = \frac{1}{3}(5z^{-1} + 1)$ and $r(z) = -4z^{-1}$.

From the results of **Example 2.1**, we see the fact that the division of two Laurent polynomials is not unique, and $b(z)$ has to match $a(z)$ at least $|a(z)| - |b(z)| + 1$ terms. Each selection of $q(z)$ corresponds to a long division algorithm, and this will turn out to be useful later.

2.1.2 Discrete wavelet transform in FIR form

The one-dimensional discrete wavelet transform can be represented as Figure 2.2. $\tilde{h}$ and $\tilde{g}$ are the low-pass and high-pass analysis filters, respectively, and h and g are the low-pass and high-pass synthesis filters, respectively. The blocks (circles) after analysis filters are subsampling units, and the blocks (circles) before synthesis filters are upsampling units. In the dissertation, all the filters in DWT are FIR filters. Equations (2.13) and (2.14) are the requirements for perfect reconstruction:

$$h(z)\tilde{h}(z^{-1}) + g(z)\tilde{g}(z^{-1}) = 2, \tag{2.13}$$

$$h(z)\tilde{h}(-z^{-1}) + g(z)\tilde{g}(-z^{-1}) = 0. \tag{2.14}$$

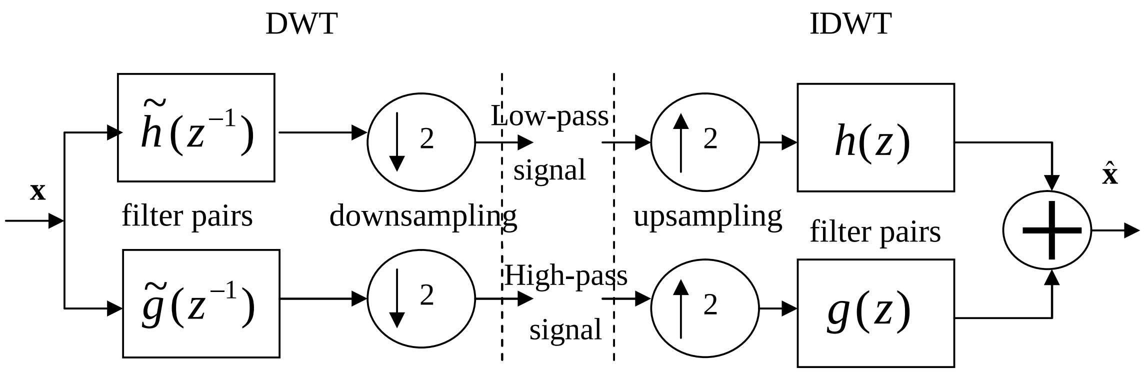


Figure 2.2 Block diagram of one-dimensional DWT. **x** is the original signal and $\hat{\mathbf{x}}$ is the reconstruction signal of **x**.

The modulation matrix *M*(*z*) and dual modulation matrix $\tilde{M}(z)$ are defined in Eqs. (2.15) and (2.16), respectively:

$$M(z) = \begin{bmatrix} h(z) & h(-z) \\ g(z) & g(-z) \end{bmatrix}, \tag{2.15}$$

$$\tilde{M}(z) = \begin{bmatrix} \tilde{h}(z) & \tilde{h}(-z) \\ \tilde{g}(z) & \tilde{g}(-z) \end{bmatrix}. \tag{2.16}$$

Hence, the perfect reconstruction conditions can be represented as

$$\tilde{M}(z^{-1})^t M(z) = 2I\,, \tag{2.17}$$

where *I* is the 2-by-2 identity matrix and $\tilde{M}^t$ is the transpose of $\tilde{M}$. Since all the filters here are FIR, the modulation and dual modulation matrices belong to GL(2, **R**[*z*, z^{-1}]) which denotes a ring whose elements are 2-by-2 matrices with Laurent-polynomial entries, and any matrix from this set is invertible and unitary.

The polyphase representation of a filter *h* is given by

$$h(z) = h_e(z^2) + z^{-1} h_o(z^2),$$

where h_e and h_o contains the even and odd coefficient terms of *h*, respectively. H_e and h_o can be represented as the form in Eqs. (2.18) or (2.19):

$$\begin{cases} h_e(z) = \sum_k h(2k) z^{-k}, \\ h_o(z) = \sum_k h(2k+1) z^{-k}, \end{cases} \tag{2.18}$$

and

$$\begin{cases} h_e(z^2) = \dfrac{h(z) + h(-z)}{2}, \\ h_o(z^2) = \dfrac{h(z) - h(-z)}{2z^{-1}}. \end{cases} \tag{2.19}$$

We define the polyphase matrix as

$$P(z) = \begin{bmatrix} h_e(z) & g_e(z) \\ h_o(z) & g_o(z) \end{bmatrix}, \tag{2.20}$$

and then

$$P(z^2)^t = \tfrac{1}{2} M(z) \begin{bmatrix} 1 & z \\ 1 & -z \end{bmatrix}. \tag{2.21}$$

Similarly, the dual polyphase matrix can be defined as that in Eq. (2.22):

$$\tilde{P}(z) = \begin{bmatrix} \tilde{h}_e(z) & \tilde{g}_e(z) \\ \tilde{h}_o(z) & \tilde{g}_o(z) \end{bmatrix}. \tag{2.22}$$

By using $\tilde{P}(z)$ and $P(z)$, the DWT can be represented as shown in Figure 2.3, and the perfect reconstruction condition is represented as

$$P(z)\tilde{P}(z^{-1})^{t} = I. \tag{2.23}$$

In the Figure 2.3, for DWT side, first, the original signal **x** is subsampled into even and odd samples, then the susampled results are applied the dual polyphase matrix. For the inverse transform, the transformed inputs are applied the polyphase matrix first, and then the even and odd results are joined to form the reconstruction signal $\hat{\mathbf{x}}$.

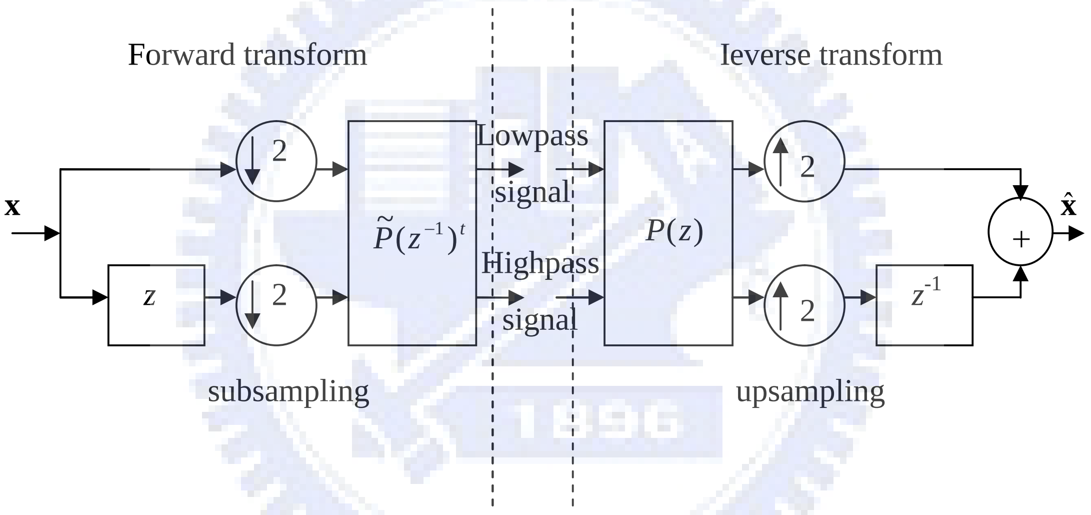


Figure 2.3 The polyphase representation of DWT and IDWT.

Because $\tilde{P}(z)$ and $P(z)$ contain only Laurent polynomials, Eq. (2.23) implies that the determinant and inverse matrix of $P(z)$ are all Laurent polynomials , and that is possible only when $det(P(z))$ is a monomial (i.e. $det(P(z)) = cz^{n}$, where $c \in \mathbf{R}$ and $n \in \mathbf{Z}$). Thus $\tilde{P}(z)$ and $P(z)$ are elements in GL(2, $\mathbf{R}[z, z^{-1}]$). If $det(P(z))$ is not equal to one, then we can divide $g_e(z)$ and $g_o(z)$ by $det(P(z))$, and then $det(P(z))$ becomes one. This means that for a specific (given) filter h, the determinant of a polyphase can always be one by scaling and shifting the filter g. Without loss of generality we assume that $det(P(z)) = 1$ and $P(z) \in$ SL[2; $\mathbf{R}[z, z^{-1}]$]. The problem to find a FIR wavelet transform is equivalent to find a polyphase matrix $P(z)$ with $det(P(z)) = 1$. For a given $\tilde{P}(z)$ and the four filters for the DWT follow immediately. Solving Eq. (2.23) we have

$$\begin{cases} \tilde{h}_e(z) = g_o(z^{-1}), \\ \tilde{h}_o(z) = -g_e(z^{-1}), \\ \tilde{g}_e(z) = -h_o(z^{-1}), \\ \tilde{g}_o(z) = h_e(z^{-1}). \end{cases} \tag{2.24}$$

Equation (2.24) implies

$$\begin{cases} \tilde{g}(z) = z^{-1}h(-z^{-1}), \\ \tilde{h}(z) = -z^{-1}g(-z^{-1}). \end{cases} \tag{2.25}$$

For the simple example $P(z) = I$, we have $h(z) = \tilde{h}(z) = 1$ and $g(z) = \tilde{g}(z) = z^{-1}$. The DWT only splits the input signal into even and odd samples and downsamples these samples. Such a DWT is named as the Lazy wavelet transform [14] or polyphase transform.

2.1.3 The lifting structure

In DWT, the lifting structure [12]-[14] is a special relationship between perfect reconstruction filter pairs (h, g) which have the same low-pass or high-pass filters. We can begin from the Lazy wavelet and apply lifting to build our way, step by step, up to a multiresolution analysis with particular features.

If the corresponding polyphase matrix $P(z)$ of a filter pair (h, g) has determinant one, then the filter pair is complementary. For a complementary filter pair (h, g), its dual filter pair $(\tilde{h}, \tilde{g})$ is also complementary. Assume that a filter pair (h, g) is complementary.

Then any other FIR filter, which is denoted as g^{new}, complementary to h is of the form in Eq. (2.26):

$$g^{new}(z) = g(z) + h(z)s(z^2)\,, \tag{2.26}$$

where $s(z)$ is a Laurent polynomial. In other words, any filter of the form in Eq. (2.26) is complementary to h.

$h_e(z)s(z)$ and $h_o(z)s(z)$ are the even and odd polyphase components of $h(z)s(z^2)$, respectively, and the new polyphase matrix is:

$$P^{new}(z) = P(z)\begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix}. \tag{2.27}$$

Since the determinant of the second term on the right side in Eq. (2.27) is one, $P^{new}(z)$ and $P(z)$ have the same determinant [12]. Figure 2.4 shows the schematic representation of lifting. In the forward DWT part, the new dual polyphase matrix is given by:

$$\tilde{P}^{new}(z) = \tilde{P}(z)\begin{bmatrix} 1 & 0 \\ -s(z^{-1}) & 1 \end{bmatrix}. \tag{2.28}$$

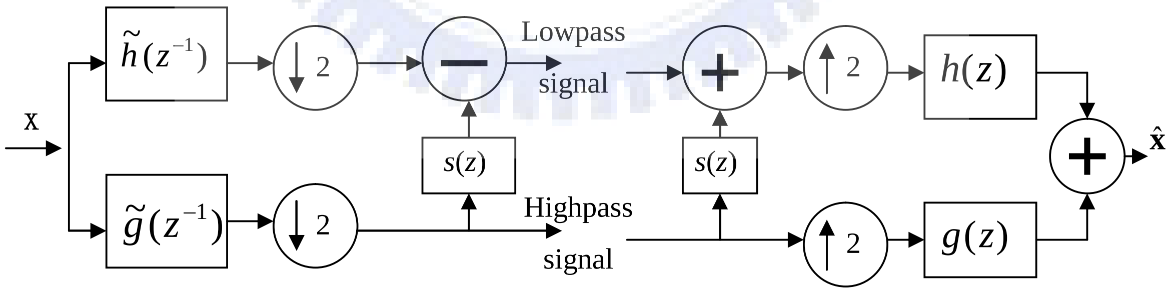


Figure 2.4 The lifting structure: A classical subband filter scheme followed by a lifting scheme which lifts the low-pass subband with the help of the high-pass subband.

A new low-pass filter $\tilde{h}$ is given by Eq. (2.29):

$$\tilde{h}^{new}(z) = \tilde{h}(z) - \tilde{g}(z)s(z^{-2})\,. \tag{2.29}$$

Similarly, if (h, g) is complementary, then any other FIR filter h^{neo} complementary to g is of the form:

$$h^{new}(z) = h(z) + g(z)t(z^2)\,, \tag{2.30}$$

where $t(z)$ is a Laurent polynomial. Conversely speaking, any filter of this form in Eq. (2.30) is complementary to g. For dual lifting, the new polyphase matrix is

$$P^{new}(z) = P(z)\begin{bmatrix} 1 & 0 \\ t(z) & 1 \end{bmatrix}. \tag{2.31}$$

Dual lifting generates a new $\tilde{g}$ which is:

$$\tilde{g}^{new}(z) = \tilde{g}(z) - \tilde{h}(z)t(z^{-2})\,. \tag{2.32}$$

The dual lifting structure is shown in Figure 2.5. Sweldens had proposed a family of lifting wavelets which starts from the Lazy wavelet followed by one dual lifting and one primal lifting step. Every h filter constructed this way is half band, and the corresponding scaling function is interpolating.

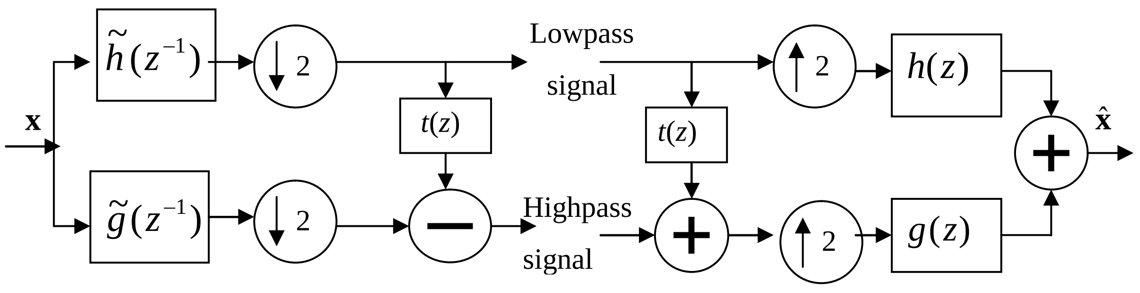


Figure 2.5 The dual lifting structure: A classical subband filter scheme followed by a lifting scheme which lifts the high-pass subband with the help of the low-pass subband.

2.1.4 The Euclidean algorithm

In this section, the Euclidean algorithm is extended to find the greatest common divisor (*gcd*) of two polynomials, and it will be used to determine the common factors of two Laurent polynomials later [20]. As we have discussed in **Example 2.1**, the *gcd* of polynomials is not unique. Actually, the *gcd* of two Laurent polynomials is defined up to a factor z^n (Note that, the *gcd* of two regular polynomials is up to a constant.). If the *gcd* of two Laurent polynomials is of degree zero, then the two Laurent polynomials are relatively prime.

Euclidean algorithm for Laurent polynomials:

Assume there are two Laurent polynomials $a(z)$ and $b(z)$ with $|a(z)| \geqq |b(z)|$ and $b(z) \neq 0$. Set $a^0(z) = a(z)$ and $b^0(z) = b(z)$ and iterate the following steps beginning from $n = 0$.

$$a^{n+1}(z) = b^n(z), \tag{2.33}$$

$$b^{n+1}(z) = a^n(z) \bmod b^n(z), \tag{2.34}$$

where the superscripts of Laurent polynomials $a(z)$ and $b(z)$ denote the iteration number. For the smallest $n = m$ that $b_m(z) = 0$, we have $a_m(z) = gcd(a(z), b(z))$. Given that

$\left|b^{n+1}(z)\right| < \left|b^{n}(z)\right|$, there is an k such that $\left|b^{k}(z)\right| = 0$. The algorithm stops for $m = k + 1$. The number of steps is bounded by m ($m \leqq |b(z)| + 1$). Let $q^{n+1}(z) = a^n(z)/b^n(z)$, then we have

$$\begin{bmatrix} a^m(z) \\ 0 \end{bmatrix} = \left(\prod_{n=m}^{1} \begin{bmatrix} 0 & 1 \\ 1 & -q^n(z) \end{bmatrix} \right) \begin{bmatrix} a(z) \\ b(z) \end{bmatrix}. \tag{2.35}$$

Therefore

$$\begin{bmatrix} a(z) \\ b(z) \end{bmatrix} = \left(\prod_{n=1}^{m} \begin{bmatrix} q^n(z) & 1 \\ 1 & 0 \end{bmatrix} \right) \begin{bmatrix} a^m(z) \\ 0 \end{bmatrix}, \tag{2.36}$$

and $a^m(z)$ divides both $a(z)$ and $b(z)$. If $a^m(z)$ is a monomial, then $a(z)$ and $b(z)$ are relatively prime.

Example 2.2 Assume that $a(z) = a^0(z) = z^{-1} + 6 + z$, $b(z) = b^0(z) = 3 + 3z$. The first division gives us

$$a^1(z) = 3 + 3z,$$
$$b^1(z) = 4,$$

and

$$q^1 = \tfrac{1}{3}(z^{-1} + 1).$$

The second iteration gives:

$$a^2(z) = 4,$$
$$b^2(z) = 0,$$

and

$$q^2 = \tfrac{3}{4}(1+z).$$

Hence, $a(z)$ and $b(z)$ are relative prime and we have:

$$\begin{bmatrix} z^{-1}+6+z \\ 3+3Z \end{bmatrix} = \begin{bmatrix} \tfrac{1}{3}(z^{-1}+1) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \tfrac{3}{4}(1+z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 0 \end{bmatrix}.$$

It takes 2 (i.e. $m = |b(z)| + 1$) steps (iterations) to complete the process.

2.1.5 The factoring algorithm

In this section, we will discuss how to factor a pair of complementary filters (h, g) into lifting steps. Note that $h_e(z)$ and $h_o(z)$ must be relatively prime, since any common factor would also divide $det(P(z))$ and $det(P(z)) = 1$ is already known. Use the Euclidean algorithm to find the monomial *gcd* of $h_e(z)$ and $h_o(z)$. Because of the non-uniqueness of the Laurent-polynomial division, we can only select the quotients such that the *gcd* is a constant. Assume the constant is c, we have that

$$\begin{bmatrix} h_e(z) \\ h_o(z) \end{bmatrix} = \prod_{n=1}^{m} \begin{bmatrix} q^n(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} c \\ 0 \end{bmatrix}. \tag{2.37}$$

If m is odd, we can multiply $h(z)$ with z and $g(z)$ with z^{-1}. This does not change the determinant of the polyphase matrix, and it flips the polyphase components of $h(z)$ and makes m even. Thus, we can always assume that m is even. Given a filter $h(z)$ we can always find a complementary filter, that is denoted as g^{cpl}, by letting:

$$P^{cpl}(z) = \begin{bmatrix} h_e(z) & g_e^{cpl}(z) \\ h_o(z) & g_o^{cpl}(z) \end{bmatrix} = \prod_{n=1}^{m} \begin{bmatrix} q^n(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix}. \tag{2.38}$$

Rewrite the second equation in Eq. (2.38) and we have

$$\begin{bmatrix} q^n(z) & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & q^n(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{2.39}$$

and

$$\begin{bmatrix} q^n(z) & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q^n(z) & 1 \end{bmatrix}. \tag{2.40}$$

When n is odd, use Eq. (2.39), and use Eq. (2.40) for n is even. Thus Eq. (2.38) becomes

$$P^{cpl}(z) = \prod_{n=1}^{m/2} \begin{bmatrix} 1 & q^{2n-1}(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q^{2n}(z) & 1 \end{bmatrix} \begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix}. \tag{2.41}$$

At last, the original filter g can be recovered by Eq. (2.26). The filter g can always be recovered from g^{cpl} with one lifting or:

$$P(z) = P^{cpl}(z) \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix}. \tag{2.42}$$

To sum up, if the complementary filter pair (h, g) is given, then there always exist Laurent polynomials $s^n(z)$ and $t^n(z)$ for $1 \le n \le k$ and a nonzero constant c such that

$$P(z) = \prod_{n=1}^{k} \begin{bmatrix} 1 & s^n(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t^n(z) & 1 \end{bmatrix} \begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix}. \tag{2.43}$$

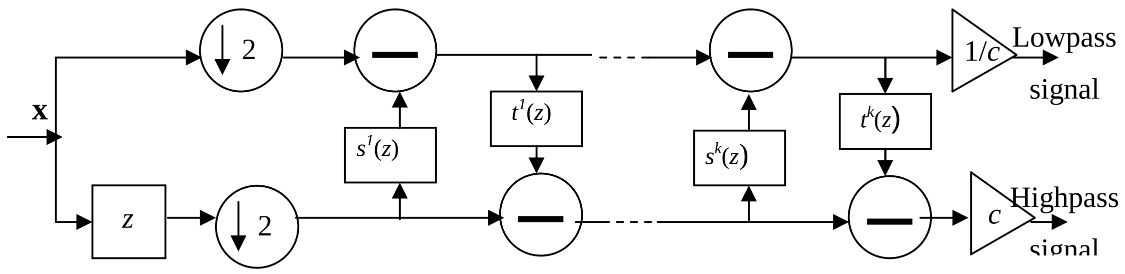


Figure 2.6 The forward DWT using lifting structure.

Equation (2.43) means that every FIR filter DWT can be obtained by beginning with the Lazy wavelet followed by k lifting and dual lifting steps followed with a scaling. Similarly, the dual polyphase matrix is given by

$$\tilde{P}(z)=\prod_{n=1}^{k}\begin{bmatrix}1 & 0\\ -s^{n}(z^{-1}) & 1\end{bmatrix}\begin{bmatrix}1 & -t^{n}(z^{-1})\\ 0 & 1\end{bmatrix}\begin{bmatrix}1/c & 0\\ 0 & c\end{bmatrix}. \tag{2.44}$$

Figures 2.6 and 2.7 show the different steps of DWT and IDWT, respectively.

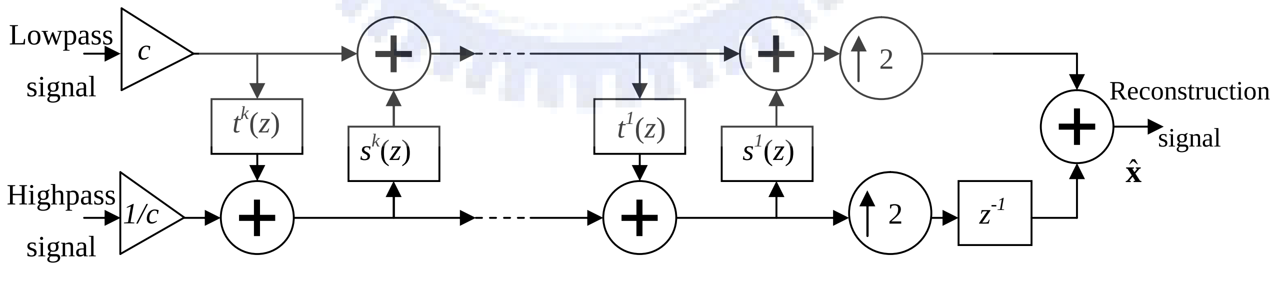


Figure 2.7 The Inverse DWT using lifting structure.

