國立交通大學

電子工程學系 電子研究所

博 士 論 文

以方向性小波轉換為基礎之影像與視訊編碼

Directional Wavelet-based Image and Video Coding

研 究 生：洪朝雄

指導教授：杭學鳴 教授

中 華 民 國 一 0 一 年 九 月

以方向性小波轉換為基礎之
影像與視訊編碼

Directional Wavelet-based Image and Video Coding

研 究 生：洪朝雄　　Student：Chao-Hsiung Hung

指導教授：杭學鳴　　Advisor：Hsueh-Ming Hang

國 立 交 通 大 學

電子工程學系 電子研究所

博 士 論 文

A Dissertation
Submitted to Department of Electronics Engineering and
Institute of Electronics
College of Electrical and Computer Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in
Electronics Engineering

September 2012
Hsinchu, Taiwan, Republic of China

中華民國一0一年九月

以方向性小波轉換為基礎之
影像與視訊編碼

研究生：洪朝雄　　　　　　　　　　　　　　　指導教授：杭學鳴

國立交通大學 電子工程學系 電子研究所博士班

摘　要

多重解析度方向性轉換(multiresolution directional transform)包含頻率領域(frequency domain)和空間領域(spatial domain)兩方面。在頻率領域的多重解析度方向性轉換中，以小波轉換為基礎的輪廓轉換(wavelet-based contourlet transform，WBCT)能針對影像中的紋路(texture)方向，提供較佳的配合度，因此用於影像編碼，但其缺點為較高的計算複雜度。另一方面在空間領域的多重解析度方向性轉換中，方向可調整性小波轉換(direction-adaptive wavelet transforms，DA-DWT)會根據影像的紋路選出適合的轉換方向，故可提供較小波轉換(discrete wavelet transform，DWT)較佳的壓縮效果。但位於平滑區域(smooth region)中的相鄰區塊(block)的最佳方向的選擇上，常會選出不一致的最佳方向，因此會造成大量且冗餘的外加資訊(side information)。

在本論文中，我們提出了數個改良演算法，用以改良以小波轉換為基礎的輪廓轉換以及方向可調整性小波轉換的編碼效果。我們提出了三個改良演算法來改善採用以小波轉換為基礎的輪廓轉換的編碼方法，特別是在減少其計算量。第一，

在方向性轉換(directional transform)上，我們提出了一組長度較短的二維濾波器(2-D filters)。第二，我們提出一個以移動平均(mean shift)為基礎的判斷方法，用來選出適合作方向性轉換的小波次頻帶(wavelet transform)。第三，我們改良算術編碼(arithmetic coder)所採用的狀態表(context table)，用來加強以小波轉換為基礎的輪廓轉換的編碼效果並減少計算複雜度。經由實驗模擬，我們可以得到和原本採用以小波轉換為基礎的輪廓轉換的編碼方法接近的影像品質，但可減少 92%以上的計算量。而和採用傳統二維小波轉換(two-dimensional discrete wavelet transform，2-D DWT)的編碼方法相比，我們能提供較佳的主觀視覺品質。

我們也提出了另外三個改良演算法來改善方向可調整性小波轉換的編碼效果。第一，我們提出了一個由小區塊擴張到大區塊的方向一致化演算法，並把各別區域中，鄰近的方塊的選擇方向作一致化。第二，我們把所提出的方向一致化演算法，延伸到次取樣及方向可調整性之小波轉換(subsampling and direction-adaptive discrete wavelet transform，SA-DWT)上。第三，我們針對動態預測殘像(motion-prediction residuals) 提出了改良式的次取樣及方向可調整性之小波轉換(modified subsampling and direction-adaptive discrete wavelet transform，MSA-DWT)。經由實驗模擬，我們所提出的方向一致化演算法可減少大約 60%的外加訊息，並且在低位元率(low-bit rate)可以增加照片壓縮效果 0.4dB 左右。而所提出的改良式的次取樣及方向可調整性之小波轉換可以增加視訊壓縮效果 0.1~0.2 dB 左右。

Directional Wavelet-based Image and Video Coding

Student: Chao-Hsiung Hong Advisor: Dr. Hsueh-Ming Hang

Department of Electronics Engineering & Institute of Electronics
National Chiao Tung University

Abstract

The multiresolution directional transform includes two types, frequency domain and spatial domain. In frequency domain, the *wavelet-based contourlet transform* (WBCT) is adopted for image coding because it matches well image textures of different orientations. However, its computational complexity is very high. In spatial domain, the *direction-adaptive discrete wavelet transform* (DA-DWT) provides better compression performance than *discrete wavelet transform* (DWT) for it selects transform directions to match image local texture. However, it often picks up inconsistent directions for neighboring blocks with similar texture and thus results in large but redundant side information.

In this dissertation, we propose some enhanced algorithms for improving the coding efficiency of WBCT and DA-DWT. we propose three algorithms to enhance the WBCT coding scheme, in particular, on reducing its computational complexity. First, we propose short-length 2-D filters for directional transform. Second, the directional transform is applied to only a few selected subbands and the selection is done by a mean-shift-based decision procedure. Third, we fine-tune the context tables used by the arithmetic coder in WBCT coding to improve coding efficiency and to reduce computation. Simulations show that, at comparable coded image quality, the proposed scheme saves over 92% computing time of the original WBCT scheme. Comparing to the conventional 2-D wavelet coding schemes, it produces clearly better subjective image quality.

We propose another three algorithmes to improve the compression performance of DA-DWT in image and video coding. First, we propose a bottom-up direction alignment algorithm to align the directions of neighboring blocks in local regions. Second, we extend alignment algorithm to the *subsampling and direction-adaptive discrete wavelet transform* (SA-DWT). Third, we propose a *modified* SA-DWT (MSA-DWT) to improve the motion-compensated residual coding. Simulations show that the proposed alignment algorithm reduces about 60% side information and improves the image coding gain up to 0.4 dB at low bit rate. The MSA-DWT scheme can also improve about 0.1 ~ 0.2 dB in video coding gain.

誌　謝

當年考上交大來這念書時，從來沒想過說自己會在這邊一待就十三年了。前面大學四年和碩士兩年的生活，感覺自己都只是在書堆中，不知道自己真正要什麼。後面七年的博士，我才真正學到很多東西，看到自己缺乏了什麼，也知道自己需要什麼，回首這七年真的很像一場很長的夢，而在這七年中我要感謝下面這些人。

謝謝我的指導教授杭學鳴老師，在這段時間除了學到作研究的方法外，更重要的是我也從老師身上學到了很多作人作事的道理，也謝謝老師對我情緒控管不佳的包容，並時時提醒我注意此問題。也謝謝老師讓我有機會去國外短暫的遊學，體驗不同的學術風氣。而且在不景氣找工作困難時，也是經由老師的介紹才找到現在的工作，真的很謝謝老師。

謝謝我的家人，如果不是你們一直給我經濟支援和精神支持，我這個博士可能會拿得更辛苦，這七年你們的精神壓力也是很大，但還是一直支持我，真的很謝謝你們。

謝謝交大諮商中心李璨如老師，感謝您一年多來的心理諮商，讓我可以去面對我心中的問題，並且陪我走過博士班最難熬的時期，也讓我可以去思考我未來想要什麼，要如何去追求我想要的事，並更積極面對我的人生。

謝謝實驗室的很多學長，蔡彰哲學長、蔡家揚學長、張豐誠學長、林鴻志等人常常跟我討論研究方向，並且指導我寫作文章和幫我修改，才能順利完成博士

學位。也謝謝吳俊容學長、陳宜寬學長、鍾翼州學長、王柏森學長、鄭榮仁學長等人，教了我很多人生應該有的智慧，這些智慧真的讓我受益良多。

謝謝我從其他親朋好友們，如果平常沒有你們的支持和給我鼓勵，這段博士我也真的很難熬。也謝謝你們時常帶我出去走走或聽我發牢騷，或給我很多建議，讓我可以舒解心中的壓力和疑惑，繼續面對自己的人生。

也謝謝系上其他教授，除了從教授們身上學到知識外，更重要的是很多人生的智慧和方向。也謝謝系上很多助理的幫忙，很多事情如果沒有大家的幫忙，自己處理起來也很費時費力。也謝謝 lab 其他學長姐和學弟妹的幫忙，大家在這個實驗室一起生活，給我留下了很多回憶。最後謝謝諸佛菩薩的幫忙，讓我可以靜下心來去面對我的人生，並且總是在我身旁看護我。

博士這個學位，讓我學到更謙卑而且積極的態度，也讓我學到更多人生的智慧，如果不是那麼多貴人的幫忙，我自己一個人是很難完成的，謝謝大家。

洪 朝 雄

謹誌於台灣新竹交通大學

2012/10/21

Tables of Contents

圖目錄(List of Figures)

表目錄(List of Tables)

Chapter 1 Introduction

DWT [1]-[5] is adopted widely in image and video coding in recent years. Wavelet-based image coding, such as JPEG2000 [6], consists of three stages: *two-dimensional discrete wavelet transform* (2-D DWT), coefficient quantization, and arithmetic coding. A digital image is first transformed by 2-D DWT to produce a set of transform coefficients. After quantization, these coefficients are compressed to a binary stream by an entropy coding tool. For video signal compression, a wavelet-based interframe coding, such as Vidwav [7], includes four stages: *motion-compensated temporal filtering* (MCTF) [8]-[13], 2-D DWT, quantization, and arithmetic coding. MCTF decomposes video frames into temporal low-pass images and high-pass residuals. Then, in addition to image coding, we process the residuals also by 2-D DWT, quantization, and arithmetic coding.

One-dimensional discrete wavelet transform (1-D DWT) represents 1-D piecewise smooth signals well in few coefficients [5]. 2-D DWT applies two 1-D DWTs along horizontal and vertical axes and ignores 2-D piecewise smooth signal continuity. It represents 2-D signals by many little coefficients and spreads the energy into the high-pass subbands [14][15]. Quantizing these coefficients to zero at low bit rates results in Gibbs artifacts at image edges [16].

Many 2-D multiresolution directional transforms have been proposed to solve this problem, including the directional filter banks [17]-[23] and the direction-adaptive wavelet transforms [27]-[32]. Directional filter banks use a set of pre-selected 2-D filters to perform multiresolution directional decomposition. Each filter corresponds to a basis function with specific spatial direction and resolution. Directional filter banks can represent 2-D directional texture patterns by relatively few large coefficients. Do and Vetterli proposed the *contourlet transform* (CT) [17], which is composed of the *Laplacian pyramid* (LP) [24] and the *directional filter bank* (DFB) [25]. Lu and Do proposed the *finer directional wavelet transform* with additional 2-D directional resolution [18]. Nguyen and Oraintara re-designed DFBs and provided enhanced directional decomposition [19][20]. Selesnick *et al.* proposed the *complex wavelet transform* with good directionality and shift invariance [21]. Eslami and Radha proposed the *wavelet-based contourlet transform* (WBCT) and its extension version by applying DFBs to 2-D DWT's high-pass subbands [22][23]. Among them, the WBCT [22] technique has the critical-sampling property, consumes comparatively less computational power, and requires no side information for decoding. Therefore, we focus on WBCT in this dissertation.

The direction-adaptive discrete wavelet transform (DA-DWT) technique partitions an image into local regions (blocks) and filters along the texture direction by 1-D DWT lifting scheme [26]. It selects the optimal direction and block size by minimizing the prediction error under the constraint bits. Thus, DA-DWT compacts more energy into the spatial low-pass subbands and provides good compression performance [15]. Chang and Girod proposed a DA-DWT based image compression scheme with integer pixel direction accuracy [27]. Ding *et al.* uses interpolation to achieve quarter pixel direction accuracy [28]. Liu and Ngan used a weighted function to avoid mismatch in the lifting scheme [29]. Dong *et al.* proposed a 2-D adaptive interpolation filter for more accurate fractional pixel accuracy [30]. Chang and Girod proposed another DA-DWT based on the quincunx subsampling pattern [31]. Xu and Wu combined different subsampling patterns together and proposed the *subsampling and direction-adaptive discrete wavelet transform* (SA-DWT) [32]. It is easy to implementing and integrating DA-DWT into wavelet-based image coding. Thus, DA-DWT becomes our second focus in this dissertation.

Arithmetic coding schemes compress the transformed/quantized coefficients into bitstream. They produce a minimum-distortion scalable bitstream under all the constrained bit rates. They consider three types of correlations among the coefficients. First, the inter-subband coding methods, such as the *set partitioning in hierarchical tree* (SPIHT) method [33] and the *embedded zerotrees of wavelet transform* (EZW) method [34], mitigate the inter-band correlations in a tree structure. Second, the intra-subband coding methods partition the coefficients in one subband to several non-overlapped coding blocks and handle only the correlations among the neighbors in one coding block (the intra-subband correlations). Examples in this category are the *embedded block coding with optimized truncation* (EBCOT) method [35], the *3-D embedded subband coding with optimized truncation* (3-D ESCOT) method [36], and the *tarp-filter-based system that classifies coefficients to achieve embedding* (TCE) method [37]. Third, the mixed inter-subband and intra-subband coding methods cover both the inter-subband and intra-subband correlations. Examples are the *embedded conditional entropy coding of wavelet coefficients* (ECECOW) method [38] and the *embedded coding using zeroblocks of wavelet coefficients and context modeling* (EZBC) method [39]. To save computing power, for single image compression, we use the intra-subband coding methods in this dissertation.

Combining WBCT and 3-D ESCOT, a WBCT image coding scheme can achieve a better coding performance than a regular 2-D DWT image coding scheme. However, there are a few issues in the existing WBCT coding schemes. They need a large amount of computations because the existing WBCT directional filters have a large support. And, we found that for a specific picture, some WBCT frequency subbands do not need further directional transform. Furthermore, the context table in 3-D ESCOT needs adjustment to match the characteristics of quantized WBCT coefficients.

To solve these issues, we propose three algorithms in this dissertation to enhance the WBCT image coding scheme. First, we suggest a set of short-length 2-D directional filters [40] and verify their performance. Second, we design a mean-shift-based decision scheme to dynamically select the proper subbands for directional transform [41]. Third, we re-design the context tables of 3-D ESCOT to match the data directionality. With these algorithms, our proposed scheme reduces 92% or higher the computational complexity of the original WBCT image coding scheme at similar visual quality [40].

DA-DWT first partitions images into non-overlapping blocks. It then applies the 1-D DWT to each block along the candidate directions and calculates the corresponding prediction errors. It finally selects the candidate direction with the minimal prediction error as the most suitable direction for the block. For the partition blocks in smooth region, each candidate direction produces similar prediction error. Thus, DA-DWT selects inconsistent directions for these blocks and increases side information. We also encounter similar situation for blocks in similar-textured region. Tanaka *et al.* pre-filtered images by 2-D filters [42]. Pre-filtering reduces candidate directions and makes selected directions more consistent. If a block has filtered output less than the threshold, it is considered in smooth region and processed by 2-D DWT.

Selecting a suitable threshold for identifying smooth region is hard. Aligning blocks in similar-textured region also helps reducing side information. Maleki *et al*. proposed an alignment cost function for entire image to align small blocks or blocks in smooth region [15]. Different local areas of the same image have different characteristics. Aligning directions based on local characteristics provides better results.

In wavelet-based video coding, we deal with motion compensated prediction residuals instead of images. Kamisli and Lim showed that prediction residuals and images have different spatial characteristics [43]. Images have 2-D anisotropic structures while prediction residuals have 1-D anisotropic structures. Kamisli and Lim proposed 1-D DCT for compressing prediction residuals [43]. They also applied 1-D DA-DWT to prediction residuals [44]. They compared the compression performance based on number of nonzero transformed coefficients instead of number of bits. Because of different spatial characteristics, 2-D DA-DWT compresses prediction residuals inefficiently. In wavelet-based video coding, temporal low-pass prediction residuals (T_L) are similar to images but high-pass ones (T_H) are similar to prediction residuals [45].

We propose another three algorithms in this dissertation to improve DA-DWT's coding performance on images and prediction residuals. First, we suggest a direction alignment algorithm to reduce DA-DWT's side information [46]. Second, we extend the suggested direction alignment algorithm to 2-D SA-DWT. Third, we analyze prediction residuals' characteristic in frequency domain and their transformed coefficients. We applied 2-D DA-DWT on T_Ls in previous research [45]. Now, we suggest a 2-D MSA-DWT for compressing T_Hs. Our suggested direction alignment algorithm saves about 60% side information at the cost of about 3% prediction error increment. It also improves DA-DWT's coding gain about 0.4 dB at low bit rate. 2-D MSA-DWT also provides better coding performance than 2-D SA-DWT on T_Hs and improves about 0.1~0.2 dB in gain.

This dissertation is organized as follows. Chapter 2 introduces the adopted directional filter banks and direction-adaptive wavelet transforms. Chapter 3 gives the introduction of temporal transform and adopted arithmetic coding. Chapters 4 and 5 desribe the proposed algorithms for WBCT and DA-DWT. Chapter 6 gives the experimental results and Chapter 7 contains the concluding remarks. The major contributions of this dissertation are listed as follows.

(1) We design short-length 2-D directional filters to save computational power of directional transform in WBCT.

(2) We propose a mean-shift-based decision scheme to dynamically select the proper subbands for directional transform.

(3) We fine-tune the context tables of 3-D ESCOT to match the data directionality.

(4) We propose a direction alignment algorithm for DA-DWT to reduce side information.

(5) We extend the proposed direction alignment algorithm to SA-DWT.

(6) We modify SA-DWT for compressing T_Hs in wavelet-based video coding.

Chapter 2 Multiresolution Directional Wavelet Transforms

2.1 Two-Dimensional Discrete Wavelet Transform

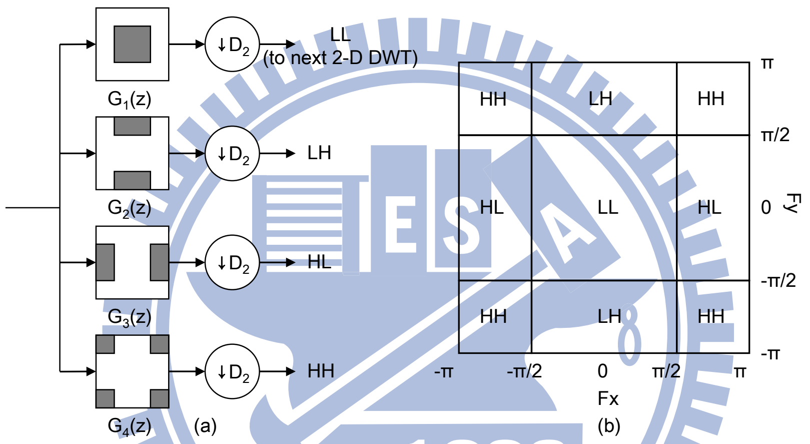


Fig. 2-1. (a) Filter bank structure of 2-D DWT. (b) Frequency partitions produced by 2-D DWT.

Fig. 2-1(a) shows the filter bank structure of 2-D DWT. After transform, it outputs four subband signals - HL (the horizontal high-pass and vertical low-pass subband signal), LH (the horizontal low-pass and vertical high-pass subband signal), HH (the horizontal high-pass and vertical high-pass subband signal), and LL (the horizontal low-pass and vertical low-pass subband signal). $G_1(z)$~$G_4(z)$ are the filters with specific pass bands and their output frequency partitions are given in Fig. 2-1(b). D_2 represents the decimation matrix, and $D_2=2I_2$, where I_2 is an identity matrix. 2-D

DWT is a critical-sampled transform that keeps the same amount of data after one level transform.

2-D DWT provides multiresolution decomposition for images. In a multi-level 2-D DWT, the subband signal LL produced by the first 2-D DWT is further processed by the sub-sequent 2-D DWT's. The first-level 2-D DWT acquires an image and generates four subbands: LL^1, HL^1, LH^1, and HH^1. Then, we filter the LL^1 subband signal again by second-level 2-D DWT to obtain LL^2, HL^2, LH^2, and HH^2. Likewise, we recursively apply 2-D DWT to the LL^i subband, and produce LL^{i+1}, HL^{i+1}, LH^{i+1}, and HH^{i+1}, wherein 'i' represents the 2-D DWT iterations.

2-D DWT is the tensor product of two 1-D DWTs, and the Daubechies 9-7 wavelet filter [2][47] is often in use. 1-D DWT can represent the piecewise smooth 1-D signals by a few coefficients [5]. But the outputs of 2-D DWT would contain many small coefficients for 2-D edges when these edges are not aligned with the vertical or the horizontal axes as shown in Fig. 2-2(a) [14]. If we quantize these coefficients to zero, the coded image shows Gibbs artifacts along the edges [16]. A multiresolution transform with directionality in Fig. 2-2(b) is more desirable for representing 2-D signals.

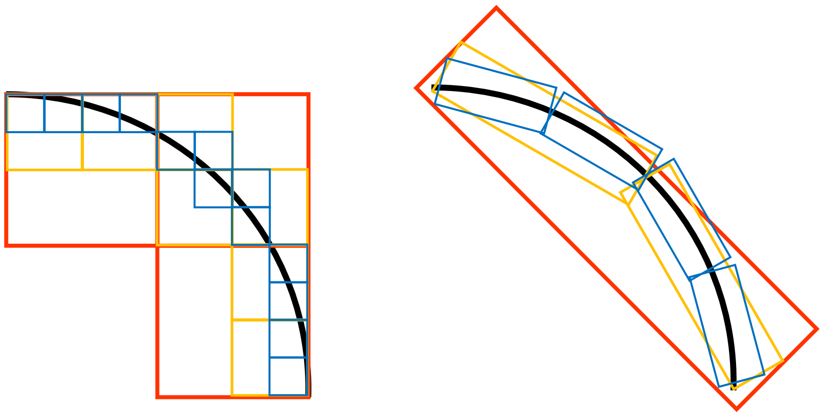


Fig. 2-2. Representing a 2-D signal by (a) 2-D DWT and (b) new transform Xlet.

2.2 Contourlet Transform

2.2.1 Laplacian Pyramid

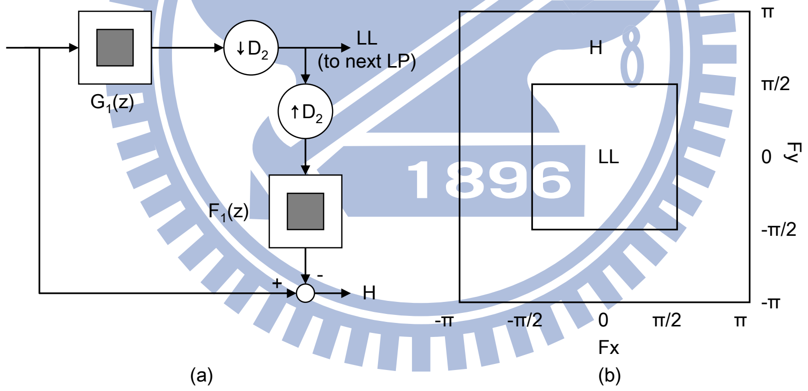


Fig. 2-3. (a) Filter bank structure of LP. (b) Frequency partitions produced by LP.

Contourlet transform (CT) [17] adopts LP [24] in Fig. 2-3 for multiresolution decomposition. The LP decomposes the input into one low-pass subband signal, LL, and one high-pass subband signal, H. $F_1(z)$ is the corresponding synthesis filter for the analysis filter $G_1(z)$ in Fig. 2-3(a). Fig. 2-3(b) shows the frequency partition of these two subbands. When the synthesized subband signal LL is subtracted from the original input, it produces the high-pass subband signal H. Without down-sampling, subband H is free from frequency scrambling [14]. Fig. 2-4 illustrates frequency scrambling in 1-D case. The high-pass signal is folded back into low frequency after down-sampling, and thus its spectrum is reflected. In CT, the LP unit behaves as an over-sampled transform and it increases 25% data size after the transform.

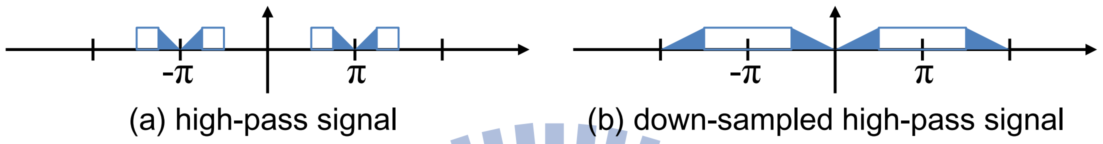


Fig. 2-4. Frequency scrambling in 1-D case.

The subband signal LL in LP (Fig. 2-3(a)) is identical to the subband signal LL in 2-D DWT (Fig. 2-1(a)) when their $G_1(z)$ and D_2 are the same. That is, these two LL signals occupy the same frequency partition as in Fig. 2-1(a) and Fig. 2-3(a), respectively. In a multi-level 2-D DWT, the subband signal LL produced by the first 2-D DWT is further processed by the sub-sequent 2-D DWT's. Likewise, in a multi-level LP, the LL subband signal may be further processed by a sub-sequent LP.

2.2.2 Directional Filter Bank

CT adopts DFB [25] in Fig. 2-5 for directional decomposition. Fig. 2-5(a) illustrates four 2-D filters and four decimation matrices. These four 2-D filters decompose the input signal to four directional subbands. Each subband has a specific directional pass band. These 2-D filters, $A_1(z)$~$A_4(z)$, are fan filters and their corresponding output frequency partitions are drawn in Fig. 2-5(b). The decimation matrices rotate and down-sample the signals along specific directions. DFB with different direction number can be constructed by different directional filters and decimation matrices [14][25].

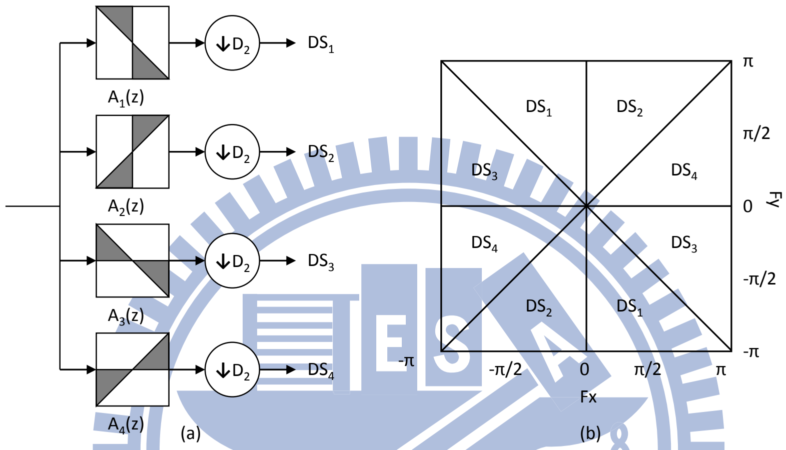


Fig. 2-5. (a) A four directional DFB structure. (b) Frequency partitions produced by the DFB in (a).

2.2.3 Contourlet Transform

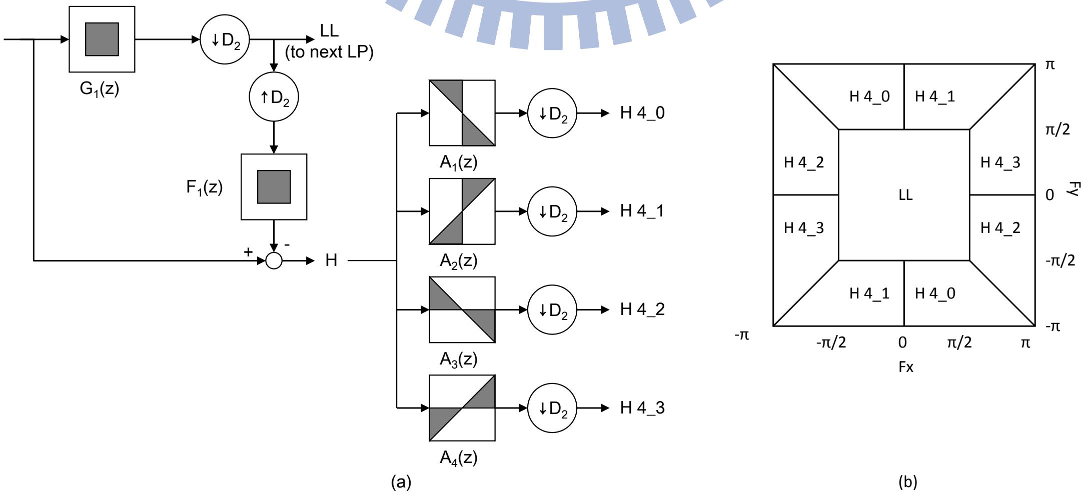


Fig. 2-6. (a) Filter bank structure of CT. (b) Frequency partition produced by CT.

CT applies DFB to the H subband in Fig. 2-6(a). Fig. 2-6(b) shows the frequency partition of each output of CT in Fig. 2-6(a). CT provides a better nonlinear approximation of 2-D signals than 2-D DWT [14]. It also provides better PSNR than 2-D DWT at low bit rate coding [48][49]. Because LP increases the data size, CT is less preferred in the compression scenario.

2.3 Wavelet-Based Contourlet Transform

Fig. 2-7(a) shows the structure of *wavelet-based contourlet transform* (WBCT) [22]. It uses the 2-D DWT to first generate four subbands, LL, HL, LH and HH. It further decomposes each of the three high-pass subband signals, HL, LH, and HH, by the DFB in Fig. 2-5(a). Fig. 2-7(b) shows the frequency partition produced by WBCT. It has the critical-sampling property and it maintains the same data size. Thus, it is more desirable for compression purpose.

The original structure of WBCT applies DFB to all high-pass subbands (HL^i, LH^i, and HH^i, $i \geq 1$). In 2-D DWT, LL^1 and its split subband signals (LL^i, HL^i, LH^i, and HH^i, where $i > 1$) contain the low and mid frequency components in the sensitive range of human visual system. When we apply the DFB to these subbands and quantize their transform coefficients, the ringing effects may appear on the smooth image regions. Thus, we tend to represent these coarse subband signals by 2-D DWT [23]. On the other hand, we apply the directional transform to HL^1, LH^1, and HH^1 to match their directional textures. But some of these subbands may be inappropriate for directional transform.

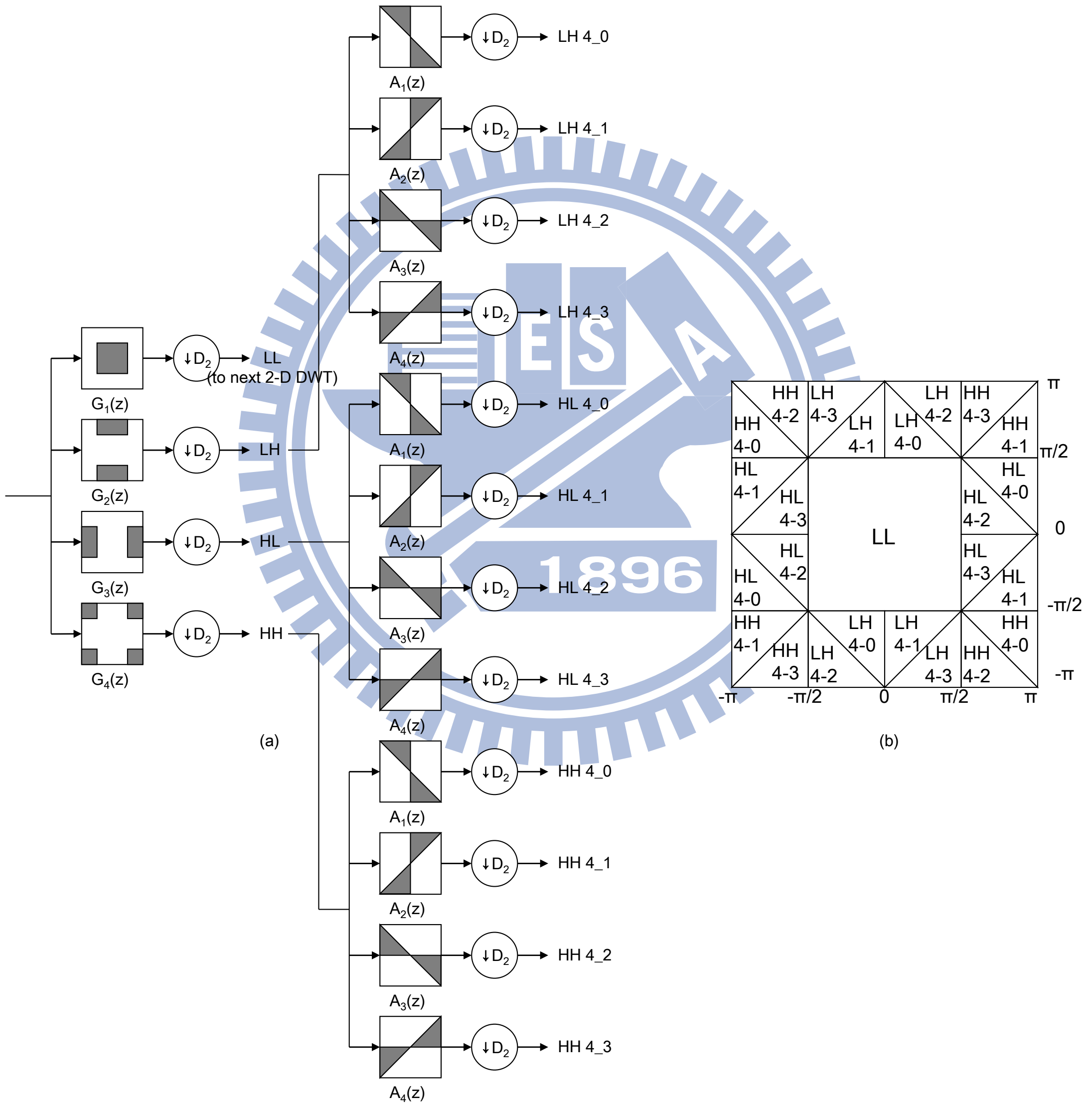


Fig. 2-7. (a) Filter bank structure of WBCT. (b) Frequency partition produced by WBCT.

2.4 Direction-Adaptive Discrete Wavelet

Transform

2.4.1 Direction-Adaptive Discrete Wavelet Transform

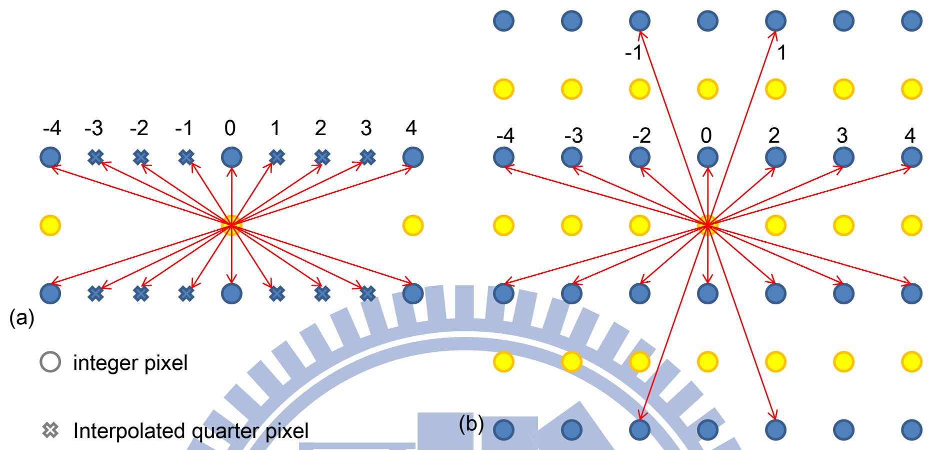


Fig. 2-8. Two sets of candidate directions (a) proposed in [28] and (b) proposed in [27]. Numbers are direction indexes.

Direction-adaptive discrete wavelet transform (DA-DWT) consists of a sets of selected candidate directions as shown in Fig. 2-8. Candidate directions in Fig. 2-8(a) and Fig. 2-8 (b) are designed for smooth and sharp textures.

Like 2-D DWT, 2-D DA-DWT first applies the first 1-D DA-DWT along the vertical candidate directions, then, it applies the second 1-D DA-DWT along the horizontal candidate directions. The first 1-D DA-DWT partitions the F_H×F_W image into non-overlapping B_H×B_W blocks. Each block $B(i, j)$ has a set of prediction errors $\{D_B(i, j; d_v)\}$, each corresponding to a vertical candidate direction d_v in Fig. 2-16(a), $1 \le i \le (F_H/B_H)$, $1 \le j \le (F_W/B_W)$, and $-4 \le d_v \le 4$. We take the sum of absolute high-pass coefficients as the prediction error. Additionally, it has been show that the sum of absolute high-pass coefficients and the sum of squared high-pass coefficients result in similar performance in coding [50].

The DA-DWT selects the best direction based on the minimum prediction errors for each $B(i, j)$. After selecting the best directions for each $B(i, j)$, the DA-DWT applied 1-D DWT along selected directions. The transforms are processed cross block boundary to avoid blocking artifact. The first DA-DWT decomposes an image into the spatial low-pass subband L (subband size is $(F_H/2)\times F_W$) and the high-pass subband H (subband size is $(F_H/2)\times F_W$) after transform. The second DA-DWT also partitions L subband into non-overlapping $(B_H/2)\times B_W$ blocks and selects the best direction for each block in a similar way. The H subband usually contains less energy. Applying the horizontal 1-D DA-DWT to it is not effective in compression [51]. Thus we apply only the horizontal 1-D DWT to H subband. We obtain four subbands, LL, LH, HL, and HH after one level of 2-D DA-DWT. We can apply another level of 2-D DA-DWT to LL for multiresolution decomposition.

2.4.2 Quadtree Partition

DA-DWT needs to transmit the side information including block partition and directional information for decoding. We adopt the quadtree partition [52] for block partition. Fig. 2-9 gives an example of quadtree partition. Each block is partitioned into four quarter sub-blocks for detail presentation. Symbol "1" and "0" present that a block is partitioned or not. The first transform and the second transform have different quadtree partitions in 2-D DA-DWT [51].

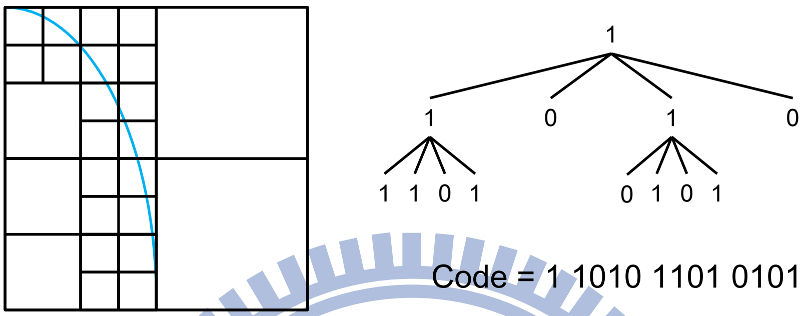


Fig. 2-9. Presenting 2-D signal by quadtree partition [52].

2.4.3 Direction Prediction Coding

The neighboring blocks usually have similar selected directions. Thus, we code the difference between two neighboring block direction indices to save bits. Fig. 2-10 gives an example of prediction of direction index [28]. θ is the direction index of the current block and α_d, α_n, and α_d are the direction indices of neighboring blocks. Because of sequential processing, we already know α_d, α_n, and α_d when decoding θ at decoder. We select a predictor θ_p from these three direction indices based on the observation that the image gradient changes smoothly in (2-1). We code the prediction error θ-θ_p as the side information and send it to decoder.

$$\theta_p = \begin{cases} \alpha_w, if \ |\alpha_d - \alpha_w| > |\alpha_d - \alpha_n| \\ \alpha_n, if \ |\alpha_d - \alpha_w| \le |\alpha_d - \alpha_n| \end{cases} \tag{2-1}$$

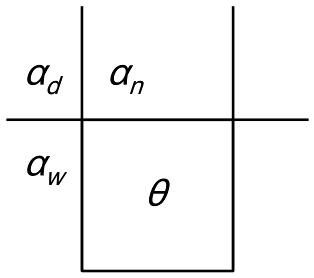
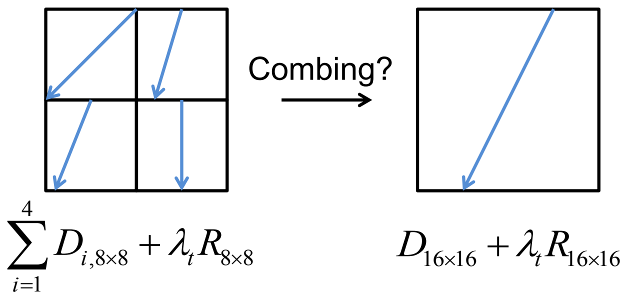


Fig. 2-10. Prediction of direction index [28].

2.4.4 Rate-Distortion Optimized Segmentation

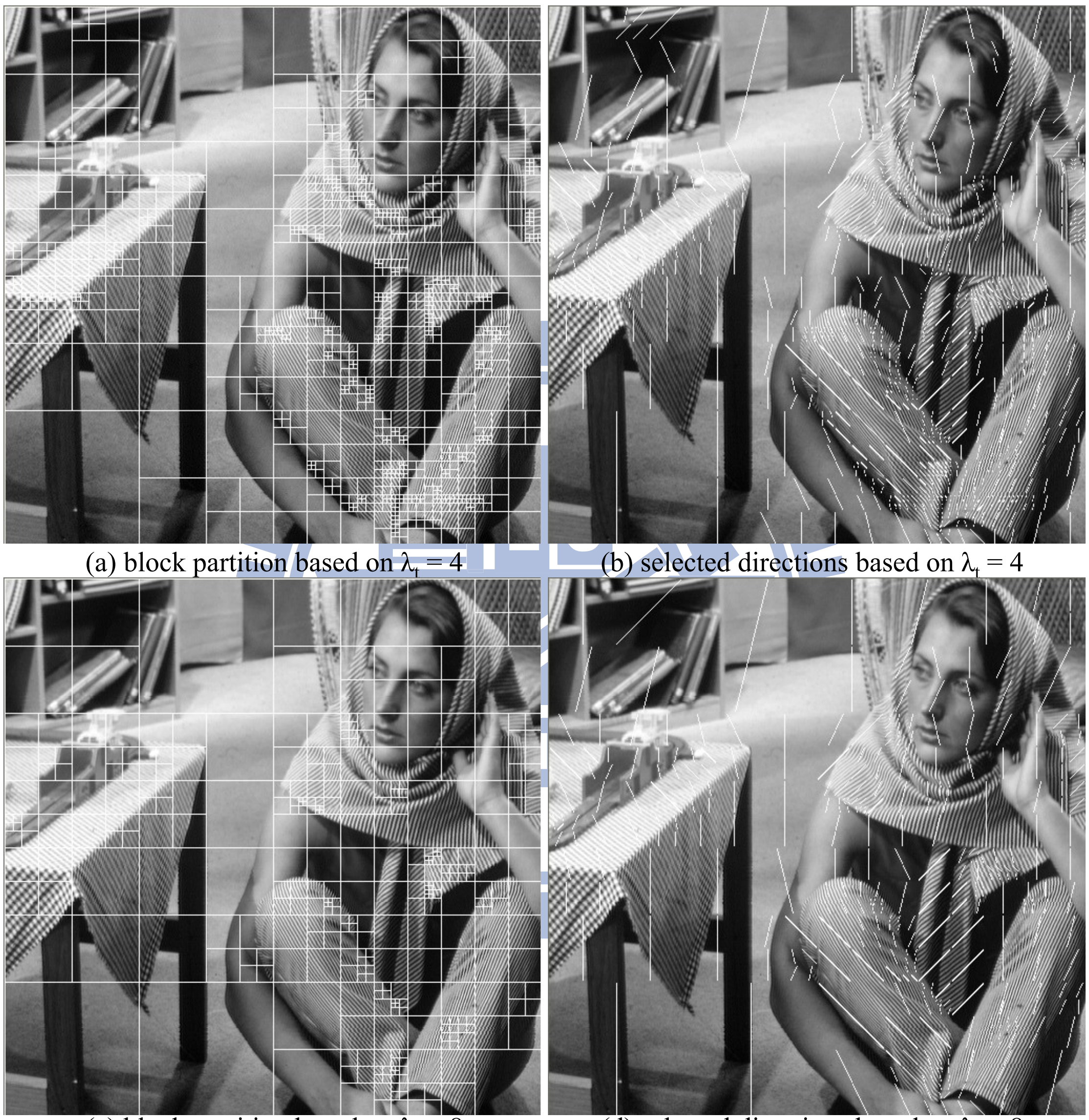
Larger partition blocks spend fewer bits for side information but it produces large prediction error. Finer partition blocks often provide good directional resolution at the cost of larger bits for side information. A good trade-off between distortion and side information is the problem of rate-distortion optimization. It is usually solved by using the Lagrangian cost function [53]. We first build a quadtree with full partition. We then calculate the cost function of every node in a quadtree partition. We finally compare these cost function values decide the partition case of each node, as an example in Fig. 2-11.



Combing?
$\sum_{i=1}^{4} D_{i,8\times8} + \lambda_t R_{8\times8}$
$D_{16\times16} + \lambda_t R_{16\times16}$

Fig. 2-11. Quadtree partition with Lagrangian cost function. λ_t is the Lagrangian multiplier.

There are two ways of count the side information. In the first count, except the information of block partition and direction, we also consider the bits of transform coefficients [27]. In the second count, the transform coefficients are not included. [28]. These two cost functions produce almost the same coding performance except for the very low bit rate cases (≤0.1 bpp) [54].

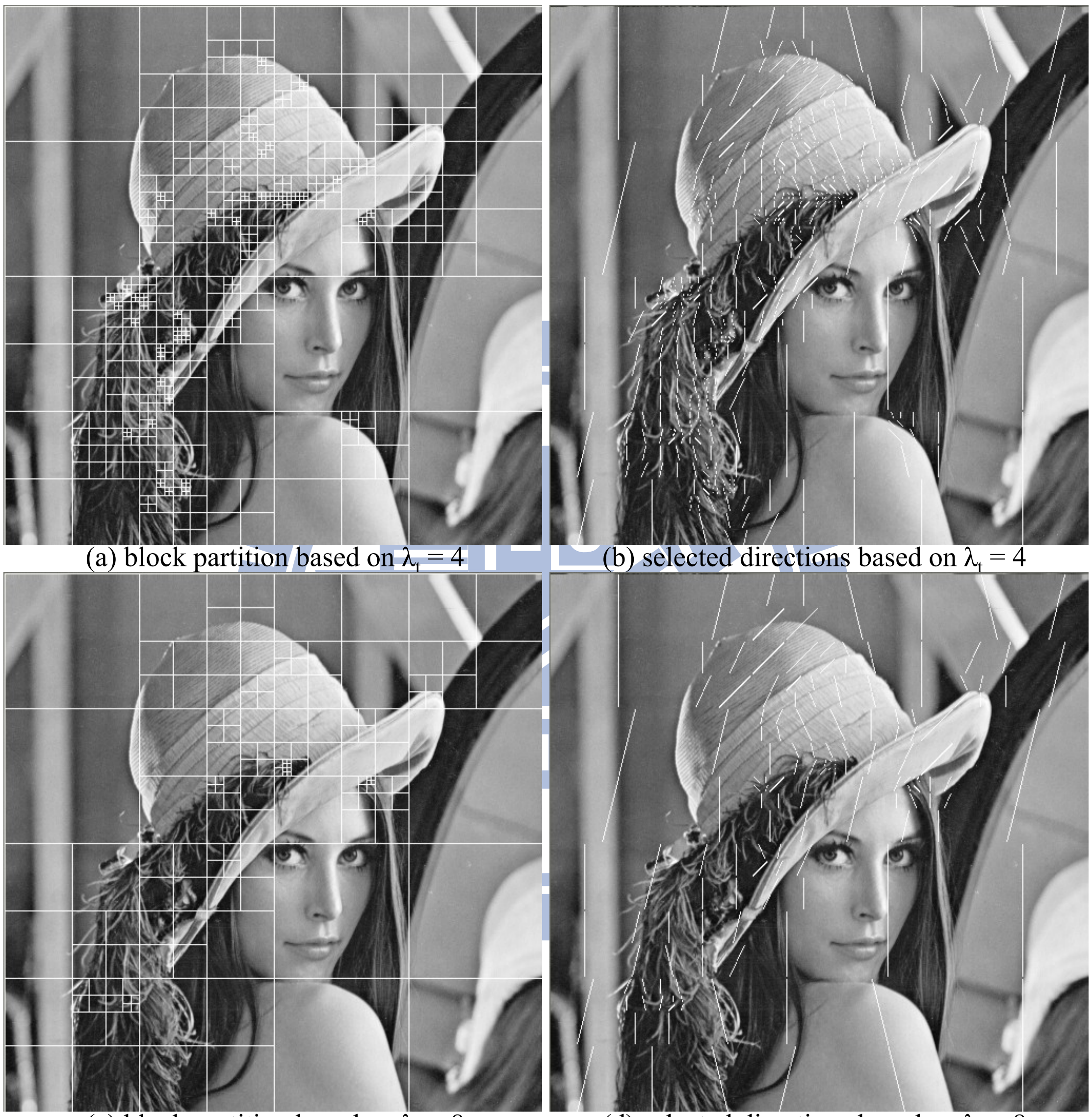


(a) block partition based on $\lambda_t = 4$ (b) selected directions based on $\lambda_t = 4$

(c) block partition based on $\lambda_t = 8$ (d) selected directions based on $\lambda_t = 8$

Fig. 2-12. Block partition and selected directions of test image *Barbara* after rate-distortion optimized segmentation based on different Lagrangian multiplier λ_t.

Fig. 2-12 and Fig. 2-13 show an example of block partition and selected directions after rate-distortion optimized segmentation of two test images. With rate-distortion optimized segmentation, we assign large blocks to the smooth regions and small blocks to the texture regions. Small Lagrangian multiplier λ_t often results in more detail block partition.


(a) block partition based on $\lambda_t = 4$ (b) selected directions based on $\lambda_t = 4$

(c) block partition based on $\lambda_t = 8$ (d) selected directions based on $\lambda_t = 8$

Fig. 2-13. Block partition and selected directions of test image *Lena* after rate-distortion optimized segmentation based on different Lagrangian multiplier λ_t.

2.5 Megablocking Partition

The quadtree partition adopts a parent-child pruning procedure to produce the optimal partition under the constrained bit budget. It ignores the correlations between neighboring nodes partitioned from different parent nodes. Thus its fails to achieve the optimal R-D performance [55]. The prune-join scheme extends the concept of pruning the child nodes to the concept of joining the similar neighbor nodes. The megablocking partition adopts this idea and achieves a better R-D performance for DA-DWT [15].

The megablocking partition first uses the quadtree partition to achieve block partition for the entire image. Each block has four neighbor blocks locating at its up, down, left, and right side [55]. The megablocking partition then joins the blocks with the same direction to form a megablock. It defines two types of blocks, inner blocks and boundary blocks, for recording the megablocking information. The Inner block has all its neighboring blocks within the same megablock. On the other hand, the boundary block has at least one neighboring block from another megablock. The megablocking partition scheme encodes one directional information for each megablock. Thus, it saves a large amount of side information. Fig. 2-14 gives an example of megablocking partition.

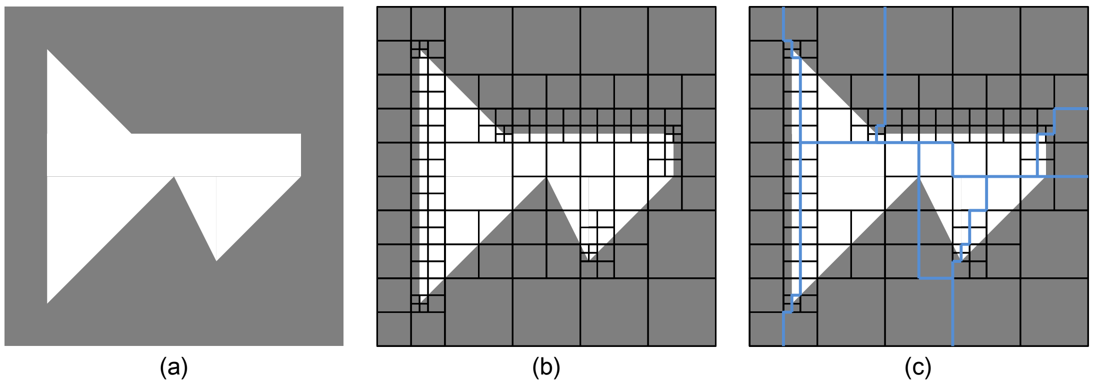


Fig. 2-14. (a) test image of polygonal model, (b) quadtree partition, (c) megablocking partition.

2.6 Subsampling and Direction-Adaptive Discrete Wavelet Transform

2.6.1 Subsampling Patterns

The 2-D DWT applies 1-D DWT along the vertical then the horizontal directions, and so does the 2-D DA-DWT. Different execution orders of these two 1-D transforms have no effect on final results of 2-D DWT, but affect that of 2-D DA-DWT. If the following conditions hold, the order of transform makes no difference. First, the pixel can be predicted by its neighboring pixels as much as possible. Thus, the candidate directions should angularly cover the whole plane. Second, we must decompose high-pass subbands fully to reduce its energy. Thus, we should find the best weighting factors of samples used for prediction by minimizing the prediction *mean square error* (MSE) [56]. However, the candidate directions of each order of transform cannot cover as whole range as supposed. We usually adopt the fixed weights (conventional lifting coefficients) for lifting scheme. Thus, we get different results by using different ordering in applying 1-D transforms in 2-D DA-DWT [31].

In a lifting-based wavelet transform, we divide the pixels of an image I into two separate subsets, I_L and I_H, where $I_L \cup I_H = I$ and $I_L \cap I_H = \emptyset$. In the vertical transform of 1-D DWT, I_L and I_H are rows of even and odd indexes. We use the prediction step (2-2) and update step (2-3) to obtain the low-pass subbands coefficients C_L and high-pass coefficients C_H.

$$C_H = I_H - P(I_L) \tag{2-2}$$

$$C_L = I_L - U(C_H) \tag{2-3}$$

$P(\ )$ and $U(\ )$ are prediction and update operators. In 2-D DWT, C_L can be decomposed into C_{LL} and C_{LH}, and C_H can be decomposed into C_{HL} and C_{HH} by another 1-D DWT.

The conventional 2-D DWT and the 2-D DA-DWT applies transform between rows (along the vertical direction) than between columns (along the horizontal direction). Thus, this transform order is called subsampling pattern *row-column* (RC) in Fig. 2-15(a). The so-called SA-DWT algorithm includes another two subsampling patterns, *column-row* (CR) and *quincunx* (QU), shown in Fig. 2-15(b) and Fig. 2-15(c). Fig. 2-16 and Fig. 2-17 show the candidate directions of different subsampling patterns. Fig. 2-16(c) and Fig. 2-17(c) show that the QU's candidate directions cover a wide range. Thus, the image rotation has no effect on QU's coding performance [31]. However, the QU provides poor coding performance for most natural images because of the far away reference pixels for prediction [57]. Each subsampling pattern has its best performed texture [32].

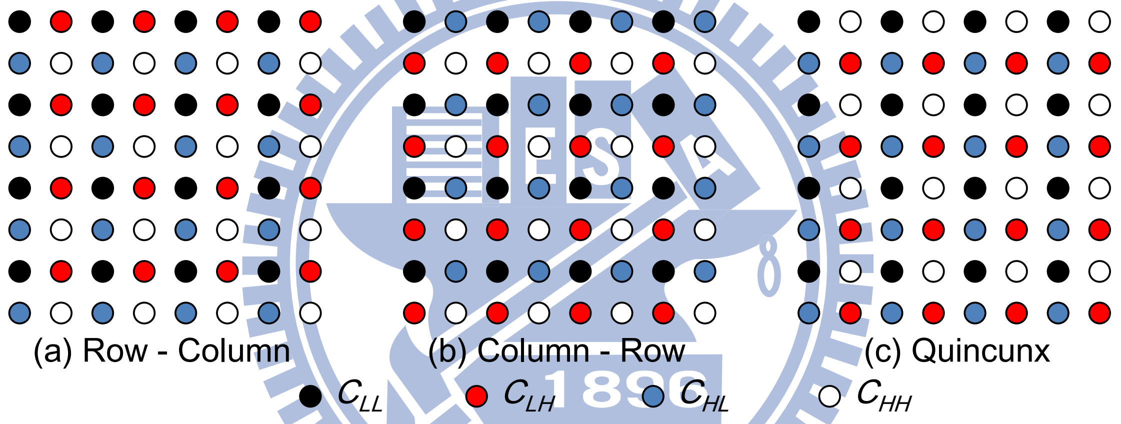


Fig. 2-15. Four subbands of different subsampling patterns.

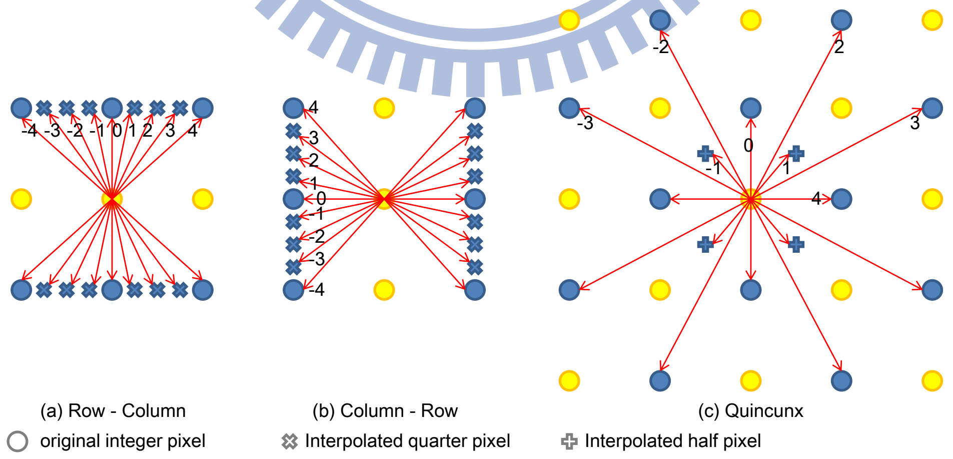


Fig. 2-16. The candidate first directions of different subsampling patterns.

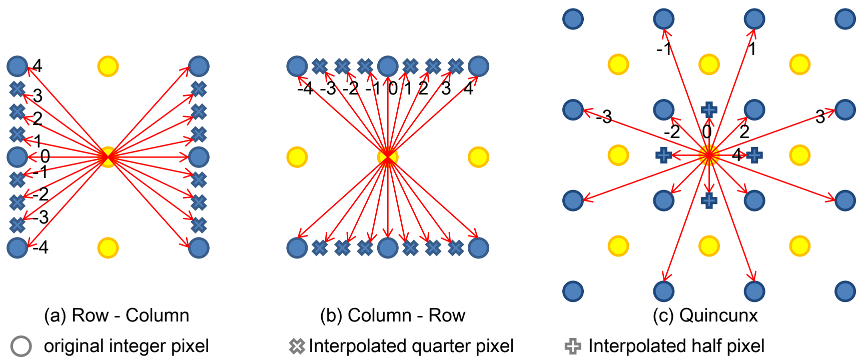


Fig. 2-17. The candidate second directions of different subsampling patterns.

2.6.2 Phase-Completion Process

When applying the the first transform of 2-D SA-DWT, we encounter a problem due to the non-uniform distribution of subset partitions between neighboring blocks with different subsampling patterns as shown in Fig. 2-18. I_L (including C_{LL} and C_{LH}) and I_H (including C_{HL} and C_{HH}) have mismatched locations in two neighboring blocks in Fig. 2-18. The pixel $\alpha \in I_H$ can be predicted by the pixel $\beta \in I_L$, but not by the pixel $\gamma \in I_H$ along the direction in Fig. 2-18. We need to estimate the I_L at location of γ to predict α. We resolve this problem by the phase-completion process $PC(\ )$ in (2-4) and (2-5).

$$C_H = I_H - P(PC(I_L)) \tag{2-4}$$

$$C_L = I_L - U(PC(C_H)) \tag{2-5}$$

The operation of $PC(I_L)$ in (2-4) is as follows. It estimated loss samples $\in I_L$ from neighboring pixels $\in I_L$. Since usually the local correlation within an image is strong, estimation of I_L from neighboring I_L is feasible. We use the average of all pixels $\in I_L$ within a window to get estimation. For example, in Fig. 2-18, we take the average of I_L within a 3×3 window centered at γ as the estimation of I_L located at γ. The operation of $PC(C_H)$ in (2-5) is similar as above.

For the second transform, we use C_{LL} to predict C_{LH}. These two phases of pixels uniformly distribute in all three subsampling patterns. Thus, we can apply the second transform without the different neighboring subsampling problem.

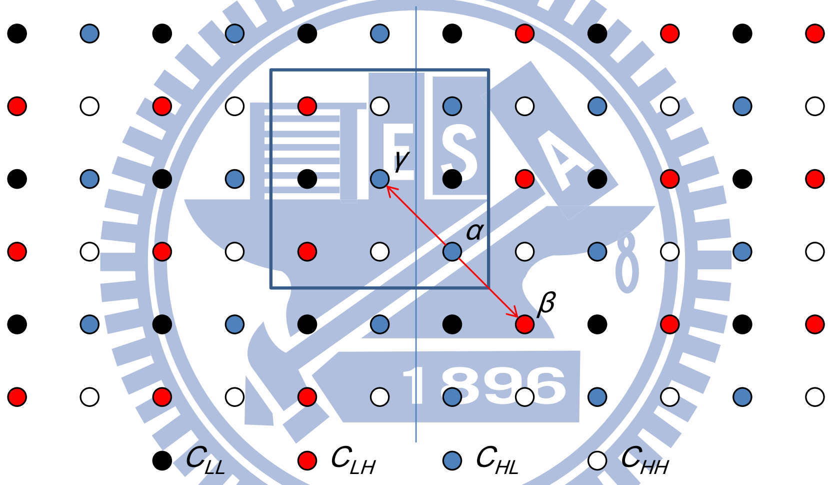


Fig. 2-18. Phase-completion between process neighboring blocks adopt different subsampling patterns. Left block uses CR while right block uses RC.