2.2 Realization of Lifting-Based Discrete Wavelet Transform

In this section, the famous Haar wavelet, 5/3 wavelet, and 9-7 wavelet and their corresponding lifting-based realizations are discussed in the following subsections.

2.2.1 Haar wavelets

For the Haar wavelets, we have that $h(z)=1+z^{-1}$, $g(z)=-\frac{1}{2}+\frac{1}{2}z^{-1}$, $\tilde{h}(z)=\frac{1}{2}+\frac{1}{2}z^{-1}$, and $\tilde{g}(z)=-1+z^{-1}$. By using the Euclidean algorithm the polyphase matrix can be represented as:

$$P(z)=\begin{bmatrix}1 & -1/2\\ 1 & 1/2\end{bmatrix}=\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}\begin{bmatrix}1 & -1/2\\ 0 & 1\end{bmatrix}.$$

Therefore, on the analysis side we have:

$$P(z)^{-1}=\tilde{P}(z^{-1})=\begin{bmatrix}1 & 1/2\\ 1 & 1\end{bmatrix}=\begin{bmatrix}1 & 0\\ -1 & 1\end{bmatrix}.$$

Hence, we have the following realization of the forward DWT:

$$\begin{cases} s^{(0)}(n)=x(2n),\\ d^{(0)}(n)=x(2n+1),\\ d(n)=d^{(0)}(n)-s^{(0)}(n),\\ s(n)=s^{(0)}(n)+\frac{1}{2}d(n),\end{cases}$$

and the IDWT is given by:

$$\begin{cases} s^{(0)}(n)=s(n)-\frac{1}{2}d(n),\\ d^{(0)}(n)=d(n)+s^{(0)}(n),\\ \hat{x}(2n+1)=d^{(0)}(n),\\ \hat{x}(2n)=s^{(0)}(n).\end{cases}$$

In the reverse transform, we use $\hat{x}$ to denote the reconstruction version of x. These signals are shown in Figs. 2.8 and 2.9.

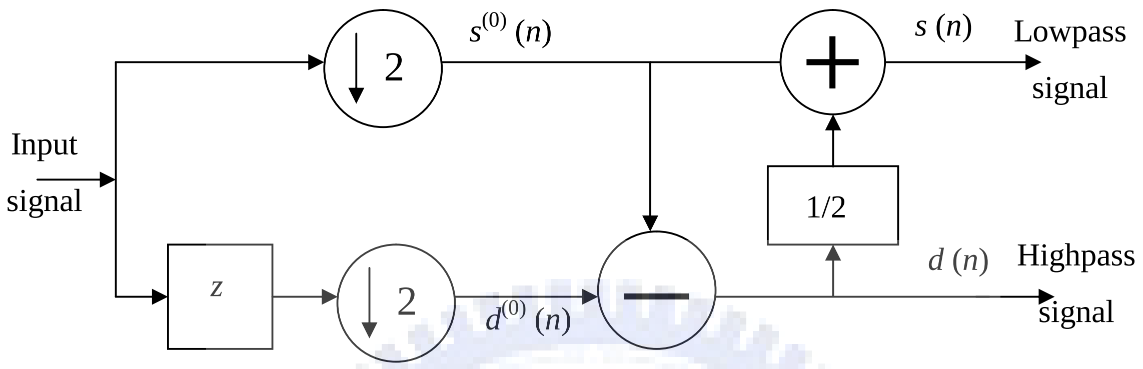


Figure 2.8 DWT with lifting Haar wavelet.

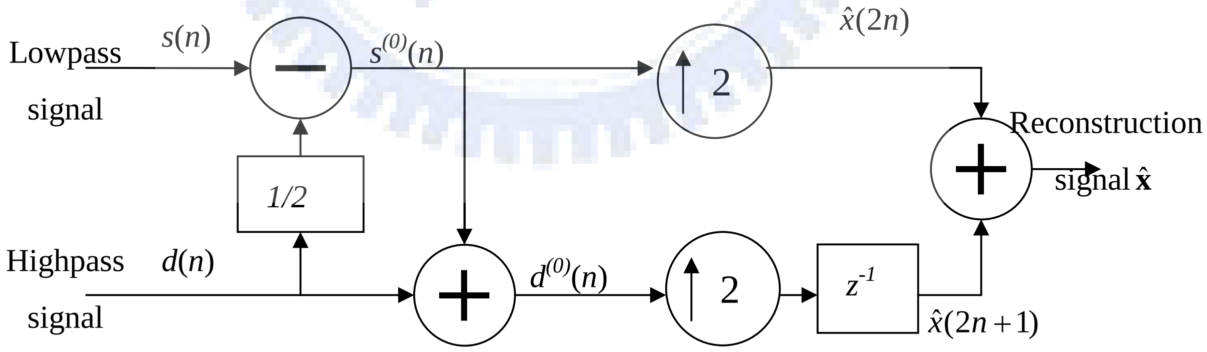


Figure 2.9 IDWT with lifting Haar wavelet.

2.2.2 The 5/3 wavelets

In this section, we will discuss the 5/3 wavelets which was recommended by the new image compression standard JPEG2000. For the 5/3 wavelets, we have that $\tilde{h}(z) = -\frac{1}{8}z^{-2} + \frac{1}{4}z^{-1} + \frac{3}{4} + \frac{1}{4}z^{1} - \frac{1}{8}z^{2}$, $\tilde{g}(z) = -\frac{1}{2}z^{-2} + z^{-1} - \frac{1}{2}$. According to Eqs. (2.18) and (2.19) we have:

$$\begin{cases} \tilde{h}_e(z^2) = -\frac{1}{8}z^{-2} + \frac{3}{4} - \frac{1}{8}z^{2}, \\ \tilde{h}_o(z^2) = \frac{1}{4} + \frac{1}{4}z^{2}, \\ \tilde{g}_e(z^2) = -\frac{1}{2}z^{-2} - \frac{1}{2}, \\ \tilde{g}_o(z^2) = 1. \end{cases}$$

The dual polyphase matrix of this filter bank is:

$$\tilde{P}(z) = \begin{bmatrix} \tilde{h}_e(z) & \tilde{g}_e(z) \\ \tilde{h}_o(z) & \tilde{g}_o(z) \end{bmatrix} = \begin{bmatrix} -\frac{1}{8}z^{-2} + \frac{3}{4} - \frac{1}{8}z^{2} & -\frac{1}{2}z^{-2} - \frac{1}{2} \\ \frac{1}{4} + \frac{1}{4}z^{2} & 1 \end{bmatrix}.$$

Assuming perfection reconstruction and complementary filters, the corresponding synthesis filters are:

$$h(z) = -z^{-1}\tilde{g}(-z^{-1}) = \frac{1}{2}z^{-1} + 1 + \frac{1}{2}z^{1},$$

and

$$g(z) = z^{-1}\tilde{h}(-z^{-1}) = -\frac{1}{8}z^{-3} - \frac{1}{4}z^{-2} + \frac{3}{4}z^{-1} - \frac{1}{4} - \frac{1}{8}z.$$

Using the Euclidean algorithm we can factor the dual polyphase matrix as:

$$\tilde{P}(z) = \begin{bmatrix} 1 & (1+z)/4 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -(1+z^{-1})/2 & 1 \end{bmatrix}.$$

Hence, we have the following realization of the forward DWT:

$$\begin{cases} s^{(0)}(n) = x(2n), \\ d^{(0)}(n) = x(2n+1), \\ d(n) = d^{(0)}(n) - \tfrac{1}{2}[s^{(0)}(n) + s^{(0)}(n+1)], \\ s(n) = s^{(0)}(n) + \tfrac{1}{4}[d(n) + d(n+1)], \end{cases}$$

and the IDWT is given by:

$$\begin{cases} s^{(0)}(n) = s(n) - \tfrac{1}{4}[d(n) + d(n+1)], \\ d^{(0)}(n) = d(n) + \tfrac{1}{2}[s^{(0)}(n) + s^{(0)}(n+1)], \\ x(2n+1) = d^{(0)}(n), \\ x(2n) = s^{(0)}(n). \end{cases}$$

2.2.3 The 9/7 wavelets

The 9/7 wavelet [22] filter bank was also proposed in Part I of JPEG2000 standard. For the 9/7 filter pair, the analysis filter $\tilde{h}$ has 9 coefficients, and the synthesis filter h has 7 coefficients. Each of the two high-pass filters $\tilde{g}$ and g has 4 vanishing moments. For a smoother scaling function, the filter with 7 coefficients is choused to be the synthesis filter. Using the Euclidean algorithm, the most efficient factorization of the dual polyphase matrix of 9/7 wavelets is as follows:

$$\tilde{P}(z) = \begin{bmatrix} 1 & a(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ b(1+z) & 1 \end{bmatrix} \begin{bmatrix} 1 & c(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ d(1+z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix},$$

where a = -1.586134342, b = -0.05298011854, c = 0.8829110762, d = 0.4435068522, and K = 1.149604398.

Both 5/3 and 9/7 wavelet filters can be represented by banded matrix operations. For the 5/3 wavelet:

$$Y_{5/3} = XM_1M_2,$$

where

$$M_1 = \begin{bmatrix} 1 & a & 0 & & & & & & \\ 0 & 1 & 0 & 0 & & & & & \\ 0 & a & 1 & a & 0 & & & & \\ & 0 & 0 & 1 & 0 & 0 & & & \\ & & 0 & a & 1 & a & 0 & & \\ & & & 0 & 0 & 1 & 0 & 0 & \\ & & & & 0 & a & 1 & a & 0 \\ & & & & & 0 & 0 & 1 & 0 \\ & & & & & & 0 & a & 1 \end{bmatrix},$$

and

$$M_2 = \begin{bmatrix} 1 & 0 & 0 & & & & & & \\ 0 & 1 & b & 0 & & & & & \\ 0 & 0 & 1 & 0 & 0 & & & & \\ & 0 & b & 1 & b & 0 & & & \\ & & 0 & 0 & 1 & 0 & 0 & & \\ & & & 0 & b & 1 & b & 0 & \\ & & & & 0 & 0 & 1 & 0 & 0 \\ & & & & & 0 & b & 1 & 0 \\ & & & & & & 0 & 0 & 1 \end{bmatrix}.$$

The IDWT can be represented as

$$X = Y_{5/3}M_2^{-1}M_1^{-1}.$$

For the 9/7 wavelet, we have:

$$Y_{9/7} = XM_1M_2M_3M_4,$$

$$M_3 = \begin{bmatrix} 1 & c & 0 & & & & & & \\ 0 & 1 & 0 & 0 & & & & & \\ 0 & c & 1 & c & 0 & & & & \\ & 0 & 0 & 1 & 0 & 0 & & & \\ & & 0 & c & 1 & c & 0 & & \\ & & & 0 & 0 & 1 & 0 & 0 & \\ & & & & 0 & c & 1 & c & 0 \\ & & & & & 0 & 0 & 1 & 0 \\ & & & & & & 0 & c & 1 \end{bmatrix},$$

and

$$M_4 = \begin{bmatrix} 1 & 0 & 0 & & & & & & \\ 0 & 1 & d & 0 & & & & & \\ 0 & 0 & 1 & 0 & 0 & & & & \\ & 0 & d & 1 & d & 0 & & & \\ & & 0 & 0 & 1 & 0 & 0 & & \\ & & & 0 & d & 1 & d & 0 & \\ & & & & 0 & 0 & 1 & 0 & 0 \\ & & & & & 0 & d & 1 & 0 \\ & & & & & & 0 & 0 & 1 \end{bmatrix}.$$

Finally, the IDWT based on 9/7 wavelet can be represented as:

$$X = Y_{9/7} M_4^{-1} M_3^{-1} M_2^{-1} M_1^{-1}.$$

2.3 Lifting-Based Direction-Adaptive DWT (DA-DWT)

The conventional separable 2-D DWT can be implemented by consecutively applying 1-D DWT in horizontal and vertical directions, or vice versa. That means if we use the lifting structure to implement 2-D DWT, the prediction and update directions are parallel to the horizontal axis or the vertical axis. The lifting-based DWT whose directions of prediction and update steps are adaptive is called a direction-adaptive DWT (DA-DWT) or adaptive directional lifting-based DWT (ADL-DWT). The DA-DWT can be decomposed into two blocks which are shown in Figure 2.10. First, an input image is partitioned into many small blocks and the best transform direction for each block is determined. Then, lifting DWT is performed on the image according to the partition and direction information obtained in the first block. Figure 2.11 shows the direction of the lifting step of a direction-adaptive DWT, and the prediction and update direction line intersects the horizontal axis by an angle θ. Applying the sub-pel technique, although the angle θ can be any value between 0 and $\pi/2$ (radians) in [3], only nine directions were used. In [4], they also used nine directions for prediction and update, but these nine directions were different from those in [3]. In this dissertation, we use the directions in [4], because this method does not involve complex sub-pel computation and has better performance. Figure 2.11 shows the nine directions and their corresponding neighbors of an odd sample in prediction step of a 5/3-wavelet DWT. In Figure 2.13, each of the even samples is updated by its two odd neighbors along the line with θ = 45 degrees.

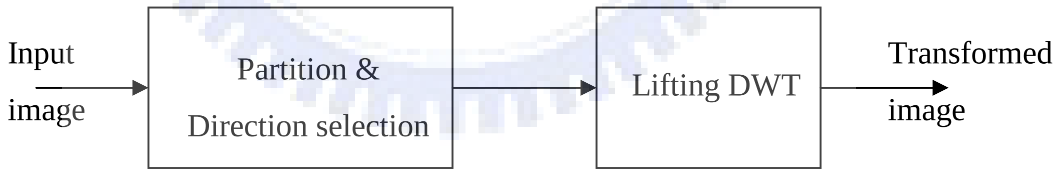


Figure 2.10 The block diagram of a DA-DWT system.

ADL-DWT [3] and DA-DWT [4] are proposed to compress a rectangular image by dividing the whole image into a lot of fixed-size small square blocks. After dividing an image into many small square blocks, the optimal direction for directional lifting DWT of each block is determined. Then, some connected blocks with the same lifting direction are grouped to form a large rectangular block with the same direction for saving the bits of side information [4]. Different from the method in [4], the method used in [3] splits a larger square block into several small rectangular blocks instead of merging some small square blocks to form a larger rectangular block. For lossless image compression, the best direction (θ) of prediction and update of the directional lifting DWT is the direction that spends the least amount of bits to compress this (square or rectangular) block. For lossy image compression, the best direction of the directional lifting DWT in a block should be the one that has the highest PSNR value for a given bit-budget. Either lossless or lossy image compression, the best direction selection should have strong energy compaction effect in the low frequency subband. On the other hand, energy compaction in low frequency subband is equivalent to that the energy left in high frequency subband is little. Hence, the optimal direction selection can be approximately determined by choosing the direction in which the directional DWT has the smallest absolute sum of the coefficients in the high frequency subband.

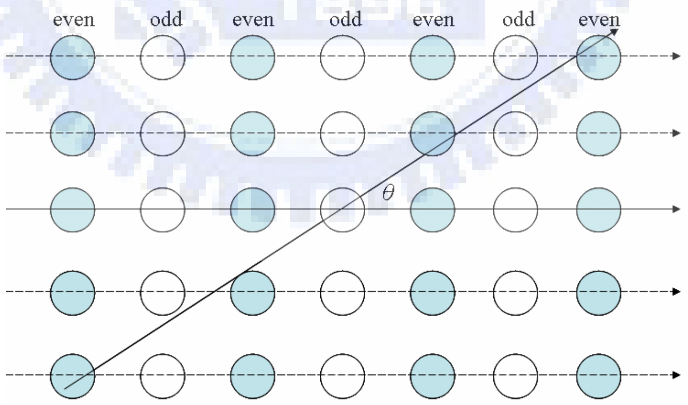


Figure 2.11 A Direction selection example with angle θ in 1-D “horizontal” DWT.

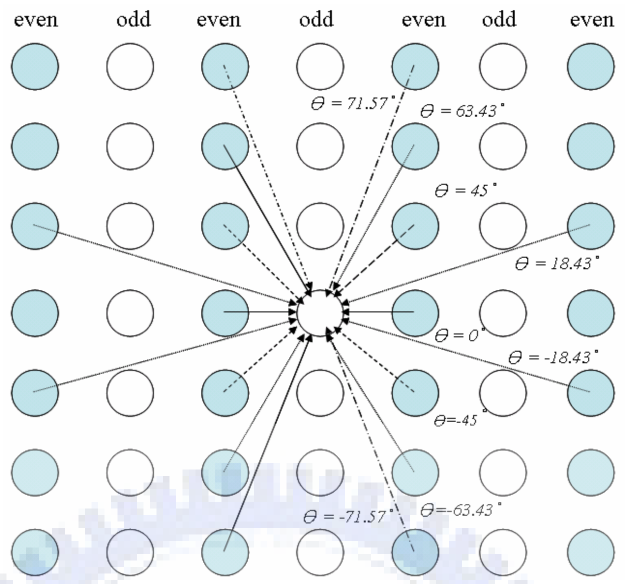


Figure 2.12 Nine prediction directions for an odd sample in 1-D “horizontal” DWT.

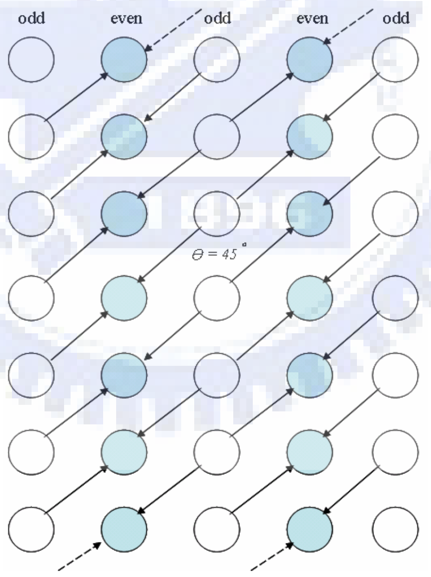


Figure 2.13 The update stage with θ = 45°.

The last step of the 1-D directional DWT is a subsampling stage, and the subsampling method is just like the way in the conventional DWT, i.e. the subsampling direction is the horizontal direction for a “row” directional adaptive DWT. After the “row” directional adaptive DWT is complete, the “column” direction DWT is performed on the whole segment block, and the last step is the subsampling step following the “column” directional DWT in a one-level direction-adaptive DWT. The realization of the “column” direction-adaptive DWT is the same as the “row” direction-adaptive DWT, if the segment block after “row” direction-adaptive DWT is rotated clockwise 90 degrees.

2.4 Shape-Adaptive DWT (SA-DWT)

Because of fast growth of multimedia applications, the needs of searching, accessing, indexing, and manipulating visual information at the semantically meaningful object level are becoming more and more urgent. The MPEG-4 standard supports such a functionality of making a visual object available in the compressed form, and this functionality provides flexibility for manipulating a visual object in multimedia applications and improves the compression efficiency in very low bit-rate coding. There are two major parts in an object-based video coding system. One is the intra frame coding, and the other is the inter frame coding. The inter frame coding involving motion prediction, which will not be discussed here, and we focus on the intra frame coding in this subsection. The intra frame coding of the object-based video coding can be divided into object shape coding and object texture coding. The alpha map (Figure 2.14 (b)) is used to represent the region that the object occupied, and the simplest alpha map can be a binary figure which has value 1 for the bits in the object and value 0 for each bit outside the object. Thus, using the alpha map, the object in an image can be easily segmented. The most popular technique for object texture coding is the shape-adaptive DCT (SA-DCT) [23], which uses 8-by-8 blocks to represent the object to be transformed and coded. Since an object usually can not be covered by 8-by-8 blocks perfectly, a lot of boundary blocks do not totally reside in the object and make this method inefficient. S. Li *et al.* proposed a shape-adaptive discrete wavelet transform (SA-DWT) for arbitrarily shaped visual object coding [1], and they used the SA-DWT for the texture coding of the intra frame part in object based video coding. Lu *et al.* also proposed an object texture coding technique [2] that combined a SA-DWT and the SPECK algorithm. The experimental results in [1] showed that the SA-DWT with extentions of zerotree entropy coding (ZTE) outperforms SA-DCT up to 0.97 dB in Y-plane *PSNR*, 1.29 dB in U-plane *PSNR*, and 0.89 dB in V-plane *PSNR*, for the Akiyo sequence (CIF) at 1.0 *bpp*. However, the SA-DWTs using conventional DWTs have the disadvantage that they need complicated rules to handle the even and odd problems of the samples. In Chapter 4, the lifting DWTs are introduced to realize the shape-adaptive functionality that we call lifting-based shape-adaptive DWT (LSA-DWT).

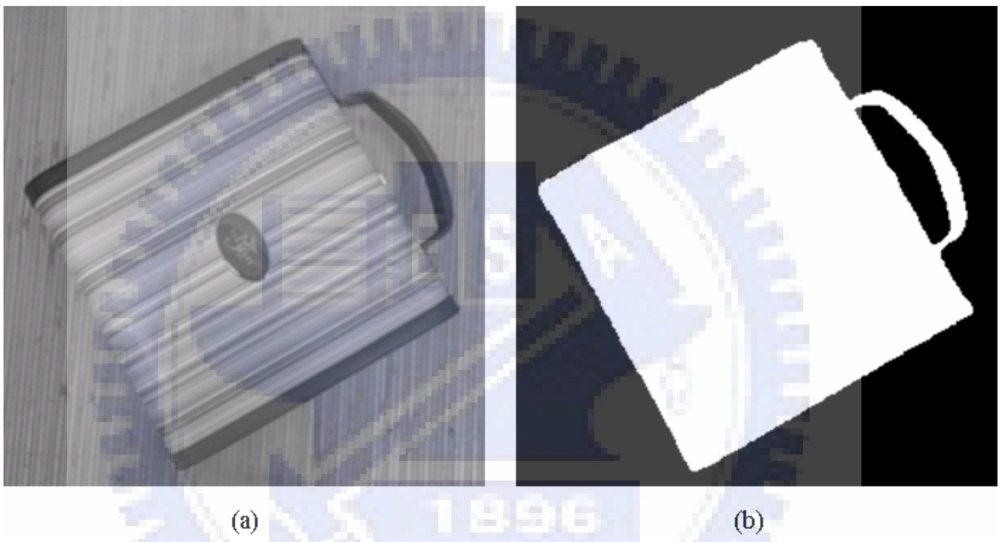


Figure 2.14 The original image and alpha map of object 1: (a) the test visual object 1 with background (256-by-256), (b) the shape mask (alpha map).

CHAPTER 3

TWO WAVELET-BASED HYBRID CODECS FOR IMAGE COMPRESSION

In this chapter, two image compression applications of DWT will be discussed. Both of them are wavelet-based applications, so the lifting-based DWT can seamlessly replace the conventional DWTs in these two applications. The first one, in Section 3.1, proposed a hybrid image coder which combines SPIHT (set partitioning in hierarchical trees) [24], DWT, and vector quantization [25][26] for improving image compression efficiency. In Section 3.2, the second application used DWT, SPECK (set-partitioning embedded block coder) [27], and residual vector quantization (RVQ) to enhance image compression performance.

3.1 Image Compression Using SPIHT and VQ

Su *et al*. presents a hybrid coding system using a combination of set partition in hierarchical trees (SPIHT) and vector quantization (VQ) for image compression [28]. In which, the wavelet coefficients of the input image are rearranged to form the wavelet trees that are composed of the corresponding wavelet coefficients from all the subbands of the same orientation; a simple tree classifier has been proposed to group these wavelet trees into two classes based on the amplitude distribution; and each class of trees is to be coded using an appropriate procedure, specifically either SPIHT or VQ. Experimental results show that advantages gained by combining the superior coding performance of VQ and efficient cross-subband prediction of SPIHT are, as expected, appreciable for the compression task, especially for many natural images with large portions of textures.