Chapter 3 Temporal-Domain Wavelet Transform and Entropy Coding

3.1 Motion-Compensated Temporal Filtering

The *Motion-compensated temporal filtering* (MCTF) is a technique that performs temporal subband decomposition on video sequences. It decomposes the original video frames into temporal low-pass residuals T_Ls and temporal high-pass residuals T_Hs. The goal of MCTF is to compact the temporal energy along motion trajectory of a video sequence.

An improved version of MCTF adopting the biorthogonal 5/3 filters for lifting schemes in is shown Fig. 3-1 [11]. There are three steps in the lifting scheme of MCTF, polyphase decomposition, prediction step, and update step. The polyphase decomposition splits the input frames F_k into odd frames F_{2i} and even frames F_{2i+1}. The prediction step generates the high-pass residuals H_i by predicting F_{2i+1} from F_{2i} and F_{2i+2}:

$$H_i = F_{2i+1} - \frac{1}{2}(MC(F_{2i}, MV_{2i+1\to 2i}) + MC(F_{2i+2}, MV_{2i+1\to 2i+2})) \qquad (3\text{-}1)$$

where $MV_{2i+1\to 2i}$ is the motion vector from frame F_{2i+1} to F_{2i}. MC(F_{2i}, $MV_{2i+1\to 2i}$) is the motion compensation process using motion vector $MV_{2i+1\to 2i}$ to generate the predicted pixels of F_{2i+1} from F_{2i}. Then, the update step generates the low-pass residuals by updating F_{2i} from H_{i-1} and H_i:

$$L_i = F_{2i} + \frac{1}{4}(MC(H_{i-1}, MV_{2i\to 2i-1}) + MC(H_i, MV_{2i\to 2i+1})) \qquad (3\text{-}2)$$

Through one level of MCTF, video frames are decomposed into T_Ls and T_Hs. Another level of MCTF decompose low-pass residuals again and iteratively to achieve temporal scalability. Fig. 3-2 shows the temporal residuals after a 4-level MCTF applied to 16 input frames, F_0~F_{15}. The result includes 1 temporal low-pass residual LLLL$_0$, and 15 high-pass residuals, LLLH$_0$, LLH$_0$~ LLH$_1$, LH$_0$~ LH$_3$, H$_0$~ H$_7$.

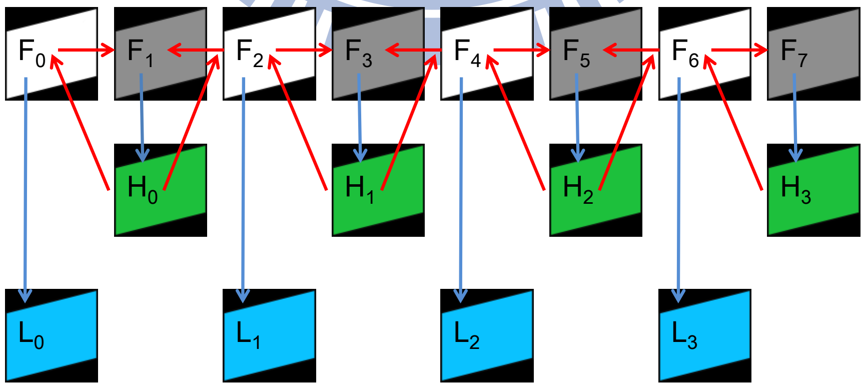


Fig. 3-1. Lifting scheme with biorthognoal 5/3 filters in MCTF. F_i, H_i, and L_i are the original video sequences, the temporal high-pass residuals, and the temporal low-pass residuals.

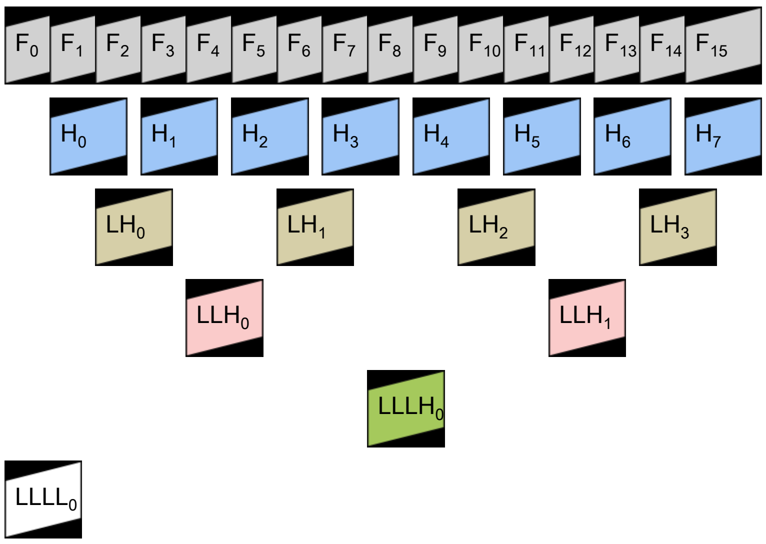

F_0 F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9 F_{10} F_{11} F_{12} F_{13} F_{14} F_{15}
H_0 H_1 H_2 H_3 H_4 H_5 H_6 H_7
LH_0 LH_1 LH_2 LH_3
LLH_0 LLH_1
$LLLH_0$
$LLLL_0$

Fig. 3-2. Temporal residuals after four levels of MCTF applied to 16 input frames, F_0~F_{15}.

3.2 Characteristics of Prediction Residuals

3.2.1 Auto-Covariance Model

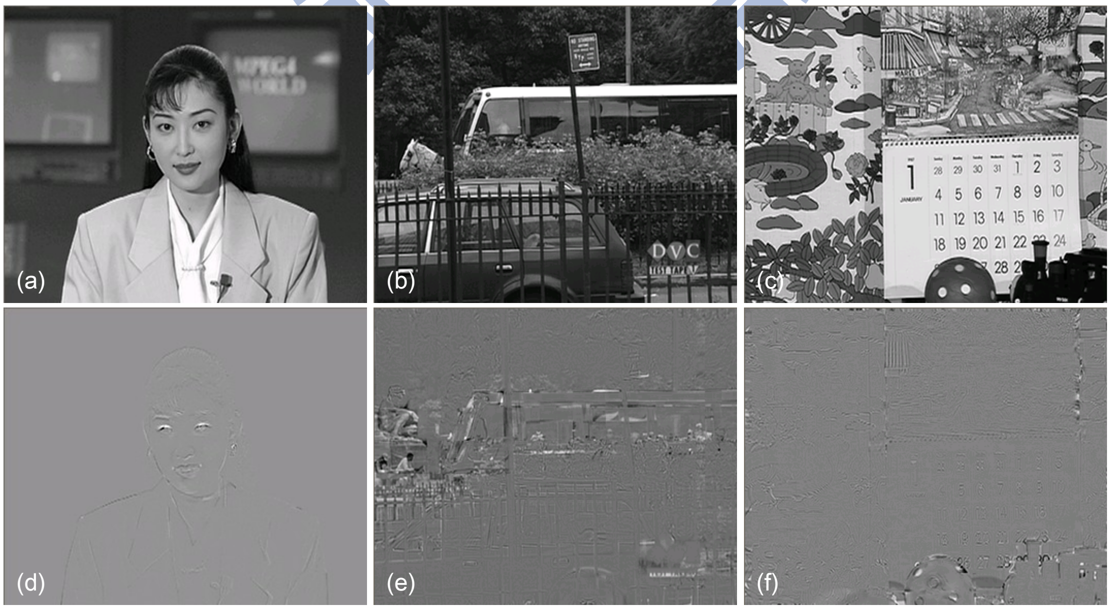


Fig. 3-3. Temporal residuals of three test video sequences. (a) $LLLL_0$ of *Akiyo*, (b) $LLLL_0$ of *Bus*, (c) $LLLL_0$ of *Mobile*, (d) $LLLH_0$ of *Akiyo*, (e) $LLLH_0$ of *Bus*, (f) $LLLH_0$ of *Mobile*.

Fig. 3-3 shows the T_Ls and T_Hs of different test video sequence. The temporal T_Ls are similar to the original images but the T_Hs are similar to the motion-compensated residuals. The motion-compensated residuals have different spatial characteristics from the original images [43]. In T_Hs, the coefficients forming large smooth regions are negligibly small. The large coefficients concentrate along object boundaries and edges to form 1-D structures.

We analyze the characteristics of image and prediction residuals by two auto-covariance models, separable model in (3-3) and generalized model in (3-4) [43]. The generalized model is the rotated version of the separable model [43]. The generalized model allows rotation of axes of the auto-covariance model and enables the capturing local anisotropic features in higher precision.

$$R_s(I,J) = \rho_1^{|I|}\rho_2^{|J|} \tag{3-3}$$

$$R_g(\theta,I,J) = \rho_1^{|I\cos(\theta)+J\sin(\theta)|}\rho_2^{|-I\sin(\theta)+J\cos(\theta)|} \tag{3-4}$$

We estimate parameters ρ_1, ρ_2, and θ for the generalized model of the images and the prediction residuals by the following steps. We first partition an image into 8×8 blocks. We calculate the auto-covariance of each 8×8 block by removing its mean, correlating the zero-mean block with itself, and dividing the correlation by the block variance. We set shifts I and J as integers between -7 to 7. Then, we minimize the MSE between the auto-covariance and the generalized model in (3-4) by adjusting the above three parameters. We obtain these parameters from with the minimal MSE.

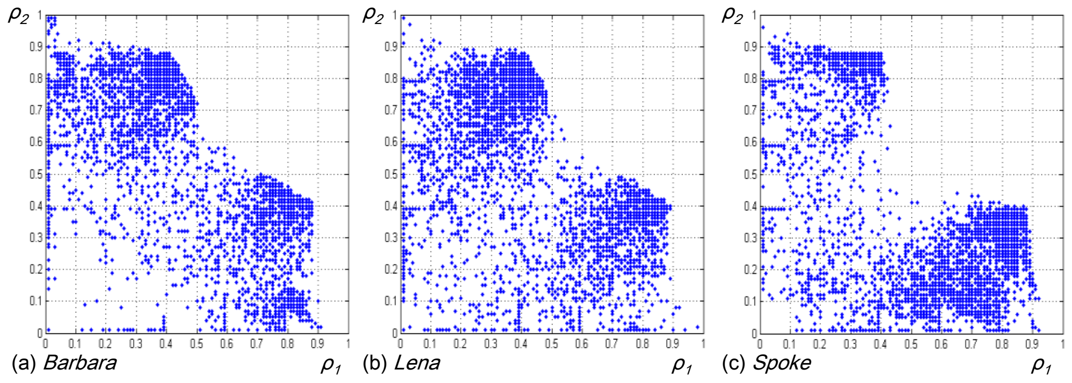


Fig. 3-4. Scatter plots of estimated (ρ_1, ρ_2) from images. Fig. 6-6 shows the original images

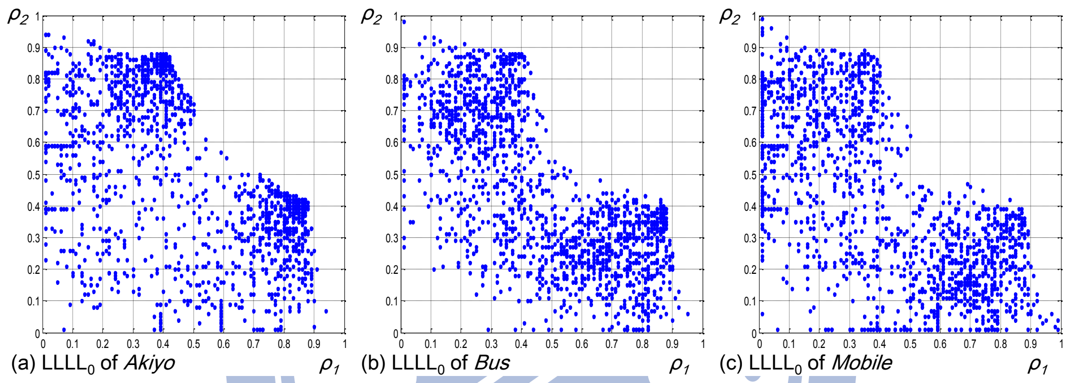


Fig. 3-5. Scatter plots of estimated (ρ_1, ρ_2) from temporal low-pass residuals.

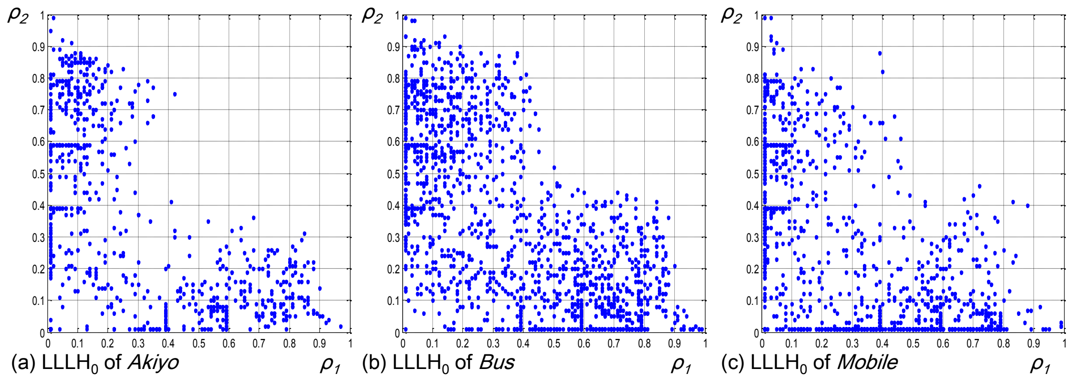


Fig. 3-6. Scatter plots of estimated (ρ_1, ρ_2) from temporal high-pass residuals.

We plot the estimated ρ_1 and ρ_2 of the generalized model in Fig. 3-4, Fig. 3-5, and Fig. 3-6. The data points (ρ_1, ρ_2) in Fig. 3-6 move closer to two axes than those in Fig. 3-4 and Fig. 3-5. It means that T_Hs have strong correlation along one direction but weak correlations along the other direction. The large coefficients in T_Hs concentrate along the object boundaries and form 1-D structures [43]. Applying 2-D transform to these 1-D structure signals results in spreading coefficients and deteriorate the coding performance [43].

3.3 Embedded Block Coding with Optimized Truncation

JPEG2000 adopts the *embedded block coding with optimized truncation* (EBCOT) technique as entropy coding unit [6][35]. EBCOT exhibits state-of-art coding scheme and it produces bitstream with scalability and random access property. It provides a set of context tables with the consideration of characteristics of transformed coefficients within different spatial subbands.

2-D DWT decomposes an image into many spatial subbands. EBCOT partition each subband into a number of 32×32 or 64×64 coding blocks. EBCOT is a bit-plane coding and accesses each bit of a coefficient from the *most significant bit* (MSB) to the *least significant bit* (LSB). Thus, EBCOT is also a binary symbol coding that encodes only “0” and “1”.

For each coefficient $x[i, j]$ at position $[i, j]$, we assign it a binary-valued state variable $\sigma[i, j]$, which indicates the significance of this coefficient. $\chi[i, j]$ is the sign of $x[i, j]$. It is 0 when the sample is positive and 1 when the sample is negative. $\sigma[i, j]$ is initialized to 0 and toggled to 1 when the $x[i, j]$'s first non-zero bit-plane value is encoded. There are 4 coding operations and they are activated by $\sigma[i, j]$.

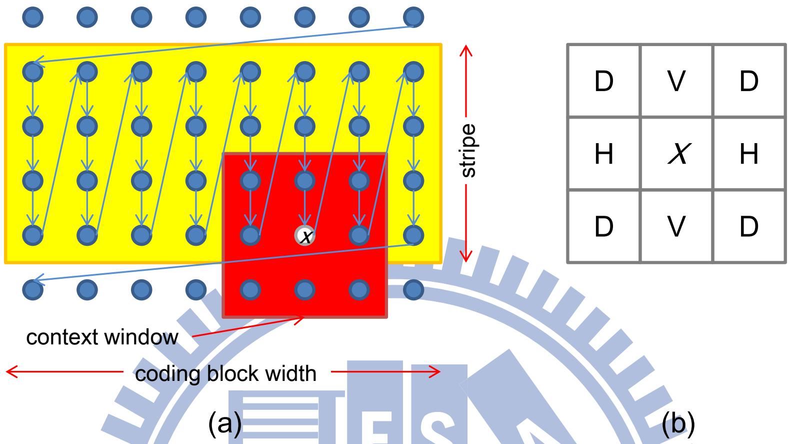


Fig. 3-7. (a) Stripe-oriented scanning path. (b) Neighbors within the context window.

For each coding pass on a bit-plane, EBCOT scans every bit along the path in Fig. 3-7(a). When encountering a bit needed to encode, it encodes this bit with the consideration of its 8 neighbors within a 3×3 context window. Fig. 3-7(b) shows the labels of these neighbors.

3.3.1 Coding Operations

EBCOT includes 4 coding operations, significance coding, sign coding, magnitude refinement coding, and clean up coding. We describe these coding operations as follows.

3.3.1.1 Significance Coding

If a coefficient $x[i, j]$ is not yet significant in the previous bit-planes, the significance coding estimates the probability of $x[i, j]$ becoming significant from its 8 neighbors. It classifies the significance situations of neighbors as the contexts given in Table 3-1. The significance coding uses the contexts in Table 3-1 to code $x[i, j]$'s significant information in the current bit-plane.

Table 3-1. Context table of significance coding. "X" means don't care.

wavelet subband	LL LH			HL			HH	
context	H	V	D	V	H	D	H+V	D
8	2	X	X	2	X	X	X	≥3
7	1	≥1	X	1	≥1	X	≥1	2
6	1	0	≥1	1	0	≥1	0	2
5	1	0	0	1	0	0	≥2	1
4	0	2	X	0	2	X	1	1
3	0	1	X	0	1	X	0	1
2	0	0	≥2	0	0	≥2	≥2	0
1	0	0	1	0	0	1	1	0
0	0	0	0	0	0	0	0	0

3.3.1.2 Sign Coding

If a coefficient $x[i, j]$ becomes significant at the current bit-plane, we set $\sigma[i, j] = 1$ and use the sign coding to code its sign $\chi[i, j]$. The sign coding calculates the horizontal contribution χ_H and the vertical contribution χ_V as follows:

$$\chi_H = \min\{1, \max\{-1, \sigma[i-1, j] \times (1-2\chi[i-1, j]) + \sigma[i+1, j] \times (1-2\chi[i+1, j])\}\} \tag{3-5}$$

$$\chi_V = \min\{1, \max\{-1, \sigma[i,j-1] \times (1-2\chi[i,j-1]) + \sigma[i,j+1]\times(1-2\chi[i,j+1])\}\} \quad (3\text{-}6)$$

Then sign coding using χ_H and χ_V to obtain the sign prediction χ_P in Table 3-2 . It codes the XOR results of χ_P and $\chi[i,j]$ using Table 3-2.

Table 3-2. Context table and sign prediction of sign coding

χ_H	χ_V	context	χ_P
1	1	13	1
1	0	12	1
1	-1	11	1
0	1	10	1
0	0	9	1
0	-1	10	-1
-1	1	11	-1
-1	0	12	-1
-1	-1	13	-1

3.3.1.3 Magnitude Refinement Coding

Table 3-3. Context table for magnitude refinement coding

H+V+D	first time for magnitude refinement coding	context
X	False	16
≥1	True	15
0	True	14

The magnitude refinement coding codes the new information of $x[i, j]$ if it becomes significant in the previous bit-plane. It uses 3 contexts for arithmetic coding in Table 3-3.

1. If $x[i, j]$ with no significant neighbors has not been coded by magnitude refinement coding, the context table of $x[i, j]$ is 14.
2. If $x[i, j]$ with at least one significant neighbor has not been coded by magnitude refinement coding, the context table of $x[i, j]$ is 15.

3. Otherwise, the context is 16.

3.3.1.4 Run Length Coding

If 4 consecutive coefficients along the stripe column in Fig. 3-7(a) are all insignificant, and their surrounding 14 coefficients are all insignificant, we code these 4 coefficients by the run length coding. When a group of 4 coefficients satisfy the above condition, the run length coding codes them by a single symbol “0”.

If one of these 4 coefficients becomes significant in the current bit-plane, the run length coding codes them by a single symbol “1”. Then, it uses two bits, “00”, “01”, “10”, or “11” to encode the position of significant coefficients.

3.3.2 Coding Passes

EBCOT includes 3 different coding passes and each coding passes includes the 4 coding operations in the above. Multiple coding passes separate the bits within a coding block into smaller subsets. The results thus form a finely embedded bitstream. We introduce these coding passes as follows.

3.3.2.1 Significance Propagation Pass

The significance propagation pass processes an insignificant coefficient with at least one significant neighbor. It uses the significance coding to code the significance information in the current bit-plane of this coefficient. If the coefficient becomes significant in the current bit-plane, then the sign coding is used to code the sign.

3.3.2.2 Magnitude Refinement Pass

This coding pass processes the coefficients that were already significant in the previous bit-planes. It uses the magnitude refinement coding to code the binary bits corresponding to these coefficients in the current bit-plane.

3.3.2.3 Cleanup Pass

This coding pass processes the coefficients that were not processed by previous two coding passes at the current bit-plane. It uses the run length coding to codes the information of 4 consecutive insignificant coefficients along the stripe column with 14 insignificant neighbors. Other un-processed coefficients are coded by significance coding and sign coding.

3.4 Three-dimensional Embedded Subband Coding with Optimized

Truncation

Three-dimensional Embedded Subband Coding with Optimal Truncation (3-D ESCOT) [36] is an extension of EBCOT used in video coding. It offers high compression efficiency and other functionalities, such as error resilience and random access.

3-D ESCOT takes the advantages of the orientation-invariant property of wavelet subbands to reduce the number of context. It codes each subband independently so that each subband can be decoded independently. Because of this feature, 3-D ESCOT can achieve a flexible spatial/temporal scalability and the R-D optimization can be done within subbands to improve compression efficiency.

In addition to the spatial wavelet subbands (LL, LH, HL, and HH) produced by 2-D DWT applying to the residuals, 3-D ESCOT also considers the orientation of temporal wavelet subbands. The temporal low-pass residuals $LLLL_0$ in Fig. 3-2 is denoted as “L” and the other high-pass residuals are denoted as “H” in the orientation consideration. Thus each subband in 3-D ESCOT has three orientations. Each subband is divided into 3-D coding blocks and these coding blocks are coded independently.

For each coefficient $x[i, j, k]$ at position $[i, j, k]$, we assign it a binary-valued state variable $\sigma[i, j, k]$, which indicates the significance of this coefficient. $\chi[i, j, k]$ is the sign of the $x[i, j, k]$. It is 0 when the sample is positive and 1 when the sample is negative. $\sigma[i, j, k]$ is initialized to 0 and toggled to 1 when the $x[i, j, k]$'s first non-zero bit-plane value is encoded. When a bit need to be coded, 3-D ESCOT encodes this bit by checking its 18 neighbors within a 3×3×3 cubic in Fig. 3-8.

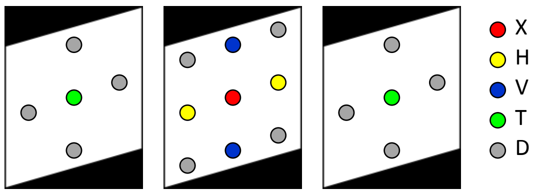


Fig. 3-8. Neighbors within the cubic.

3.4.1 Coding Operations

There are 3 coding operations in 3-D ESCOT and their use is controlled by $\sigma[i, j, k]$. The zero coding (ZC) and the sign coding (SC) are used to code $x[i, j, k]$ if $\sigma[i, j, k] = 0$ and magnitude refinement (MR) is used if $\sigma[i, j, k] = 1$. We will describe these 3 coding operations as follows.

3.4.1.1 Zero Coding

If a coefficient $x[i, j, k]$ is not yet significant in the previous bit-planes, ZC estimates the probability of $x[i, j, k]$ becoming significant from its 18 neighbors in Fig. 3-8. It classifies the significance situations of neighbors as contexts in Table 3-4. ZC uses the contexts in Table 3-4 to code the significance information of $x[i, j, k]$.

Table 3-4. Context table of zero coding. “X” means don’t care.

wavelet subband	LLL LLH				LHH			HHH	
Context	H	V	T	D	H	V+T	D	D	H+V+T
0	2	X	X	X	2	X	X		
0	1	≥1	X	X	1	≥3	X	≥6	X
1	1	0	≥1	X	1	≥1	≥4	≥4	≥3
2	1	0	0	X	1	≥1	X	≥4	X
3	0	2	0	X	1	0	≥4	≥2	≥4
4	0	1	0	X	1	0	X	≥2	≥2
5	0	0	≥1	X	0	≥3	X	≥2	X
6	0	0	0	3	0	≥1	≥4	≥0	≥4
7	0	0	0	2	0	≥1	X	≥0	≥2
8	0	0	0	1	0	0	≥4	≥0	1
9	0	0	0	0	0	0	X	≥0	0

3.4.1.2 Sign Coding

Table 3-5. Context table and sign prediction of sign coding.

$\chi_H = -1$				$\chi_H = 0$				$\chi_H = 1$			
χ_V	χ_T	χ_P	context	χ_V	χ_T	χ_P	context	χ_V	χ_T	χ_P	context
-1	-1	0	0	-1	-1	0	9	-1	-1	1	8
-1	0	0	1	-1	0	0	10	-1	0	1	7
-1	1	0	2	-1	1	0	11	-1	1	1	6
0	-1	0	3	0	-1	0	12	0	-1	1	5
0	0	0	4	0	0	0	13	0	0	1	4
0	1	0	5	0	1	1	12	0	1	1	3
1	-1	0	6	1	-1	1	11	1	-1	1	2
1	0	0	7	1	0	1	10	1	0	1	1
1	1	0	8	1	1	1	9	1	1	1	0

If a coefficient $x[i, j, k]$ becomes significant at the current bit-plane, we set $\sigma[i, j, k] = 1$ and use SC to code its sign $\chi[i, j, k]$. SC calculates the horizontal contribution χ_H, the vertical contribution χ_V, and the temporal contribution χ_T, as following:

$$\chi_H = \min\{1, \max\{-1, \sigma[i\text{-}1, j, k] \times (1\text{-}2\chi[i\text{-}1, j, k]) + \sigma[i\text{+}1, j, k] \times (1\text{-}2\chi[i\text{+}1, j, k])\}\} \tag{3-7}$$

$$\chi_V = \min\{1, \max\{-1, \sigma[i, j\text{-}1, k] \times (1\text{-}2\chi[i, j\text{-}1, k]) + \sigma[i, j\text{+}1, k] \times (1\text{-}2\chi[i, j\text{+}1, k])\}\} \tag{3-8}$$

$$\chi_T = \min\{1, \max\{-1, \sigma[i, j, k\text{-}1] \times (1\text{-}2\chi[i, j\text{-}1, k]) + \sigma[i, j, k\text{+}1] \times (1\text{-}2\chi[i, j, k\text{+}1])\}\} \tag{3-9}$$

Then, SC uses χ_H, χ_V, and χ_T to get the sign prediction χ_P in Table 3-5. It codes the XOR results of χ_P and $\chi[i, j, k]$ using Table 3-5.

3.4.1.3 Magnitude Refinement Coding

Table 3-6. Context table for magnitude refinement coding

H+V+D+T	first time for magnitude refinement coding	context
X	False	2
≥1	True	1
0	True	0

MR encodes the new information about $x[i, j, k]$ if it became significant in the previous bit plane. It uses three contexts in Table 3-6 for arithmetic coding.

1. If $x[i, j, k]$ with no significant neighbors has not been coded by MR, the context table of $x[i, j, k]$ is 0.
4. If $x[i, j, k]$ with at least one significant neighbor has not been coded by MR, the context table of $x[i, j, k]$ is 1.
5. Otherwise, the context is 2.

3.4.2 Coding Passes

3-D ESCOT provides a high coding gain due to the use of fractional bit-plane coding. The fractional bit-plane coding provides a practical means of scanning the wavelet coefficients within each bit-plane for rate-distortion (R-D) optimization at different rates. There are 3 different fractional bit-plane coding passes and the scanning order in each of them is along the *i*-direction firstly, then the *j*-direction and the *k*-direction lastly.

3.4.2.1 Significance Propagation Pass

If the coefficients which are not yet significant but have "preferred neighborhood" are processed by this pass. A coefficient has a "preferred neighborhood" if and only if the coefficient has at least one significant immediate diagonal neighbor for the HHH subband or the horizontal, vertical, temporal neighbor for the other types of subbands. For these coefficients, we apply the ZC to code their significance information in the current bit-plane of this coefficient. If the coefficient becomes significant in the current bit-plane, then SC is used to code the sign.

3.4.2.2 Magnitude Refinement Pass

If the coefficient became significant in the previous bit-plane, it will be coded in this pass. The binary bits corresponding to these coefficients in the current bit-plane are coded by MR.

3.4.2.3 Normalization Pass

It is used to code the coefficients if it is not coded in the previous two passes.

Because these coefficients are not yet significant, they are only processed by ZC and SC.

Chapter 4 Enhanced Wavelet-based Contourlet Image Coding

Combining WBCT and 3-D ESCOT, a WBCT image coding scheme can achieve a better coding performance than a regular 2-D DWT image coding scheme. However, there are a few issues in the existing WBCT coding schemes. They need a large amount of computations because the existing WBCT directional filters have a large support. And, we found that for a specific picture, some WBCT frequency subbands do not need further directional transform. Furthermore, the context table in 3-D ESCOT needs adjustment to match the characteristics of quantized WBCT coefficients.

To solve these issues, we propose three algorithms in this paper to enhance the WBCT image coding scheme. First, we suggest a set of short-length 2-D directional filters and verify their performance. Second, we design a mean-shift-based decision scheme to dynamically select the proper subbands for directional transform. Third, we re-design the context tables of 3-D ESCOT to match the data directionality. With these algorithms, our proposed scheme reduces 92% or higher the computational complexity of the original WBCT image coding scheme at similar visual quality.

4-1 Short-Length 2-D Filters

To reduce computational load of the current WBCT, we design new short-length 2-D filters (SLF). The design procedure is as follows. We first choose an appropriate 1-D filter, up-sample it, and map it to a 2-D filter.