3.1.1 Overview of SPIHT

The SPIHT algorithm has received a lot of attention since its introduction for image compression in 1996. It contains two passes: sorting pass and refinement pass, which can be combined to form a single scan pass. Three symbols: zero tree (ZT), insignificant pixel (IP) and significant pixel (SP) are used to code the wavelet tree structure of images, which are stored in their respective lists: list of insignificant sets (LIS), list of insignificant pixels (LIP) and list of significant pixels (LSP). Below is the encoding algorithm presented in four steps [24].

1) Compute $b = \lfloor \log_2(\max_{(m,n)} |c_{m,n}|) \rfloor$, where $c_{m,n}$ is the wavelet tree node at coordinate (m,n). Set the initial threshold $T = 2^b$.

2) Sorting pass: identify the coefficients such that $T \le |c_{m,n}| < 2T$; output their respective coordinates and signs.

3) Refinement pass: output the b-th (most significant) bit of all the tree nodes with $|c_{m,n}| \ge 2T$ following the same order used to output the coordinates in previous sorting passes.

4) Decrease b by one, halve the threshold T and go to step 2.

The scan pass (i.e. Step 2 followed by Step 3) of SPIHT is performed in a recursive manner until the expected bit rate is reached. In sorting pass, the coefficients in LIS and LIP are evaluated as follows. For coefficients whose magnitudes are greater than or equal to the current threshold, they become significant and will be moved to LSP. For insignificant coefficients whose magnitudes are less than the current threshold, they will be stored in LIS if all their descendants are also insignificant with respective to the same threshold; or otherwise, stored in LIP. A sequence of successively smaller thresholds can be obtained by using the following recursive equation:

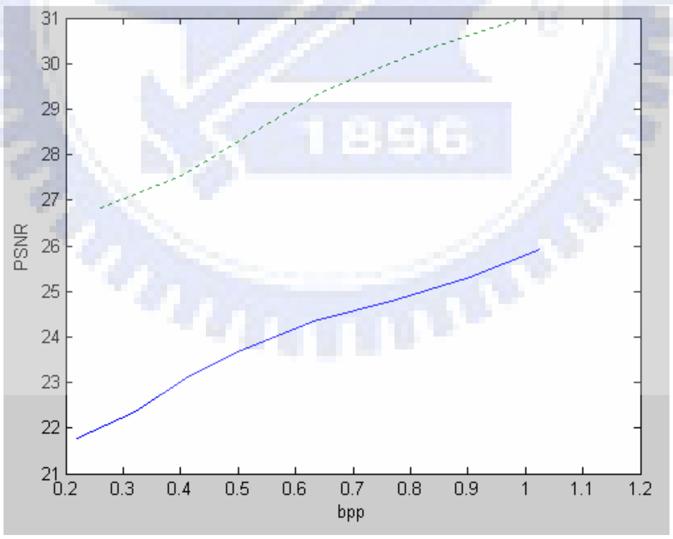
$$T_k = 0.5T_{k-1}, \tag{3.1}$$

where the initial threshold T_1 must be greater than or equal to half the maximum magnitude of the transform coefficients. After the k-th sorting pass, tree nodes whose magnitudes are in the range: $[T_k, T_{k-1})$ for $k > 1$ (or $[T_1, \infty)$ for $k = 1$) will be stored in LSP with one bit per node to indicate their respective signs. In refinement pass, the significant nodes stored in LSP are refined with one bit per node to update their respective information. The great success of SPIHT is attributed to the important hypothesis of wavelet transform: if a parent node is insignificant, then all its descendants are likely to be insignificant with respect to the same threshold and therefore these insignificant nodes can be efficiently coded with a single symbol ZT.

3.1.2 Proposed hybrid coding

For images with textures composed mainly of the middle and high frequency components, there are many significant nodes whose ancestors are insignificant. It follows that zero trees of insignificant nodes are very rare. Figure 3.1 (a), for example shows a 256×256 grayscale Mandrill image with large portions of high frequency textures. Empirically, we have classified the wavelet trees into two classes based on the magnitude distribution. The compression performance of SPIHT is evaluated for each class of wavelet trees. As shown in Figure 3.1 (b), where the horizontal and vertical axes are the compression rates measured in bits per pixel (*bpp*) and peak signal to noise ratio (*PSNR*) values measured in dB, respectively, the SPIHT algorithm is much more effective for one class of wavelet trees than the other.

(a)



(b)

Figure 3.1 Rate-distortion curves of the low frequency (dotted line) and high frequency (solid line) wavelet trees of Mandrill image by using the SPIHT algorithm.

3.1.3 Wavelet tree classification

High quality image compression at low bit rates can be achieved by coding each individual wavelet tree using a distinct, suitable procedure. It is noted that the SPIHT algorithm will not be suitable for coding wavelet trees with a large amount of significant nodes scattered in the higher frequency subbands, and therefore a different coding strategy is desirable. Moreover, a tree classifier that can efficiently divide the wavelet trees of images into two classes based on the magnitude distribution of the dominant wavelet coefficients is required.

For computation simplicity, a tree classifier based on the average magnitude of wavelet coefficients of each subband has been utilized to divide the wavelet trees of images into two classes: low frequency tree and high frequency tree, which is given as follows:

$$Arg\left\{ \max_{\ell} \quad \alpha_{\ell} \cdot Avg\left\{ \left| D_{\ell}^{d}(m,n) \right| \right\} \right\} < L_{tr} \quad \rightarrow \quad \text{high frequency tree} \ , \qquad (3.2)$$

where $\left| D_{\ell}^{d}(m,n) \right|$ is the wavelet coefficient magnitude at tree node coordinate (m,n), ℓ = 1, 2, ⋯, L denotes the resolution level with larger meaning coarser, L is the number of decomposition levels, d = 1, 2, 3 denotes the wavelet subband orientation in the horizontal, vertical and diagonal directions, respectively, α_{ℓ} is a weighting factor with respect to the resolution level, L_{tr} is a threshold value, and $Avg\{\cdot\}$ is an average operator.

3.1.4 MVQ Coding for High Frequency Wavelet Trees

Even though it is noted that wavelet transform provides de-correlation property, i.e. most of the correlation between image pixels can be removed in the wavelet domain, there may still be some residual correlation between neighboring coefficients across subbands of the same orientation. In order to get a good quality of the reconstructed images at relatively low bit rates, the residual correlation between wavelet coefficients must be exploited. According to the Shannon's theory, VQ can significantly reduce the coding bits of signals over scalar quantization. The VQ approach is therefore suitable for coding the high frequency wavelet trees of images.

Our strategy is as follows. First, group the high frequency wavelet trees into three categories according to their respective subband orientations: horizontal, vertical or diagonal. Second, partition each category of high frequency wavelet trees into small vectors based on the standard deviation distribution. Third, encode the small vectors of high frequency wavelet trees by using multistage VQ (MVQ). Figure 3.2 shows the MVQ structure with successive refinements. Here, the input vector is quantized at the first stage, and the residual information is quantized at the following stages in a recursive manner.

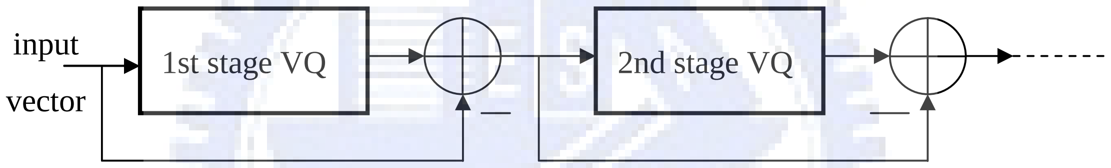


Figure 3.2 Multistage VQ structure.

3.1.5 MVQ codebook generation

A representative collection of images is utilized as training images for codebook generation. After 2-D DWT, the high frequency wavelet trees will be partitioned into small vectors to alleviate the computation complexity. The partitions chosen for each of the three categories of high frequency trees are determined in such a manner that tree nodes that have similar standard deviations are grouped into a single vector. Thereafter, a unique codebook is constructed for each vector because the intrinsic statistics and dimensions of vectors are different.

By taking into account that one of the key issues of the proposed hybrid image coding system, which is presented in the next subsection, is the bit allocation between two different coding procedures, the codebook size for each MVQ stage is 2. In other words, each vector will be coded in a progressive manner by using MVQ with one code bit per stage. The MVQ codebooks are constructed by using the LBG algorithm [26], stored in tables on both encoder and decoder sides, and therefore not transmitted along with the bit stream header.

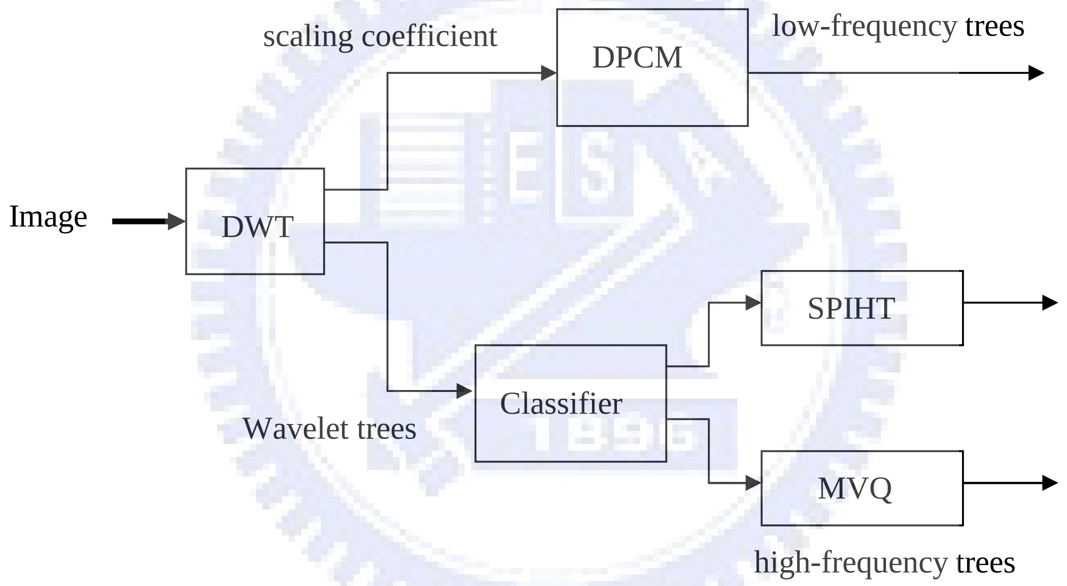


Figure 3.3 Block diagram of the proposed hybrid image coder by combining SPIHT and MVQ for coding the low and high frequency wavelet trees, respectively.

3.1.6 Hybrid image coding

After wavelet tree classification, the low frequency trees can be efficiently coded by SPIHT, and the high frequency trees are to be coded by MVQ. A hybrid coding system that combines SPIHT and MVQ is then proposed to improve the overall compression performance. Figure 3.3 shows the block diagram. The input image is decomposed into a set of subbands with orientation selectivity using 2-D DWT. The scaling coefficients at the coarsest resolution are coded by using the differential pulse code modulation algorithm. The corresponding wavelet coefficients taken from all the subbands of the same orientation (i.e. horizontal, vertical or diagonal) are rearranged to form wavelet trees. While SPIHT coding is suitable only for wavelet trees with a large amount of significant nodes in the lower frequency subbands, the MVQ approach seems promising for coding wavelet trees with many nodes of significance in the higher frequency subbands.

In MVQ, the high frequency wavelet trees are partitioned into small vectors. These partitions have been determined previously in the training process for codebook generation. Initially, each vector is progressively coded using the MVQ algorithm with B_1 stages, which is determined in such a manner that the norm of the residual quantization error will not be greater than the initial threshold T_1 of SPIHT. After one MVQ pass followed by one scan pass of SPIHT (for coding the high frequency wavelet trees and low frequency wavelet trees, respectively), the threshold of SPIHT is halved, and based on which the corresponding parameter of MVQ is determined similarly for the next MVQ pass. The alternate coding of high frequency wavelet trees and low frequency wavelet trees will go on until the expected bit rate (or the quality of the reconstructed image) is reached.

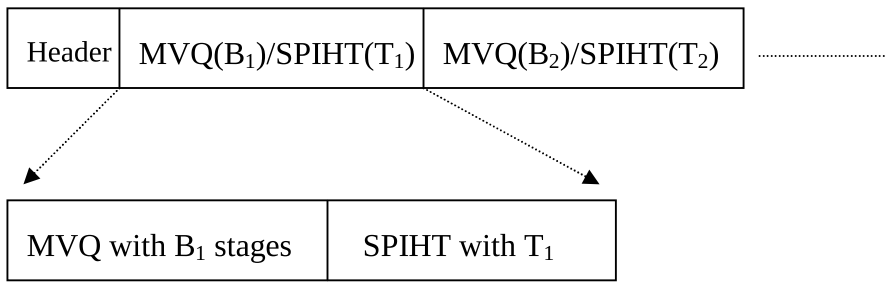


Figure 3.4 Bitstream structure.

Bit allocation between the SPIHT and MVQ coding procedures needs to be truly adaptive in order to generate an improved, embedded bit stream. During hybrid coding, both sequences of SPIHT thresholds: T_k and MVQ parameters: B_k are adapted to the input image since T_k is determined by the wavelet coefficient magnitudes of the input image, and B_k is determined by T_k. In our experiments, the parameter: B_k is often about 1 or 2 for $k = 2, 3, \cdots$, if the sequence of successively smaller thresholds: T_k is obtained by using Eq. (3.1), therefore, they can be set constant. The side information that is required to be transmitted along with the bit stream will be the number of wavelet decomposition levels, the tree classification threshold, one bit per wavelet tree to indicate the tree class, the initial SPIHT threshold: T_1, and the MVQ parameter: B_1. Figure 3.4 shows the embedded bit stream structure, where the side information is stored in the header portion.

3.1.7 Experimental results and conclusions

The proposed hybrid coding system is evaluated on natural 256×256 grayscale images. A set of nine training images is utilized to determine the partitions of the high frequency wavelet trees into small vectors, and to construct the MVQ codebooks for encoding these vectors. The partition strategy is as follows. For each of the three categories of high frequency wavelet trees, the standard deviation values are uniformly quantized with nine quantization levels. All the wavelet coefficients that have the same standard deviation level are grouped into a single vector. Consequently, the high frequency wavelet trees are partitioned into nine small vectors. The test 256×256 grayscale images: Mandrill (shown in Figure 3.1(a)), Bridge, and Lena, which represent natural images with a large amount of high frequency, middle frequency, and low frequency components, respectively, are outside the training set.

The compression performance is compared with the SPIHT coding algorithm. The compression rate is measured in *bpp*. The distortion is measured by peak signal to noise ration (*PSNR*), which is given by

$$PSNR\,[dB] = 20\log_{10}\frac{255}{RMSE} \quad , \tag{3.3}$$

where *RMSE* is the root mean squared error between the original and reconstructed images. The computed compression rates and *PSNR* values are collected to generate the rate distortion curves. The linear phase, biorthogonal wavelet with 9/7-coefficient filter set is utilized. The number of wavelet decomposition levels is 4. The tree classification threshold L_{tr} is 3. The weighting factors are empirically obtained by $\alpha_{\ell+1} = 0.5\alpha_{\ell}$ with $\alpha_1 = 1$. The infinite norm is used to compute the norms of the residual quantization error vectors. The maximum wavelet coefficient magnitude is halved and then used for the initial SPIHT threshold T_1, and the successively smaller thresholds are obtained by using Eq. (3.1).

Figure 3.5 shows the rate-distortion curves. The horizontal and vertical axes are the compression rates (in *bpp*) and *PSNR* values (in dB), respectively. For Lena image that is relatively smooth and most of the significant wavelet coefficients are in the lower frequency subbands, only a small number of wavelet trees are classified into the high frequency class. The performances of the hybrid coder and SPIHT coder are comparable, as expected. For the texture-rich images, e.g., Bridge and Mandrill that contain a large amount of significant wavelet coefficients in the middle and high frequency subbands, the hybrid coder is superior to the SPIHT algorithm in general. As indicated by the simulation results, the hybrid coder outperforms SPIHT by 0.38 dB at 0.5 *bpp* for Bridge image, by 0.74 dB at 0.5 *bpp* for Mandrill image.

It has been shown that when the textured images are encoded, 2-D DWT is unlikely to yield many large zero trees due to lack of homogeneous regions. Thus, the advantage of encoding zero trees of insignificant wavelet coefficients by using SPIHT is weakened. On the other hand, the high frequency wavelet trees can be efficiently sought out by using the proposed tree classifier, and then can be encoded by using MVQ to improve the overall compression performance.

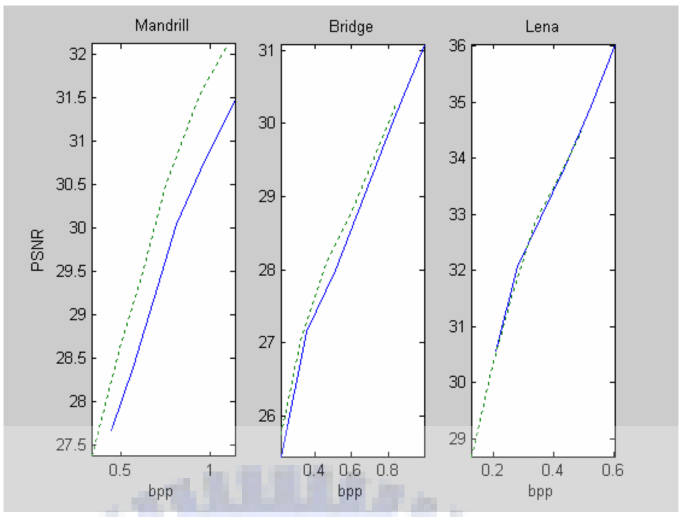


Figure 3.5 Rate-distortion curves of the test images: Mandrill, Bridge and Lena (from left to right) by using the proposed hybrid coder (dotted lines) and SPIHT (solid lines).

Wavelet transform provides an efficient multi-resolution analysis. It decomposes images into subbands with orientation selectivity as well as joint space-frequency localization. Many efficient wavelet coders, e.g. EZW and its improved version: SPIHT have been developed by taking advantage of the following hypothesis: if a wavelet coefficient is insignificant with respect to a given threshold, then all the corresponding wavelet coefficients at the finer resolutions are likely to be insignificant with respect to the same threshold. However, natural images are often composed of textures with rapid variations in grayscales. For such images, there are many significant wavelet coefficients scattered in the higher frequency subbands. Consequently, the coding performances of both EZW and SPIHT are usually not adequate since groups of insignificant wavelet coefficients in the tree structure are very rare. A hybrid image coder by combining SPIHT and MVQ with their respective advantages is proposed. In which, trees with a large amount of significant wavelet coefficients in the higher frequency subbands are to be identified by using a simple tree classifier, and then coded by using a different, more suitable method instead of SPIHT. Experimental results show that the proposed hybrid coding is superior to SPIHT coding for images with textures composed of the middle and high frequency components. It improves the overall compression performance at the cost of additional computations, i.e., the computation of Eq. (3.2) for classifying wavelet trees into two classes, and the computation of norms of MVQ error vectors for determining the MVQ parameters: B_k .

3.2 Image Compression Based on Set-Partitioning Embedded Block Coder and Residual Vector Quantization

A hybrid image coding scheme based on the set-partitioning embedded block coder (SPECK) and residual vector quantization (RVQ) is proposed for image compression. In which, the scaling and wavelet coefficients of an image are coded by using the original SPECK algorithm and the SPECK with RVQ, respectively. The use of hybrid coding strategy by combining SPECK with RVQ for high frequency wavelet coefficients is to take account of the energy clustering property of wavelet transform. Experimental results show that, for gray-level still images, the proposed hybrid SPECK-RVQ coder outperforms SPECK, e.g. the peak-signal-to-noise-ratio (*PSNR*) values can be improved by 1.67 dB and 0.69 dB at compression rate of 1 bit per pixel for the Lena and Barbra images, respectively. The application for chroma subsampling images is also presented in this subsection, and the proposed method usually outperforms color SPECK method. The *PSNR* values can be improved by 1.11 dB for the Y plane, 0.99 dB for the U plane, and 2.31 dB for the V plane at the bit budget of 81,920 bits for the test image Goldhill. In addition to high coding efficiency, the proposed method also preserves the features of embeddness, low computation complexity, and exact bitrate control.

3.2.1 Set-partitioning embedded block coder (SPECK)

The SPECK algorithm [27] is a simple, efficient image coder with coding scalability. In which, two symbols, namely insignificant pixel (IP) and significant pixel (SP) are stored in their respective lists called list of insignificant sets (LIS) and list of significant pixels (LSP). SPECK consists of four steps: initialization, sorting, refinement, and quantization passes. In sorting pass, four functions are performed on sets of two types: ***S*** and ***I*** (Figure 3.6), which are sets of significant pixels and insignificant pixels, and to be partitioned by using quadtree partition (Figure 3.7) and octave band partition (Figure 3.8), respectively. Figure 3.9 shows the flow chart of SPECK.

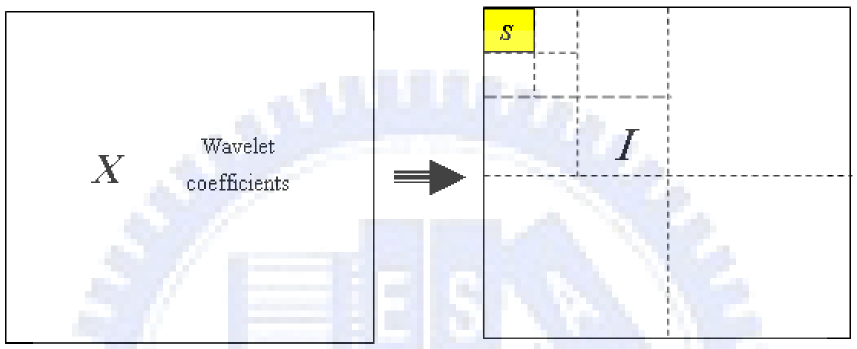


Figure 3.6 Partitioning a transformed image ***X*** into sets ***S*** and ***I***.

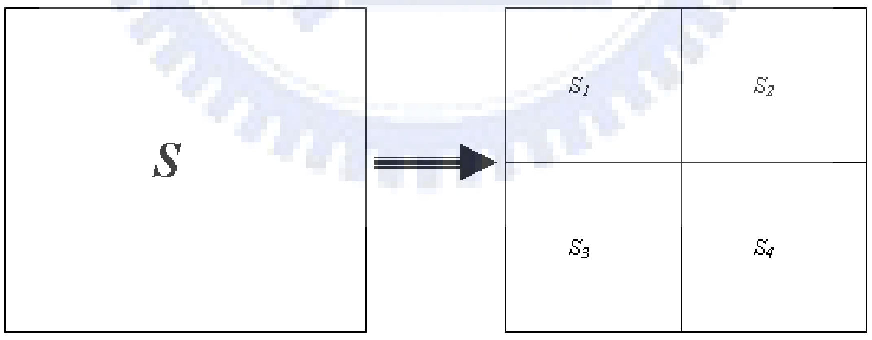


Figure 3.7 Quadtree partition: partitioning set ***S*** into $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, $\boldsymbol{S}_3$, and $\boldsymbol{S}_4$.