We begin our design from a 1-D type-II linear phase finite impulse response filter [60][65]. Eq. (4-1) is a 1-D prototype filter $\beta(z)$, wherein the coefficients $\{v_k\}$ satisfy (4-2) so that $\beta(e^{j0})=1$. When $N_1=1$ (short filter), $\beta(z)$ has a wide transition band. To keep a good balance between the transition band width and the filter length, we select $N_1=2$, and thus, $v_1=0.5916$ and $v_2=-0.0982$. Fig. 4-1 (a) and (b) show the magnitude and the phase responses of $\beta(z)$. We up-sample $\beta(z)$ by 2 and get $\beta(z^2)$. Fig. 4-1 (c) and (d) show the magnitude and the phase responses of $\beta(z^2)$. In Fig. 4-1(d), $\beta(z^2)$ contains a phase discontinuity of π at frequency 0.5π. Because of this phase discontinuity, the left-side and the right-side amplitudes in Fig. 4-1(c) have different signs.

$$\beta(z)=\sum_{k=1}^{N_1} v_k \cdot (z^{-N_1+k}+z^{-N_1-k+1}) \tag{4-1}$$

$$\sum_{k=1}^{N_1} v_k = 0.5 \tag{4-2}$$

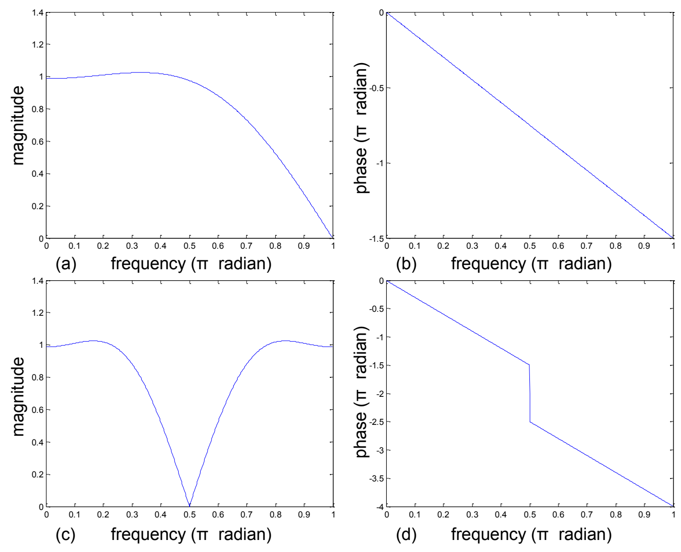


Fig. 4-1. (a) Magnitude response of $\beta(z)$. (b) Phase response of $\beta(z)$. (c) Magnitude response of $\beta(z^2)$. (d) Phase response of $\beta(z^2)$.

We up-sample $\beta(z)$ by 2 and get $\beta(z^2)$. Fig. 4-1 (c) and (d) show the magnitude and the phase responses of $\beta(z^2)$. In Fig. 4-1 (d), $\beta(z^2)$ contains a phase discontinuity of π at frequency 0.5π. Because of this phase discontinuity, the left-side and the right-side amplitudes in Fig. 4-1 (c) have different signs.

We then map $\beta(z^2)$ to a 2-D filter [61]. From $\beta(z^2)$, we derive the quadrant filters and rotate them by 45 degrees to construct the hourglass filters [18]. In Fig. 4-2, the symbol z_h denotes the horizontal frequency, and z_v denotes the vertical one. In Fig. 4-2(a), we shift $\beta(z^2)$ by 0.5π along the negative frequency axis and the shifted $\beta(z^2)$ in horizontal direction is denoted by $\alpha(z_h^2)$. Similarly, the shifted $\beta(z^2)$ in vertical direction is denoted by $\alpha(z_v^2)$ in Fig. 4-2(b). In Fig. 4-2(c), we multiply $\alpha(z_h^2)$ and $\alpha(z_v^2)$ together to obtain a quadrant filter $\alpha(z_h, z_v)$. Accordingly, the four acquired quadrant filters are defined by (4-3), (4-4), (4-5), and (4-6 [18]. We rotate these quadrant filters by (4-7) to obtain the hourglass filters. In (4-7), an hourglass filter *A'*(ω) is obtained from a quadrant filter *A*(ω) [14], wherein Q_0 and Q_1 are the quincunx sampling matrices specified by (4-8) [58].

$$H_0(z_h, z_v) = (1 + \alpha(z_h, z_v)) / \sqrt{2} \tag{4-3}$$

$$H_1(z_h, z_v) = z_0(\sqrt{2} - (\sqrt{2}H_0(z_h, z_v) - 1)H_0(z_h, z_v)) \tag{4-4}$$

$$F_0(z_h, z_v) = -z_h^{-1} H_1(-z_h, z_v) \tag{4-5}$$

$$F_1(z_h, z_v) = z_h^{-1} H_0(-z_h, z_v) \tag{4-6}$$

$$A'(\omega) = A(Q_0^{-T}\omega) = A(\frac{1}{2}Q_1^{T}\omega) = A(\frac{1}{2}Q_0\omega) \tag{4-7}$$

$$Q_0 = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \quad Q_1 = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \tag{4-8}$$

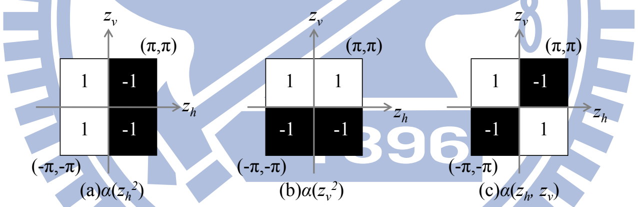


Fig. 4-2. Derivation of quadrant filters.

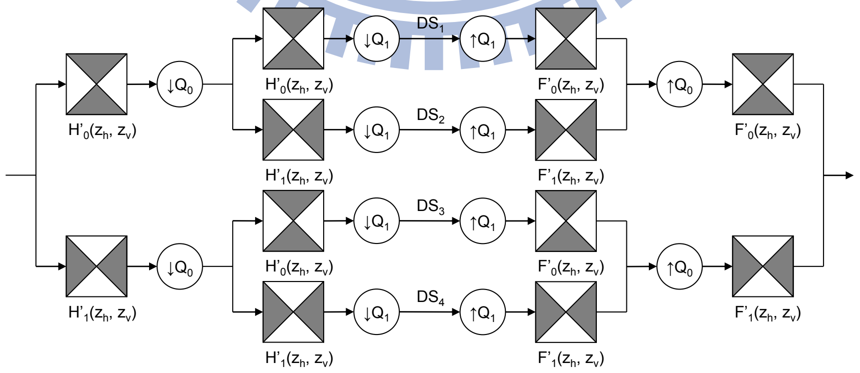


Fig. 4-3. A four-channel cascaded DFB.

Fig. 4-3 shows a cascaded DFB structure [14]. The left half, $H'_0(z_h, z_v)$ and $H'_1(z_h, z_v)$, is the analysis filters, and the right half, $F'_0(z_h, z_v)$ and $F'_1(z_h, z_v)$, is the corresponding synthesis filters. The signals DS$_1$~DS$_4$ are identical to those in Fig. 2-5(a) and their frequency partitions are in Fig. 2-5(b). This two-level analysis DFB structure consists of hourglass filters and quincunx sampling matrices. We rotate the quadrant filter $H_0(z_h, z_v)$ in (4-3) to obtain the hourglass filter $H'_0(z_h, z_v)$. $H'_1(z_h, z_v)$, $F'_o(z_h, z_v)$ and $F'_1(z_h, z_v)$ are designed similarly.

The sizes of our proposed 2-D hourglass short-length filters (SLF) are 7×7 and 13×13. They are much smaller than the sizes (23×23 and 45×45) of their corresponding long-length filters (LLF) [59]. Fig. 4-4 shows the magnitude responses of SLF and LLF. Although the transition band of SLF seems wider than that of the LLF, SLF matches the image local variation well due to its small size.

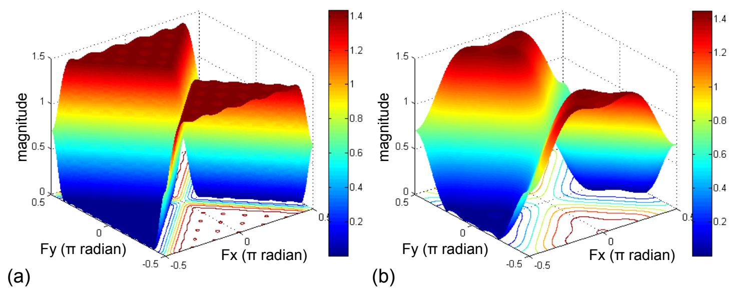


Fig. 4-4. (a) LLF, whose size=23×23 [59]. (b) SLF, whose size=7×7.

Table 4-1 shows the impacts of SLF and LLF on the DFB computational complexities. We compare two DFB implementations, direct structure and ladder structure, on the non-zero SLF/LLF coefficients. S is the size of input image. The numbers of multiplications and additions are proportional to S. The runtime is measured by running Matlab r2008b on a PC with Intel Core 2 Quad Q9400 CPU.

The numbers of multiplications and additions include both convolution and down-sampling operations. When the sizes of the hourglass filters are 23×23, 45×45, 7×7 and 13×13, the numbers of nonzero coefficients are 145, 649, 17 and 65, respectively. For both the direct and the ladder structures, the SLF-based DFB takes approximately only 10% multiplications and additions of those of the LLF-based DFB. In the runtime profile, the SLF-based DFB saves roughly 80% computation time in both structures. The performance gap between our theoretical estimates (multiplications and additions) and experimental measurements (runtime) are largely due to data transfer (disk access).

Table 4-1. The computational complexity and run time measured on the non-zero filter coefficients.

	LLF		SLF	
	Direct structure	Ladder structure	Direct structure	Ladder structure
Number of Multiplications	4S(145+649+2) =3124S	4S(144+2) =584S	4S(17+65+2) =336S	4S(16+2) =68S
Number of Additions	4S(145+649+2) =3124S	4S(144+2) =584S	4S(17+65+1) =336S	4S(16+2) =68S
S=512×512	43.656 sec	14.938 sec	9.078 sec	3.953 sec
S=256×256	10.906 sec	3.813 sec	1.797 sec	0.854 sec
S=128×128	2.859 sec	0.953 sec	0.438 sec	0.219 sec

4-2 Mean-Shift-Based Decision on Subband Selection

In the WBCT image coding scheme, we apply the directional transform to the LH^1, HL^1, and HH^1 subbands. Yet, only the subband signal with significant energy in that direction would benefit from the directional transform. We thus try to locate the subbands with this property. Essentially, we identify the energy peaks and find their locations.

Mean shift technique is adopted to locate the energy peaks in the frequency spectrum. Mean shift is an iterative, nonparametric estimator of the peak location [62][63]; it finds a path to local maximum [64]. Let $\{x_i\}_{i=1\ldots n}$ be an arbitrary n-point data set in the d-dimensional Euclidean space R^d. First, we calculate the mean shift vector $m(x)$ by (4-9), wherein x is the center of current window, h is the window radius, and $K(x)$ is the flat kernel defined by (4-10). Then, we update the center by setting $m(x)+x$ as the center of the next window. We repeat this process until $m(x)$ converges to 0.

$$m(x) = \frac{\sum_{i=1}^{n} x_i K(\frac{x-x_i}{h})}{\sum_{i=1}^{n} K(\frac{x-x_i}{h})} - x \tag{4-9}$$

$$K(x) = \begin{cases} 1, if \ \|x\| \le 1 \\ 0, if \ \|x\| > 1 \end{cases} \tag{4-10}$$

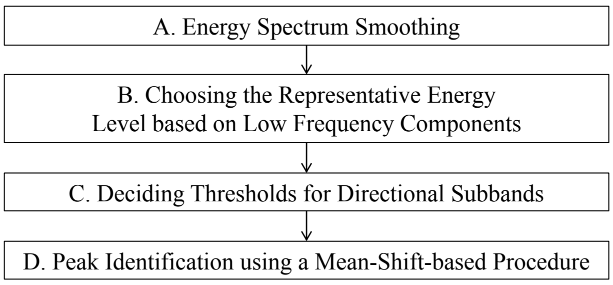

A. Energy Spectrum Smoothing
B. Choosing the Representative Energy Level based on Low Frequency Components
C. Deciding Thresholds for Directional Subbands
D. Peak Identification using a Mean-Shift-based Procedure
Fig. 4-5. The flowchart of the proposed mean-shift-based decision algorithm.

Fig. 4-5 shows our proposed mean-shift-based decision process for selecting the subbands. To illustrate the decision flow, we use a 512×512 pixel, 256 gray-level image as the input.

4.2.1 Energy Spectrum Smoothing

We calculate the input image frequency spectrum by the *2-D discrete Fourier transform* (2-D DFT). The frequency spectrum comprises 512×512 *discrete frequency components* (DFC). The DFC is generally a complex number with the form in (4-11) and their energy levels are in form of (4-12). Herein, (x, y) represents the coordinate pair of a DFC, $1 \leq x \leq 512$, and $1 \leq y \leq 512$.

$$m(x, y)=a(x, y)+b(x, y)i \qquad (4\text{-}11)$$

$$c(x, y)=(a(x, y))^2+(b(x, y))^2 \qquad (4\text{-}12)$$

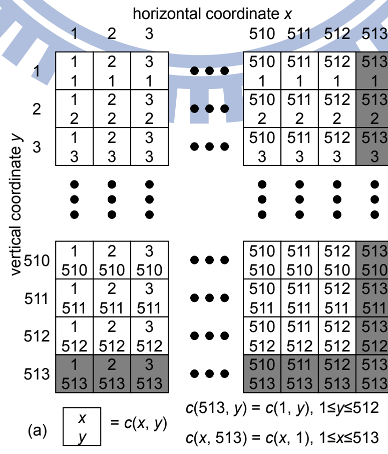
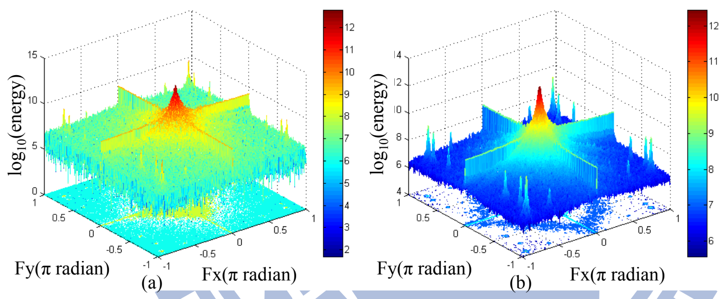


Fig. 4-6. The coordinates of energy coefficients $c(x, y)$. The padded data are in gray background.

In Fig. 4-6, we copy the left-most column to the right-most column border and copy the upmost row to the bottom-most row border in order to get a symmetric energy spectrum. The zero frequency DFC is at (257, 257).

Fig. 4-7(a) shows the energy spectrum $c(x, y)$ of the input image Pepper, wherein the energy levels are in $\log_{10}$ scale, i.e., $\log_{10}(c(x, y))$. It contains many small peaks. These small peaks may cause misjudgment on cluster identification. Therefore, we use a smoothing operator (defined in Fig. 4-7(c)) to reduce small peaks [41]. Fig. 4-7(b) shows the smoothed energy spectrum. The large energy peaks typically stand out after smoothing.



1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

(c)

Fig. 4-7. (a) Energy spectrum of image Pepper. (b) Smoothed energy spectrum of image Pepper. (c) Smoothing operator.

4.2.2 Choosing the Representative Energy Level based on the Low Frequency Components

Fig. 4-7(b) shows natural images contain strong low frequency components. We choose it as the basis for calculating the threshold value for identifying energy peaks. Fig. 4-8(a) shows the subband signals generated by WBCT and Fig. 4-8(b) shows the

DFC coordinates in the upper half subband LH 4-0. The gray area is called the *low frequency zone*, and the white area is the *high frequency zone*. Because the upper half subband is symmetric to its lower half, we only look at the DFC in the upper half of LH[1]. The upper half of LH[1] is the region bounded by $129 \le x \le 385$ and $1 \le y \le 129$. Along each column x of LH[1], we calculate the mean $\mu(x)$ and the variance $\sigma(x)$ of the DFC by (4-13) and (4-14). We find that the DFC magnitudes in the center three columns ($256 \le x \le 258$) usually have large means and small variances. Similar property holds for HL[1]. Therefore, we set the width of low frequency zone in LH[1] and HL[1] to 3 when the input image size is 512×512.

$$\mu(x) = \sum_{y=1}^{129} \frac{\log_{10} c(x,y)}{129}, 129 \le x \le 385 \tag{4-13}$$

$$\sigma(x) = \sqrt{\sum_{y=1}^{129} \frac{(\log_{10} c(x,y))^2}{129} - \left(\sum_{y=1}^{129} \frac{\log_{10} c(x,y)}{129}\right)^2}, 129 \le x \le 385 \tag{4-14}$$

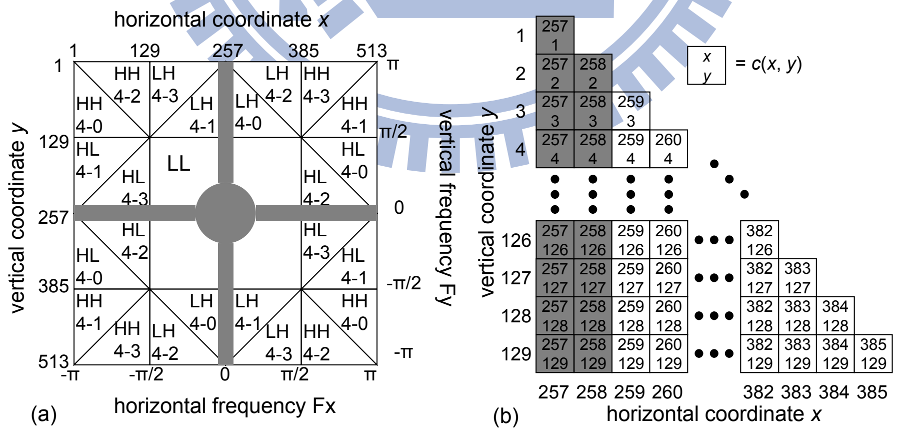


Fig. 4-8. (a) The subband frequency domain partition produced by WBCT. (b) The DFC coordinates in the upper half subband LH 4-0. The gray area in (a) and (b) is the low frequency zone.

To detect the peaks, we calculate the *representative* energy levels of the low frequency components. Eq. (4-15) computes the DFC mean of the LH[1] low frequency zone, and (4-16) computes that of the HL^1 low frequency zone. With these DFC means, we define the representative energy level LH_L for LH^1 by (4-17), and HL_L for HL^1 by (4-18). Essentially, we like to select a threshold that identifies the peaks with "significant" energy. In (4-17), when the average energy level of low frequency components in HL subband is at least four times larger than that in the LH subband, we use the former as the threshold; otherwise, the latter. The parameter "$\log_{10}(4)$" denotes the case that the large energy is at least 4 times of small ones. Correspondingly, the absolute magnitude of the large energy is at least twice of that of the small energy because the energy is the square of the absolute value. In this case, the difference in bit plan coding is significant.

$$\mathrm{LH}\_\mu = \frac{\sum_{y=1}^{129}\sum_{x=256}^{258}\log_{10} c(x,y)}{3\times 129} \tag{4-15}$$

$$\mathrm{HL}\_\mu = \frac{\sum_{y=256}^{258}\sum_{x=1}^{129}\log_{10} c(x,y)}{3\times 129} \tag{4-16}$$

$$\text{if}((\mathrm{HL}\_\mu - \mathrm{LH}\_\mu) < \log_{10}(4))\ \mathrm{LH}\_\mathrm{L} = \mathrm{LH}\_\mu,\ \text{else}\ \mathrm{LH}\_\mathrm{L} = \mathrm{HL}\_\mu \tag{4-17}$$

$$\text{if}((\mathrm{LH}\_\mu - \mathrm{HL}\_\mu) < \log_{10}(4))\ \mathrm{HL}\_\mathrm{L} = \mathrm{HL}\_\mu,\ \text{else}\ \mathrm{HL}\_\mathrm{L} = \mathrm{LH}\_\mu \tag{4-18}$$

4.2.3 Deciding Thresholds for Directional Subbands

A directional subband sometimes contains stronger energy level than the low frequency components. We consider this situation and adjust threshold in this step. We try to determine a peak detection threshold for every WBCT subband. Take the subband LH 4-0 as an example. We only look at the upper half of LH 4-0 because the DFCs in the upper half of LH 4-0 are symmetric to those in the lower half of LH 4-0. In the upper half of LH 4-0, we first consider only the DFC outside the low frequency zone. We calculate the mean LH_4-0_μ and the variance LH_4-0_σ outside the low frequency zone in LH 4-0, i.e., the $c(x, y)$ of white area in Fig. 4-8(b), and construct a Gaussian distribution using the calculated mean and variance. In Fig. 4-9, each Gaussian distribution approximates its corresponding energy histogram well. Thus, the peak detection threshold for LH 4-0 is set by (4-19). The parameter b in (4-19) is chosen to be 0.7 because we like to eliminate the 75% DFC candidates. Together with the representative energy level LH_L defined earlier, 25% or fewer DFC candidates may be identified as energy peaks. We repeat similar procedures on LH 4-1~LH 4-3, and HL 4-0~HL 4-3.

Generally, the transmission priority of HH^1 is lower than the other subbands due to its lower information contents. Because of its low energy, we use the thresholds of its neighboring subbands to identify the energy peaks in HH^1. For example, we set the threshold HH_4-0_T of HH 4-0 by (4-20) using the parameters of HL 4-1.

$$\text{LH\_4-0\_T} = \text{Max}(\text{LH\_L}, \text{LH\_4-0\_}\mu + b \times \text{LH\_4-0\_}\sigma) \tag{4-19}$$

$$\text{HH\_4-0\_T} = \text{Max}(\text{HL\_L}, \text{HL\_4-1\_}\mu + b \times \text{HL\_4-1\_}\sigma) \tag{4-20}$$

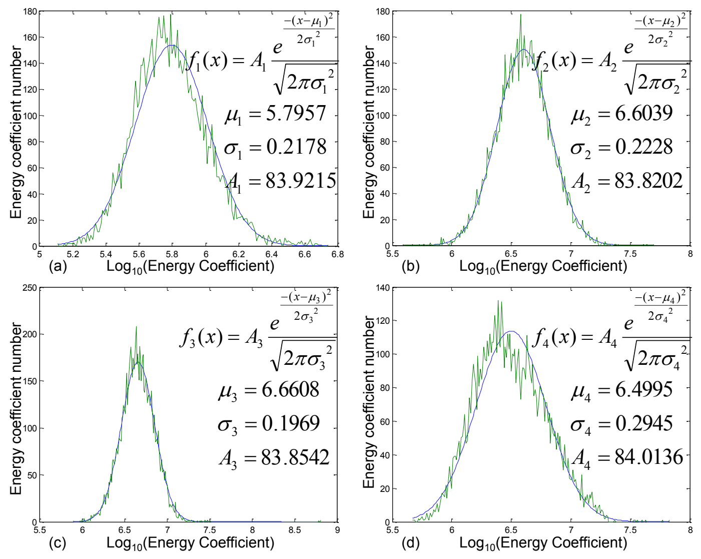


Fig. 4-9. The DFC energy histograms of some directional subbands. Each histogram is approximated by a Gaussian distribution. The directional subbands and the corresponding images are (a) LH 4-0 of Boat, (b) HL 4-3 of Lena, (c) LH 4-3 of Pepper, and (d) HL 4-0 of Fingerprint.

4.2.4 Peak Identification using a Mean-Shift-based Procedure

We like to identify a directional band that has significant energy by examining the discrete frequency components (DFC) of an image. This typically is caused by periodic texture patterns. And its corresponding DFC pattern is a cluster of DFCs with high energy. Thus, an energy peak in this paper is defined as a cluster of coefficients ($c(x, y)$ in a neighborhood) whose energy level is larger than the threshold. It has two properties: the energy level is high and these high-energy DFC coefficients are clustered in a small neighborhood. We use an image cluster identification scheme, Mean-Shift technique, to allocate them.

1) When a $c(x, y)$ within a directional subband and outside the low frequency zone is greater than the threshold of that subband, its location (x, y) is set to be the center of a search window. We then calculate its mass center coordinates $(x_{\mathrm{mass}}, y_{\mathrm{mass}})$ by (4-21). The window size is chosen to be 11×11, or, roughly, its radius r=5, because a small radius often leads to too many small peaks and a large radius sometimes misses peaks. In the search procedure, we extend the coefficients outside the subband boundary by periodic extension

$$(x_{mass}, y_{mass}) = \left(\frac{\sum_{m=x-5}^{x+5}\sum_{n=y-5}^{y+5} m\cdot c(m,n)}{\sum_{m=x-5}^{x+5}\sum_{n=y-5}^{y+5} c(m,n)}, \frac{\sum_{m=x-5}^{x+5}\sum_{n=y-5}^{y+5} n\cdot c(m,n)}{\sum_{m=x-5}^{x+5}\sum_{n=y-5}^{y+5} c(m,n)}\right) \tag{4-21}$$

We round x_{mass} and y_{mass} to the nearest integers and set the rounded (x_{mass}, y_{mass}) as the center of next search window. Then, we use (4-21) again to update the mass center. We repeat this procedure until the rounded (x_{mass}, y_{mass}) converges. Thus, a *peak candidate* is identified.

2) The number of the peak candidates is recorded by a table $d(x, y)$. The initial values of all entries of $d(x, y)$ are 0. When we identify a DFC at (x, y) as an energy peak candidate, we increase $d(x, y)$ by 1. When the table value of a specific location (x, y) is greater than 10 and it is also the largest $d(x, y)$ within a 3×3 window, the DFC located at (x, y) is judged as an *energy peak*. When one subband contains one or more energy peaks in the high frequency zone, it is considered to be suitable for directional

decomposition.

Table 4-2. Some test images, their max energy peak location in each subband ((*x*, *y*)) and the decision result for each subband (suitable for DT).

wavelet subband / test image	LH^1		HL^1		HH^1	
	(x, y)	Suitable for DT	(x, y)	Suitable for DT	(x, y)	Suitable for DT
Barbara	(213,126)	N	(130,366)	Y	(92,384)	N
Fingerprint	(257,129)	N	(128,200)	Y		N
Lena	(154,121)	N	(122,259)	N	(129,390)	N
Pepper		N		N	(32,24)	Y
Boat		N	(118,259)	Y		N
Couple	(236,96)	N	(109,259)	Y		N
Elaine	(177,123)	Y	(107,212)	Y	(83,32)	Y

Table 4-2 shows some representative test images and their band-decomposition decision results for each subband. All images are images of 256 gray levels, and their sizes are 512×512 pixels. For each subband, the "(*x, y*)" column denotes the max energy peak location, and the "suitable for DT" column denotes the decision result. As Table 4-2 shows, the directional transform is inadequate for all subbands of the test image *Lena*; some subbands of *Barbara*, *Fingerprint*, *Pepper*, *Boat*, and *Couple* are suitable for directional transform, and all subbands of *Elaine* benefit from the directional transform. Fig. 4-10 shows the identified peaks by red dots. We fail to identify some peaks for two reasons. First, some peaks contain energy lower than the threshold. Second, when a peak is near the low frequency zone, clusters identified by the mean Mean-Shift scheme are occasionally in the low frequency zone. Fig. 6-4(a) shows a portion of test images *Barbara* and *Elaine*. They contain periodic signals. Identifying these signals in the spatial domain is hard. These periodic signals are corresponding to energy peaks in the frequency domain and thus we perform peak identification in frequency domain.

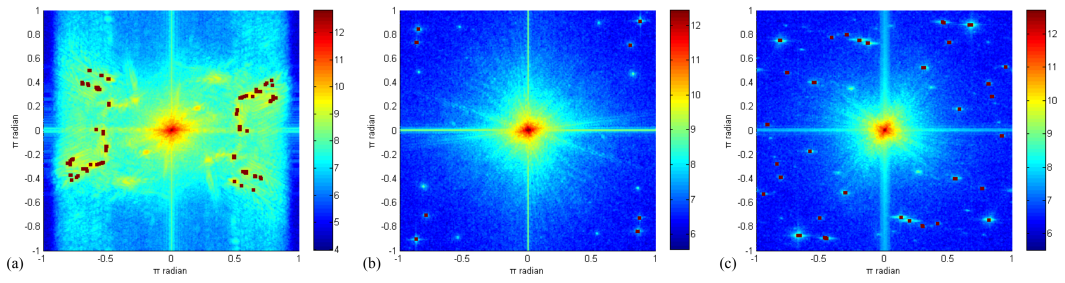


Fig. 4-10. Energy spectrum of test images (a) *Barbara*, (b) *Pepper*, and (c) *Elaine*. Horizontal axis and vertical axis represent horizontal frequency and vertical frequency, respectively. The energy spectrums are all in $\log_{10}$ scale. The red squares are the locations of the identified energy peaks.

4.2.5 Computational Complexity

We now look at the computational complexity issue of our decision algorithm. We examine the amount of multiplications and additions for the steps in Fig. 4-5. We assume that the input image size is $S=W\times H$. Herein, W is the width of the input image and H is the height. We also assume that W and H are all power of 2 and we can implement the 2-D DFT in the radix-2 fast Fourier transform (FFT) structure.

1) In Step A of Fig. 4-5, we apply 2-D DFT to the input image, obtain its energy spectrum, and then apply a smoothing filter to the spectrum. The 2-D DFT is implemented by the radix-2 FFT, and thus the required numbers of real-value additions and multiplications are given by (4-22) and (4-23), in which ceil(x) means the smallest integer greater than or equal to x. Next, Eq. (4-12) needs 1 real addition and 2 real multiplications to calculate the energy of a DFC. For the entire image, the

required numbers of real additions and real multiplications are in (4-24) and (4-25). The smoothing operator in Fig. 4-7(c) requires 8 real additions and 1 real multiplication for each $c(x, y)$. Thus, the total numbers of real additions and multiplications are given by (4-26) and (4-27). Finally, the overall numbers of real additions and real multiplications in Step A are (4-28) and (4-29).

$$\begin{aligned} &N_{real\_addition\_in\_DFT} \\ &= 2 \times N_{complex\_multiplication\_in\_DFT} + 2 \times N_{complex\_addition\_in\_DFT} \\ &= 3 \times W \times H \times (\text{ceil}(\log_2 W) + \text{ceil}(\log_2 H)) \end{aligned} \tag{4-22}$$

$$\begin{aligned} &N_{real\_multiplication\_in\_DFT} \\ &= 4 \times N_{complex\_multiplication\_in\_DFT} \\ &= 2 \times W \times H \times (\text{ceil}(\log_2 W) + \text{ceil}(\log_2 H)) \end{aligned} \tag{4-23}$$

$$N_{real\_addition\_in\_calculating\_power} = W \times H \tag{4-24}$$

$$N_{real\_multiplication\_in\_calculating\_power} = 2 \times W \times H \tag{4-25}$$

$$N_{real\_addition\_in\_smoothing\_spectrum} = 8 \times W \times H \tag{4-26}$$

$$N_{real\_multiplication\_in\_smoothing\_spectrum} = 2 \times W \times H \tag{4-27}$$

$$N_{real\_addition\_in\_stepA} = 3 \times W \times H \times (\text{ceil}(\log_2 W) + \text{ceil}(\log_2 H)) + W \times H + 8 \times W \times H \tag{4-28}$$

$$N_{real\_multiplication\_in\_stepA} = 2 \times W \times H \times (\text{ceil}(\log_2 W) + \text{ceil}(\log_2 H)) + 2 \times W \times H + W \times H \tag{4-29}$$

2) Step B chooses the representative energy levels based on the low frequency zone. Eqs. (4-15) and (4-16) calculate the mean of the DFC energy in the low frequency zone. The heights of the low frequency zones in LH^1 and HL^1 are (ceil(H/4)+1) and (ceil(W/4)+1), respectively. The width is W_{lfz}. Thus, the mean calculation (Step B) needs 2 divisions and $N_{real\_addition\_in\_stepB}$ real additions as shown

in (4-30). We choose W_{lfz}=3 when S=512×512.

$$N_{real\_addition\_in\_stepB} = W_{lfz} \times \left(\text{ceil}\left(H/4\right) + \text{ceil}\left(W/4\right) + 2\right) - 2 \tag{4-30}$$

3) Step C decides the thresholds for directional subbands. The DFC number in each directional subband is $W \times H/16$, thus the DFC number in each half directional subband is $W \times H/32$. In addition to 2 real divisions, we need W×H/32 real multiplications and (W×H/16-2) real additions to calculate the mean and the variance of each half directional subband. LH^1 and HL^1 together have 8 directional subbands in total. The numbers of real additions and real multiplications in Step C are, therefore, given by (4-31) and (4-32).

$$N_{real\_addition\_in\_stepC} = 8 \times (W \times H/16 - 2) = W \times H/2 - 16 \tag{4-31}$$

$$N_{real\_multiplication\_in\_stepC} = 8 \times W \times H/32 = W \times H/4 \tag{4-32}$$

4) Step D identifies the energy peaks. Eq. (4-21) needs 1 division, 242 multiplications and 480 additions. In total, the numbers of real additions and real multiplications in Step D are in (4-33) and (4-34), wherein N_{it} is the iteration number. In our experiments, the minimal N_{it} is 11 (test image *Baboon*), the maximal N_{it} is 12487 (test image *Barbara*), and the average N_{it} is 1697.

$$N_{real\_addition\_in\_stepD} = N_{it} \times 480 \tag{4-33}$$

$$N_{real\_multiplication\_in\_stepD} = N_{it} \times 242 \tag{4-34}$$

All in all, (4-35) and (4-36) give the total number of multiplications and additions in the decision procedure. When S=W×H=512×512, W_{lfz}=3, N_{in}=1697, the total number of real additions and real multiplications are 17,461,460 and 10,699,826.

$$N_{total\_real\_addition} = W\times H\times(3\times \text{ceil}(\log_2 W)+3\times \text{ceil}(\log_2 H)+9+1/2)-16+W_{lfz}\times(\text{ceil}(H/4)+\text{ceil}(W/4)+2)-2+N_{it}\times 480 \tag{4-35}$$

$$N_{total\_real\_multiplication} = W\times H\times(2\times \text{ceil}(\log_2 W)+2\times \text{ceil}(\log_2 H)+3+1/4)+N_{it}\times 242 \tag{4-36}$$

Table 4-3. Computational complexity and run time for the systems with and without decision when LLF is adopted.

	LLF without decision	LLF with decision (fastest)	LLF with decision (slowest)	Ratio (fastest)	Ratio (slowest)
Number of Multiplications	114,819,072	10,699,826	125,518,898	9.32%	109.32%
Number of Additions	114,819,072	17,461,460	132,280,521	15.21%	115.21%
Run Time	11.613 sec	1.385 sec	13.012 sec	11.93%	112.05%

Table 4-4. Computational complexity and run time for the systems with and without decision when SLF is adopted.

	SLF without decision	SLF with decision (fastest)	SLF with decision (slowest)	Ratio (fastest)	Ratio (slowest)
Number of Multiplications	13,369,344	10,699,826	24,069,170	80.03%	180.03%
Number of Additions	13,369,344	17,461,460	30,830,804	130.61%	230.61%
Run Time	2.662 sec	1.385 sec	4.055 sec	52.03%	152.33%

Table 4-3 and Table 4-4 show the computational complexity and the run time of the entire system with and without decision, wherein the directional filters are LLF and SLF, respectively. With decision, the fastest case occurs when no directional transform is conducted on LH^1, HL^1, and HH^1. And the slowest case occurs when we apply the directional transform to all subbands. In Table 4-3, the image coding scheme with LLF and decision may save over 84% computational load or 88% run time in the fastest case. In the slowest case, the decision process requires an additional 16% computational load or 13% run time. In Table 4-4, the image coding scheme with SLF and decision saves about 48% run time in the fastest case and consumes 52% extra run time in the slowest case. On the average, the image coding schemes with decision require less run time.

4-3 New ZC Context Tables for 3-D ESCOT

Arithmetic coding methods encode the transformed/quantized coefficients into a bit-stream. 3-D ESCOT is a bit-plane coding method and it uses its neighbors for the *context model*. Let the sequence $x^N = \{x_N, x_{N-1}, \ldots, x_2, x_1\}$ represents one bit-plane of a coefficient block. Because the bit-plane consists of binary symbols, i.e., $x_i \in \{0,1\}$, the minimum code length of a binary sequence estimated based on the information theory is shown in (4-37), wherein $P(x_i|x^{i-1})$ is the conditional probability of x_i given $x^{i-1} = \{x_{i-1}, x_{i-2}, \ldots, x_2, x_1\}$. Clearly, x^{i-1} is the subset of x^N. Assuming x^N is a Markov random sequence of some finite order, we then can reduce the size of x^{i-1} down to $\underline{x}^{i-1}$, which is a subsequence of x^{i-1}. This $\underline{x}^{i-1}$ is the context model support [36][38]. Typically, $\underline{x}^{i-1}$ includes the neighbors and the (bit-plane) parents of x_i. Ideally, the optimal context model gives the maximum mutual information [68].

$$L = -\log_2 \prod_{i=1}^{n} P(x_i \mid x^{i-1}) \tag{4-37}$$

The original 3-D ESCOT considers only the 2-D DWT coefficients in the horizontal and the vertical directions. Yet, the coefficients in a certain directional subband may cluster along one specific direction (different from the vertical or horizontal directions). The original context table fails to handle this case well. Therefore, we redesign the context models of 3-D ESCOT.

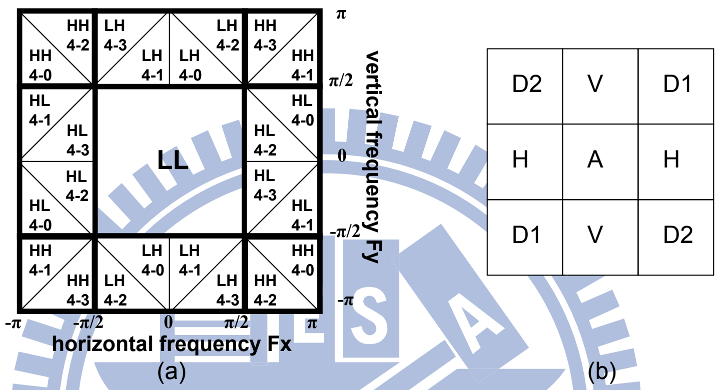


Fig. 4-11. (a) The directional subbands produced by WBCT. (b) The spatial neighbor directions for coefficient *A*.

In Fig. 4-11(a), the 13 subbands produced by WBCT are labeled as “LL”, “HH 4-0”, “LH 4-0”, “HL 4-0”, and likewise. In Fig. 4-11(b), the edges passing through *A* can be H-A-H (0^O), V-A-V (90^O), D1-A-D1 (45^O), and D2-A-D2 (-45^O). We denote the 0^O, 90^O, 45^O, -45^O directions as “H”, “V”, “D1”, and “D2”, respectively.

In Fig. 4-12, we examine the effect of the directional filter LH 4-0 (DF_LH 4-0). A concentric-circle pattern, which has edges of all directions, is used as the input pattern. Fig. 4-12(a) and (b) show this input signal and its frequency spectrum. Fig. 4-12(c) shows the spatial filter impulse response of DF_LH 4-0, which is roughly along the H direction (slightly tilted to the D2 direction). Fig. 4-12(d) shows the filter frequency magnitude response of DF_LH 4-0, whose energy clusters mainly along the vertical axis. In Fig. 4-12(e), the filtered output image contains mainly the spatial edges aligned with the H direction (slightly tilted to the D2 direction). Fig. 4-12(f) shows the frequency spectrum of filtered signals. Evidently, the dominated directions of the LH 4-0 outputs are H and D2. Hence, "H and D2" are the filtered directions of LH 4-0.

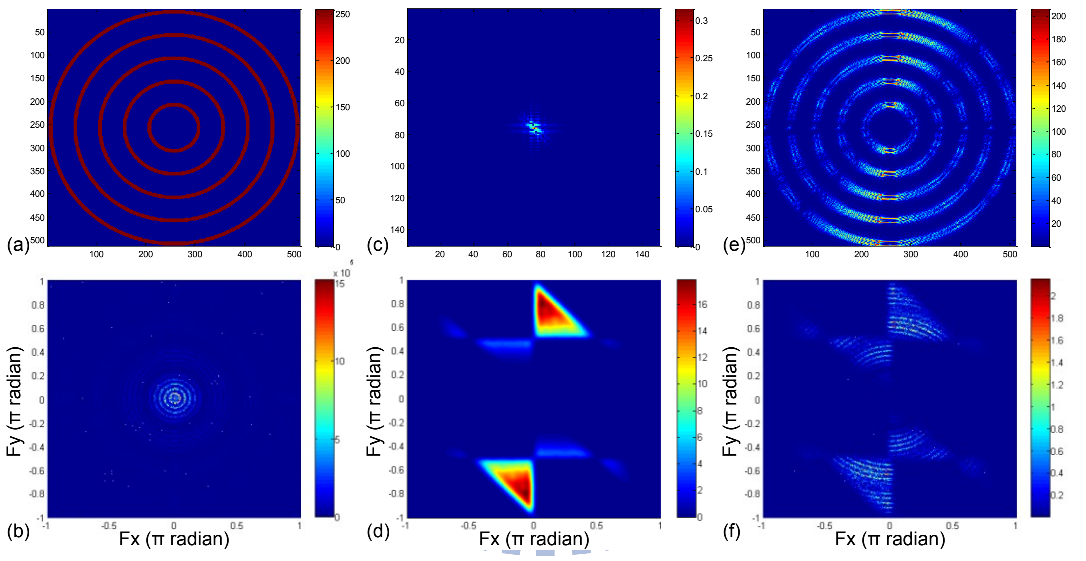


Fig. 4-12. (a) Input signal in spatial domain. (b) Input signal in frequency domain. (c) Filter response of DF_LH 4-0 in spatial domain. (d) Filter response of DF_LH 4-0 in frequency domain. (e) Output signal in spatial domain. (f) Output signal in frequency domain.

Similarly, we identify the filtered directions of the other directional subbands. The filtered directions of LH 4-1 are "H and D1", those of HL 4-2 are "V and D2", and those of HL 4-3 are "V and D1". The filtered directions of the four corner subbands (LH 4-2, HH 4-3, HH 4-1, and HL 4-0) are D2. And those of the other four corner subbands (LH 4-3, HH 4-2, HH 4-0, HL 4-1) are D1.

3-D ESCOT uses three types of context models or *context tables* – the zero coding tables (ZC), the sign coding tables (SC) and the magnitude refinement tables (MR). 3-D ESCOT codes bit-planes from the most significant bit-plane to the least significant bit-plane. 3-D ESCOT starts with ZC to code the beginning zeros until it hits the first non-zero bit. 3-D ESCOT uses ZC to code the magnitude of the first non-zero bit and SC to code its sign. For the remaining bits, 3-D ESCOT uses MR to code their magnitudes. To match the characteristics of the WBCT coefficients, we alter the ZC context table in 3-D ESCOT. For the coefficients in the ordinary 2-D wavelet subbands, we adopt the ZC context table (Table 4-5) in EBCOT [35]. But for the coefficients in the directional subbands, the proposed Table 4-6 is the ZC context table.

In Table 4-5 and Table 4-6, each "context" denotes a model, and the numbers of non-zero coefficients are listed under the directions, H, V, and D1+D2, and X denotes "Don't care". Fig. 4-11(b) shows the neighbors and their notations we use in the entropy coding. The neighbors include vertical neighbors (V), horizontal neighbors (H), left-lower and right-upper neighbors (D1), and left-upper and right-lower neighbors (D2). To code coefficient A in a wavelet subband of a bit-plane, we first calculate the number of non-zero coefficients in all directions. For 2-D wavelets, based on the subband location and the non-zero coefficient patterns, we decide which context in Table 4-5 is to be used to code this bit of coefficient A. Similarly, we code the coefficients in the other directional subbands using Table 4-6.

Table 4-5. ZC context table for 2-D wavelet subbands

wavelet subband	LL LH			HL			HH	
context	H	V	D1+D2	V	H	D1+D2	H+V	D1+D2
8	2	X	X	2	X	X	X	≥3
7	1	≥1	X	1	≥1	X	≥1	2
6	1	0	≥1	1	0	≥1	0	2
5	1	0	0	1	0	0	≥2	1
4	0	2	X	0	2	X	1	1
3	0	1	X	0	1	X	0	1
2	0	0	≥2	0	0	≥2	≥2	0
1	0	0	1	0	0	1	1	0
0	0	0	0	0	0	0	0	0

Table 4-6. ZC context table for directional subbands

Directional subband	LH 4-0			LH 4-1			HL 4-2			HL 4-3			LH 4-3 HL 4-1 HH 4-0 HH4-2			LH 4-2 HL 4-0 HH 4-1 HH 4-3		
context	D2+H	V	D1	D1+H	V	D2	D2+V	H	D1	D1+V	H	D2	D1	H+V	D2	D2	H+V	D1
8	≥2	X	X	≥2	X	X	≥2	X	X	≥2	X	X	2	X	X	2	X	X
7	1	≥1	X	1	≥1	X	1	≥1	X	1	≥1	X	1	≥1	X	1	≥1	X
6	1	0	≥1	1	0	≥1	1	0	≥1	1	0	≥1	1	0	≥1	1	0	≥1
5	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
4	0	2	X	0	2	X	0	2	X	0	2	X	0	≥2	X	0	≥2	X
3	0	1	X	0	1	X	0	1	X	0	1	X	0	1	X	0	1	X
2	0	0	2	0	0	2	0	0	2	0	0	2	0	0	2	0	0	2
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 4-13 shows the frequency responses of the WBCT directional filters. We notice the aliasing phenomenon in WBCT [69]. Because the directional filters are not ideal filters, their outputs contain aliasing components. Thus, the outputs of a certain filter populate not only along one direction but also along another direction (with less energy). Consequently, the context model in arithmetic coding becomes less accurate or its coding efficiency is reduced. We may reduce aliasing by adopting a sharper (and thus longer) filter but the computation time would then increase.

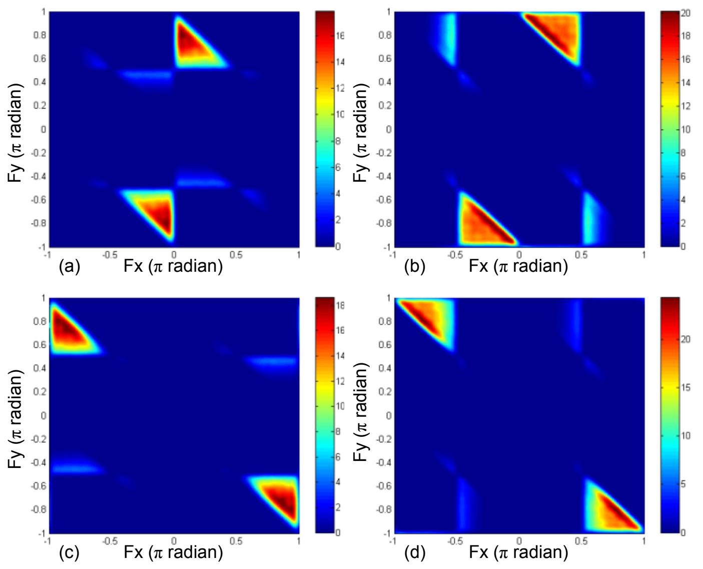


Fig. 4-13. Frequency magnitude responses of (a) LH 4-0 (b) LH 4-2 (c) HH 4-0 (d) HH 4-2.

Chapter 5 Enhanced Direction-Adaptive Wavelet Image and Video Coding

5.1 Direction Alignment Algorithm

A typical 2-D DWT conducts two 1-D DWTs sequentially. For example, it applies a 1-D DWT vertically (along column) to an image and then down-samples the low-pass output and high-pass output vertically to produces the *L* subband (even rows) and *H* subband (odd rows). The image rows are split into two subbands and thus this operation is called *row transform* in [32]. Then, the horizontal 1-D DWT and down-sampling process is applied along the rows of these two subbands. The columns of *L* are spilt into *LL* (even columns) and *LH* (odd columns) subbands and those of *H* are split into *HL* (even columns) and *HH* (odd columns) subbands (column transform). This transform and subsampling order is called *row-column* (RC) subsampling pattern in [32]. We can reverse the order of the above two transform and down-sampling processes and the result is the *column-row* (CR) subsampling pattern. The 2-D DA-DWT also conducts two separable 1-D DA-DWTs using either the RC or CR subsampling pattern. Other subsampling patterns are possible but we consider only these two subsampling patterns in this study. In this section, the direction alignment algorithm designed for 2-D DA-DWT is based on the RC subsampling pattern. It can easily be extended to the CR subsampling pattern in a similar way.

We first apply the vertical 1-D DA-DWT to an image. Nine wavelet candidate directions in [28] are used and they are labeled from -4 to 4 in Fig. 5-1(a). We partition an $F_H \times F_W$ image into non-overlapping $B_H \times B_W$ blocks. We label a block with its coordinates (i, j) as $B(i, j)$, $1 \le i \le (F_H/B_H)$, $1 \le j \le (F_W/B_W)$. Each block $B(i, j)$ is associated with a set of prediction errors $\{D_B(i, j; d_v)\}$; each corresponds to a vertical candidate direction d_v in Fig. 5-1(a) $-4 \le d_v \le 4$. We choose the sum of absolute high-pass coefficients as the prediction error because the absolute value and the squared value result in similar performance for video coding [50]. The DA-DWT selects the best direction based on the minimum $\{D_B(i, j; d_v)\}$ for each $B(i, j)$. It often produces different directions of nearby blocks as shown in Fig. 5-2, which leads to higher side information bits. According to the rate-distortion theory, the optimal direction should be decided based on both bits and distortion. Therefore, we thus try to align the directions of neighboring blocks using the Lagrangian cost function. Fig. 5-3 shows the flow chart of proposed direction alignment algorithm. Step A1 aligns the block directions in similar-texture regions. It scans through the entire image. Steps A2 and A3 align the directions of isolated blocks and small-cluster blocks. They adjust directions of local areas.

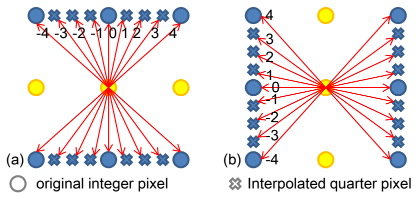


Fig. 5-1. Direction indices: (a) vertical candidate direction d_v and (b) horizontal candidate direction d_h [28].

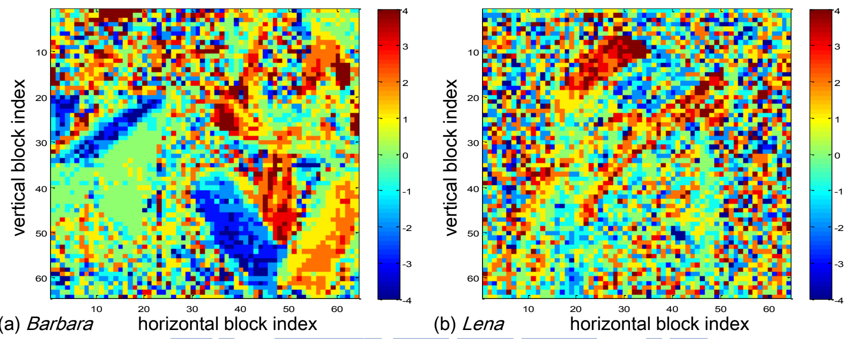


Fig. 5-2. T The best vertical direction d_v of each 8×8 block based on minimal prediction errors. The indices of direction d_v are specified by Fig. 5-1(a). Fig. 6-6 shows the original images of *Barbara* and *Lena*.

$$\begin{tikzcd}
\boxed{\text{Step A1. Aligning Block Directions in Similar-Texture Regions}} \arrow[r] & \boxed{\text{Step A2. Adjusting Directions of Isolated Blocks}} \arrow[r] & \boxed{\text{Step A3. Adjusting Directions of Small-Cluster Blocks}}
\end{tikzcd}$$

Fig. 5-3. Flow chart of proposed direction alignment algorithm.

5.1.1. Step A1: Aligning Block Directions in Similar-Texture Regions

We follow the left-right, top-down scanning order in processing image blocks. We choose the best direction for each block first based on the minimal prediction error. In similar-texture regions, some candidate directions have prediction errors of similar magnitudes. In this case, choosing one direction or the other does not change the distortion much. We like to merge the directions of these blocks into one.

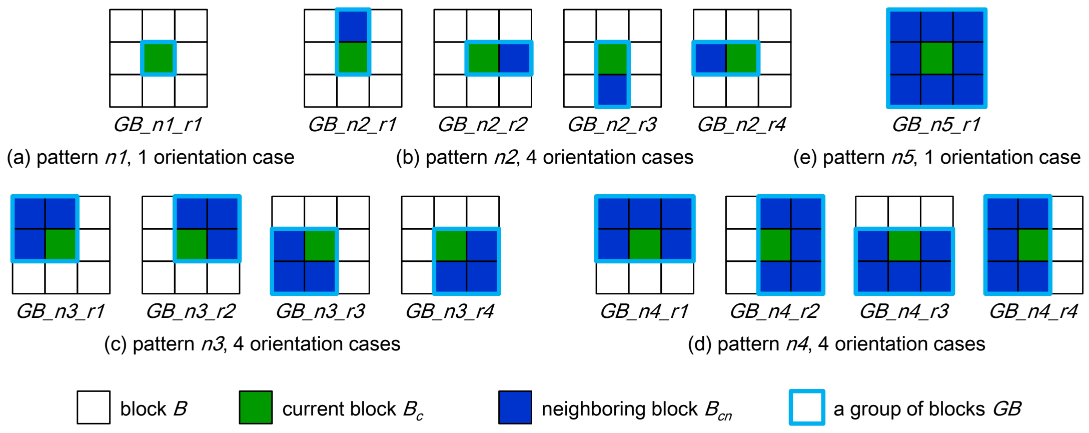


Fig. 5-4. Patterns and orientation cases of *GB*.

We examine the optimal R-D wavelet direction of a *group of blocks* (*GB*) in Fig. 5-4. It consists of 5 patterns: 1-block, 2-block, 4-block, 6-block, and 9-block. Each pattern may contain several *cases* due to different pattern orientations. For example, the 2-block pattern (*n2*) has 4 cases corresponding to the 4 possible orientations shown in Fig. 5-4(b), where the green block is the current block under consideration. We define other patterns and their associated orientation cases similarly. Fig. 5-5 and Fig. 5-6 show the pseudo code of Step A1.

We adopt *GB_n3* in Step A1. *GB_n3*(m, n) is made of 4 $B(i, j)$s: $\{B(i, j), m\text{-}1 \leq i \leq m\text{+}1,\ n\text{-}1 \leq j \leq n\text{+}1,\ B(i, j) \in GB\_n3(m, n)\}$. For a given *GB_n3*($m$, n), a candidate direction d_v produces a *GB distortion* defined by (5-1). This GB distortion, $D_{GB\_n3}(m, n; d_v)$, consists of 4 components – 4 block distortion, $D_B(i, j; d_v)$, using the current candidate direction.

$$D_{GB\_n3}(m, n; d_v) = \sum_{i=m-1}^{m+1} \sum_{j=n-1}^{n+1} D_B(i, j; d_v), \quad B(i, j) \in GB\_n3(m, n) \tag{5-1}$$

The best group direction, $d_{GB\_n3(m,\ n)}$, of a $GB\_n3(m,\ n)$ is chosen based on the minimum prediction error among all candidate directions. However, one block can belong to four $GB_n3(m,\ n)$s depending on the choice of origin. That is, if we slide the $GB\_n3(m,\ n)$ pattern over an image, we get its four position or orientation cases, GB_n3_r1 ~ GB_n3_r4, defined in Fig. 5-4(c). Thus, a block $B(i, j)$ has four possible best GB directions using four orientation cases: GB_n3_r1 ~ GB_n3_r4. We count the occurrence of GB directions of four orientation cases. The maximum value of occurrence number, *moc*, ranges from 1 to 4. Fig. 5-7 shows the *moc* of two test images. The larger *moc* appears in the smooth or the similar-texture regions and the smaller *moc* often locates at region boundaries.

```
moc_table = zeros(($F_H/B_H$), ($F_W/B_W$));
d_offset = 5;    % $d_v$ = -4 ~ 4, thus we set d_offset = 5 to match the array index.

% (Part A) Find the $moc$ of a block $B$ using $GB\_n3$
for $i$ = 1 : ($F_H/B_H$)
  for $j$ = 1 : ($F_W/B_W$)
    d_temp_buffer = zeros(1, 9);

    for $GB$ = $GB\_n3$
      $D_{GB}$ = zeros(1, 9);
      for $d$ = $d_v$
        for $m$ = $i$-1 : $i$+1
          for $n$ = $j$-1 : $j$+1
            if($B(m, n) \in GB$)
              $D_{GB}$($d$ + d_offset) += $D_B(m, n, d)$;
              % $D_B(m, n, d)$ is prediction error of $B(m, n)$ corresponding to direction $d$
            end
          end
        end
      end
      find the best direction $d_{GB}$ of $GB$ based on the minimum candidate in $D_{GB}$;
      d_temp_buffer($d_{GB}$ + d_offset)++;    % accumulate the occurrence of $d_{GB}$
    end

    moc_table($i$, $j$) = the maximum element of d_temp_buffer;
  end
end
```

Fig. 5-5. The pseudo code of Step A1, part A.

```
% (Part B) Find the $d_{B\_A1}$ of a block $B$
for $i$ = 1 : ($F_H$/$B_H$)
  for $j$ = 1 : ($F_W$/$B_W$)
    % set the considered $GB$ based on $moc$
    if(moc_table($i$, $j$) ≤ 2)
      considered_GB = $GB\_n1$ && $GB\_n2$;
    end
    if(moc_table($i$, $j$) == 3)
      considered_GB = $GB\_n1$ && $GB\_n2$ && $GB\_n3$ && $GB\_n4$;
    end
    if(moc_table($i$, $j$) == 4)
      considered_GB = $GB\_n1$ && $GB\_n2$ && $GB\_n3$ && $GB\_n4$ && $GB\_n5$;
    end

    cost_temp_buffer = zeros(1, 9);
    cost_temp_buffer = cost_temp_buffer + 10000000000;
    % We assume 10000000000 is the up limit of the Lagrangian cost.

    % calculate the corresponding Lagrangian cost of each considered $GB$
    for $GB$ = considered_GB
      $D_{GB}$ = zeros(1, 9);
      for $d = d_v$
        for $m = i$-1 : $i$+1
          for $n = j$-1 : $j$+1
            if($B(m, n) \in GB$)
              $D_{GB}(d$ + d_offset) += $D_B(m, n, d)$;
            end
          end
        end
      end

      find the best direction $d_{GB}$ of $GB$ based on the minimum candidate in $D_{GB}$;
      set $B(i, j) = B_c$ of $GB$;   $L_{GB\_A1} = D_{B_c}(d_{GB}) + \lambda_{A1}(R_{A1}/N_{GB})$;
      if(cost_temp_buffer($d_{GB}$ + d_offset) < $L_{GB\_A1}$)
        cost_temp_buffer($d_{GB}$ + d_offset) = $L_{GB\_A1}$;
      end
    end
    find the aligned direction $d_{B\_A1}$ of $B(i, j)$ based on the minimum candidate in   cost_temp_buffer;
  end
end
```

Fig. 5-6. The pseudo code of Step A1, part B.

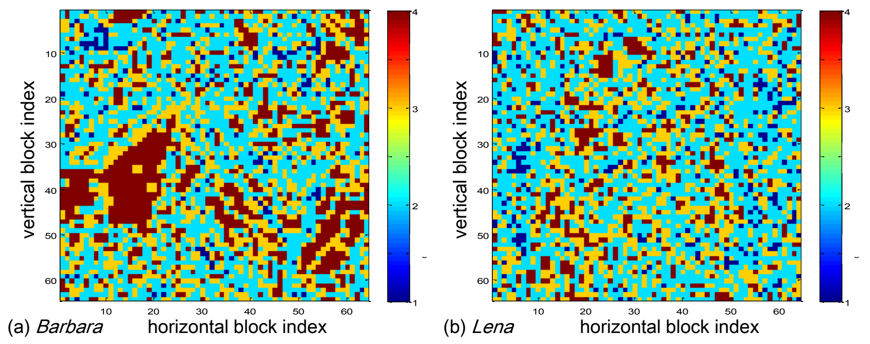


Fig. 5-7. *moc* of each block (8×8 block size).

To simplify the notations, we abbreviate *GB_n3*(m, n) as *GB_n3* when its coordinates are not important. The other symbols are abbreviated in a similar way. The aligned direction d_{B_A1} of a block B is derived by using the following procedure. Assuming B_c is the *current block* (green color) in Fig. 5-4. We calculate the Lagrangian cost function of each orientation case in Fig. 5-4. We take the 4-block pattern case of *GB_n3_r1* (Fig. 5-4(c)) as an example. Its Lagrangian cost function $L_{GB\_n3\_r1\_A1}$ is calculated by (5-2).

$$L_{GB\_n3\_r1\_A1} = D_{B_c}(d_{GB\_n3\_r1}) + \lambda_{A1}(R_{A1} / N_{GB\_n3\_r1}) \qquad (5\text{-}2)$$

where $D_{Bc}(d_{GB\_n3\_r1})$ is the prediction error of B_c using $d_{GB\_n3\_r1}$, the best direction of *GB_n3_r1,* and $N_{GB\_n3\_r1}$ is the total number of blocks within *GB_n3_r1* and thus $N_{GB\_n3\_r1} = 4$. The bit rate is derived under the assumption that 9 candidate directions occur equally likely with the same probability 1/9 and thus R_{A1} is $\log_2(9)$ for all cases. The Lagrangian multiplier λ_{A1} is chosen empirically. Each orientation case in Fig. 5-4 results in one Lagrangian cost function. We select the aligned direction d_{B_A1} for a block B based on the minimum Lagrangian cost function. The adopted *GB* patterns for a block B depends on *moc* of B as follows.

Next, we examine the distribution of directions of all cases for the current block. The maximum occurrence (of a candidate direction) is called *moc* as defined earlier. The *moc* gives the texture orientation information surrounding that block. A larger *moc* implies most blocks in its neighborhood have similar-texture, and in contrast, a smaller *moc* (say, $moc \leq 2$) often indicates it locates at object corners or boundaries. In the case of complex surrounding texture, the smaller patterns (say, the 2-block pattern) show more consistent directions. However, *moc* ties appear often and we use the following rules to make the final decision. If $moc \leq 2$, we choose the direction associated with *GB* pattern *n1* or *n2*, because the current block is likely located in a complex region. When $moc = 3$, the direction comes from *GB* pattern *n1, n2, n3* and *n4*. When $moc = 4$, the direction comes from all *GB* patterns.