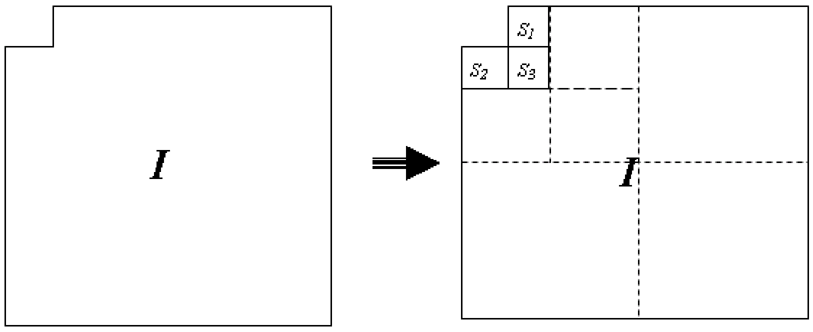


Figure 3.8 Partitioning set ***I*** into ***S***$_1$, ***S***$_2$, ***S***$_3$, and a smaller ***I***.

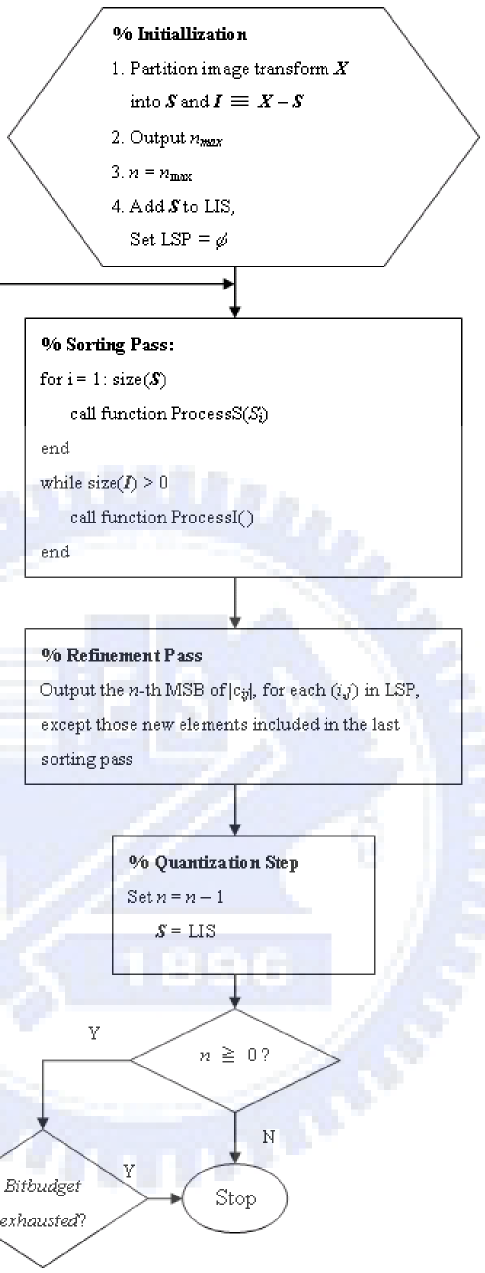
In the initialization pass of SPECK, ***X*** denotes the wavelet transform image and parameter n_{max} is determined by Eq. (3.4):

$$n_{\max} = \left\lfloor \max_{c_{ij} \in \chi} \log_2(|c_{ij}|) \right\rfloor . \tag{3.4}$$

The lowest frequency subband *LL* (Figure 3.17) of a transformed image is taken as ***S***. LIS is set to be equal to ***S***, and LSP is set to be empty.

In the second pass, the significant coefficients with magnitudes in range of $[2^n, 2^{n+1}]$ are identified. If there are significant coefficients found, the corresponding significant maps with sign bits are to be coded efficiently. Functions *ProcessS*( ) (Figure 3.10) and *ProcessI*( ) (Figure 3.11) are in charge of partitioning sets ***S*** and ***I***, respectively, which are to be coded by using their respective functions *CodeS*( ) (Figure 3.12) and *CodeI*( ) (Figure 3.13). The significant test function *SigTest*( ) in procedures *ProcessS*( ), *ProcessI*( ), and *CodeS*( ) is defined by Eq. (3.5):

$$SigTest(A,n) = \begin{cases} 1 & |c_{ij}| \geq 2^n, \quad \forall c_{ij} \in A, \\ 0 & |c_{ij}| < 2^n, \quad \forall c_{ij} \in A. \end{cases} \tag{3.5}$$



% Initiallization
1. Partition image transform $\boldsymbol{X}$ into $\boldsymbol{S}$ and $\boldsymbol{I} \equiv \boldsymbol{X} - \boldsymbol{S}$
2. Output n_{max}
3. $n = n_{max}$
4. Add $\boldsymbol{S}$ to LIS, Set LSP = ϕ
% Sorting Pass:
for i = 1 : size($\boldsymbol{S}$)
call function ProcessS(S_i)
end
while size($\boldsymbol{I}$) > 0
call function ProcessI()
end
% Refinement Pass
Output the n-th MSB of $|c_{ij}|$, for each (i,j) in LSP, except those new elements included in the last sorting pass
% Quantization Step
Set $n = n - 1$
$\boldsymbol{S}$ = LIS
$n \geqq 0$?
Y
N
Bitbudget exhausted?
N
Y
Stop

Figure 3.9 Flow chart of the SPECK algorithm.

SigTest(A,n) = 1 implies that set A is significant, and *SigTest*(A,n) = 0 means set A is insignificant. The elements of $\boldsymbol{S}$ are tested according to the special order of SPECK. If an element of $\boldsymbol{S}$ is insignificant, it is stored in LIS and a coding bit 0 is therefore outputted to the code bitstream. On the other hand, for a significant element of $\boldsymbol{S}$, say $\boldsymbol{S}_i$, if $\boldsymbol{S}_i$ is a point, it is stored in LSP and a coding bit 1 together with the sign bit are outputted to the code bitstream. If $\boldsymbol{S}_i$ is a significant set, output a coding bit 1 and divide $\boldsymbol{S}_i$ into four subsets (Figure 3.7) for locating the significant points in the iterative manner. After the process of $\boldsymbol{S}$, with respect to a given bit-plane n, is finished, the process of $\boldsymbol{I}$ proceeds with respect to the same bit-plane. If $\boldsymbol{I}$ is still insignificant, output a coding bit 0 and proceed the next coding step. If set $\boldsymbol{I}$ becomes significant, it is partitioned into three subsets: $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, and $\boldsymbol{S}_3$ of type $\boldsymbol{S}$ and a smaller $\boldsymbol{I}$ by the use of octave band partition (Figure 3.8). This procedure is performed iteratively until set $\boldsymbol{I}$ becomes empty.

In the third pass of SPECK, which is known as refinement coding pass, the n-th most significant bit of the significant points that had been found in previous coding passes and stored in LSP is updated one bit per point.

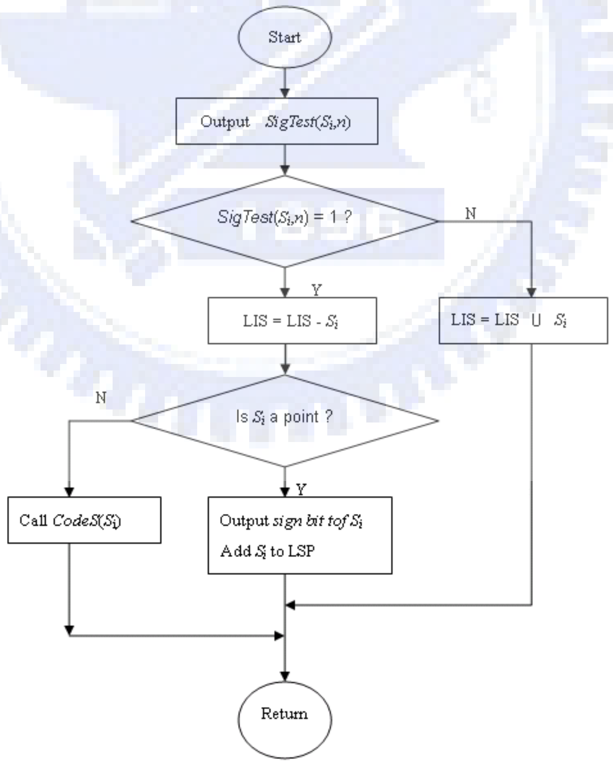


Figure 3.10 The flow chart of procedure *ProcessS*( ).

The last pass of SPECK is the quantization pass. In which, ***S*** is set to be the current LIS and *n* is decreased by one. In addition, if a set of type ***S*** becomes significant and its first three offspring subsets are still insignificant, then the fourth subset must be significant and therefore the coding bits can be omitted for saving bit budget. Moreover, the output bit stream of SPECK consists of three parts: significant maps, sign bits, and refinement bits. The significant maps can be coded by using a context-based arithmetic coder such that the compression performance can be improved at the cost of increasing computational complexity.

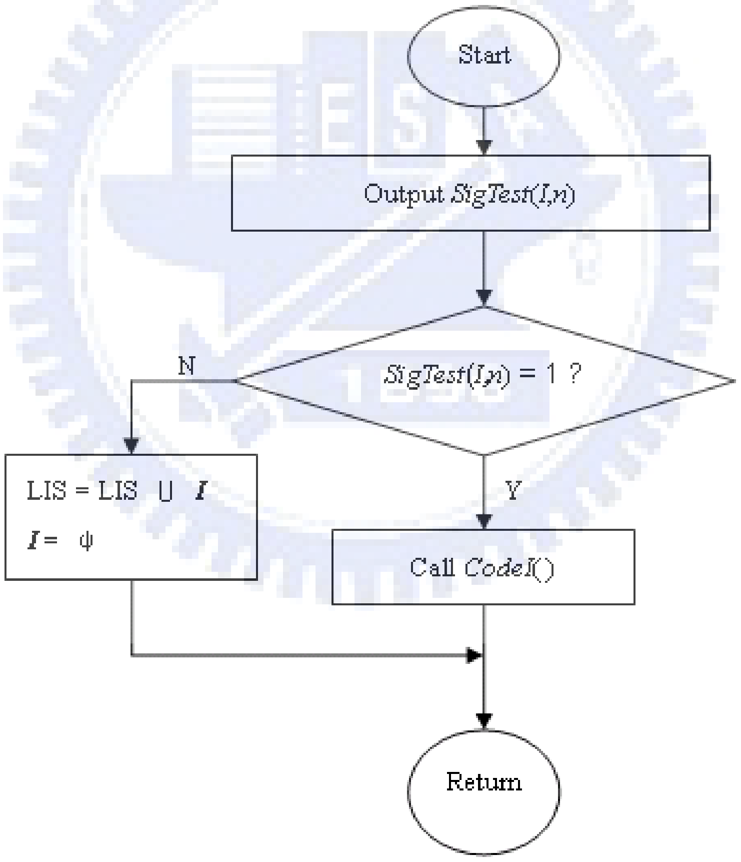


Figure 3.11 The flow chart of procedure *ProcessI*( ).

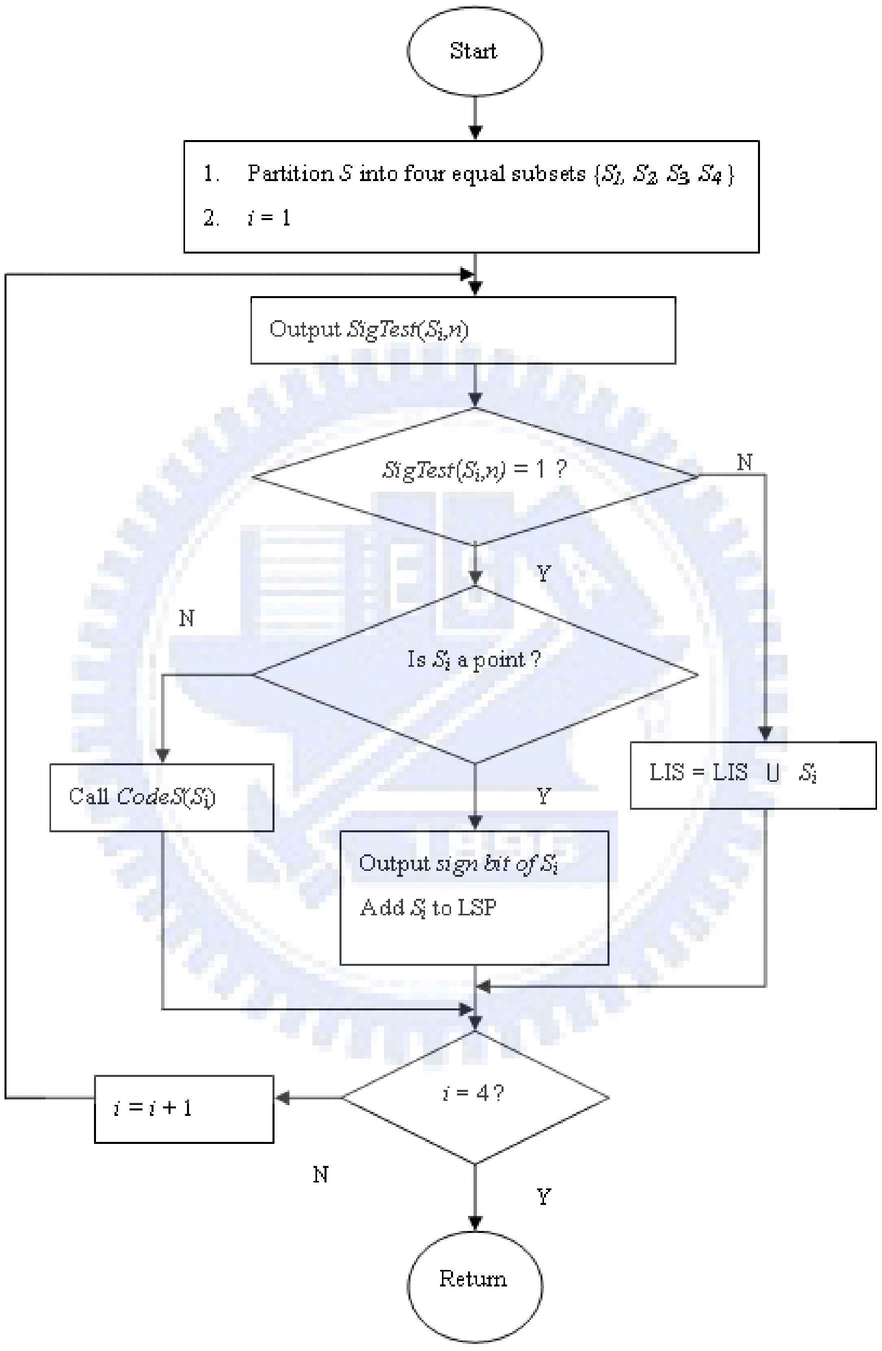


Figure 3.12 The flow chart of procedure *CodeS*( ).

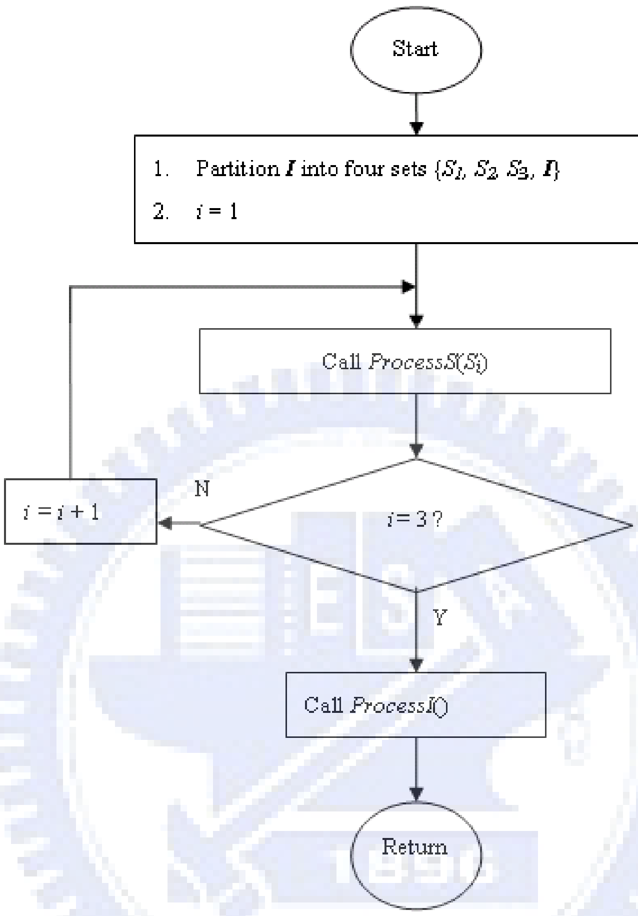


Figure 3.13 The flow chart of procedure *CodeI*( ).

3.2.2 Residual vector quantization (RVQ)

Vector quantization (VQ) plays an important role in many applications, e.g. speech recognition, volume rendering, and image compression. Gupta *et al*. utilized VQ to compress multispectral satellite images [30]. Su *et al*. developed a hybrid coding system by using SPIHT and VQ for image compression in [28]. Abdel-Galil *et al*. applied VQ to power systems for classifying power quality disturbances [31]. A vector quantizer Q with dimension m and size k can be defined as a mapping from a vector in m-dimensional Euclidean space, R^m, into a finite set $\boldsymbol{B}$ with k output elements called code words, which is given by Eq. (3.6):

$$Q: R^m \rightarrow B, \tag{3.6}$$

where $B = \{x_1, x_2, \cdots, x_k\}$ and $x_i \in R^m$, $i = 1, 2, \cdots, k$ and B is called the codebook that has size k. The code rate, resolution, of a quantizer is defined by Eq. (3.7):

$$r = \frac{\log_2 k}{m}, \tag{3.7}$$

which implies the bit number needed to represent a codeword if the codebook is well designed. When the code vector size becomes large, the distortion of the vector quantizer approaches the lower bound of the distortion-rate relation [32]. However, both the computation complexity and memory requirements, associated the vector quantizer, increase exponentially. Hence, an unconstrained full search vector quantizer usually uses small vectors. For reducing the computation complexity and memory requirements of VQ, several variants of the original VQ had been proposed in literature, such as residual vector quantization (RVQ) [33][34], hierarchical VQ [35], and tree-structured VQ (TSVQ) [25]. Each VQ variant makes a compromise between the computation complexity and performance.

RVQ or multistage VQ [36] is a VQ variant with less computation complexity. Because the decoder of a RVQ is constrained by a direct-sum codebook structure and the encoder typically uses a suboptimal stage-sequential search procedure, the RVQ results in performance degradation. The signal flow diagram of a p-stage

RVQ is shown in Figure 3.14, where x_i ($1 \leqq i \leqq p$) is the input vector of the i-th VQ stage in the p-stage RVQ, and $\hat{x}_i$ is the code vector which has the smallest distance to x_i. The residual of $x_i - \hat{x}_i$ is x_{i+1} that is the input vector for the (i+1)-th VQ stage in the RVQ system.

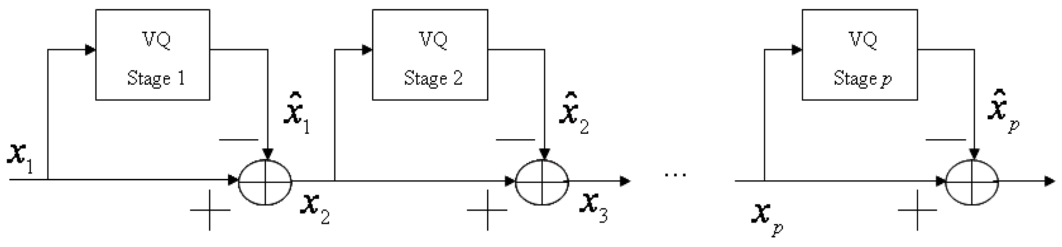


Figure 3.14 The signal flow diagram of a p-stage RVQ.

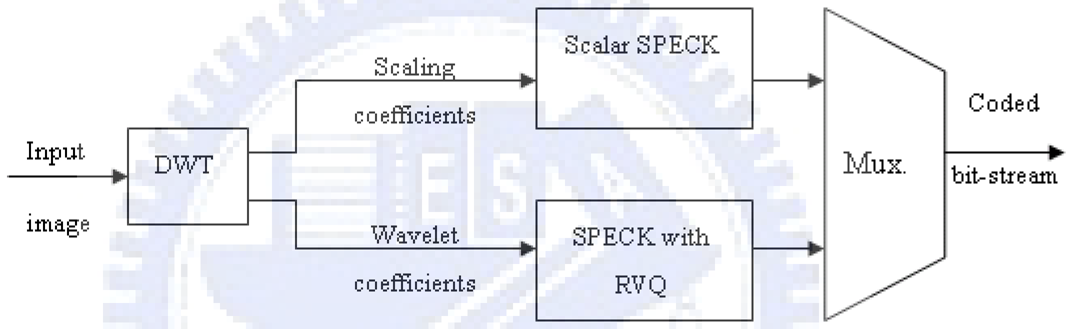


Figure 3.15 The proposed hybrid image coder.

3.2.3 The Proposed Hybrid Image Compression Method

In SPECK, the wavelet transform coefficients of an image are classified into one of two-type blocks, namely ***I*** and ***S***. Block ***I*** denotes acollection of coefficients that are likely to be insignificant with respect to a given threshold value, whereas at least one of the block samples of type ***S*** is likely to be significant with respect to the same threshold value. If block ***I*** becomes significant (with respect to the next smaller threshold value), it is partitioned into three sub-blocks of type ***S*** and one sub-block of type ***I***; this procedure is performed iteratively on the successively smaller ***I*** until it is insignificant or its block size reaches the limit. If block ***S*** becomes significant, it is partitioned into four sub-blocks of type ***S*** for zooming into the high energy areas; this procedure is also performed iteratively until the significant samples are found and thereby coded. Even though wavelet transform possesses the decorrelation property, i.e. most of the correlation between image-pixels can be removed, there may still be some residual correlation between neighboring wavelet coefficients, especially in the high frequency subbands. This residual information also known as the energy clustering of wavelet transform needs to be taken into account to improve the compression performance. Moreover, according to Shannon's theory, vector quantization (VQ) can reduce the coding bits effectively over scalar quantization. Thus, from the compression point of view, the VQ approach, which is expected to be gaining favor, motivates the development of the vector version of SPECK.

A hybrid image coding system by combining SPECK with RVQ is therefore proposed to improve the compression performance, and Figure 3.15 shows the block diagram. In which, the wavelet transform coefficients of the input image are classified into two parts. One is the *LL* subband, which contains the scaling coefficients, and is taken as the initial set ***S***. The other coefficients of ***X*** form the initial set ***I***. Whereas the scaling coefficients representing the lowest frequency component of an image can be coded efficiently by using the original SPECK algorithm, the wavelet coefficients with energy clustering are coded by using the SPECK with RVQ. Finally, the coded bit-stream can be obtained by the use of multiplex operation. The coefficients of *LL* are normalized before feeding into the (original) scalar SPECK, and the small vectors of the wavelet coefficients are normalized such that their respective L-2 norms are less than or equal to one. As the information of the lowest frequency subband *LL* of an image is usually more important than that of the high frequency wavelet subbands, the bitplane resolution involved in the scalar SPECK is therefore higher than that involved in the SPECK with RVQ. Thus, the transmission rate of the scalar SPECK is usually faster than that of the SPECK with RVQ. Based on the simulation results, the transmission rate of the scalar SPECK is set empirically twice of the SPECK-RVQ transmission rate, i.e. two coding bitplanes of the scalar SPECK are outputted with one coding bitplane of the SPECK with RVQ.