Fig. 5-8 shows d_{B_A1} of all blocks after Step A1. The neighboring blocks in similar-texture or smooth regions now have consistent directions. But still there are some isolated blocks, which are expensive in sending side information. Therefore, we design the Step A2 next to reduce isolated blocks.

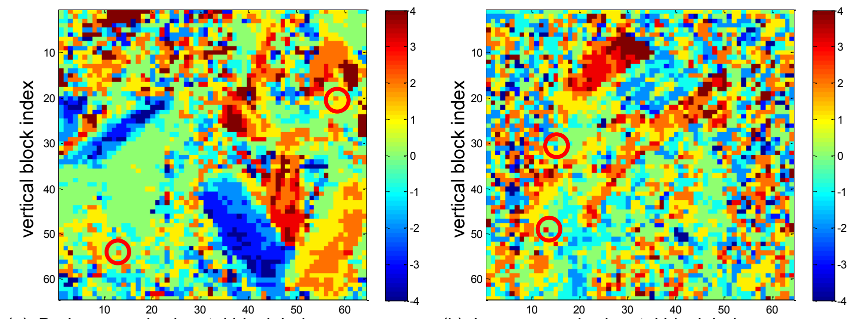


Fig. 5-8. Aligned directions after Step A1 (8×8 block). The circles indicate isolated blocks.

5.1.2. Step A2: Adjusting Directions of Isolated Blocks

Let *B* be an *Isolated Block* (IB) if its direction d_{B_A1} differs from all its 4 neighboring blocks after Step A1. Fig. 5-8 shows some *IB*s in circles. If a block *B* is *IB*, we adjust its direction by Step A2 and the new direction is denoted as d_{IB_A2}. Fig. 5-9 shows the pseudo code of Step A2.

```
d_offset = 4;   IB_table = zeros(($F_H/B_H$), ($F_W/B_W$));

% record the location of $IB$
for $i$ = 1 : ($F_H/B_H$)
  for $j$ = 1 : ($F_W/B_W$)
    if($B(i, j)$ == $IB$)
      IB_table($i, j$) = 1;
    end
  end
end

% Find the $d_{B\_A2}$ of $IB$
while(sum of IB_table ~= 0)
  for $i$ = 1 : ($F_H/B_H$)
    for $j$ = 1 : ($F_W/B_W$)
      if(IB_table($i, j$) == 1)
        cost_temp_buffer = zeros(1, 9);
        cost_temp_buffer = cost_temp_buffer + 10000000000;

        % calculate the corresponding Lagrangian cost of each considered $GB$
        for $GB$ = $GB\_n1$ && $GB\_n2$ && $GB\_n3$ && $GB\_n4$ && $GB\_n5$
          if($B_{cn}$ of $GB$ have identical direction $d_{B_{cn}\in GB}$ && IB_table of $B_{cn}$ == 0)
            %$B_{cn}$ can be IB with $d_{B\_A2}$ but not $d_{B\_A1}$.
            set $B(i, j) = B_c$ of $GB$;   $L_{GB\_A2} = D_{B_c}(d_{B_{cn}\in GB}) + \lambda_{A2}(R_{A2}/N_{GB})$;
            if(cost_temp_buffer($d_{GB}$ + d_offset) < $L_{GB\_A2}$)
              cost_temp_buffer($d_{GB}$ + d_offset) = $L_{GB\_A2}$;
            end
          end
        end

        if(the minimum candidate of cost_temp_buffer < 10000000000)
          find the aligned direction $d_{B\_A2}$ of $B(i, j)$ based on the minimum candidate in cost_temp_buffer;
          IB_table($i, j$) = 0;
        end
      end
    end
  end
end
```

Fig. 5-9. The pseudo code of Step A2.

We set *IB* as B_c in Fig. 5-4 and calculate d_{IB_A2} as follows. We take GB_n3_r1 as an example. The Lagrangian cost function, $L_{GB\_n3\_r1\_A2}$, is defined by (5-3).

$$L_{GB\_n3\_r1\_A2} = D_{B_c}(d_{B_{cn}\in GB\_n3\_r1}) + \lambda_{A2}(R_{A2} / N_{GB\_n3\_r1}) \qquad (5\text{-}3)$$

We consider the case that all *the neighboring blocks*, B_{cn}, within GB_n3_r1 ($B_{cn} \in GB\_n3\_r1$) in Fig. 5-4(c) must have identical directions $d_{Bcn \in GB\_n3\_r1}$. $B_{cn} \in GB\_n3\_r1$ may have inconsistent directions after Step A1. We discard the case with inconsistent B_{cn} directions. An *IB* may still be *IB* after Step A2 and we re-define its direction as d_{IB_A2}. We also adopt these *IB*s as B_{cn} for future processing patterns in Fig. 5-4. If all $B_{cn} \in GB\_n3\_r1$ have consistent direction, we define it as $d_{Bcn \in GB\_n3\_r1}$. Let $D_{Bc}(d_{Bcn \in GB\_n3\_r1})$ be the prediction error of B_c using the direction $d_{Bcn \in GB\_n3\_r1}$. $N_{GB\_n3\_r1} = 4$ as discussed earlier. We now estimate the side information bits R_{A2} based on the known direction information. For the current *GB*, we use the direction index of its neighboring blocks (up, left, left-up) as a direction index predictor. If a block *B* is not an *IB*, we choose its Step A1 aligned direction d_{B_A1} as a predictor. For an *IB*, we choose its Step A2 aligned direction d_{IB_A2} if we already known it. We encode the prediction index differences of the current *GB*, which gives R_{A2} as in [28]. In calculating the Lagrangian cost function in Step A2, λ_{A2} is the Lagrangian multiplier obtained empirically. For an *IB*, we use all *GB* patterns in Fig. 5-4 and calculate the L_{GB_A2} in each orientation case. We select a direction d_{IB_A2} for *IB* based on the minimum L_{GB_A2}. We repeat the above procedure on all *IB*s following the scanning order.

After Step A2, we denote the directions of all blocks as d_{B_A2}. Fig. 5-10 shows the d_{B_A2} of all blocks. Most *IB*s in Fig. 5-8 are eliminated. If an *IB* remains, it must have a large R-D cost reason. Next, we find a few clustered blocks that have different directions from their neighbors. If the cluster size is small, we like to re-examine their R-D cost.

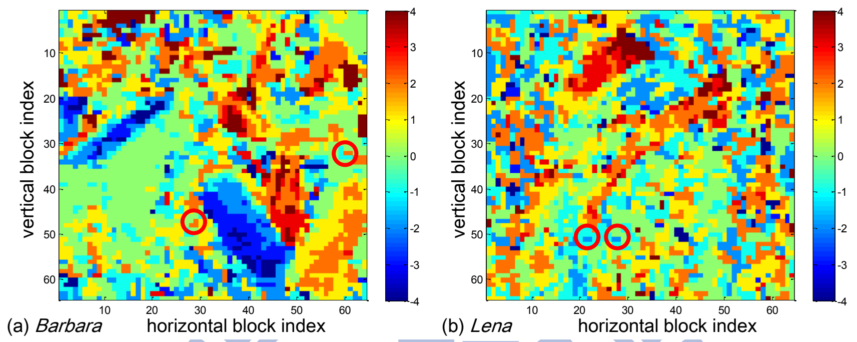


Fig. 5-10. Aligned directions after Step A2 (8×8 block). The circles indicate small-cluster blocks.

5.1.3. Step A3: Adjusting Directions of Small-Cluster Blocks

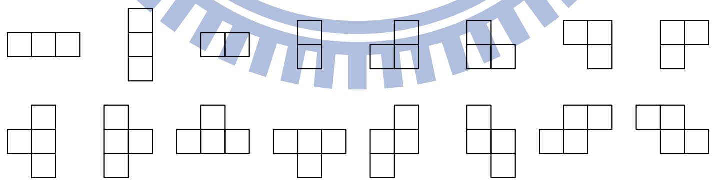


Fig. 5-11. Different types of small-cluster blocks. These blocks cannot be presented by a large square block in quadtree partition.

The shapes of *Small-Cluster Blocks* (*SCB*) are defined by Fig. 5-11. An *SCB* is a small group of blocks with a consistent direction, but their direction differs from those of their surrounding blocks, as shown in Fig. 5-10. We are unable to present these *SCB*s by a larger square block in quadtree partition. If a block *B* belongs to a *SCB*, we adjust its direction to d_{SCB_A3} by Step A3 below. Each *SCB* contains a number of blocks. We process each block within *SCB* from left to right, top to down by Step A3. Fig. 5-12 shows the pseudo code of Step A3.

```
d_offset = 4;    SCB_table = zeros(($F_H/B_H$), ($F_W/B_W$));

% record the location of SCB
for $i$ = 1 : ($F_H/B_H$)
   for $j$ = 1 : ($F_W/B_W$)
      if($B(i, j)$ == $SCB$)
         SCB_table($i, j$) = 1;
      end
   end
end

% Find the $d_{B\_A3}$ of SCB
while(sum of SCB_table ~= 0)
   for $i$ = 1 : ($F_H/B_H$)
      for $j$ = 1 : ($F_W/B_W$)
         if(IB_table($i, j$) == 1)
            cost_temp_buffer = zeros(1, 9);
            cost_temp_buffer = cost_temp_buffer + 10000000000;

            % calculate the corresponding Lagrangian cost of each considered GB
            for $GB$ = $GB\_n1$ && $GB\_n2$ && $GB\_n3$ && $GB\_n4$ && $GB\_n5$
               if($B_{cn}$ of $GB$ have identical direction $d_{B_{cn}\in GB}$ && SCB_table of $B_{cn}$ == 0)
                  %$B_{cn}$ can be SCB with $d_{B\_A3}$ but not $d_{B\_A2}$.
                  set $B(i, j) = B_c$ of $GB$;    $L_{GB\_A3} = \mathrm{D}_{\mathrm{B_c}}(d_{B_{cn}\in GB}) + \lambda_{A3}(R_{A3}/N_{GB})$;
                  if(cost_temp_buffer($d_{GB}$ + d_offset) < $L_{GB\_A3}$)
                     cost_temp_buffer($d_{GB}$ + d_offset) = $L_{GB\_A3}$;
                  end
               end
            end

            if(the minimum candidate of cost_temp_buffer < 10000000000)
               find the aligned direction $d_{B\_A3}$ of $B(i, j)$ based on the minimum candidate in cost_temp_buffer;
               SCB_table($i, j$) = 0;
            end
         end
      end
   end
end
```

Fig. 5-12. The pseudo code of Step A3.

The directions of its surrounding blocks are the candidates, and the direction alignment procedure is similar to that of Step A2. If block B_c belongs to a *SCB*, we calculate its Lagrangian cost function (eqn. (5-4)) of all cases in Fig. 5-4 for all its surrounding directions. Let us take *GB_n3_r1* as an example again. Its Lagrangian cost function $L_{GB\_n3\_r1\_A3}$ is as follows.

$$L_{GB\_n3\_r1\_A3} = D_{B_c}(d_{B_{cn}\in GB\_n3\_r1}) + \lambda_{A3}(R_{A3} / N_{GB\_n3\_r1}) \qquad (5\text{-}4)$$

We calculate the side information R_{A3} in (5-4) in a similar way to R_{A2} in (5-3). Since a *SCB* often locates in the complex texture region, we adopt the value of λ_{A2} for λ_{A3}. Similar to Step A2, $B_{cn} \in GB\_n3\_r1$ must have a consistent direction $d_{Bcn\in GB\_n3\_r1}$. Step A3 may still keep some *SCB*s unchanged. We define directions of these *SCB*s as d_{SCB_A3}. We also adopt the *SCB*s with directions d_{SCB_A3} as B_{cn} for future process. $D_{Bc}(d_{Bcn\in GB\_n3\_r1})$ is the prediction error of B_c corresponding to $d_{Bcn\in GB\_n3\_r1}$. We select a direction d_{SCB_A3} for *SCB* based on the minimum L_{GB_A3}.

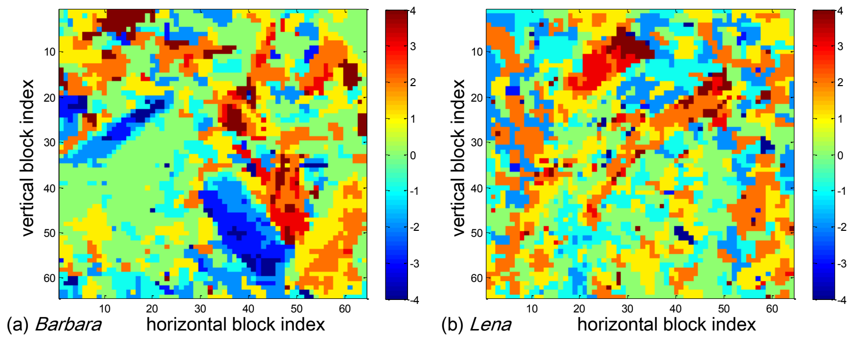


Fig. 5-13. Aligned directions after Step A3 (8×8 block).

Fig. 5-13 shows the results after Step A3. Clearly, compared to Fig. 5-10, adjusting the directions of *SCB* helps in forming larger connected blocks. These large connected blocks reduce the side information and thus improve the coding performance. Step A2 adjusts the directions of *IB*s and Step A3 adjusts those of *SCB*s.

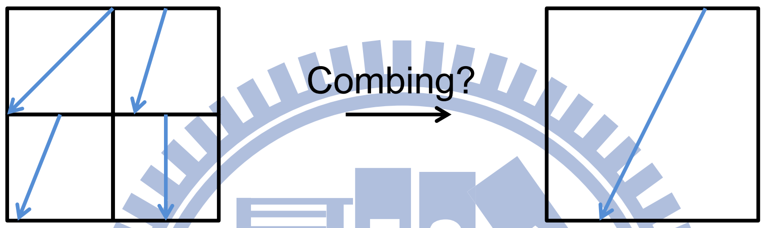
Images with low-resolution may contain complex textures inside a small region. Aligning directions of these regions significantly increases prediction error. Thus, applying Step A2 and Step A3 to low-resolution images is less desirable.

5.1.4. Step A4: Adjusting Directions of the Second 1-D DA-DWT

Similar to the vertical 1-D DWT, the vertical 1-D DA-DWT decomposes an $F_H \times F_W$ image into the spatial low-pass and high-pass subbands with size $(F_H/2) \times F_W$ each. Then, we apply the horizontal 1-D DA-DWT to the spatial low-pass subband and then followed by a direction alignment algorithm similar to that of the vertical 1-D DA-DWT describe in the above. We start with partitioning the spatial low-pass subband into non-overlapping $(B_H/2) \times B_W$ blocks. The nine candidate directions for d_h are defined by Fig. 5-1(b). Then, the 3-step direction alignment algorithm can be applied to the low-pass subband blocks in a similar way. The spatial high-pass subband usually contains little energy. Applying the horizontal 1-D DA-DWT to it is inefficient. Thus, we apply the conventional horizontal 1-D DWT to vertical high-pass subband to save side information [51].

A popular DA-DWT image coding structure adopts the quadtree partition to represent the partition information as illustrated by Fig. 5-14 [27][28][50]. Four individual blocks (block size is $B_H \times B_W$) often provide less prediction error but at the cost of higher side information bits. A larger block (block size is $(2B_H) \times (2B_W)$) needs fewer side information bits but produce large prediction errors. A well-designed DA-DWT uses the Lagrangian cost function (R-D optimization) to find the optimal trade-off between prediction errors and side information bits. For a multi-layer quadtree, we need to calculate the cost functions of all possible partitions and then choose the one with the minimum overall cost. This exhaustive search is generally impractical because the total possible combinations are too huge. Also, the search for the optimal Lagrangian multiplier λ_t is another highly complicated job. Typically, a larger λ_t puts more weights on bits and results in larger block partitions while a smaller λ_t does the opposite. In general, a 2-D DA-DWT scheme [51] may have different block partition for the first (vertical) and the second (horizontal) 1-D DA-DWT. In the experimental section, we adopt some existing quadree partition [52], megablocking partition [15], and direction prediction techniques in coding the side information.

As for the parameter selections, it may worth noting that both the distortion model and bits model (bit estimation) adopted in the previous discussions do not match exactly the coding distortion and the coding bits at the end. The exact models are too complicated in real images even if they exist. Therefore, a lot of the above formulas are approximations and the parameters are tuned empirically. Although the words "best" or "optimal" are sometimes used, they describe the cases under the given assumptions or models. We cannot guarantee that they are the ultimate best or optimal choices for the final coding results. With more accurate distortion and bits model, the coding performance of the proposed scheme may be further improved.



$$\sum_{i=1}^{4} D_{i,B_W\times B_H} + \lambda_t R_{B_W\times B_H} \qquad D_{(2B_W)\times(2B_H)} + \lambda_t R_{(2B_W)\times(2B_H)}$$

Fig. 5-14. Quadtree combination with Lagrangian cost function. λ_t is the Lagrangian multiplier.

5.2 Direction Alignment Algorithm for SA-DWT

The 2-D DWT applies two 1-D DWTs to an image along the vertical direction and then the horizontal direction. The conventional 2-D DA-DWT applies two 1-D DA-DWTs in a similar way. Different orders of these two 1-D DWTs have no effect on the final results of 2-D DWT. But for DA-DWAT, if we reverse the order of two 1-D DA-DWTs, we may obtain different final results [31][56]. Under certain conditions given in [56], the order of applying 1-D DA-DWT is irrelevant; however, those conditions are not satisfied practically. We thus discuss the selection of subsampling patterns.

Three most commonly used subsampling patterns are RC, CR, and quincunx (QU). The QU subsampling pattern is most effective for the strongly anisotropic images. It is reported that for most nature images, its performance is not as good as RC or CR [31]. Therefore, we only consider the RC and CR patterns in this study.

We can examine individual block separately for the RC and CR subsampling patterns and select one with better performance. This is the key idea behind the subsampling and direction-adaptive DWT (SA-DWT) [32]. If two neighboring blocks have different subsampling patterns, it uses a phase-completion process to handle the transform across their boundaries. Fig. 5-15 shows the 4 spatial subbands of different subsampling patterns [32]. In this section, we propose an extended direction alignment algorithm for 2-D SA-DWT.

We again partition an $F_H \times F_W$ image into non-overlapping blocks with $B_H \times B_W$ block size. For the first 1-D SA-DWT, each block $B(i, j)$ now has 18 candidate directions, d_s, including 9 possible d_v and 9 possible d_h in Fig. 5-1. The corresponding prediction errors are denoted as $\{D_B(i, j; d_s)\}$. The SA-DWT selects the best direction based on the minimum $\{D_B(i, j; d_s)\}$ among all candidates. Fig. 5-16 shows the best direction of two test images. Neighboring blocks have inconsistent directions and subsampling patterns. It results in a large amount of side information of directions and subsampling patters. We extend the proposed direction alignment algorithm for the 2-D SA-DWT with some modifications on the original SA-DWT. Fig. 5-17 and Fig. 5-18 show the flow charts of our proposed direction alignment algorithms.

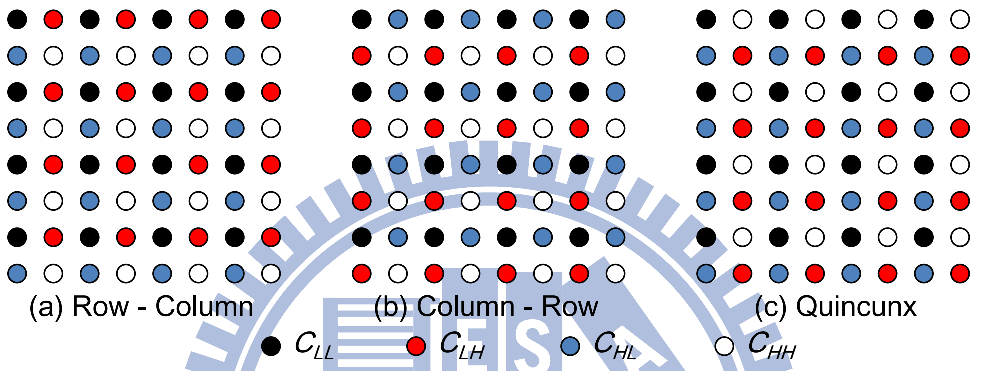


Fig. 5-15. Four spatial subbands of different subsampling patterns [32].

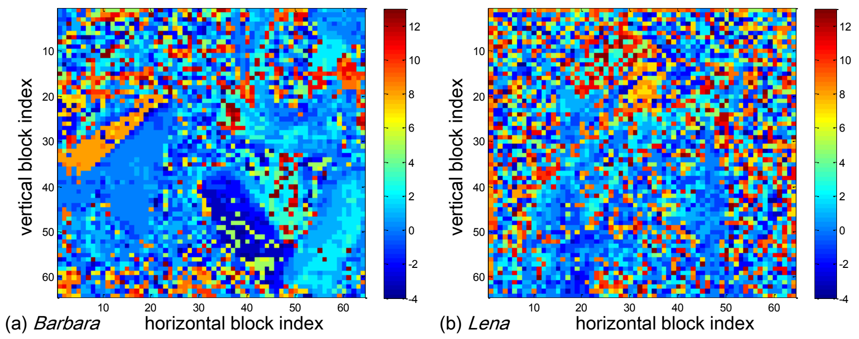


Fig. 5-16. The best direction d_s of each 8×8 block. The direction indexes -4 ~ 4 correspond to d_v (-4 ~ 4) and 5 ~ 13 correspond to d_h (-4 ~ 4).

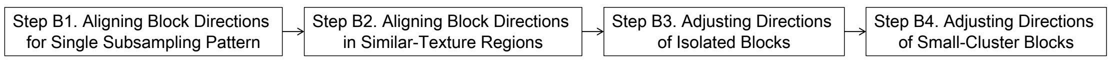

Step B1. Aligning Block Directions for Single Subsampling Pattern
Step B2. Aligning Block Directions in Similar-Texture Regions
Step B3. Adjusting Directions of Isolated Blocks
Step B4. Adjusting Directions of Small-Cluster Blocks

Fig. 5-17. Flow chart of proposed direction alignment algorithm for the first 1-D SA-DWT.

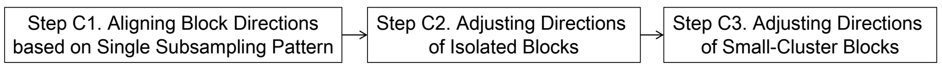

Step C1. Aligning Block Directions based on Single Subsampling Pattern
Step C2. Adjusting Directions of Isolated Blocks
Step C3. Adjusting Directions of Small-Cluster Blocks

Fig. 5-18. Flow chart of proposed direction alignment algorithm for the second 1-D SA-DWT.

To simplify the process and reduce the distortion increased due to subsampling pattern change at block boundaries, we start with only one single subsampling pattern (either RC or CR) applied to the entire image. We propose four steps to align directions of the first 1-D SA-DWT in Fig. 5-17. Step B1 aligns block directions based on a single subsampling pattern. We then choose the row transform or the column transform (and the best direction) for each individual block. Step B2 aligns directions in similar-texture regions. Step B3 and B4 aligns directions of isolated blocks and small-cluster blocks.

Except best directions, the first 1-D SA-DWT also decide subsampling patterns of each block. Thus, after the first 1-D SA-DWT, we execute the second 1-D SA-DWT based on double subsampling patterns, RC and CR, interlacing together. A 3-step procedure similar to Algorithm A aligns the directions of the second 1-D SA-DWT in Fig. 5-18. Step C1 aligns block directions based on a single subsampling pattern. It also calculates the prediction errors for Steps C2 and C3. Steps C2 and C3 align the directions of isolated blocks and small-cluster blocks.

5.2.1. Step B1: Aligning Block Directions for Single Subsampling Pattern

We adopt the direction alignment Algorithm A to align the directions of the first transform based on one single subsampling pattern. We repeat the transform and alignment procedure twice: for RC and for CR. After the procedure is completed, each $B(i, j)$ has two candidate first directions, $d_{B(i,\ j)\_RC\_B1} \in d_v$ and $d_{B(i,\ j)\_CR\_B1} \in d_h$, corresponding to RC and CR. Fig. 5-19 shows separately the aligned first directions for two subsmapling patterns. In fact, Fig. 5-19(a) and Fig. 5-19(c) are identical to Fig. 5-10(a) and Fig. 5-10(b).

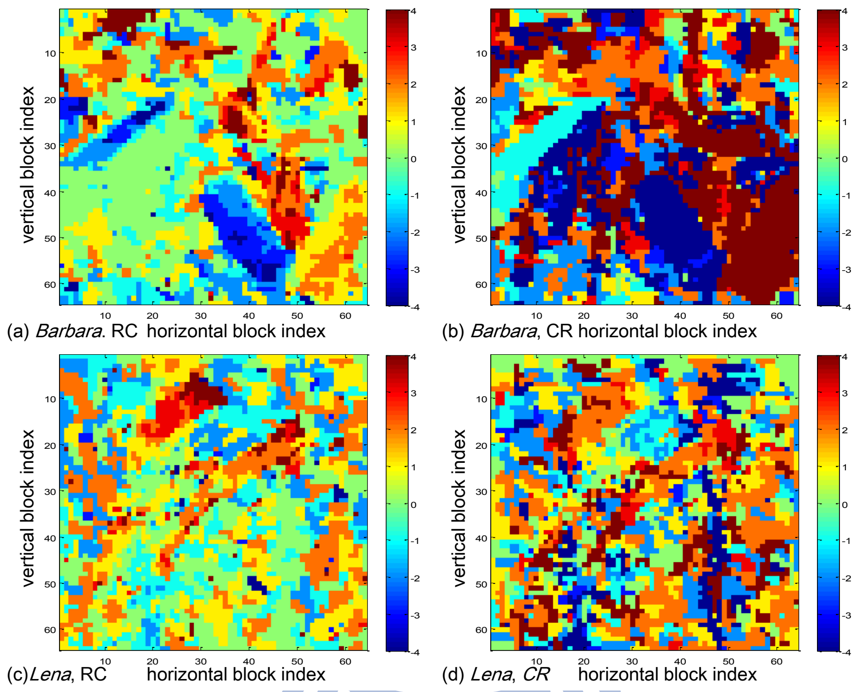


Fig. 5-19. The aligned first direction of the entire image (8×8 block). Direction indexes in (a)(c) and (b)(d) are specified by Fig. 5-1(a)(b). (a)(c) are the same as Fig. 5-10(a)(b).

5.2.2. Step B2: Aligning Block Directions in Similar-Texture Regions

Next, we choose both the direction and subsampling pattern together in Step B2. Fig. 5-20 and Fig. 5-21 show the pseudo code of Step B2. For simplicity, we use only one GB pattern in this step, namely, *GB_n3*, which is a 4-block pattern defined in Fig. 5-4(c). Each orientation of *GB_n3* has a set of 18 prediction errors $\{D_{GB\_n3}(m, n; d_s)\}$, which is calculated using (5-1) in a similar way. We choose the best direction for an orientation of *GB_n3* based on the minimum $\{D_{GB\_n3}(m, n; d_s)\}$. The best direction belongs to one of two subsampling patterns, RC or CR. Each *GB_n3* has four orientation cases, *GB_n3_r1* ~ *GB_n3_r4* in Fig. 5-4(c). We count the occurrence number of the best orientation-case directions that belong to RC and call it *ocrc*.

```
ocrc_table = zeros(($F_H/B_H$), ($F_W/B_W$));
d_offset_v = 5;    d_offset_h = 14;
% $d_v$ = -4 ~ 4, $d_h$ = -4 ~ 4, thus we set d_offset_v = 5 and d_offset_h = 14 to match the array index.

subsampling_table = zeros(($F_H/B_H$), ($F_W/B_W$));
% This table records the subsampling pattern of each block, RC is 1, CR is 2.

% (Part A) Find the ocrc of a block B using GB_n3
for $i$ = 1 : ($F_H/B_H$)
  for $j$ = 1 : ($F_W/B_W$)
    for $GB$ = $GB\_n3$
      $D_{GB}$ = zeros(1, 18);
      for $d$ = $d_v$ and $d_h$
        for $m$ = $i$-1 : $i$+1
          for $n$ = $j$-1 : $j$+1
            if($B(m, n) \in GB$)
              if($d \in d_v$)
                $D_{GB}(d$ + d_offset_v) += $D_B(m, n, d)$;
              else
                $D_{GB}(d$ + d_offset_h) += $D_B(m, n, d)$;
              end
            end
          end
        end
      end
      find the best direction $d_{GB}$ of $GB$ based on the minimum candidate in $D_{GB}$;
      if($d_{GB} \in d_v$)
        ocrc_table($i$, $j$)++;    % accumulate the occurrence of $d_{GB} \in d_v$
      end
    end
  end
end
```

Fig. 5-20. The pseudo code of Step B2, part A.

Fig. 5-22 shows the *ocrc* distributions of two test images. Often, the blocks with *ocrc* ≥ 3 and those with *ocrc* ≤ 1 form two different regions. The blocks locating at region boundaries have *ocrc* = 2. Thus, we decide the subsampling pattern (RC or CR) of $B(i, j)$ based on its *ocrc* value. If *ocrc* ≥ 3, it is RC; and if *ocrc* ≤ 1, it is CR.

```
% (Part B) Find the $d_{B\_B2}$ of a block $B$
considered_GB = $GB\_n1$ && $GB\_n2$ && $GB\_n3$;
for $i = 1 : (F_H/B_H)$
  for $j = 1 : (F_W/B_W)$
    if(ocrc_table($i, j$) ≥ 3)
      $d_{B\_B2} = d_{B\_RC\_B1}$ ;   subsampling_table($i, j$) = 1;   % subsampling pattern of B($i, j$) is RC
    end
    if(ocrc_table($i, j$) ≤ 1)
      $d_{B\_B2} = d_{B\_CR\_B1}$ ;   subsampling_table($i, j$) = 2 ;   % subsampling pattern of B($i, j$) is CR
    end
    if(ocrc_table($i, j$) == 2)

      cost_temp_buffer = zeros(1, 18);
      cost_temp_buffer = cost_temp_buffer + 10000000000;
      % We assume 10000000000 is the up limit of the Lagrangian cost.

      % calculate the corresponding Lagrangian cost of each considered $GB$
      for $GB$ = considered_GB
        $D_{GB}$ = zeros(1, 18);
        for $d = d_v$ and $d_h$
          for $m = i-1 : i+1$
            for $n = j-1 : j+1$
              if($d \in d_v$)
                $D_{GB}(d$ + d_offset_v) += $D_B(m, n, d)$;
              else
                $D_{GB}(d$ + d_offset_h) += $D_B(m, n, d)$;
              end
            end
          end
        end

        find the best direction $d_{GB}$ of $GB$ based on the minimum candidate in $D_{GB}$;
        set $B(i, j) = B_c$ of $GB$;   $L_{GB\_B2} = D_{B_c}(d_{GB}) + \lambda_{B2}(R_{B2}/N_{GB})$;
        if($d_{GB} \in d_v$ && cost_temp_buffer($d_{GB}$ + d_offset_v) < $L_{GB\_B2}$)
          cost_temp_buffer($d_{GB}$ + d_offset_v) = $L_{GB\_B2}$;
        end
        if($d_{GB} \in d_h$ && cost_temp_buffer($d_{GB}$ + d_offset_h) < $L_{GB\_B2}$)
          cost_temp_buffer($d_{GB}$ + d_offset_h) = $L_{GB\_B2}$;
        end
      end

      find the aligned direction $d_{B\_B2}$ of $B(i, j)$ based on the minimum candidate in cost_temp_buffer;
      if($d_{B\_B2} \in d_v$)
        subsampling_table($i, j$) = 1;
      else
        subsampling_table($i, j$) = 2;
      end
    end
  end
end
```

Fig. 5-21. The pseudo code of Step B2, part B.

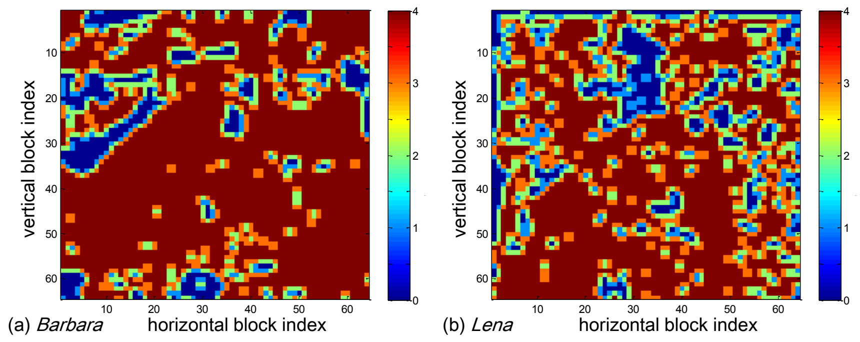


Fig. 5-22. *ocrc* of each 8×8 block.

We now try to align the blocks with *ocrc* = 2. If a block has an *ocrc* ≥ 3 or ≤ 1, its direction, d_{B_B2}, is set to $d_{B\_RC\_B1}$ or $d_{B\_CR\_B1}$ as discussed previously. For a block with *ocrc* = 2, similar to Step A1, we set it as B_c in Fig. 5-4 then compute the corresponding Lagrangian cost function, for instance, $L_{GB\_n3\_r1\_B2}$ by (5-5).

$$L_{GB\_n3\_r1\_B2} = D_{B_c}(d_{GB\_n3\_r1}) + \lambda_{B2}(R_{B2} / N_{GB\_n3\_r1}) \tag{5-5}$$

Again, $N_{GB\_n3\_r1}$ = 4 and λ_{B2} is obtained empirically. We assume 18 directions occurring with the same probability and set R_{B2} as $\log_2(18)$ for all cases. $d_{GB\_n3\_r1}$ is the best direction of *GB_n3_r1*. It is either d_v or d_h in Fig. 5-1. $D_{Bc}(d_{GB\_n3\_r1})$ is the prediction error of B_c corresponding to $d_{GB\_n3\_r1}$. A block with *ocrc* = 2 usually locates on the region boundaries in Fig. 5-22. We thus consider only the GB patterns of *n1*, *n2*, and *n3* here. We calculate L_{GB_B2} for each case and pick up the best direction with the minimum L_{GB_B2} as d_{B_B2} for the current block under consideration. Fig. 5-23 shows the aligned first directions after Step B2. As expected, the neighboring blocks with similar-texture regions have consistent directions and subsampling patterns. Again, a few isolated or small-cluster blocks still remain.

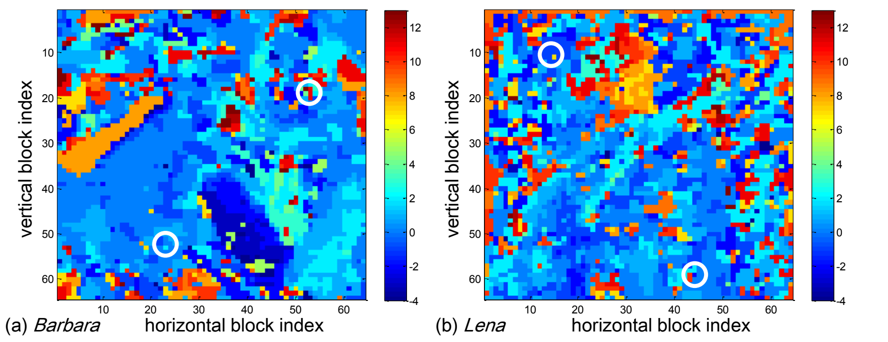


Fig. 5-23. The aligned first directions after Step B2 (8×8 block). The direction indexes -4 ~ 4 correspond to d_v (-4 ~ 4) and 5 ~ 13 correspond to d_h (-4 ~ 4) in Fig. 5-1. The circles indicate isolated blocks.

5.2.3. Step B3: Adjusting Directions of Isolated Blocks

The *isolated block* (*IB*) definition here is the same as that in Step A2 except that each block has 18 possible directions (not 9 directions). Fig. 5-23 shows some IBs in circles. Fig. 5-24 shows the pseudo code of Step B3. We adopt the procedure of Step A2 for Step B3 except that the Lagrangian cost function (eq. (5-3)) is replaced by (5-6).

$$L_{GB\_n3\_r1\_B3} = D_{B_c}(d_{B_{cn}\in GB\_n3\_r1}) + \lambda_{B3}(R_{B3} / N_{GB\_n3\_r1}) \tag{5-6}$$

In Step B3, the subsampling pattern (RC or CR) is decided by its *ocrc* (*ocrc* ≥ 3 or ≤ 1). If the *ocrc* of the current B_c is ≥ 3 or ≤ 1, we align its direction to those B_{cn} with the same subbsampling pattern. If *ocrc* = 2, we align its direction to the directions of all B_{cn} in *GB* without considering the subsampling pattern. All cases in Fig. 5-4 are included in the selection process. The conditions of B_{cn} ∈ *GB_n3_r1* are similar as those in Step A2. $B_{cn} \in GB\_n3\_r1$ must have a consistent subsampling pattern and directions $d_{Bcn \in GB\_n3\_r1}$. $D_{Bc}(d_{Bcn \in GB\_n3\_r1})$ is the prediction error of B_c corresponding to $d_{Bcn \in GB\_n3\_r1}$. The final selection is based on the minimum Lagrangian cost function and the final selected direction is denoted as d_{IB_B3}. Again, in calculating R_{B3}, we assume a direction index predictor is in use [28], similar to the calculation of R_{A2}. λ_{B3} is the Lagrangian multiplier obtained empirically.

```
d_offset_v = 5;    d_offset_h = 14;    IB_table = zeros(($F_H$/$B_H$), ($F_W$/$B_W$));

% record the location of $IB$
for $i$ = 1 : ($F_H$/$B_H$)
  for $j$ = 1 : ($F_W$/$B_W$)
    if($B(i, j)$ == $IB$)
      IB_table($i$, $j$) = 1;
    end
  end
end

while(sum of IB_table ~= 0)
  for $i$ = 1 : ($F_H$/$B_H$)
    for $j$ = 1 : ($F_W$/$B_W$)
      if(IB_table($i$, $j$) == 1)
        cost_temp_buffer = zeros(1, 18);
        cost_temp_buffer = cost_temp_buffer + 10000000000;

        % calculate the corresponding Lagrangian cost of each considered $GB$
        for $GB$ = $GB\_n1$ && $GB\_n2$ && $GB\_n3$ && $GB\_n4$ && $GB\_n5$
          if($B_{cn}$ of $GB$ have identical direction $d_{B_{cn}\in GB}$ and subsampling pattern && IB_table of $B_{cn}$ == 0)
            %$B_{cn}$ can be IB with $d_{B\_B3}$ but not $d_{B\_B2}$.
            set $B(i, j) = B_c$ of $GB$;    $L_{GB\_B3} = D_{B_c}(d_{B_{cn}\in GB}) + \lambda_{B3}(R_{B3}/N_{GB})$;
            if($d_{GB} \in d_v$ && cost_temp_buffer($d_{GB}$ + d_offset_v) < $L_{GB\_B3}$ && ocrc_table($i$, $j$) $\geq$ 2)
              cost_temp_buffer($d_{GB}$ + d_offset_v) = $L_{GB\_B3}$;
            end
            if($d_{GB} \in d_h$ && cost_temp_buffer($d_{GB}$ + d_offset_h) < $L_{GB\_B3}$ && ocrc_table($i$, $j$) $\leq$ 2)
              cost_temp_buffer($d_{GB}$ + d_offset_h) = $L_{GB\_B3}$;
            end
          end
        end

        if(the minimum element of cost_temp_buffer < 10000000000)
          find the aligned direction $d_{B\_B3}$ of $B(i, j)$ based on the minimum candidate in cost_temp_buffer;
          if($d_{B\_B3} \in d_v$ && ocrc_table($i$, $j$) == 2)
            subsampling_table($i$, $j$) = 1;
          else
            subsampling_table($i$, $j$) = 2;
          end
          IB_table($i$, $j$) = 0;
        end
      end
    end
  end
end
```

Fig. 5-24. The pseudo code of Step B3.

After Step B3, the directions of all blocks are renamed as d_{B_B3}. Fig. 5-25 shows d_{B_B3} of two test images. Most of IBs in Fig. 5-23 are eliminated.

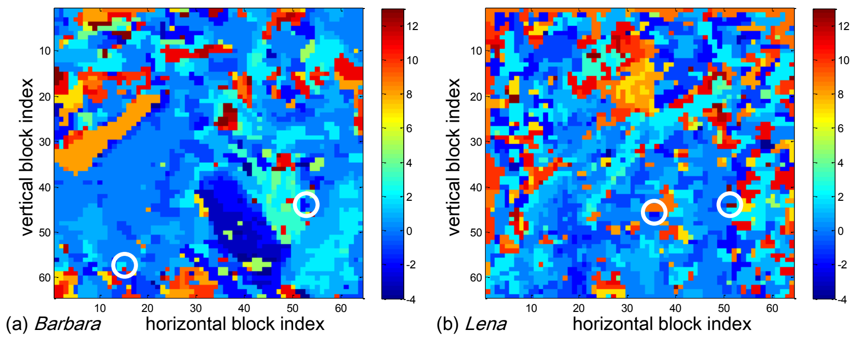


Fig. 5-25. The aligned first directions after Step B3 (8×8 block). The direction indexes -4 ~ 4 correspond to d_v (-4 ~ 4) and 5 ~ 13 correspond to d_h (-4 ~ 4). The circles indicate small-clustered blocks.

5.2.4. Step B4: Adjusting Directions of Small-Cluster Blocks

The *small-cluster block* (*SCB*) definition in Step A3 is also adopted here. Fig. 5-25 shows some examples of *SCB*s in circles. We adopt the procedure of Step A3 except that the Lagrangian cost function (eq.(5-3) is replaced by (5-7) in Step B4. Fig. 5-26 shows the pseudo code of Step B4.

$$L_{GB\_n3\_r1\_B4} = D_{B_c}(d_{B_{cn} \in GB\_n3\_r1}) + \lambda_{B4}(R_{B4} / N_{GB\_n3\_r1}) \tag{5-7}$$

For the subsampling pattern selection of B_{cn}, it decided by the *ocrc* of the current block in a similar way to Step B3. After picking up the best direction based on the Lagrangian cost function, the directions of all blocks are denoted as d_{B_B4}. Two aligned direction samples after Step B4 are shown in Fig. 5-27.

```
d_offset_v = 5;   d_offset_h = 14;   SCB_table = zeros(($F_H$/$B_H$), ($F_W$/$B_W$));

% record the location of $SCB$
for $i$ = 1 : ($F_H$/$B_H$)
  for $j$ = 1 : ($F_W$/$B_W$)
    if($B(i, j)$ == $IB$)
      SCB_table($i, j$) = 1;
    end
  end
end

while(sum of SCB_table ~= 0)
  for $i$ = 1 : ($F_H$/$B_H$)
    for $j$ = 1 : ($F_W$/$B_W$)
      if(SCB_table($i, j$) == 1)
        cost_temp_buffer = zeros(1, 18);
        cost_temp_buffer = cost_temp_buffer + 10000000000;

        % calculate the corresponding Lagrangian cost of each considered $GB$
        for $GB$ = $GB\_n1$ && $GB\_n2$ && $GB\_n3$ && $GB\_n4$ && $GB\_n5$
          if($B_{cn}$ of $GB$ have identical direction $d_{B_{cn} \in GB}$ and subsampling pattern && SCB_table of $B_{cn}$ == 0)
            %$B_{cn}$ can be SCB with $d_{B\_B4}$ but not $d_{B\_B3}$.
            set $B(i, j) = B_c$ of $GB$;   $L_{GB\_B4} = \mathrm{D}_{\mathrm{B_c}}(d_{B_{cn} \in GB}) + \lambda_{B4}(R_{B4}/N_{GB})$;
            if($d_{GB} \in d_v$ && cost_temp_buffer($d_{GB}$ + d_offset_v) < $L_{GB\_B4}$ && ocrc_table($i, j$) $\geq$ 2)
              cost_temp_buffer($d_{GB}$ + d_offset_v) = $L_{GB\_B4}$;
            end
            if($d_{GB} \in d_h$ && cost_temp_buffer($d_{GB}$ + d_offset_h) < $L_{GB\_B4}$ && ocrc_table($i, j$) $\leq$ 2)
              cost_temp_buffer($d_{GB}$ + d_offset_h) = $L_{GB\_B4}$;
            end
          end
        end

        if(the minimum element of cost_temp_buffer < 10000000000)
          find the aligned direction $d_{B\_B4}$ of $B(i, j)$ based on the minimum candidate in cost_temp_buffer;
          if($d_{B\_B4} \in d_v$ && ocrc_table($i, j$) == 2)
            subsampling_table($i, j$) = 1;
          else
            subsampling_table($i, j$) = 2;
          end
          SCB_table($i, j$) = 0;
        end
      end
    end
  end
end
```

Fig. 5-26. The pseudo code of Step B4.

We have aligned the first directions and subsampling patterns using Steps B1 to B4. This helps in reducing the side information and improving the coding performance. The phase-completion process estimates the missing pixel from its neighboring pixels with the same sampling phase. It induces a bit of the boundary effect between two different subsampling patterns due to mismatch. The aligned subsampling patterns can also reduce this boundary effect. Similar to Steps A2 and A3, Steps B3 and B4 are also less desirable for the low-resolution images.

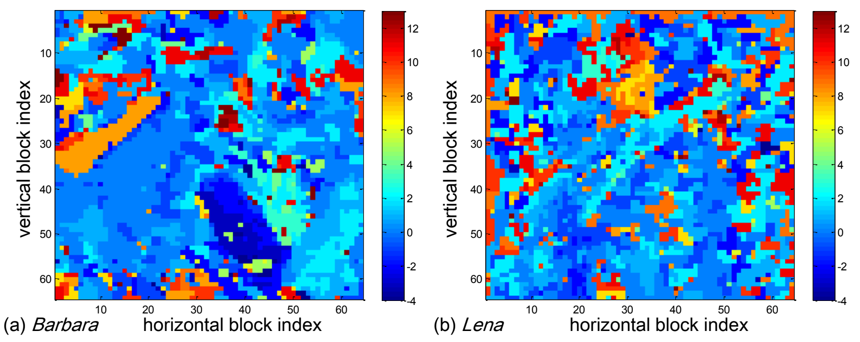


Fig. 5-27. The aligned first directions after Step B4 (8×8 block). The direction indexes -4 ~ 4 correspond to d_v (-4 ~ 4) and 5 ~ 13 correspond to d_h (-4 ~ 4).

5.2.5. Step C1: Aligning Block Directions based on Single Subsampling Pattern

Now, we apply the second 1-D SA-DWT to all blocks. Because the first 1-D SA-DWT has decided the subsampling pattern of every block, the sampling pattern of the second 1-D SA-DWT is thus decided, which is the complementary to the first one. Consequently, the candidate directions under consideration must be consistent with the specified subsampling pattern. In principle, we copy Steps A1 to A3 to Steps C1 to C3 for aligning the directions of the second 1-D SA-DWT. Because different subsampling patterns interweave each other, the prediction error calculation for aligning the second directions is tedious.

```
for $i = 1 : (F_H/B_H)$
  for $j = 1 : (F_W/B_W)$
    if(subsampling_table $(i, j)$ == 1)
      the first direction of $B_{RC}(i, j) = d_{B\_B4}$ of $B(i, j)$;
      the first direction of $B_{CR}(i, j)$ = direction 0 of $d_h$;
    else
      the first direction of $B_{RC}(i, j)$ = direction 0 of $d_v$;
      the first direction of $B_{CR}(i, j) = d_{B\_B4}$ of $B(i, j)$;
    end
  end
end

% The process based on RC. $B_{RC}(i, j)$'s first direction $\in d_v$ and second direction $\in d_h$.
Applying vertical 1-D DA-DWT to the test image in the process based on RC.
The size of two spatial subbands are $(F_H / 2) \times F_W$.
Partitioning the spatial low-pass subband into $(B_H / 2) \times B_W$ blocks.
Applying Step A1 ~ Step A3 to the spatial low-pass subband.
Setting the aligned second directions to $B_{RC}(i, j)$.
Each $B_{RC}(i, j)$ has a set of prediction errors corresponding to the second direction from Step A1.

% The process based on CR. $B_{CR}(i, j)$'s first direction $\in d_h$ and second direction $\in d_v$.
Applying horizontal 1-D DA-DWT to the test image in the process based on CR.
The size of two spatial subbands are $F_H \times (F_W / 2)$.
Partitioning the spatial low-pass subband into $B_H \times (B_W / 2)$ blocks.
Applying Step A1 ~ Step A3 to the spatial low-pass subband.
Setting the aligned second directions to $B_{CR}(i, j)$.
Each $B_{CR}(i, j)$ has a set of prediction errors corresponding to the second direction from Step A1.

for $i = 1 : (F_H/B_H)$
  for $j = 1 : (F_W/B_W)$
    if(subsampling_table $(i, j)$ == 1)
      the second direction of $B(i, j)$ = the second direction of $B_{RC}(i, j)$;
      the prediction errors of $B(i, j)$ for Step C2 ~ Step C3
      = the prediction errors of $B_{RC}(i, j)$ corresponding to the second direction;
    else
      the second direction of $B(i, j)$ = the second direction of $B_{CR}(i, j)$;
      the prediction errors of $B(i, j)$ for Step C2 ~ Step C3
      = the prediction errors of $B_{CR}(i, j)$ corresponding to the second direction;
    end
  end
end
```

Fig. 5-28. The pseudo code of Step C1.

Fig. 5-28 shows the pseudo code of Step C1. We assume the blocks with different subsampling patterns can be processed independently. Thus, we calculate the directions and prediction error of the second 1-D SA-DWT in two parallel processes, one based on RC and the other based on CR, as illustrated by Fig. 5-29. We have decided the directions of the first 1-D SA-DWT for every block. We label a block by $B_{RC}(i, j)$ in the process based on RC and $B_{CR}(i, j)$ in the process based on CR. If the subsampling pattern of $B(i, j)$ is RC, we set the first direction of $B_{RC}(i, j)$ as $d_{B(i, j)\_B4}$. Otherwise, we set the first direction of $B_{RC}(i, j)$ as 0 of d_v, the vertical 1-D DWT. We set the first directions of $B_{CR}(i, j)$ in the other process in a similar way. The purpose is to make the cost function calculation easily. Certainly, this is an approximation. Fig. 5-30 shows the first directions of these two parallel processes.

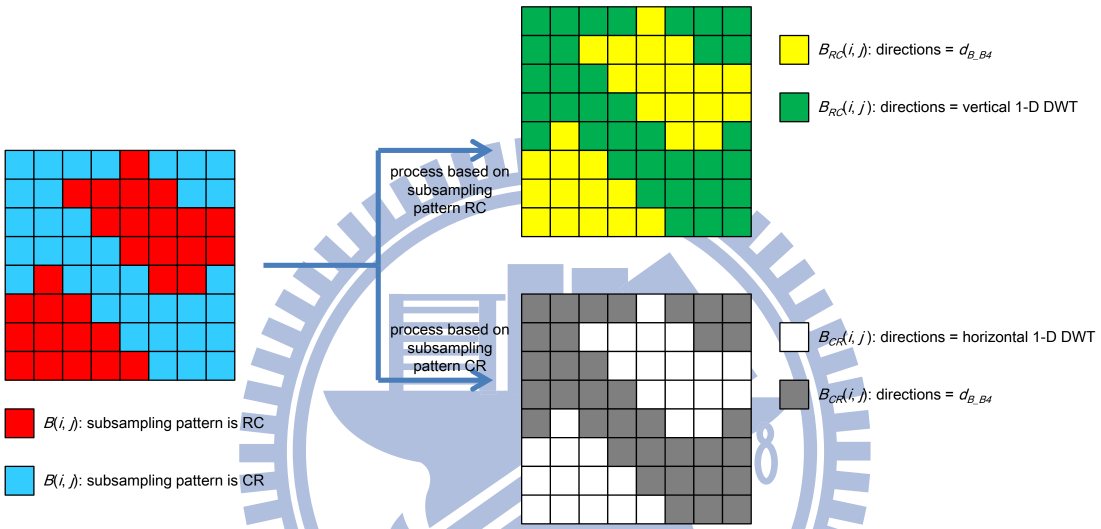

process based on subsampling pattern RC
process based on subsampling pattern CR
$B_{RC}(i, j)$: directions = d_{B_B4}
$B_{RC}(i, j)$: directions = vertical 1-D DWT
$B_{CR}(i, j)$: directions = horizontal 1-D DWT
$B_{CR}(i, j)$: directions = d_{B_B4}
$B(i, j)$: subsampling pattern is RC
$B(i, j)$: subsampling pattern is CR

Fig. 5-29. The parallel processes for handling an image's first direction.

After setting up the directions of the first transform in each of the two parallel processes, we apply the specified the first 1-D DA-DWT and subsampling pattern to a test image for each process. It results in the spatial low-pass and high-pass subbands located in different rows for the first process and located in different columns in the second process. We then can start the procedure of selecting the second transform directions. This procedure is identical to Steps A1 ~ Step A3. For the second transform, we use 1-D DA-DWT for the spatial low-pass subband and use 1-D DWT for the high-pass subband. Applying the alignment algorithm Step A1 ~ Step A3 to each process separately, Fig. 5-31 shows the aligned second directions of spatial low-pass subbands of each process.

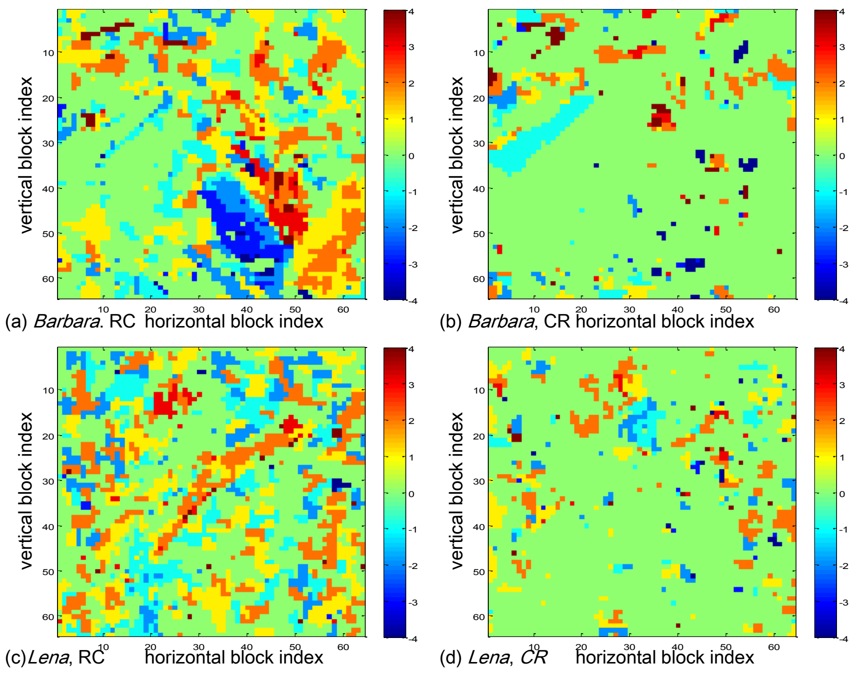


Fig. 5-30. The first direction of each 8×8 block. (a)(b) are the two parallel processes of *Barbara* and (c)(d) are those of *Lena*. Direction indexes in (a)(c) and (b)(d) are defined by Fig. 5-1(a) and Fig. 5-1(b).

Next, we need to merge the two processed processes into one image. In the first parallel process, block $B_{RC}(i, j)$ has both valid the first transform and the second transform. In the second process, block $B_{CR}(i, j)$ also has valid the first transform and the second transform. Therefore, we pick up these blocks and put them into the final image as illustrated by Fig. 5-32. The merged image with selected directions is shown in Fig. 5-33. The aligned second transform direction of block $B(i, j)$ is denoted as $d_{B(i, j)\_C1}$ after Step C1. We can get the prediction error of the second transform when aligning directions. We assign the prediction errors to each block $B(i, j)$ similarly for following alignment.

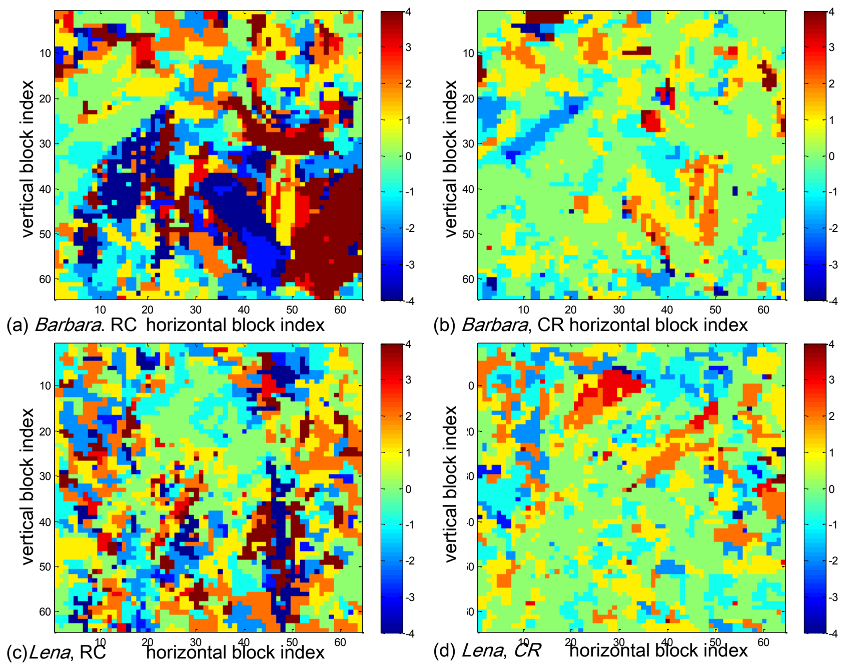
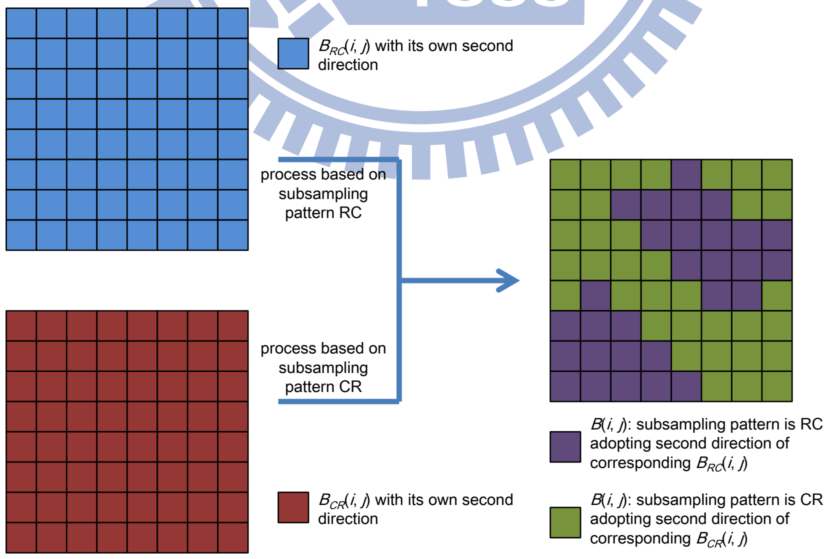


Fig. 5-31. The aligned second directions of each 8×8 block. (a)(b) are two parallel processes of *Barbara* and (c)(d) are those of *Lena*. Direction indexes in (a)(c) and (b)(d) are defined by Fig. 5-1(b) and Fig. 5-1(a).



$B_{RC}(i, j)$ with its own second direction
process based on subsampling pattern RC
process based on subsampling pattern CR
$B_{CR}(i, j)$ with its own second direction
$B(i, j)$: subsampling pattern is RC adopting second direction of corresponding $B_{RC}(i, j)$
$B(i, j)$: subsampling pattern is CR adopting second direction of corresponding $B_{CR}(i, j)$

Fig. 5-32. The parallel processes for handling the second direction.

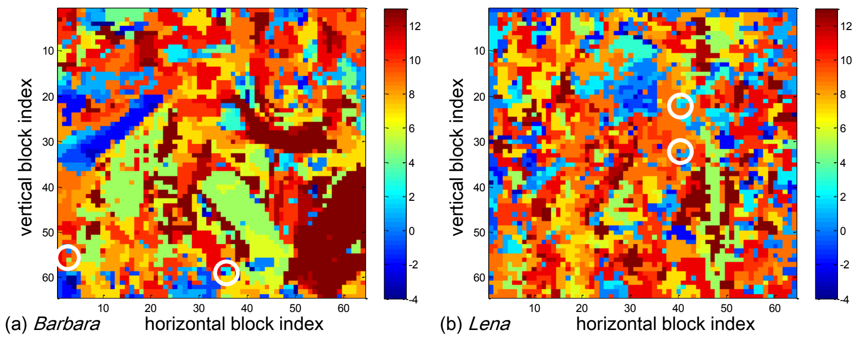


Fig. 5-33. The aligned second directions after Step C1 (8×8 block). The direction indexes -4 ~ 4 correspond to d_v (-4 ~ 4) and 5 ~ 13 correspond to d_h (-4 ~ 4). The circles indicate isolated blocks.

5.2.6. Step C2: Adjusting Directions of Isolated Blocks

The *isolated block* (*IB*) definition of the second *1-D SA_DWT* is similar to the previous one. Fig. 5-33 shows some *IB*s in circles. We adjust the second directions of *IB*s in a similar way to adjusting those of the first transform *IB*s in Step B3. We align the current block direction to the neighboring blocks with the same subsampling pattern. The aligned second direction is denoted as d_{B_C2} after Step C2; two test images are shown in Fig. 5-34.