In our proposed method, the coefficients of set ***I*** are coded by using the SPECK with RVQ; each vector contains four entries forming a 2× 2 coefficient block. The sorting step performed on these vectors is similar to that of the original SPECK, however, with the smallest block of 2×2 coefficients rather than a scalar coefficient. As a result, the elements of LSP in the RVQ are vectors of 4 entries, and in the refinement pass, the corresponding codewords of the vectors in LSP are outputted into a bitstream. Finally, the compression file which contains the necessary overhead and the bitstream is formed. The decoder of the proposed method can be implemented by simply reverseing the processing steps of the encoder. Besides the overhead of the compression file, the bits in the compression file are ordered in importance, so the proposed method is embedded. The proposed encoder (decoder) can terminate the coding (decoding) process at any point, so it can achieve the exact bit-rate control which is an important requirement of modern codecs.

It is noted that the compression performance can be improved by the use of arithmetic coding after SPECK, however, at the cost of increasing computational complexity. As an example, the *PSNR* value of the decoded 512×512 Lena image can be improved by 0.22 dB at compression rate of 1 *bpp* by using SPECK with arithmetic coding [37]. For system simplicity, the operation of arithmetic coding is not performed in our experiments.

3.2.4 Experimental results and conclusions

The compression performance of the proposed hybrid image coder is compared to the scalar SPECK in terms of the *PSNR-bpp* curves. The compression rate is measured in bits per pixel (*bpp*), and the peak signal to noise ratio (*PSNR*) measured in dB is utilized to evaluate the decoded image quality. Computer simulations are performed on several monochrome images of 256×256 pixels. A set of 41 images is used to train the codebooks of the RVQ, and the test images shown in Figure 3.16 are not included in the training images.

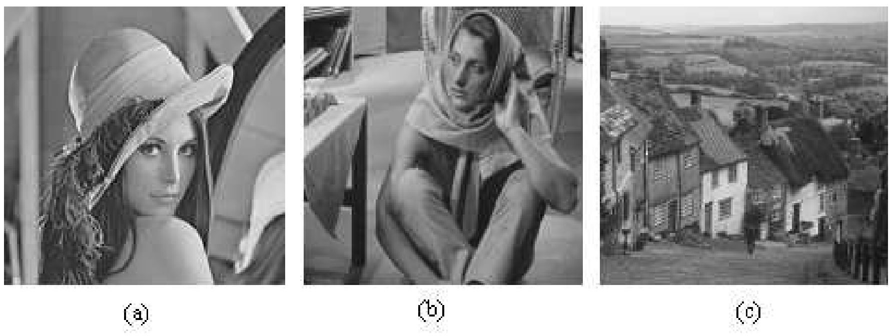


Figure 3.16 Three 256×256 gray-level test images: (a) Lena, (b) Babara, (c) Goldhill.

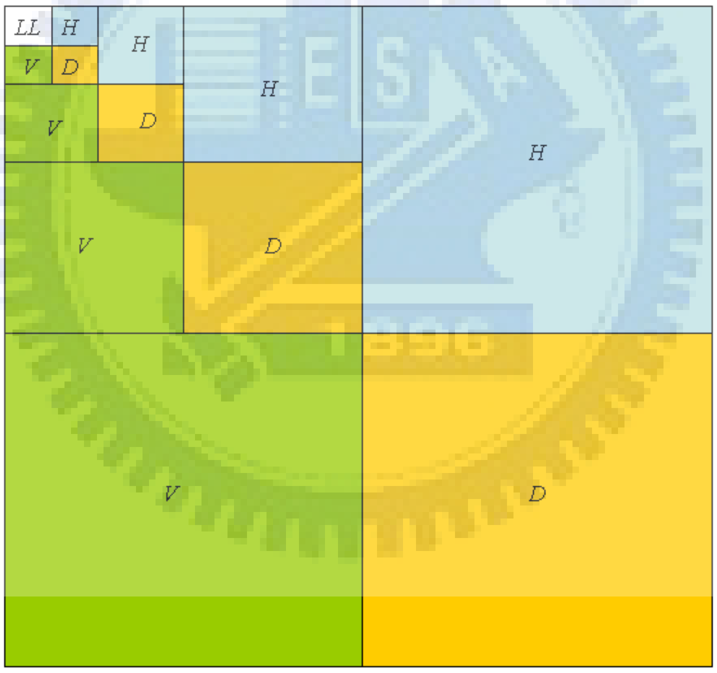
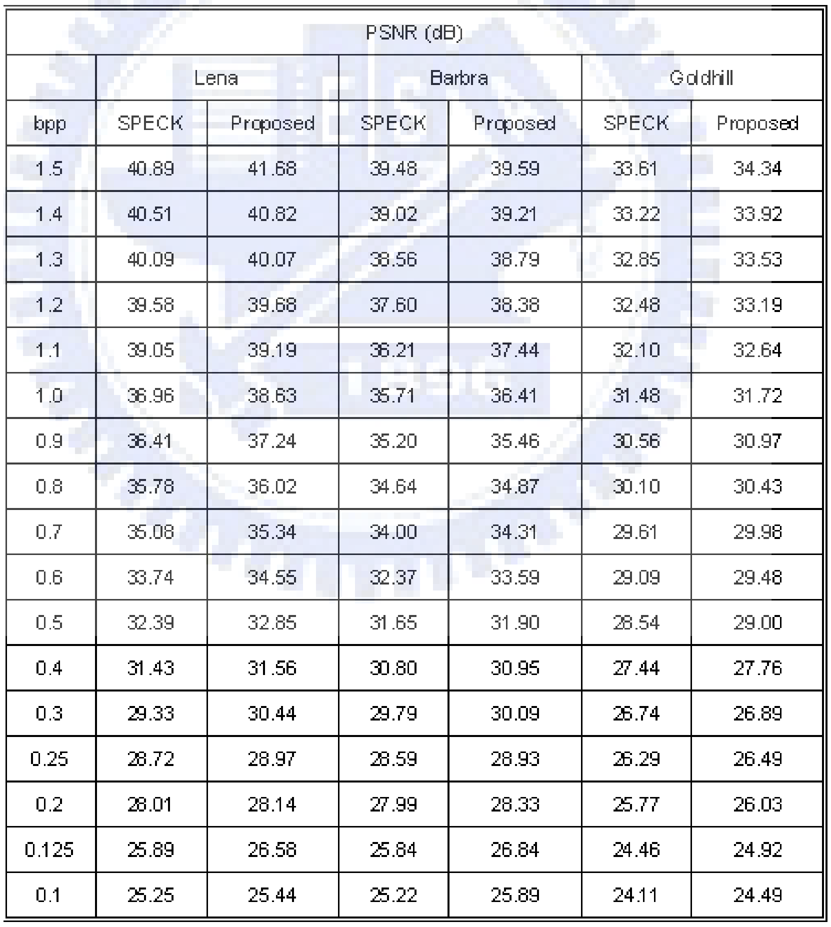


Figure 3.17 The partition and assignment of a 4-decomposition-leveltransformed image.

Linear phase biorthogonal wavelet filters with 9/7-coefficients are used. The number of wavelet decomposition levels is 4. Figure 3.17 depicts the wavelet decomposition of an image into subbands of four types: *LL*, *H*, *V*, and *D*. The lowest frequency coefficients in subband *LL* are normalized such that their respective magnitudes are in the range of [0, 1], which are coded by using the scalar SPECK. The wavelet coefficients in subbands of types *H*, *V*, and *D* are coded by using SPECK with RVQ. For the coefficients of set ***I***, the number of threshold values in the RVQ part is 9. It is noted that the characteristics of subbands of types *H*, *V*, and *D* are different, each type of subbands are therefore coded with their respective codebooks. A total number of 30 codebooks are constructed by using the K-means algorithm. The codebook size of the first decomposition level is 64, and that of the other decomposition levels is 32.

Table 3.1 The results of SPECK and the proposed method for 3 test images.



PSNR (dB)						
	Lena		Barbra		Goldhill	
bpp	SPECK	Proposed	SPECK	Proposed	SPECK	Proposed
1.5	40.89	41.68	39.48	39.59	33.61	34.34
1.4	40.51	40.82	39.02	39.21	33.22	33.92
1.3	40.09	40.07	38.56	38.79	32.85	33.53
1.2	39.58	39.68	37.60	38.38	32.48	33.19
1.1	39.05	39.19	36.21	37.44	32.10	32.64
1.0	36.96	38.63	35.71	36.41	31.48	31.72
0.9	36.41	37.24	35.20	35.46	30.56	30.97
0.8	35.78	36.02	34.64	34.87	30.10	30.43
0.7	35.08	35.34	34.00	34.31	29.61	29.98
0.6	33.74	34.55	32.37	33.59	29.09	29.48
0.5	32.39	32.85	31.65	31.90	28.54	29.00
0.4	31.43	31.56	30.80	30.95	27.44	27.76
0.3	29.33	30.44	29.79	30.09	26.74	26.89
0.25	28.72	28.97	28.59	28.93	26.29	26.49
0.2	28.01	28.14	27.99	28.33	25.77	26.03
0.125	25.89	26.58	25.84	26.84	24.46	24.92
0.1	25.25	25.44	25.22	25.89	24.11	24.49

Table 3.1 shows the simulation results of the test images. Meanwhile Figs. 3.18-3.20 show their respective *PSNR-bpp* curves, where the horizontal and vertical axes are the compression rates in *bpp* and *PSNR* values in dB, respectively. For the monochrome image Lena, the proposed hybrid coder outperforms the SPECK coder by 1.67 dB at 1.0 *bpp*, and 0.48 dB, on average, from 0.1 *bpp* to 1.5 *bpp*. For the monochrome image Barbra, the proposed hybrid coder outperforms the SPECK coder by 1.23 dB at 1.1 *bpp*, and 0.49 dB on average. For the monochrome image Goldhill, the proposed hybrid coder outperforms the SPECK coder by 0.73 dB at 1.5 *bpp*, and 0.43 dB on average. The experimental results of more test images obtained from the USC database are shown in Figure 3.21. In which, the curve denotes the average improvement by using the proposed hybrid coder compared to the pure SPECK coder. It is shown that the proposed hybrid coder is preferable to the SPECK coder in terms of the *PSNR-bpp* curves.

Wavelet transform has been adopted by JPEG2000 due to many desirable properties, e.g. multi-resolution representation, embedded coding and progressive transmission. For a texture-rich image, the high frequency wavelet coefficients are likely to become significant after few coding passes of the SPECK algorithm, which degenerate the compression performance. These high frequency wavelet coefficients representing the high-detail textures of an image can be coded efficiently by exploiting the energy clustering of wavelet transform, which motivates the development of hybrid coding strategy. In this subsection, a hybrid image coder using SPECK (CSPECK for color images) and RVQ is proposed. In which, the lowest frequency scaling coefficients of an image are coded by using the original scalar SPECK (CSPECK), whereas the high frequency wavelet coefficients are coded by using SPECK (CSPECK) with RVQ. Experimental results show that the proposed hybrid coder outperforms SPECK for a broad range of compression rates.

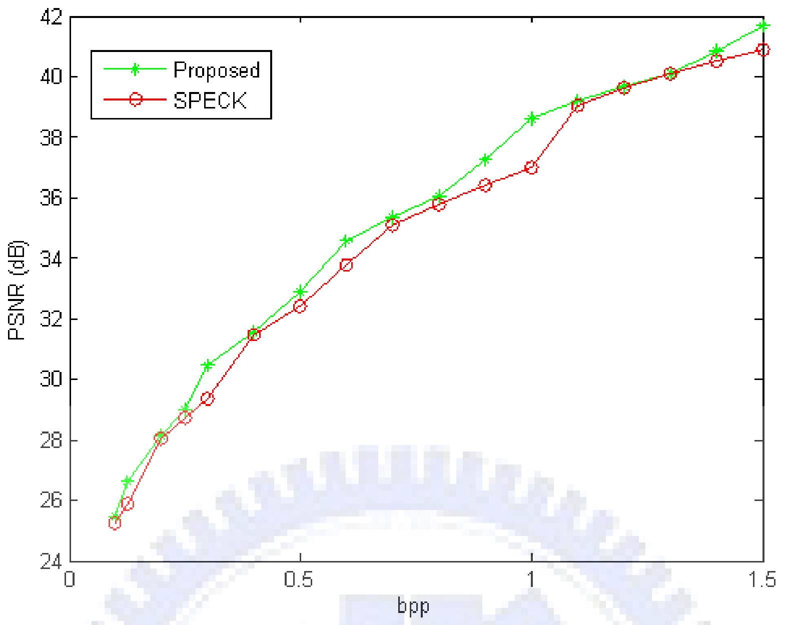


Figure 3.18 The experimental results of the gray-level image Lena.

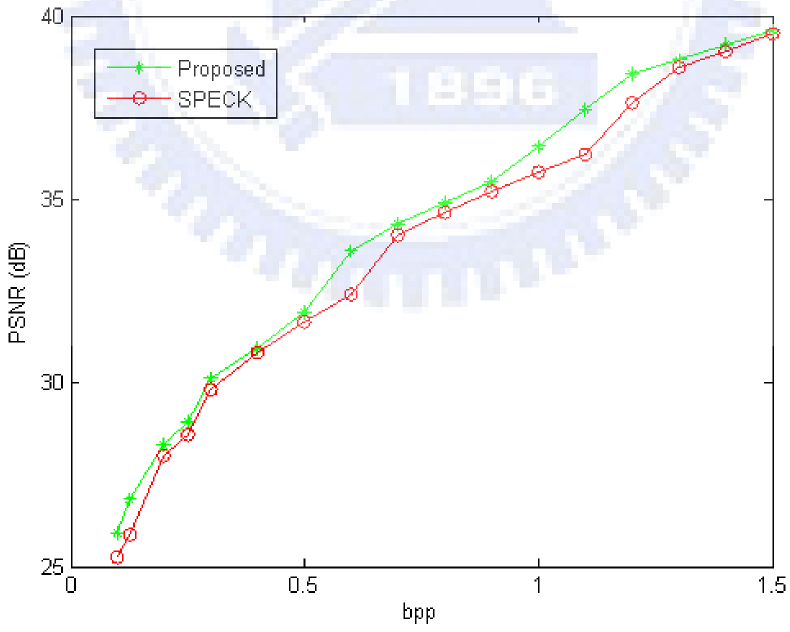


Figure 3.19 The experimental results of the gray-level image Barbara.

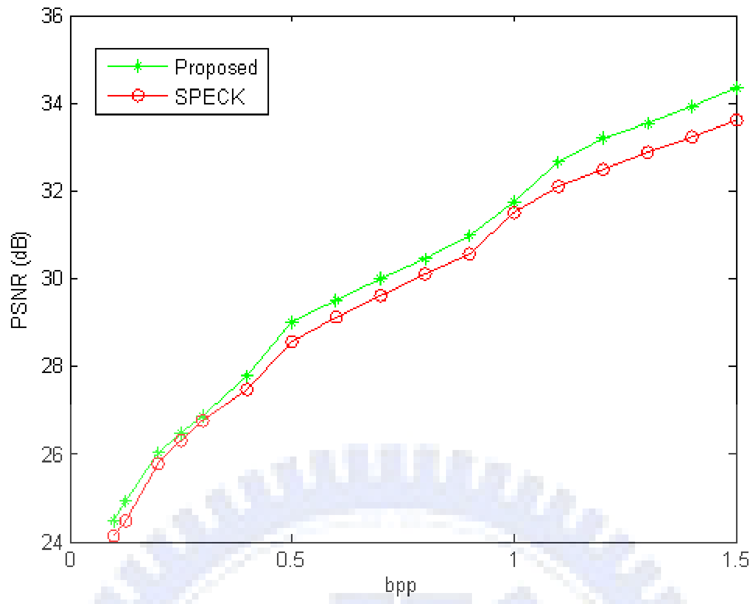


Figure 3.20 The experimental results of the gray-level image Goldhill.

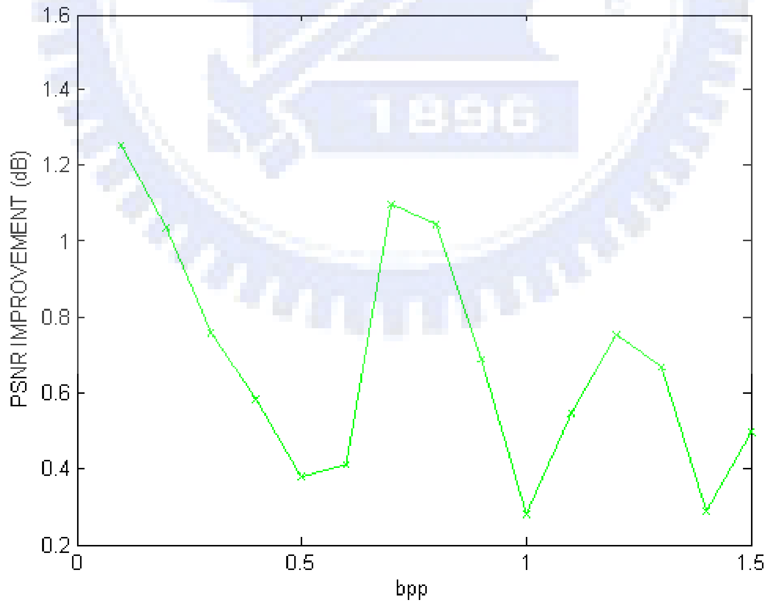


Figure 3.21 The average improvements of the proposed hybrid coder compared to the original SPECK on more test images.

CHAPTER 4

THE PROPOSED METHODS

In this chapter we will propose two methods. One is called lifting-shape-adaptive DWT (LSA-DWT) and the other is named as shape-direction-adaptive DWT (SDA-DWT). LSA-DWT is different from SA-DWT for that the former uses flexible lifting DWT to simplify implementation and shorten computation time. The major proposed method, SDA-DWT, can well exploit the correlation of texture orientation and segment shape in object image to contribute higher compression efficiency in object-based image compression.

For the lifting structure, each finite impulse response (FIR) wavelet filter is factored into several pairs of lifting steps (Figs. 2.1, 2.6, and 2.7). One pair of lifting steps includes a prediction step followed by one update step. In this dissertation, we only use the lifting 5/3 wavelet to implement the proposed methods, and the lifting 5/3 wavelet can be realized with only one pair of lifting steps. The block diagram in Figure 2.1 shows the structure of the lifting-based DWT containing one prediction step and one update step. For one-dimension lifting-based DWTs, the input samples are classified into two categories (even and odd) first. Then, each odd sample is predicted by some specific even-neighbor samples (This depends on the wavelet type.), and replace each original odd sample with the residual, which is obtained from subtracting the odd sample by the prediction value. After the prediction step, in the following update step, each of the even samples is replaced by the result that obtained by adding its original value and the updated value generated from its odd-neighbor samples. Note that, in the update step, the odd samples are not the original inputs, and they have been changed in the preceding prediction step. Finally, the outputs are downsampled to produce the low frequency subband and the high frequency subband and complete a lifting-based DWT with one pair of lifting steps. The lifting-based IDWT can be implemented by reversing the steps in the corresponding lifting-based DWT.

For the 5/3 wavelet used in proposed methods, Eqs. (4.1) and (4.2) are corresponding to the prediction and update steps, respectively:

$$y(2k+1) := x_{ext}(2k+1) - \tfrac{1}{2}[x_{ext}(2k) + x_{ext}(2k+2)], \tag{4.1}$$

$$y(2k) := x_{ext}(2k) + \tfrac{1}{4}[y(2k-1) + y(2k+1) + 2], \tag{4.2}$$

where $x(k)$ is the 1-D input data, $x_{ext}(k)$ means the symmetric extended version of $\mathbf{x}$. The equation y := K means that assign y as K. Signals **s** and **d** are the lowpass and highpass output signals of the lifting DWT. Assume that

$$\mathbf{x} = \{1, 2, 3, 4, 5\},$$

and then we have

$$\mathbf{x}_{ext} = \{\cdots, 4, 3, 2, 1, 2, 3, 4, 5, 4, 3, 2, \cdots\}.$$

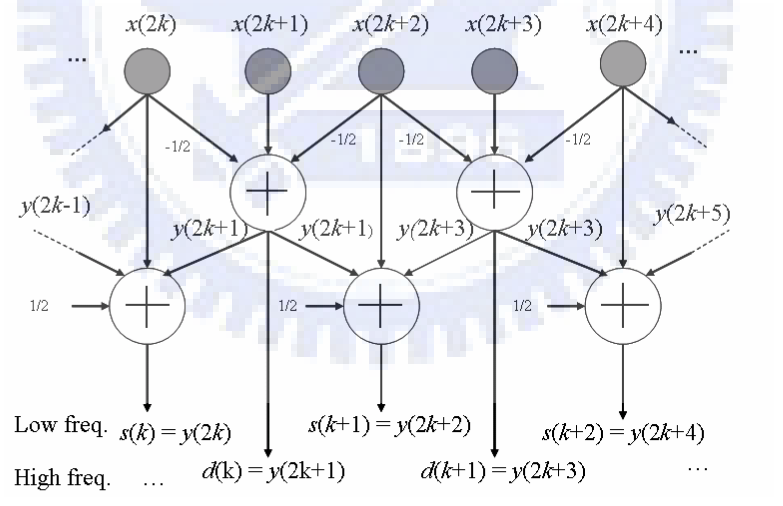


Figure 4.1 The structure of a lifting-based one-dimensional 5/3- wavelet DWT.

The operations corresponding to Eqs. (4.1) and (4.2) can be also represented as Figure 4.1. The range of k is: $0 \leqq k \leqq$ (length of $\mathbf{x}$)/2, assuming the length of $\mathbf{x}$ is even. Equations

(4.3) and (4.4) are two down-sampling relations used to generate the low frequency subband output $s(k)$ and the high frequency subband output $d(k)$, respectively:

$$s(k) := y(2k), \tag{4.3}$$

$$d(k) := y(2k+1), \tag{4.4}$$

The lifting-based IDWT of the 5/3 wavelet is described by Eqs. (4.5)-(4.8):

$$y(2k) := d(k), \tag{4.5}$$

$$y(2k+1) := s(k), \tag{4.6}$$

$$\hat{x}(2k) := y_{ext}(2k) - \tfrac{1}{4}[y_{ext}(2k-1) + y_{ext}(2k+1) + 2], \tag{4.7}$$

$$\hat{x}(2k+1) := y_{ext}(2k+1) + \tfrac{1}{2}[x(2k-1) + x(2k+2) + 2], \tag{4.8}$$

where $\mathbf{y}_{ext}$ is the symmetric extended version of $\mathbf{y}$ and $\hat{\mathbf{x}}$ is the reconstruction of the original signal $\mathbf{x}$. The lifting-based IDWT begins from applying Eqs. (4.5) and (4.6) for upsampling $\mathbf{d}$ and $\mathbf{s}$ to produce $\mathbf{y}$. Then, the even samples of $\mathbf{x}$ are reconstructed by using Eq. (4.7). Finally, Eq. (4.8) is used to reconstruct the odd samples of $\mathbf{x}$. The flow chart in Figure 4.2 shows the relation among Eqs. (4.1)-(4.8) and the steps in DWT and IDWT of a lifting 1-D 5/3 DWT. Follow the steps in Figure 4.2, and a lifting 1-D 5/3 DWT (IDWT) can be implemented by software easily. For a 2-D data matrix (e.g. a graylevel image), apply 1-D lifting 5/3 DWT on the row data, and then apply it again on the new column data. Thus, we have done a 2-D lifting 5/3 wavelet transform on the 2-D data matrix. The corresponding 2-D IDWT can be realized by reversing the steps in forward DWT.

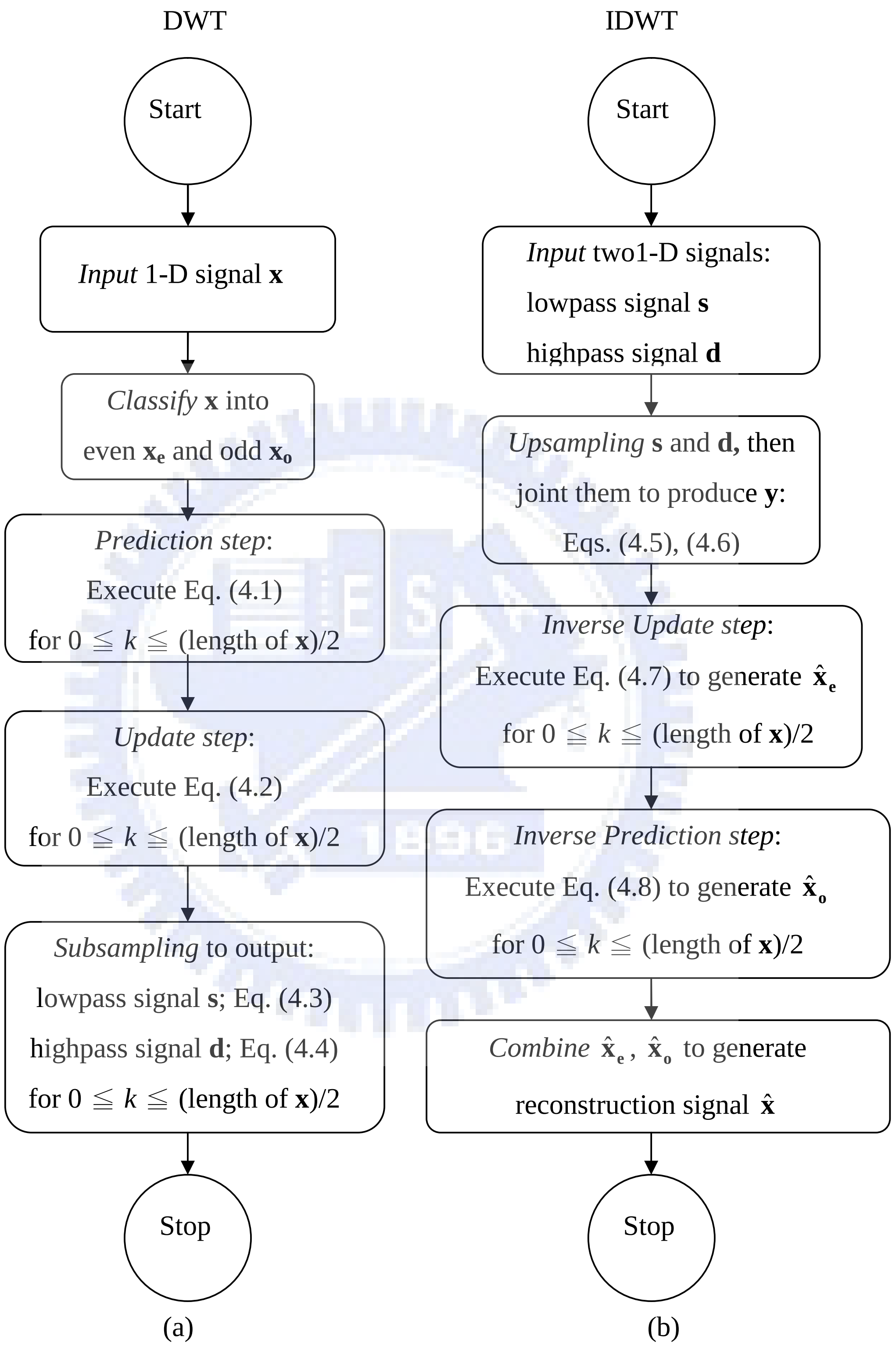

DWT
Start
Input 1-D signal **x**
Classify **x** into even $\mathbf{x_e}$ and odd $\mathbf{x_o}$
Prediction step: Execute Eq. (4.1) for 0 ≦ *k* ≦ (length of **x**)/2
Update step: Execute Eq. (4.2) for 0 ≦ *k* ≦ (length of **x**)/2
Subsampling to output: lowpass signal **s**; Eq. (4.3) highpass signal **d**; Eq. (4.4) for 0 ≦ *k* ≦ (length of **x**)/2
Stop
(a)
IDWT
Start
Input two1-D signals: lowpass signal **s** highpass signal **d**
Upsampling **s** and **d,** then joint them to produce **y**: Eqs. (4.5), (4.6)
Inverse Update step: Execute Eq. (4.7) to generate $\hat{\mathbf{x}}_e$ for 0 ≦ *k* ≦ (length of **x**)/2
Inverse Prediction step: Execute Eq. (4.8) to generate $\hat{\mathbf{x}}_o$ for 0 ≦ *k* ≦ (length of **x**)/2
Combine $\hat{\mathbf{x}}_e$, $\hat{\mathbf{x}}_o$ to generate reconstruction signal $\hat{\mathbf{x}}$
Stop
(b)

Figure 4.2 Flow charts of lifting 1-D 5/3 wavelet DWT and IDWT: (a) DWT, (b) IDWT.

4.1 Lifting-Based Shape-Adaptive DWT (LSA-DWT)

The works of [1] and [2] used the convolution-based DWT, and both of them involved complicated computation. Here we propose to use lifting DWT and global even-odd relation to implement the shape-adaptive DWT which is called LSA-DWT. Although using the global even-odd relation sometimes will leave some energy in the high frequency subband, it is simple for implementing LSA-DWT and can preserve the spatial correlation.

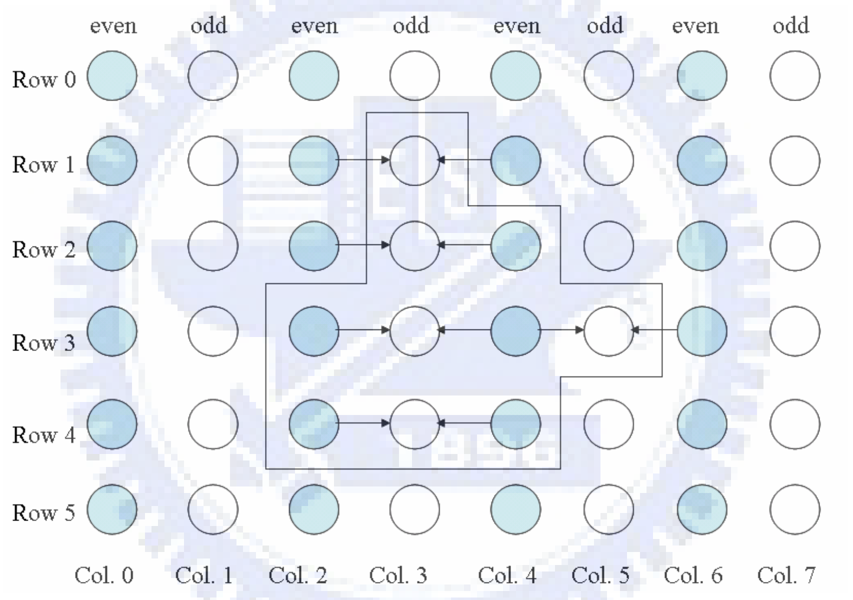


Figure 4.3 An arbitrarily shaped segment and the relation of its even and odd pixels in the prediction step of the 1-D row direction lifting-based DWT.

Figure 4.3 shows an arbitrarily shaped segment which contains 10 pixels in a 6-by-8 image, and it also shows the relation of even and odd pixels in the prediction step. In Figure 4.3, the two arrows, pointing to each odd pixel, is used to indicate that the odd pixel’s two nearest even neighbors in the same row which will be used to predict the odd pixel. For the 5/3-wavelet DWT, the prediction value of each odd pixel in the lifting-based horizontal 1-D DWT is the mean value of its right and left neighbors. After the prediction step, the residual that each odd pixel subtracts its prediction value is stored in the position of the odd pixel. If the even neighbor does not exist in the segment, symmetric extension is used to generate the new even pixel value for prediction. For the single point in a row (e.g. the pixel at row 1 and column 3), its two neighbors for prediction are set to zero. Figure 4.4 shows the update stage of the arbitrarily shaped segment when the lifting-based horizontal 1-D 5/3 DWT is performed on the segment. Each even pixel in the segment is updated by using Eq. (4.2) in the update step, and the corresponding pixels (coefficients) are its left and right odd neighbors. The processing of symmetric extension and single points are the same as the methods in the prediction step. The last step of the lifting-based 1-D DWT is subsampling step by which the transformed 1-D data are classified into the high frequency and the low frequency subbands, and the result is shown in Figure 4.5. Then, the transformed image in Figure 4.5 is transformed by the lifting-based vertical 5/3-wavelet transform. The 1-D lifting-based vertical 5/3-wavelet DWT is just like the processing methods in the 1-D lifting-based horizontal 5/3-wavelet DWT.

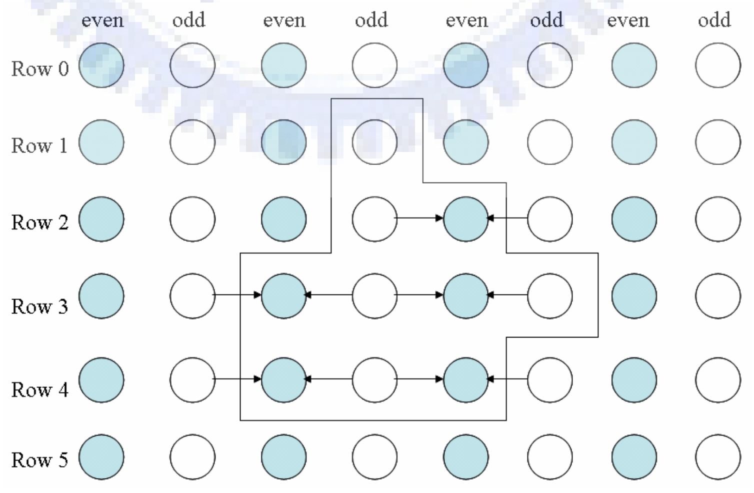


Figure 4.4 An arbitrarily shaped segment and the relation of its even and odd pixels in the update step of the 1-D horizontal lifting-based DWT.

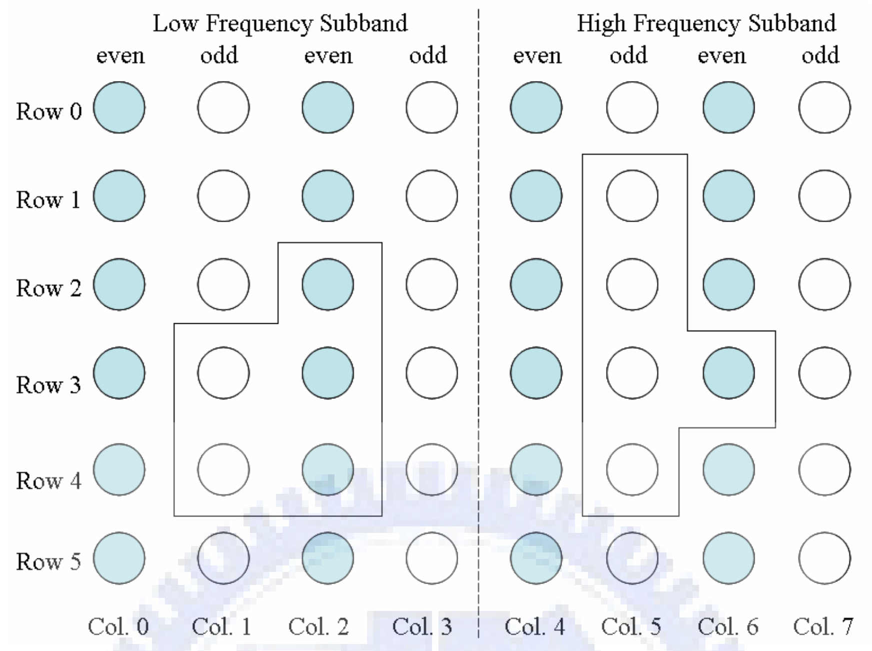


Figure 4.5 The subsampling result of the arbitrarily shaped segment in Figure 4.3.

4.2 Shape-Direction-Adaptive DWT (SDA-DWT)

Shape-adaptive and direction-adaptive functionalities are two important improvements of DWT, and the experimental results in [1]-[4] show that they are very efficient for still image coding. The DAL-DWT [3] and the DA-DWT [4] were designed for processing rectangular images, so they can not process an arbitrarily shaped segment directly. On the other hand, The SA-DWT [1] and the method proposed in [2] can process arbitrarily shaped object, but they do not offer the direction-adaptive functionality. In this dissertation, we propose a new DWT which has both the shape-adaptive and direction-adaptive abilities, and we call it the shape-direction-adaptive DWT [38]. The inputs of the SDA-DWT, proposed in this dissertation, are the image containing the object (segments) to be transformed and the corresponding shape mask (alpha map), and the outputs are the transformed image containing these segments and the corresponding shape masks after SDA-DWT.

4.2.1 Illustrations of SDA-DWT

The proposed SDA-DWT can be described by using Figs. 4.6-4.11, and the same arbitrarily shaped segment in Figs. 4.3 or 4.4 and the 5/3 wavelet are used for illustration. Compared to the LSA-DWT step in Figure 4.3, the corresponding SDA-DWT step is shown in Figure 4.6. In Figure 4.6, assume that the 45-degree direction is selected, so each odd pixel in the segment is predicted by two nearest even neighbors on the 45-degree line passing through this pixel. Then, each odd pixel is replaced by the residual obtained from subtracting the pixel value to the prediction value. If the prediction is good, the residual will be a small value. In the prediction step, symmetric extension method is used for generating those even samples not in the segments, and the symmetric relation is to the perpendicular line, passing through the odd pixel to be predicted, of the 45-degree line. According to Eq. (4.2), the update step in Figure 4.7 is corresponding to Figure 4.4 of LSA-DWT, and every even sample in the segment is updated by its two nearest odd neighbors (They have already been replaced by the residual values in the previous prediction step.) on the 45-degree line. After performing a pair of lifting steps (i.e. a prediction and an update steps), the transformed image is subsampled, and the result is shown in Figure 4.8. The subsampling process is the same as the conventional horizontal subsampling method, and the subsampled coefficients are classified into the low-frequency subband and the high-frequency subband.

When the horizontal subsampling step is complete, the second part (corresponding to the vertical conventional 1-D DWT) of the SDA-DWT begins from a prediction step (the second prediction step in SDA-DWT). Each odd sample in columns of the segment is predicted by its upper (right) and lower (left) even neighbors on the 45-degree line compared to the vertical line (Figure 4.9). Then, the second update step of SDA-DWT is performed on the even samples in columns of the segment (Figure 4.10). Finally, a conventional subsampling along the vertical direction is performed on the coefficients of Figure 4.10, and the image is transformed and divided into four subbands *LL*, *HL*, *LH*, and *HH* subbands (Figure 4.11). The symmetrical extension is used to generate the even samples and odd samples, not in the segment, for prediction and update, respectively. From Figure 4.6 through Figure 4.11, the one-level SDA-DWT is performed, and the *LL* subband can be used to be further transformed. The new alpha map is generated by subsampled the input shape mask along the horizontal direction and followed by a vertical subsampling step.

The flow chart in Figure 12 shows an *l*-decomposition-level SDA-DWT. Assume that a digital image, its object shape mask, and the decomposition number *l* are given. First of all, the object segmented by using the shape mask has to be partitioned, and then the direction of each partition segments is determined in the second step. For the third step, those connected segments with the same direction are combined to form a large segment. Then, one-decomposition-level SDA-DWT is performed on the visual object image, and the parameter *l* is decreased by one. If *l* is not equal to 0, the low-frequency subband *LL* of the transformed image is used for further decomposition until *l* equals to 0. In every decomposition level, the new direction of each partitioned segment should be determined in advance.

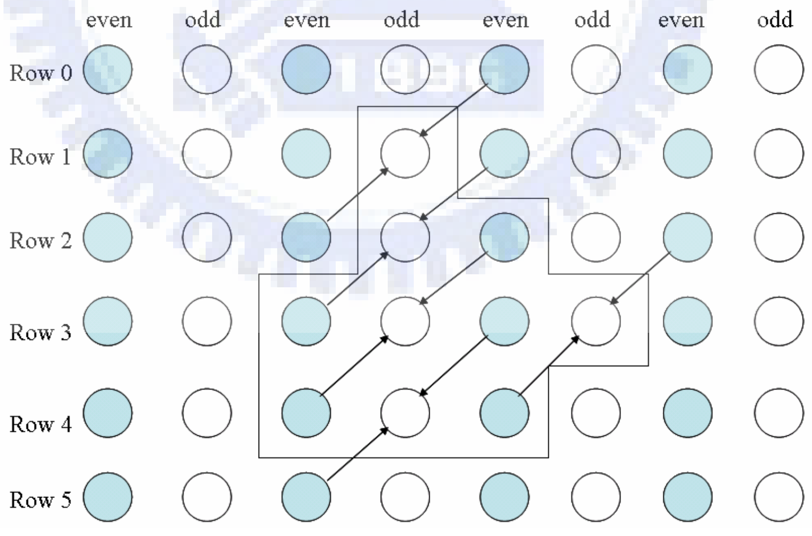


Figure 4.6 The first prediction step of the 2-D shape-direction-adaptive DWT (θ = 45^o) performed on an arbitrarily shaped segment.

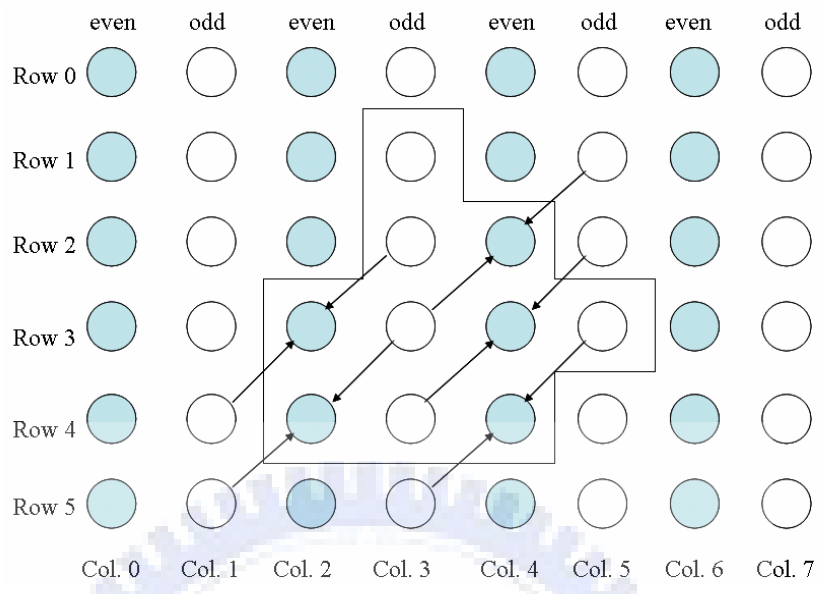


Figure 4.7 The first update step of the 2-D shape-direction-adaptive DWT ($\theta = 45^o$) performed on an arbitrarily shaped segment.

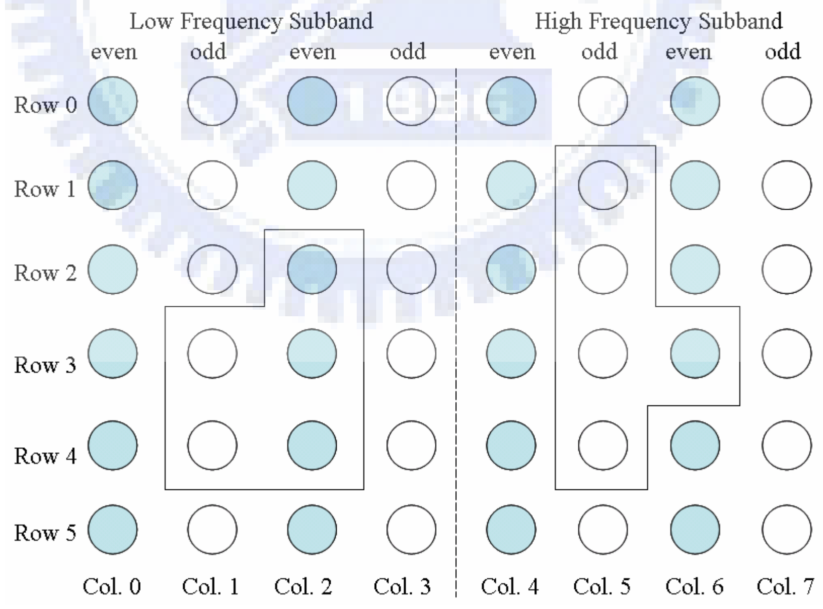


Figure 4.8 The horizontal subsampling result of Figure 4.7 in SDA-DWT.

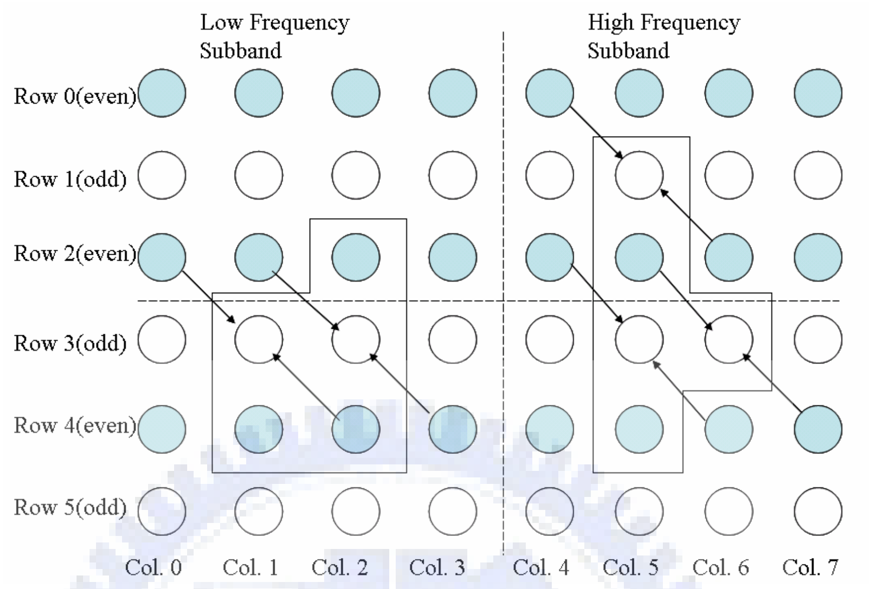


Figure 4.9 The second prediction step of the 2-D SDA-DWT on an arbitrarily shaped segment.

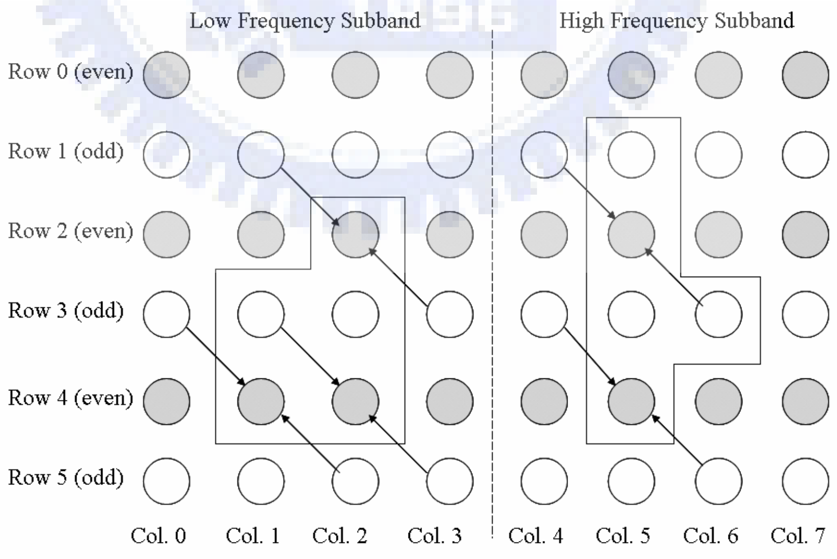


Figure 4.10 The second update step of the 2-D SDA-DWT on an arbitrarily shaped

segment.

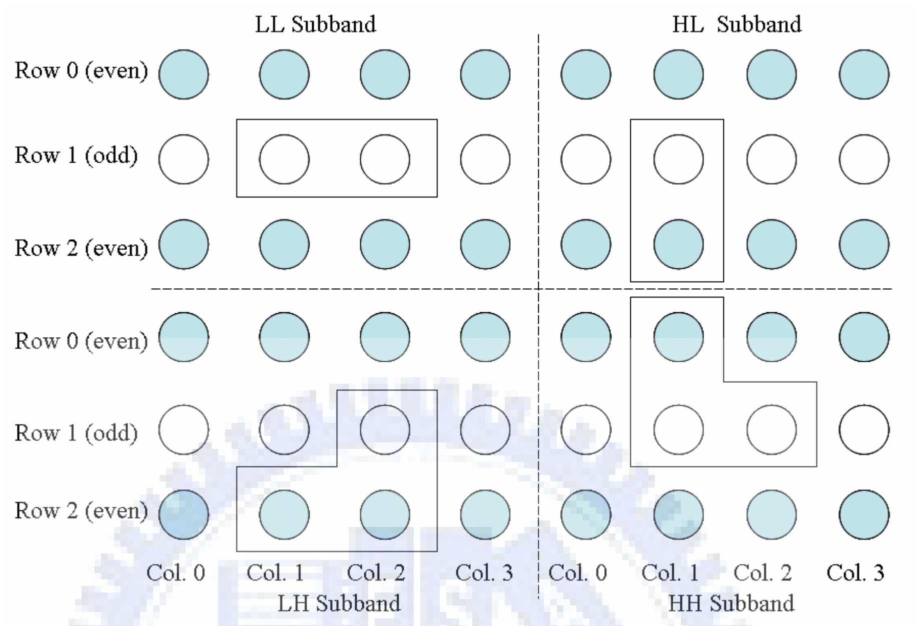


Figure 4.11 The vertical subsampling result of Figure 4.10 in SDA-DWT.

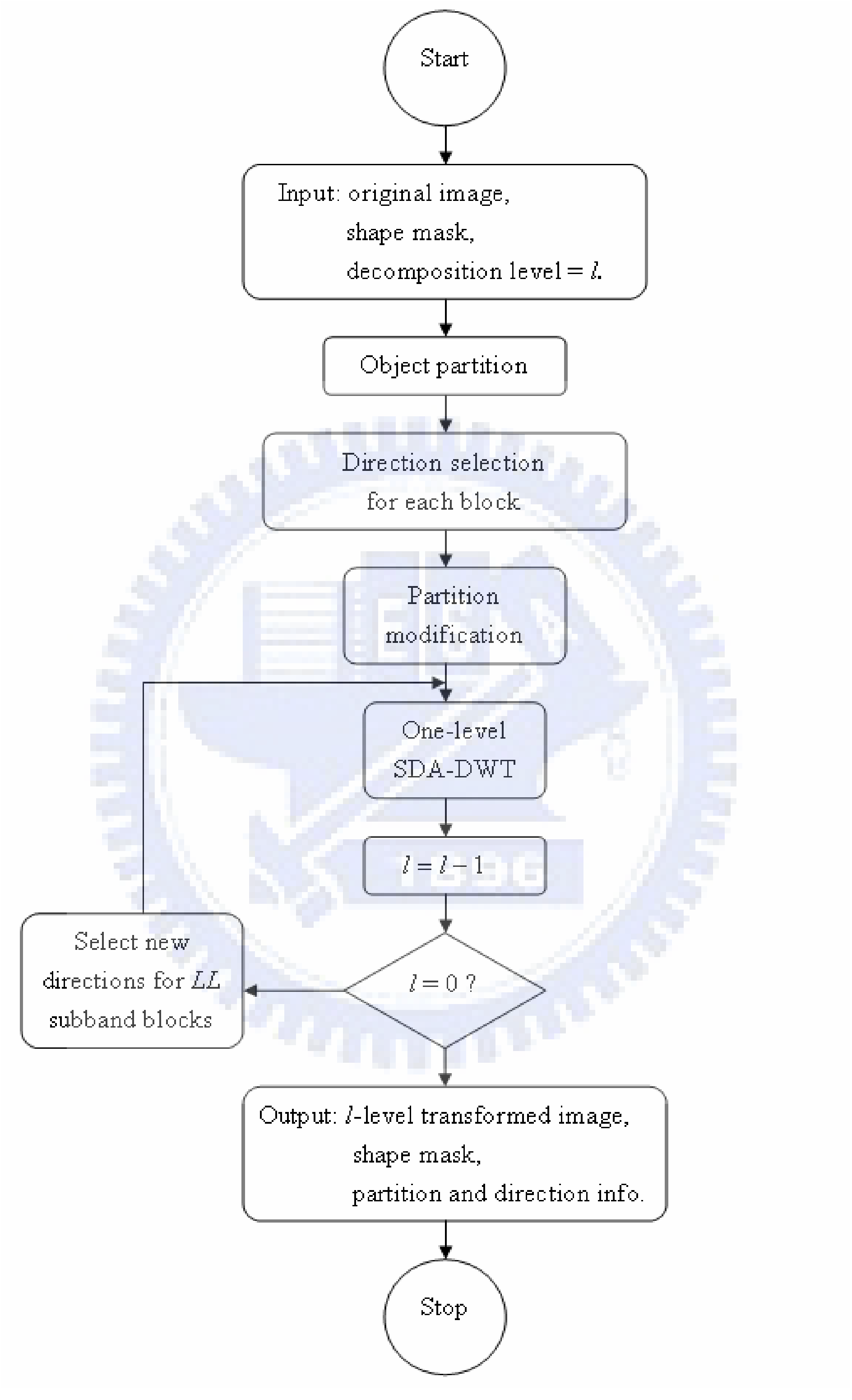


Figure 4.12 The flow chart of a multilevel SDA-DWT.

4.2.2 Filter direction determination in SDA-DWT

There are nine direction selections (as shown in Figure 2.12) in 1-D SDA-DWT. The filter direction in SDA-DWT should be selected to minimize the distortion of the reconstruction object image, for object-based image compression, under a bit-budget. The bit budget is used for representing the side information, which include the partition and filter directions of every partition segment, the object shape mask, and the transformed coefficients. Hence, to find the best filter direction is equivalent to find the direction which results in the least amount of bits for a specified distortion. In the dissertation, the absolute sum of high frequency subband (*HH*) coefficients is used as an approximation objection function for determining the filter direction in SDA-DWT. The filter direction which has the smallest absolute high-frequency subband coefficient sum is the selected filter direction in the segment.

Since the absolute sum of high-frequency subband coefficients is used as an object function, the computation time for filter direction determining is increased when the segment size is large and the direction choices are many. For reducing the time to select the filter direction in each object segment, we can only do the prediction step computations without doing the update step computations, since the high-frequency subband coefficients epend only on the prediction step in the lifting 5/3 DWT.

4.2.3 The importance of shape-adaptive and direction-adaptive functionalities in object-based image compression

For object-based image compression, the visual object images are usually neither rectangular nor with height and width which are powers of 2. If the convolution-based DWT is used for object image compression, then two techniques are often adopted in early days. One method is to pad zeros to become a rectangular image that can be processed by the convolution-based DWT, and the other method is to reallocate the object pixels for applying the convolution-based DWT. For example, we can pad 6 zeros (or other predefined values) to the object in Figure 4.3 to form a 4×4 image, or we can rearrange the object to be a 1-D signal with 10 discrete samples. It is clear that, both methods are not efficient. Therefore, SA-DWT and LSA-DWT were designed for object-based image compression.

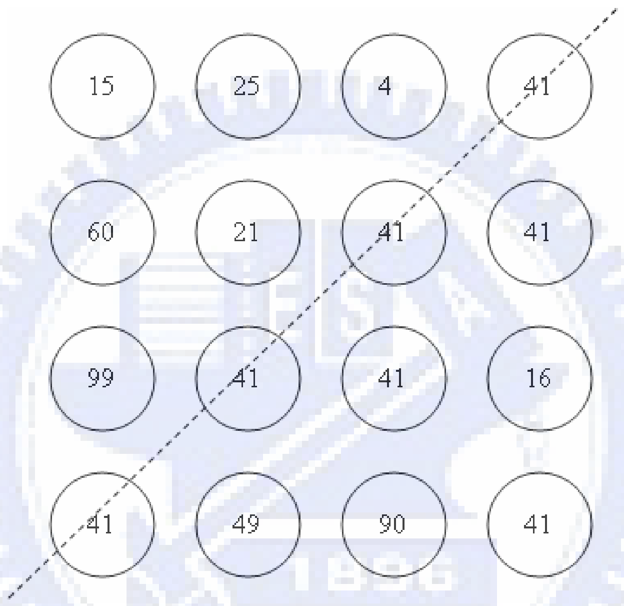


Figure 4.13 An 4×4 image segment and the direction that the odd samples can be predicted perfectly.

The importance of direction-adaptive functionality can be well explained by using the 4×4 image segment in Figure 4.13, assuming the values denoting the graylevel values. For wavelet-based image compression, a DWT transformed image can be coded efficiently by some coders (such as SPIHT or SPECK), if its high frequency subbands contain coefficients with small amplitude and low frequency subband has coefficients with large amplitude. We know that, in lifting 1-D 5/3 DWT, the high frequency subband coefficients are generated in the prediction step only. If the prediction is perfect, after prediction step the odd samples will be zeros which will make the coefficients in high frequency subband become zeros after subsampling step. The direction shown by the dash line in Figure 4.13 can have perfect prediction, since the graylevels of pixels along this line are all 41. Thus the prediction value is also 41, and the residual value (Eq. 4.1) is zero for each odd sample along this line.

CHAPTER 5

EXPERIMENTAL RESULTS OF SDA-DWT

The original application of SDA-DWT was for the object image compression of the intra frames of videos, but its applications can be extended to any wavelet-based fields. The object-based image compression application is introduced in Section 5.1 first. Then, in Section 5.2 we will apply SDA-DWT to the regular still image compression.

5.1 Object Image Compression

In this section, three test object images (Figs. 2.14, 5.1, and 5.2) are used for simulation to evaluate the performance of SDA-DWT, LSA-DWT, and DA-DWT. The original sizes of test images 1 (Figure 2.14) and 2 (Figure 5.1) are 256-by-256 pixels, and the third test image (Figure 5.2) is 128-by-128 pixels. Although the video frame size in MPEG-4 is 360-by-288, we choose square images in order to reduce the bits used for coding the paths in SPECK coding. For comparison, all methods (i.e. LSA-DWT, DA-DWT, and SDA-DWT) use the same 5/3 wavelet, and both LSA-DWT and SDA-DWT use symmetric extension for transform calculation while DA-DWT uses symmetric extension for transform calculation only on the boundary between the object image and background. For the partition boundaries in the object image, DA-DWT uses the practical values at the extension points. Here, we ignore the bits for side information (i.e. the partition of DA-DWT, the shape masks of LSA-DWT, and the partition and shape mask of SDA-DWT) for simplification and focusing on the main problem. The decomposition-level decision in wavelet transform is important and difficult. For a suitable design of decomposition levels, energy clustering effect will make compression efficient. However, excessively many levels can not improve the overall compression efficiency, since the *LL* subband becomes a very small region that may degrade the overall compression efficiency. The suitable number of wavelet decomposition levels mainly depends on the image size, image content, and the coder/decoder used. In most cases, for a 512-by-512-pixel image, we select 3, 4, or 5 levels empirically. In this dissertation, 4 decomposition levels were used because the test images are small size. In the followings, *PSNR* (peak-signal-to-noise ratio) values and the lengths of bit streams after SPECK coding are used as two performance measures. The *PSNR* calculation is based on a 256-by-256-pixel image (objects 1 and 2) or a 128-by-128-pixel image (object 3), and the *bpp* (bit/pixel) calculation is based on the pixel number in an object image.

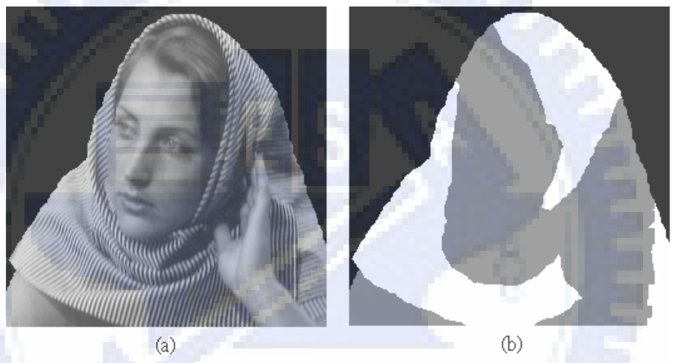


Figure 5.1 The 256 × 256 gray-level object image and its shape mask with partition: (a) the test object image, (b) the mask with partition. (Object 2 contains 45,012 pixels.)

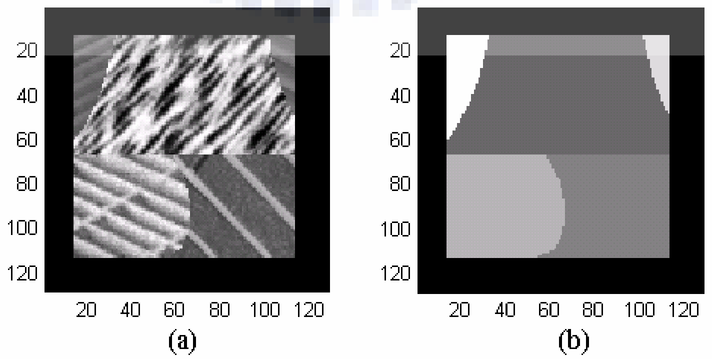


Figure 5.2 The 128 × 128 gray-level object image and its shape mask with partition: (a) the test object image, (b) the mask with partition. (Object 3 contains 10,000 pixels.)

For the first object image (Figure 2.14), our interested object is a suitcase covered with many line textures, and the object occupies 30,353 pixels in a 256-by-256-pixel image. SDA-DWT and LSA-DWT are evaluated by compressing object-1 image. Since the orientations of lines in object 1 are almost the same, we do not partition the object into small segments, i.e. the whole visual object is a large segment. After performing 4-level SDA-DWT on the visual object, the transformed object image is coded by using the SPECK algorithm, and the resulted bit-stream can represent a compression file of the object image. The same procedures are performed on test image 1 except that SDA-DWT is replaced by SA-DWT, and we have another compression file of the object image by using LSA-DWT. Table 5.1 shows the sizes (in bits) of each object image for each method, and it tells us that SDA-DWT is more efficient than LSA-DWT is. The bit number of SDA-DWT compression file is about 77.8% size of LSA-DWT compression file. Table 5.2 shows that SDA-DWT outperforms LSA-DWT up to 5.88 dB under 2.15-*bpp* (256×256 bits) condition. In this case, the performance of SDA-DWT is always better than that of LSA-DWT because of the directional line textures on the object. For the characteristic of the textures on object 1, if we choose $+45^{o}$ direction in the prediction step of the 1-D 'horizontal' transform, the predicted values will very close to the actual values of odd pixels. Thus, much energy is clustered in the low-frequency subband, and that makes the wavelet transform very successful, which makes the overall compression scheme very efficiently.

Table 5.1 The bit numbers of the bit stream of each test object image after SPECK coding. (SDA^1 and SDA^2 represent SDA-DWT without object partition and with object partition, respectively. LSA means LSA-DWT and DA is DA-DWT.)

Object image	Object 1	Object 2	Object 3
SDA^1	123,341 bits	243,002 bits	NA
SDA^2	NA	244,729 bits	67,726 bits
LSA	158,530 bits	245,330 bits	71,556 bits
DA	NA	NA	77,209 bits

Table 5.2 The *PSNR* results for lossy compression of object image 1. (Object 1 contains 30,535 pixels.)

Rate (*bpp*)	1.00	2.15	3.22
SDA	36.82 dB	42.75 dB	48.30 dB
LSA	31.06 dB	36.87 dB	42.54 dB

Table 5.3 The *PSNR* results for lossy compression of object image 2. (Object 2 contains 45,012 pixels. SDA^1 and SDA^2 represent SDA-DWT without object partition and with object partition, respectively.)

Rate (*bpp*)	1.00	1.46	2.91
SDA^1	28.05 dB	28.88 dB	39.45 dB
SDA^2	28.11 dB	29.02 dB	39.45 dB
LSA	27.52 dB	28.73 dB	39.30 dB

Table 5.4 The *PSNR* results for lossy compression of object image 3. (Object 3 contains 10,000 pixels. SDA^2 represents SDA-DWT with object partition.)

Rate (*bpp*)	1.00	1.64	3.28	4.92
SDA^2	22.41 dB	22.91 dB	33.29 dB	43.02 dB
LSA	18.10 dB	22.58 dB	31.79 dB	40.55 dB
DA	17.53 dB	22.29 dB	27.85 dB	38.01 dB

For the test image of object 2 (Figure 5.1(a)), SDA-DWT and LSA-DWT are simulated and compared by their *PSNR* values and file sizes. The gray-level object 2 is segmented from the famous test image Barbara, and Figure 5.1 (b) shows the shape mask of the visual object. Two cases are simulated for evaluating SDA-DWT. First, the whole object 2 without partition is used for simulation, and second, object 2 is partitioned into two parts (Figure 5.1 (b); the white region and the gray part) for simulation. The partition shown in Figure 5.1 (b) is an example, which is partitioned manually, for arbitrarily shaped partition which is not the optimal one. Table 5.1 shows that, for compression-file size, SDA-DWT with object-image partition is the most efficient case among these cases, SDA-DWT without object partition is second place, and LSA-DWT is third place. SDA-DWT with object partition reduces 0.95% bit-budget of LSA-DWT's, and SDA-DWT without object partition reduces 0.24% bit-budget. On the other hand, the *PSNR* values in Table 5.3 show that SDA-DWT with partition has the best performance.

The results show that for a texture rich (especially, non-horizontal or non-vertical edges) image, the performance of lossy compression can be enhanced by suitably partitioning the object image. The proposed method offers much flexibility for partition, since it can handle segments with any shape. The reconstruction object images of SDA-DWT with object partition and LSA-DWT, under 1.46-*bpp* condition, are shown in Figure 5.3. We also performed the experiments on the object images segmented from Lena, Claire, and Akiyo. Since these object images lack non-horizontal or non-vertical edges or the directions of textures are random, without suitable object partition, the performance of SDA-DWT and LSA-DWT are almost the same.

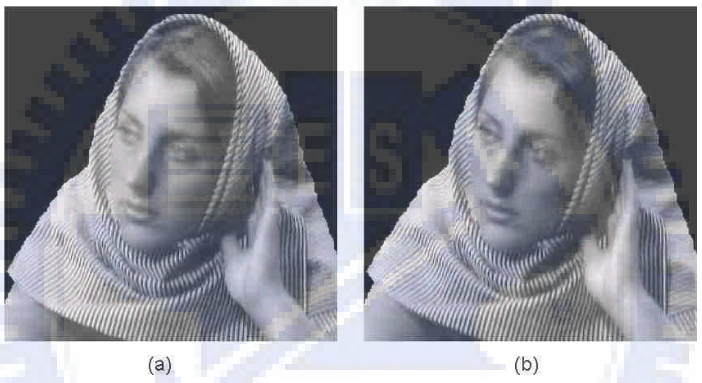


Figure 5.3 The object-2 reconstruction images under 1.46-*bpp* condition: (a) the result of SDA-DWT with object partition according to Figure 5.1 (b), (b) the result of LSA-DWT.

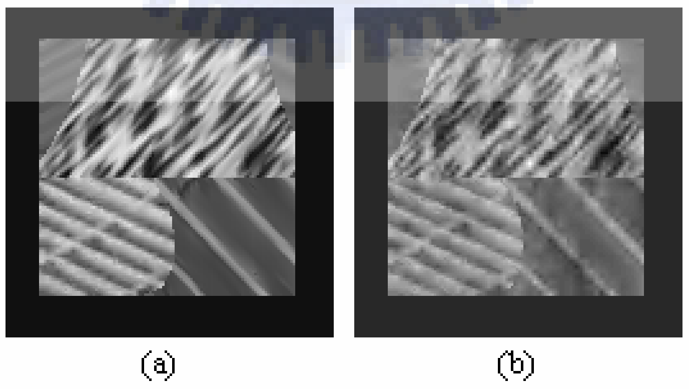


Figure 5.4 The reconstruction object images, under 1-*bpp* condition: (a) the result of SDA-DWT, (b) the result of LSA-DWT.

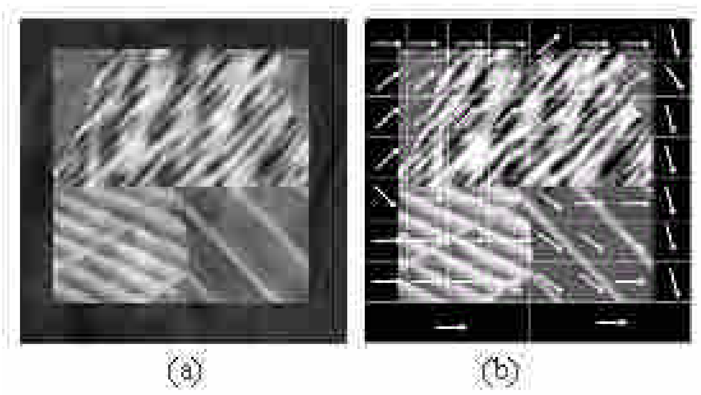


Figure 5.5 The reconstruction object image and the partition and direction in DA-DWT: (a) the reconstruction result under 10,000-bit condition. (b) mask partition and block directions of partition used in DA-DWT.

For the third gray-level object image (Fig 5.4 (a)), all the three methods (LSA-DWT, DA-DWT, and SDA-DWT) are evaluated. The object-3 image (synthesized from the images from USC image database) contains 10,000 pixels in a 128-by-128-pixel area, and there are five different textures on the object. Hence, the object image is partitioned into 5 segments (Figure 5.4 (b), this is synthesized from the image in the USC image database, so it is given in this experiment.) for SDA-DWT. Although object-3 image is rectangular, SDA-DWT can handle any shaped objects. DA-DWT is originally designed for processing a rectangular image, but object-3 image can be viewed as a squared 128-by-128-pixel image containing an object 3. DA-DWT partitions the object image into many small blocks (Figure 5.5 (b)) to discover the texture direction which can not be seen in large scale. Table 5.1 shows that, for lossless compression, SDA-DWT uses the least amount of bits, and DA-DWT is the most bit consuming one. For *PSNR* comparison, Table 5.4 shows that SDA-DWT outperforms LSA-DWT up to 4.31 dB in *PSNR* under 1.00-*bpp* (bit / object pixel) condition, and reduces the bit-budget up to 5.7% for lossless compression. SDA-DWT also outperforms DA-DWT up to 5.44 dB in *PSNR* under 3.28-*bpp* condition, and reduces the bit-budget up to 14.0%. The reconstruction results under 1-*bpp* condition are shown in Figure 5.4 and Figure 5.5 (a). From the experiments of object 3, we understand that LSA-DWT can not well exploit the correlation of the directional textures, so it has poor performance for this test object image. For DA-DWT, since its resolution is not high enough (the smallest partition block is 16-by-16) and can not approximating the non-rectangular segment boundaries well, DA-DWT has the poorest performance for the special object-image.

5.2 Regular Image Compression

SDA-DWT was originally designed for object image compression, but it can be used for a regular (rectangular or square) image by extending the mask (or alpha maps) to cover the whole image. For example, Figure 2.14 (b) is a partition mask for the visual object, and we can use Figure 5.6 as a partition mask of the whole image for SDA-DWT. In Figure 5.6, the regular image is partitioned manually into three large connected segments which are distinguished by three different graylevel values. For the test image (Figure 2.14 (a)), three compression methods, which are SDA-DWT, DA-DWT, and conventional-direction lifting DWT, will be simulated and compared. All the three methods use the lifting 5/3 DWT discussed in Chapter 4, and the third method uses the conventional filter directions, i.e. horizontal and vertical directions. First, the lossless compressed file sizes of these three methods are compared. Then, we will compare the *PSNR* values (dB) under several *bpp* conditions. Finally, we also want to examine the reconstruction results of these three methods, since, sometimes, the *PSNR* values are not reflect the real visual quality.

Test image Figure 2.14 is a 256×256 graylevel image, and it use 8 bits to represent the gray levels of a pixel. Since the test image is small size, we choose to use 4 decomposition levels. After transformed by any one of the three methods, the transformed image is coded by using the SPECK coder. The result bitstream of the SPECK coder is our lossless compression file for each method. The experimental results showed that the file sizes are 245,440 bits, 268,631 bits, and 273,254 bits for SDA-DWT, DA-DWT, and conventional-direction lifting DWT, respectively. Ignoring the side information, SDA-DWT has the smallest lossless compression file.

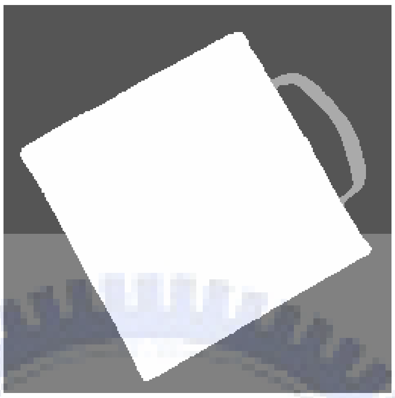


Figure 5.6 An example of partitioning the image in Figure 2.14 (a) for regular image compression. The suitcase image is partitioned into background, the handle, and the box manually.

Table 5.5 *PSNR* values of three method under 0.1-*bpp*, 0.25-*bpp*, 0.5-*bpp*, and 1.0-*bpp* conditions, where CD-DWT means the conventional-direction lifting DWT.

bpp	SDA-DWT	DA-DWT	CD-DWT
0.1	22.52 dB	21.92 dB	22.43 dB
0.25	22.94 dB	22.52 dB	22.96 dB
0.5	28.54 dB	27.51 dB	27.82 dB
1.0	34.10 dB	32.85 dB	32.96 dB

From the data in Table 5.1, we know that SDA-DWT has the best (highest) *PSNR* values in all cases, but it is interesting that DA-DWT is third place. The results showed that the partition in direction DWTs is critical, and locally optimal is not equal to global optimal. Although DA-DWT is inferior to lifting conventional-direction DWT for *PSNR* value in this experiment, but, for 0.25-*bpp* and 0.5-*bpp*, the reconstruction images of DA-DWT have better visual quality than lifting conventional-direction DWT's do.

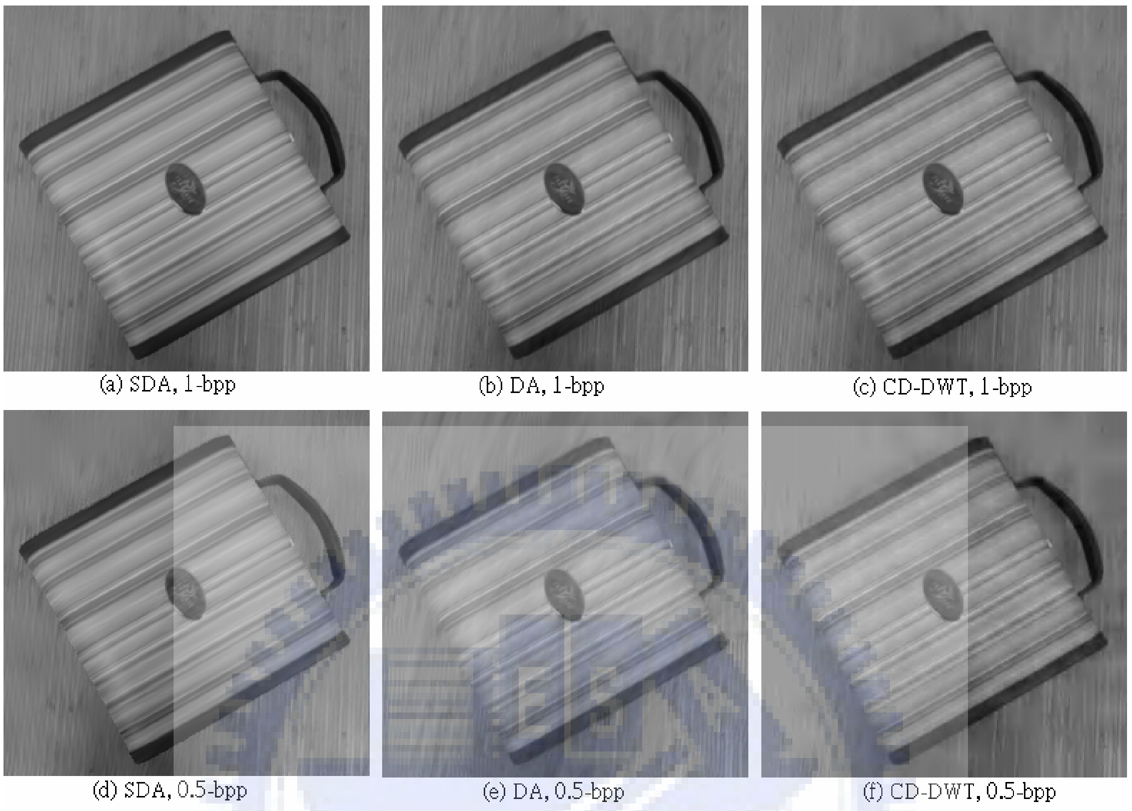


Figure 5.7 Reconstruction images of three methods under 1-bpp and 0.5-bpp conditions respectively, where SDA means SDA-DWT, DA is DA-DWT, and CD-DWT denotes lifting conventional-filter-direction DWT.

Figure 5.9 shows another example whose original image is the Pentagon. For the Pentagon image (Figure 5.9 (a)), there are two partition methods shown in Figure 5.9 (b) and (c), respectively. The partition shown in Figs. 5.9 (b) is based on the partition method of [4] which partitions an image into some small rectangular blocks, and we joint the blocks with the same local filter direction to form 9 types of segments. Note that se4gments belonged to the same filter direction are not necessarily connected. The gray-level value in Figs. 5.9 (b) and (c) is used to represent the filter direction of each of the nine types of segments. The smallest pixel value corresponds to θ = -71.5°, and the highest gray-level corresponds to θ = 71.5°. From Figure 5.9, we can see the segmentation- direction relation. Since SDA-DWT does not require the boundary of each segment to be vertical or horizontal, we can partition the image according to any requirement need. For example, a human body can be partitioned based on his body shape or other ways that is meaningful to us.

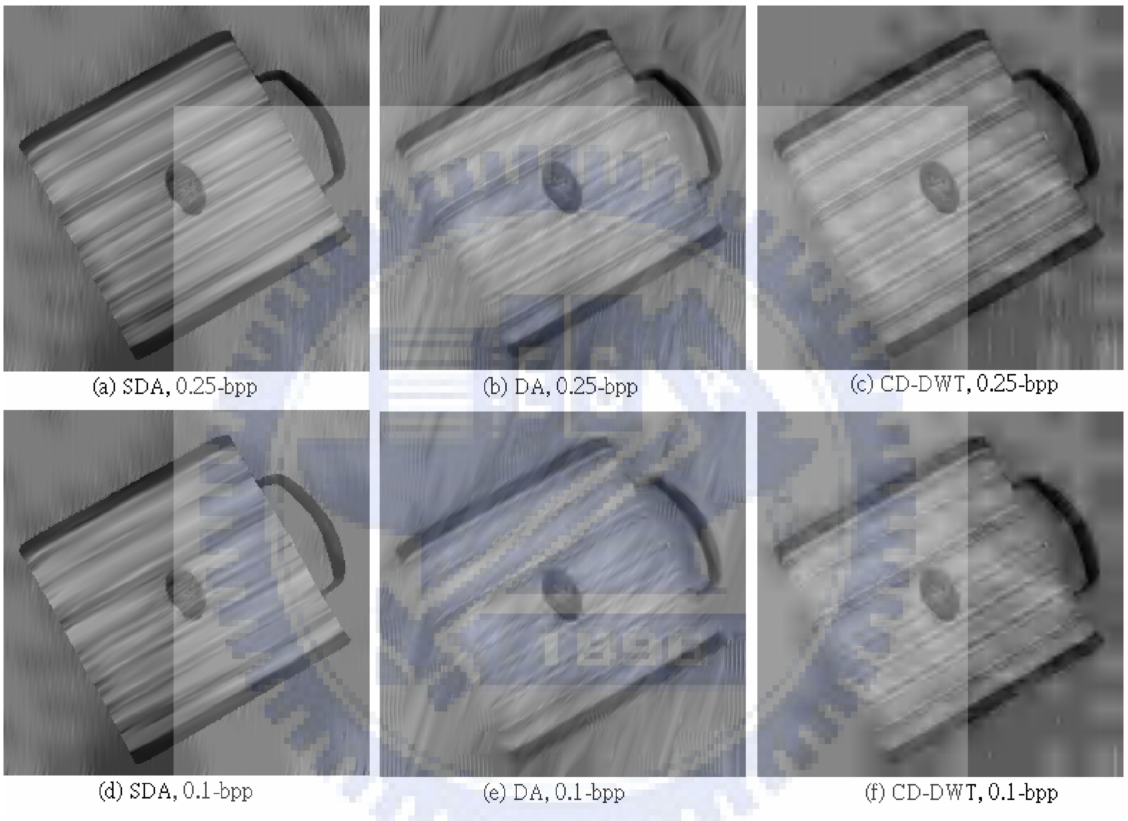


Figure 5.8 Reconstruction images of three methods under 0.25-bpp and 0.1-bpp conditions respectively, where SDA means SDA-DWT, DA is DA-DWT, and CD-DWT denotes lifting conventional-filter-direction DWT.

SDA-DWT outperforms DA-DWT because the former allows flexible partition on shape and size. A simple method can guarantee that SDA-DWT obtains better performance than DA-DWT can do is that use the partition of DA-DWT as a initial partition of SDA-DWT, then modify the partition to be a better one. For example, we can partition the image into many small triangular segments which should be able to better exploit the directional correlation in the image. For many natural or artificial images, the best partition boundaries are usually not vertical or horizontal lines, so SDA-DWT are suitable for such cases.

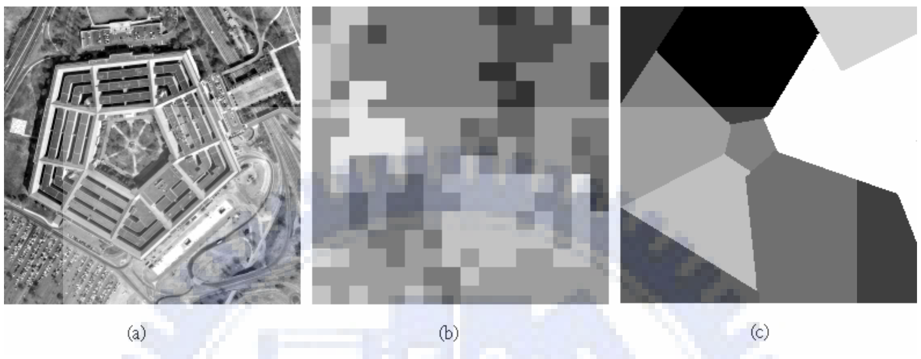
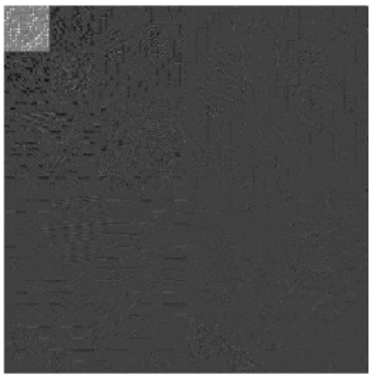
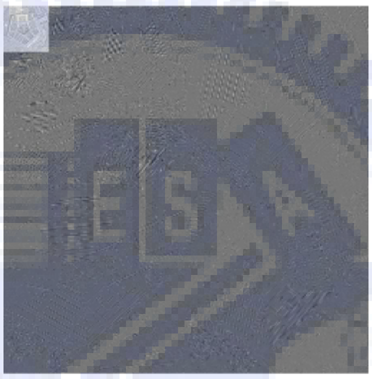


Figure 5.9 A 256×256 image and its partitions: (a) the original image Pentagon, (b) partitioning the image into 9 types of blocks, (c) another non-block partition.

Finally, SDA-DWT (5/3 wavelet) is compared with the lifting 5/3-wavelet conventional-directional DWT by transforming Figure 5.7 (a) with 3 decomposition levels. The results are shown in Figure 5.10, and we can see that less energy is left in the high frequency subbands for SDA-DWT because the high-frequency-subband image of Figure 5.9 (a) are darker than that in Figure 5.9 (b). Hence, generally speaking, SDA-DWT can well exploit the correlation existing in the image and achieve higher coding efficiency in image compression. The maximum amplitude of the transformed coefficients in Figure 5.9 (a) is 423.5, and the maximum coefficient value in Figure 5.9 (b) is 222.8. It usually means good energy compaction and a better transform in image compression, if the coefficient amplitudes are large in the low-frequency subband.



(a)



(b)

Figure 5.10 Transformed images of Figure 5.7 (a): (a) by SDA-DWT, (b) by conventional-direction lifting DWT. (both use 5/3 wavelet and 3 decomposition levels) Note that the high-frequency subband of (a) is smoother than (b), and it means that the former transform is usually more efficient than the later.

CHAPTER 6

CONCLUSIONS

In this dissertation we discuss lifting-based DWT and propose SDA-DWT, which can be used for arbitrarily shaped image segments, and whose direction of prediction and update are adaptive. The conclusions and comments (or future works) of these two topics are presented in the followings in this chapter.

For lifting-based DWTs, we discussed how every wavelet filter pair can be decomposed into lifting steps. The decomposition is equivalent to present the polyphase matrix and dual polyphase matrix as products of elementary matrices (i.e. lower triangular and upper triangular matrices), which was known to be possible by mathematicians long time ago. Compared to conventional implementation, lifting structure can lead to a speed-up. The lifting structure also allows for an in-place realization of the fast wavelet transform, so the wavelet transform can be computed without allocating auxiliary memory. In a lifting step, all operations can be done totally parallel, and the only sequential part is the order of the lifting operations. For hardware implementation and lossless image compression, lifting structure is important because it is easier to build on non-linear wavelet transform and wavelet transforms which map integers to integers. And, it is possible to integrate biorthogonal wavelets with scalar quantization and also keep cubic quantization cells which are optimal like in orthogonal cases, by using both the lifting and integer-to-integer transforms. At last, the special feature of lifting, which helps us to develop SDA-DWT, is that lifting allows for adaptive wavelet transform. Therefore, one can start the analysis of a function form the coarsest levels and build the finer levels by refining the region of interest.

Since the results of factoring lifting steps are not unique, we should know what the optimal case is in the future work. The Euclidean algorithm, which is used to factor polyphase and dual polyphase matrices, can not work for integer or dyadic-number filter coefficients. Hence, we can not assure that filters with binary coefficients are able to be factored into lifting steps with binary filter coefficients. Any invertible polyphase matrix which has a non-identity polynomial on the diagonal can be obtained using lifting, but some of the advantages of lifting structure discussed above rely on the identity diagonal requirement.

For the proposed SDA-DWT, since it can well exploit the correlation because of spatial orientation and handle regions with any shape and size, SDA-DWT has superior performance than SA-DWT or DA-DWT does for visual objects with non-horizontal or non-vertical edge textures. SDA-DWT can be applied to any wavelet-based application, although, in this dissertation, we only give three application examples. The extra costs of SDA-DWT compared to SA-DWT are the increased complexity and the storing and processing of the side information of the directions in each segment of the object image.

For convenience, we focus on how to compress the partitioned still-object image while assuming that the partition of the object image has been done in this work. In order to achieve the optimal result, a good texture-segmentation method is necessary. The optimal partition depends on the image to be compressed, and it is usually not the case in DA-DWT (i.e. rectangular blocks). Partitioning an image into many small rectangular blocks is usually not the optimal partition, and the reconstruction result is possible to suffer from blocking effect when the bit rate is low. The optimal partition usually partitions an image into many irregular segments, so it is difficult to determine a global optimal partition for SDA-DWT. Thus, solving the optimal partition problem is the most important work in future, and the partition method used in DA-DWT can be used as a starting point.

REFERENCES

[1] S. Li and W. Li, "Shape-adaptive discrete wavelet transforms for arbitrarily shaped visual object coding," *IEEE Trans. Circuit and Syst.*, vol. 10, no. 5, pp. 725-743, 2000.

[2] Z. Lu and W.A. Pearlman, "Wavelet coding of video object by object-based SPECK algorithm," *Picture Coding Symposium* (*PCS 2001*), pp. 413-416, April 2001.

[3] W. Ding, F. Wu, X. Wu, S. Li and H. Li, "Adaptive directional lifting-based wavelet transform for image coding," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 416-427, Feb. 2007.

[4] C.-L. Chang and B. Girod, "Direction-adaptive discrete wavelet transform for image compression," *IEEE trans. Image Process.*, vol. 16, no. 5, pp.1289-1302, May 2007.

[5] S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd *ed.*, Academic Press, San Diego, CA, 1999.

[6] Y. Meyers, *Wavelet: Algorithms and Applications*. SIAM, Philadelphia, 1993 (Translated by Robert D. Ryan).

[7] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE Trans. on Info. Theory*, vol. 36, no. 5, pp. 961-1005, 1990.

[8] G. Strang and T. Nguyen, *Wavelets and filter banks*, Wellesley-Cambridge, MA, USA, 1996.

[9] T. Acharya and P.-S. Tsai, *JPEG2000 Standard for Image Compression, Concepts, Algorithms and VLSI Architectures*, John Wiley & Sons, Hoboken, New Jersey, 2005.

[10] K. Weierstrass, *Mathematische Werke*, *vol. II*, Mayer & Muller, Berlin, 1895.

[11] R. N. Bracewell, *The Fourier Transform and Its Applications*, 2nd *ed.*, McGraw-Hill, Singapore, 1986.

[12] I. Daubechies and W. Swendens, "Factoring wavelet transforms into lifting steps," *J. of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247-269, 1998.

[13] W. Sweldens, "The lifting scheme: a construction of second generation wavelets," *SIAM I. Math. Anal.*, vol. 29, no. 2, pp. 511-546, 1997.

[14] W. Sweldens, "The lifting scheme: a custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.*, vol. 3, no. 2, pp. 186-200, 1996.

[15] S. G. Mallat, “Multi frequency channel decompositions of images and wavelet models,” *IEEE Trans. Acoust. Speech Signal Process.*, vol. 37, no. 12, pp. 2091-2110, 1989.

[16] S. Mallat, “A theory for multiresolution signal decomposition: the wavelet representation,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674-693, 1989.

[17] Y. Meyers,*Wavelets: Algorithms and Applications*, SIAM, Philadelphia, 1993 (translated by Robert D. Ryan).

[18] A. Cohen, I. Daubeches, and J. Feauveau, “Bi-orthogonal bases of compactly support wavelets,” *Comm. Pure Appl. Math.*, Vol. 45, pp. 485-560, 1992.

[19] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, Englewood Cliffs, N.J., 1989.

[20] S. Lang, *Undergraduate Algebra, 2nd ed.*, Springer-Verlag, New York, 1990.

[21] R. E. Blauht, *Fast Algorithms for Digital Signal Processing*. Addison-Wesley, Reading, MA, 1984.

[22] T. Acharya and C. Chakrabarti, “A survey on lifting-based discrete wavelet transform architectures,” *J. of VLSI Signal Process.*, vol. 42, pp. 321-339, 2006.

[23] T. Sikora and B. Makai, “Shape-adaptive DCT for generic coding of video”, *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 1, pp. 59-62, Feb. 1995.

[24] A. Said and W. A. Pearlman, “A new, fast, and efficient image codec based on set partitioning in hierarchical trees,” *IEEE Trans. Circuits Syst. Video Tech.*, vol. 6, no. 3, pp. 243-250, 1996.

[25] A. Gersho and R. M. Gray, *Vector quantization and signal compression*, Kluwer Acdemic Publishers, MA. USA, 1992.

[26] Y. Linde, A. Buzo, and R. Gray, “An algorithm for vector quantizer design,” *IEEE Trans. Commun.*, COM-28, pp. 84-95, 1980.

[27] W. A. Pearlman, A. Islam, N. Nagaraj, and A. Said, “Efficient, low-complexity image coding with a set-partitioning embeded block coder,” *IEEE Trans. on Circuits Syst. Video Tech.*, vol. 14, no. 11, pp. 1219-1235, 2004.

[28] C.-K. Su, H.-C. Hsin, and S.-F. Lin, “Wavelet tree classification and hybrid coding for image compression,” *IEE Proc.-Vis. Image Signal Process.*, vol. 152, no. 6, pp. 752-756, 2005.

[29] J. M. Shapiro, “Embedded image coding using zero-trees of wavelet coefficients,” *IEEE Trans. on Signal Proces.*, vol. 41, pp. 3445-3462 ,1993.

[30] S. Gupta and A. Gersho, “Feature predictive vector quantization of multispectral images,” *IEEE Trans. Geoscience and Remote Sensing*, vol. 30, no. 3, pp. 491-501, 1992.

[31] T. K. Abdel-Galil, E. F. El-Saadany, A. M. Youssef, and M. M. Salama, “Disturbance classification using hidden Markov models and vector quantization,” *IEEE Trans. Power Delivery*, vol. 20, no. 3, pp. 2129-2135, 2005.

[32] T. Berger, *Rate Distortion Theory*, Prentice Hall, NJ, USA, 1971.

[33] C. F. Barnes, *Residual Quantizer*, Ph.D. dissertation, BrigHam Young University, Provo, UT, Dec. 1989.

[34] F. Kossentini, M. J.T. Smith, and C. F. Barnes, “Image coding using entropy-constrained residual vector quantization,” *IEEE Trans. Image Process.*, vol. 4, no. 10, pp. 1349-1357, 1995.

[35] Y. Shoham, *Hierachical Vector Quantization with Application to Speech Waveform Coding*, Ph.D. dissertation, University of California at Santa Barbara, Mar. 1985.

[36] B. H. Juang and A. H. Gray, “Multiple stage vector quantization for speech coding,” *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. Mag.*, vol. 1, pp. 597-600, Apr. 1982.

[37] G. Xie and H. Shen, “Highly scalable, low-complexity image coding using zeroblocks of wavelet coefficients,” *IEEE Trans. on Circuits Syst. Video Tech.*, vol. 15, no. 6, pp. 762-770, 2005.

[38] C.-K. Su and S.-F Lin, “Shape-Direction-Adaptive Lifting-Based Discrete Wavelet Transform for Arbitrarily Shaped Segments in Image Compression,” *IEICE Trans. Information and Syst.* (Accepted, 2008).