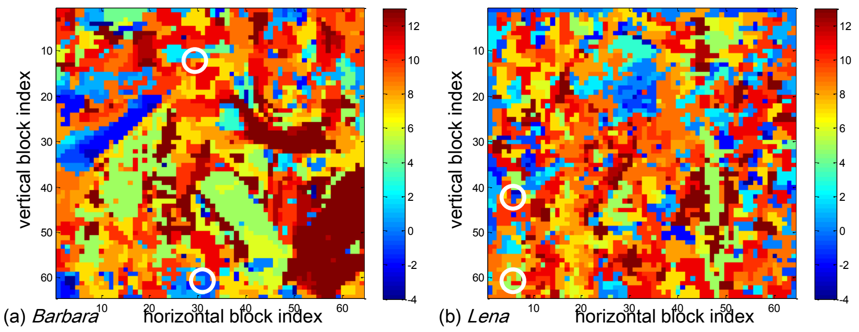


Fig. 5-34. The aligned second directions after Step C2 (8×8 block). The circles indicate small-clustered blocks.

5.2.7. Step C3: Adjusting Directions of Small-Cluster Blocks

The *small-cluster blocks* (*SCB*) definition is similar to the previous one. Their second directions are adjusted in a similar way to Step B4. Fig. 5-34 shows some *SCB*s. Fig. 5-35 shows the aligned second directions after this step.

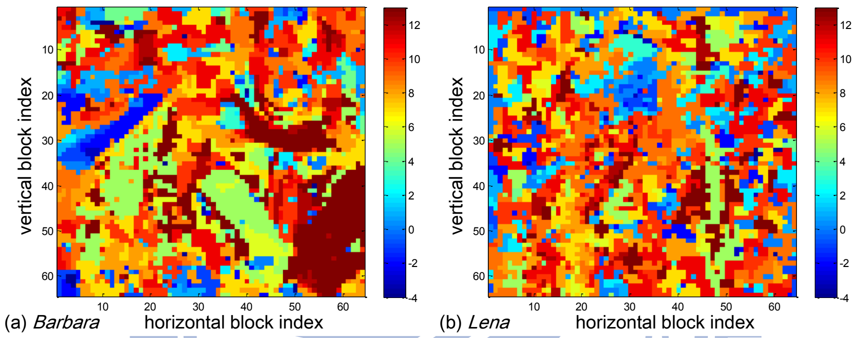


Fig. 5-35. The aligned second directions after Step C3 (8×8 block).

The basic concepts of Steps C1 to C3 are similar to those of B2 to B4, individually. The 2-D SA-DWT needs to consider two subsampling patterns at the same time and this complicates quite a bit the entire direction alignment process. Note that the two-parallel processes operation is added into Step C1 to reduce the prediction error calculation. Also, Steps C2 and C3 are also less desirable for low-resolution images

5.3 Prediction Residual Characteristics

and 2-D MSA-DWT

5.3.1. Predication Residuals in Frequency Domain

In this section, we examine the frequency-domain energy distribution of the temporal high-pass (prediction residual) signals. Because the DA-DWT partitions an image into blocks and find the best filtering direction for each block, we thus study the block characteristics. We partition natural images, T_Ls, and T_Hs into 8×8 blocks and we apply 64×64 2-D DFT to each block.

Fig. 5-36 shows the energy spectrum of image blocks. Blocks with smooth texture have narrow and strong energy peak located at low frequencies. DWT provides good compression performance for these blocks. Some blocks contain edges along specific directions. Their energy spectrums have energy peaks spreading over a short line segment at a specific angle (decided by the edge orientation). The zero-frequency component has powerful energy. For repeated line patterns such as the pants of Barbara (Fig. 5-36(a)), their spectrum contains periodic peaks spreading along a line. The DA-DWT filters can be adjusted along specific directions and thus it can represent edges and line patterns more efficiently. Blocks of T_Ls show similar property in Fig. 5-37. Therefore, DA-DWT also compresses T_Ls well and improves the coding performance of wavelet-based image coding [45].

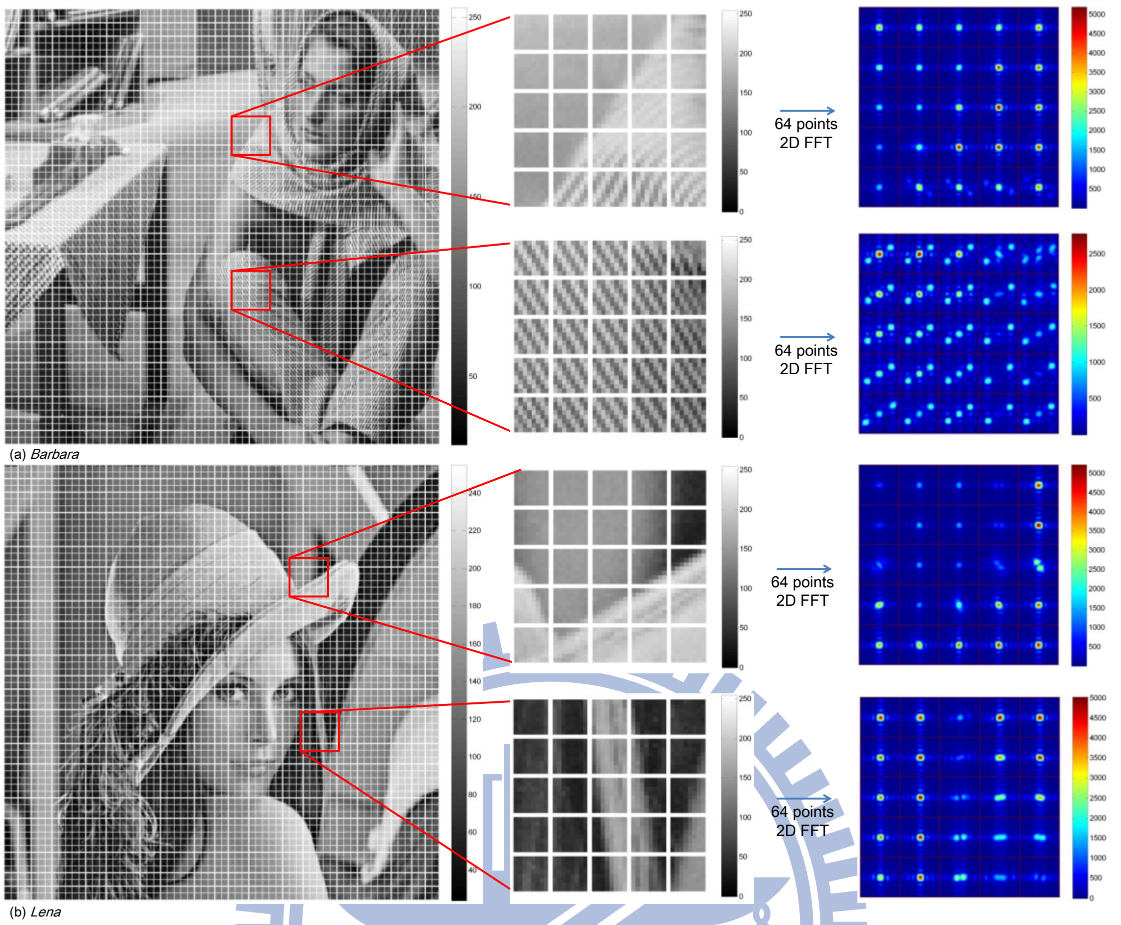


Fig. 5-36. Frequency domain spectrum of some image blocks.

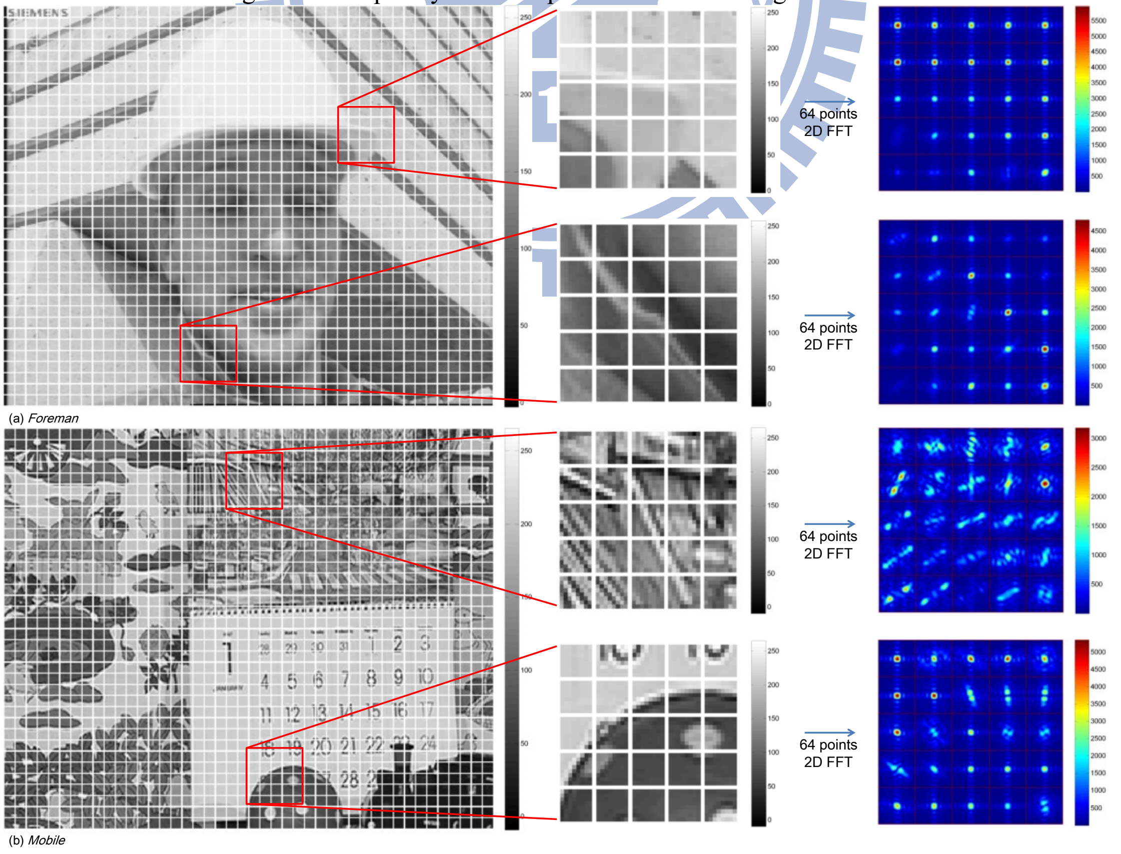


Fig. 5-37. Frequency domain spectrum of some T_L blocks.

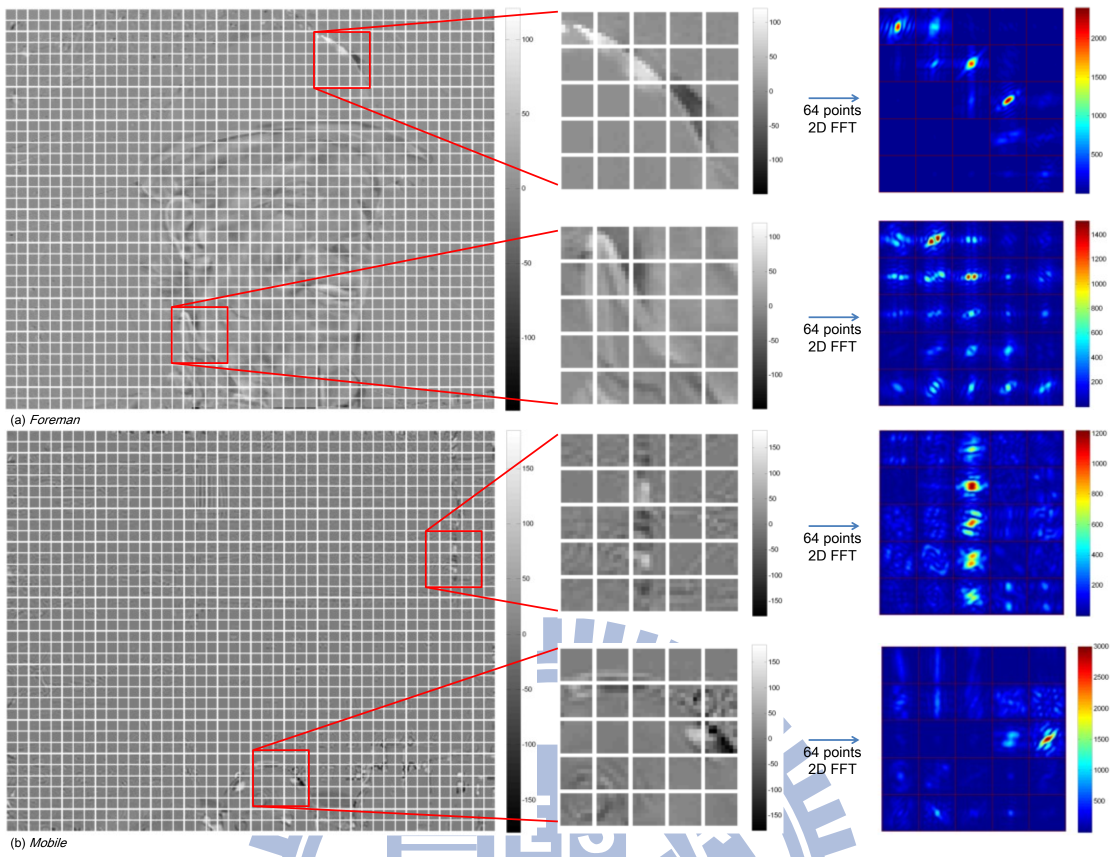


Fig. 5-38. Frequency domain spectrum of some T_H blocks.

Fig. 5-38 shows the energy spectrum of T_H blocks. The energy of most blocks is quite low. For blocks with a somewhat significant amount of energy, a few blocks have energy peak locating at low frequency. These energy peaks contain much less energy than those in Fig. 5-36 and Fig. 5-37. A number of blocks have edge or line–type spectrums. Potentially, these blocks can be well represented by DA-DWT or SA-DWT. There are many other blocks having spectrums spreading over a wide region or even nearly the entire frequency plane. DA-DWT and SA-DWT do not seem to offer more coding gains than the ordinary DWT on these blocks. Overall, the edge and line-type blocks are not yet dominate the T_H signals and the total energy of

T_H signals is much less than that of the T_L signals; therefore, the advantage of T_H band, DA-DWT or SA-DWT on the overall coding efficiency is not dramatic although they do provide some gains on the T_H signal compression.

5.3.2. Transform Coefficients

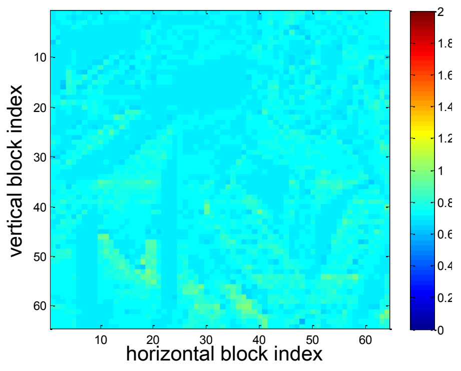
It is reported that 1-D DCT with adaptive orientation compresses T_H better than 2-D DCT because the latter spreads the energy to a larger number of transformed coefficients [43]. We examine the transformed coefficients of 2-D SA-DWT on natural images, T_Ls, and T_Hs. The 1-D SA-DWT decomposes a block B into spatial low-pass subband B_L and spatial high-pass subband B_H. Another 1-D SA-DWT decomposes B_L into B_{LL} and B_{LH}. For all the transform coefficients in B, we calculate the sum of absolute values and it is denoted as SAV_B. The other quantities, SAV_{BL}, SAV_{BH}, SAV_{BLL}, and SAV_{BLH}, are similarly defined for the coefficients in various subbands. We then define the ratios between these quantities in (5-8) and (5-9).

$$B\_ratio\_1 = (SAV_{B_L} / \sqrt{\omega_L} + SAV_{B_H} / \sqrt{\omega_H}) / SAV_B \tag{5-8}$$

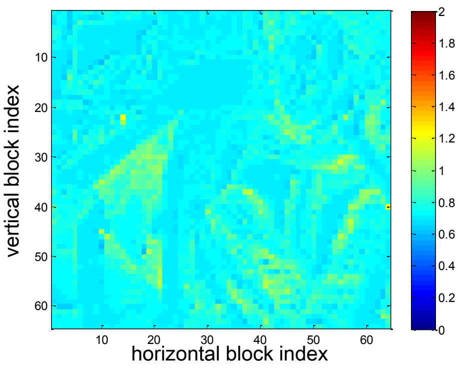
$$B\_ratio\_2 = (SAV_{B_{LL}} / \sqrt{\omega_L} + SAV_{B_{LH}} / \sqrt{\omega_H}) / SAV_{B_L} \tag{5-9}$$

where ω_L and ω_H are the energy responses of the low-pass and high-pass wavelet filters. Because the bit-plane coding technique is adopted for entropy coding, SAV is in a way in proportional to the coding bits. Thus, this *SAV ratio* gives an indication of coding bits before and after transform. A ratio < 1 usually implies higher compression efficiency.

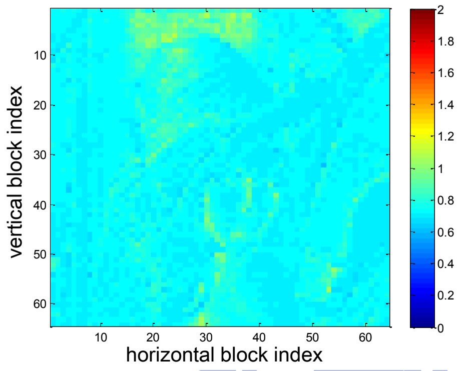
These two ratios calculated for images, T_Ls, and T_Hs are shown in Fig. 5-39, Fig. 5-40, and Fig. 5-41. For images and T_Ls (Fig. 5-39 and Fig. 5-40), the SA-DWT produces both ratios at around 75% in average. In other words, after the wavelet decomposition, the coding bits are generally fewer. It saves about 20% ~ 28% in *SAV* in the first and the second transform. For T_H in Fig. 5-41, the SA-DWT saves about 10% in *SAV* at most. Thus, the SA-DWT is less efficient in coding T_Hs. Many blocks of T_H have the second transform ratios larger than 1 in Fig. 5-41(b)(d); that is, the *SAV* value is increased after the second transform. The effect of the second transform on T_H will be further examined in the next sub-section.



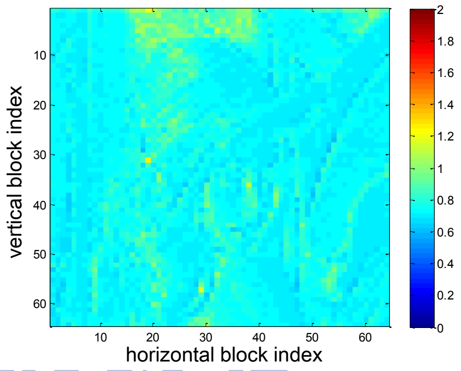
(a) *Barbara, ratio_1st,* average = 0.7357



(b) *Barbara, ratio_2nd,* average = 0.7578

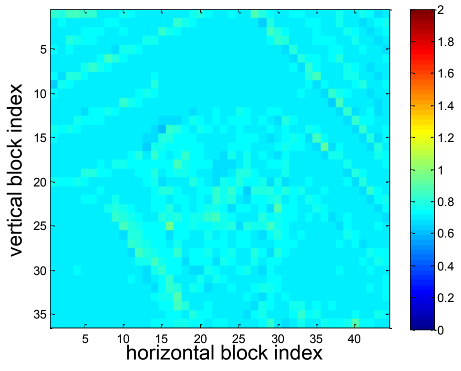


(c) *Lena, ratio_1st,* average = 0.7461

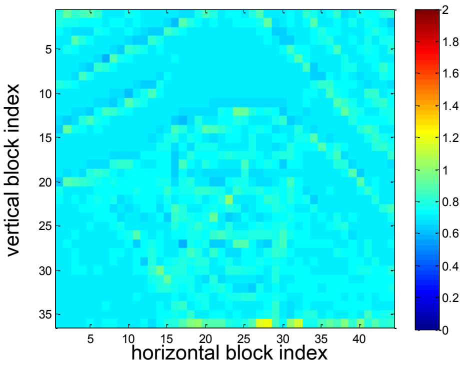


(d) *Lena, ratio_2nd,* average = 0.7512

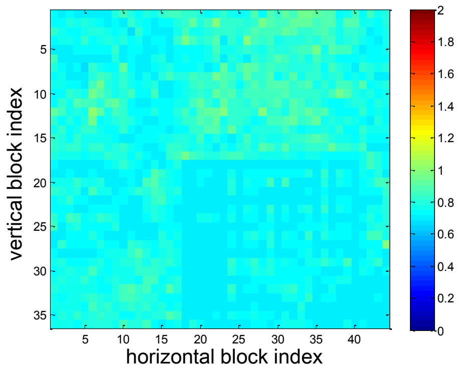
Fig. 5-39. The first and the second transform ratios of images.



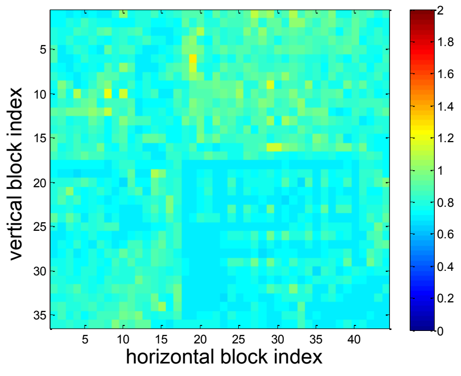
(a) *Foreman, ratio_1st,* average = 0.7212



(b) *Foreman, ratio_2nd,* average = 0.7314

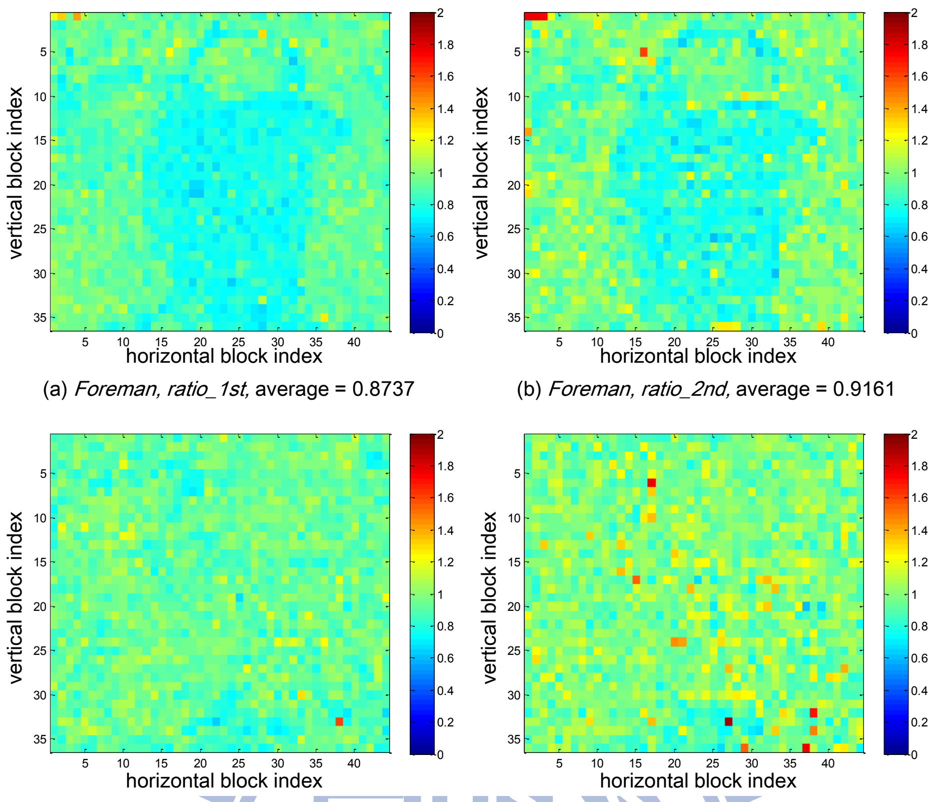


(c) *Mobile, ratio_1st,* average = 0.7670



(d) *Mobile, ratio_2nd,* average = 0.8011

Fig. 5-40. The first and the second transform ratios of T_Ls.



(a) *Foreman, ratio_1st,* average = 0.8737 (b) *Foreman, ratio_2nd,* average = 0.9161

(c) *Mobile, ratio_1st,* average = 0.9345 (d) *Mobile, ratio_2nd,* average = 0.9882

Fig. 5-41. The first and the second transform ratios of T_Hs

5.3.3. The Second Transform

We apply 2-D SA-DWT to blocks of T_Hs and show the transform coefficients in Fig. 5-42. Typically, the first 1-D SA-DWT concentrates energy into B_L but the second 1-D SA-DWT often spreads the energy into both B_{LL} and B_{LH}. This is consistent with the report that 2-D transforms compresses T_H inefficiently because of 1-D structures of T_H [43]. Often, the 2-D transform spreads the energy of coefficients and results in more coefficients.

We thus modify the original 2-D SA-DWT and call it 2-D MSA-DWT (modified subsampling and direction-adaptive DWT). The 2-D MSA-DWT has the same first transform as 2-D SA-DWT but it may or may not perform the second transform on B_L. The second transform is turned on when (5-10) holds.

$$SAV_{B_L} > SAV_{B_{LL}} \big/ \sqrt{\omega_L} + SAV_{B_{LH}} \big/ \sqrt{\omega_H} \tag{5-10}$$

When (5-10) is not valid, we split the samples of B_L in Fig. 5-15 into B_{LL} and B_{LH} without executing transform. This "no second transform" case is labeled by a direction index of "5".

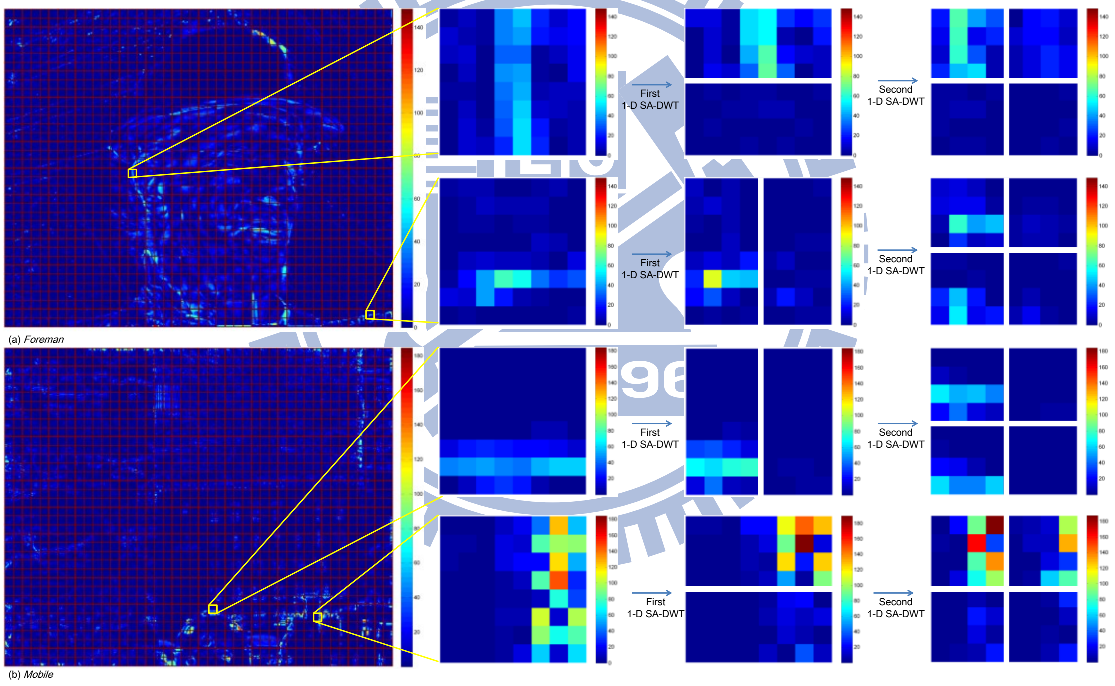


Fig. 5-42. The transform coefficients in T_Hs after 2-D SA-DWT. The coefficients are displayed in absolute value.

Chapter 6 Experimental Results

6.1 Experimental Results of FMDT

We have discussed the three proposed algorithms that enhance a WBCT image coding scheme in computation and/or complexity reduction. They are short length 2-D filters, a mean-shift-based decision, and new ZC context tables for ESCOT. In this section, we examine the impact of each algorithm towards the system performance. And, putting them together, we compare the overall performance between the 2-D DWT image coding scheme, the original WBCT image coding scheme, and the proposed WBCT image coding scheme with three new aglorithms.

A few abbreviations are explained below. The original WBCT image coding scheme can apply directional filtering to either all subbands (NDS1) or no subband (NDS2). With our decision mechanism (WDS), we adaptively choose the subbands for directional filtering. Moreover, the original WBCT scheme uses long length directional filters (LLF), and our proposed image coding scheme uses short length directional filters (SLF) instead. The no directional filtering (NDF) situation appears when either the WDS declares that no subband needs directional filtering or the NDS2 strategy is adopted. There are two options for ESCOT: the original context tables (O) designed for 2-D DWT coefficients or the proposed context tables (P) fine-tuned for the WBCT coefficients. Table 6-1 summarizes all the aforementioned abbreviations.

Table 6-1. Abbreviations for the adopted algorithms in the image coding scheme

Directional Transform	
SLF	Short Length directional Filter.
LLF	Long Length directional Filter.
NDF	No directional Filter.
Decision	
NDS1	No Decision, applying directional transform on all subbands (LH^1, HL^1, and HH^1).
NDS2	No Decision, directional transform not applied.
WDS	With Decision, applying directional transform on the chosen wavelet subbands.
Entropy Coder	
O	ESCOT with the Original ZC context tables.
P	ESCOT with the Proposed ZC context tables.

The notation of an image coding scheme consists of three parts: the directional transform type, the decision, and the coder tables. For example, the 2-D DWT image coding scheme is “NDF+NDS2+O”, the original WBCT image coding scheme is “LLF+NDS1+O”, and our proposed coding scheme with three algorithms is “SLF+WDS(HL^1, HH^1)+P”. Note that the subbands selected by WDS are listed in the parenthesis after WDS, and thus “WDS(LH^1, HL^1, and HH^1)” is the same as “NDS1”.

6.1.1. Short Length Directional Filters

Our test images are listed in Table 4-2. The experimental platform is Matlab r2008b on a PC with Intel Core 2 Quad Q9400 CPU. First, we show the impacts of filter length in terms of PSNR and run time by comparing “SLF+WDS+O” and “LLF+WDS+O”. Fig. 6-1 shows their PSNR at various bitrates (bit per pixel, bpp). Obviously, the image coding scheme with SLF has similar PSNR performances as that with LLF. Table 6-2 shows the run time of these two schemes and the image coding scheme with SLF consumes only 10%~20% computational time of that with LLF.

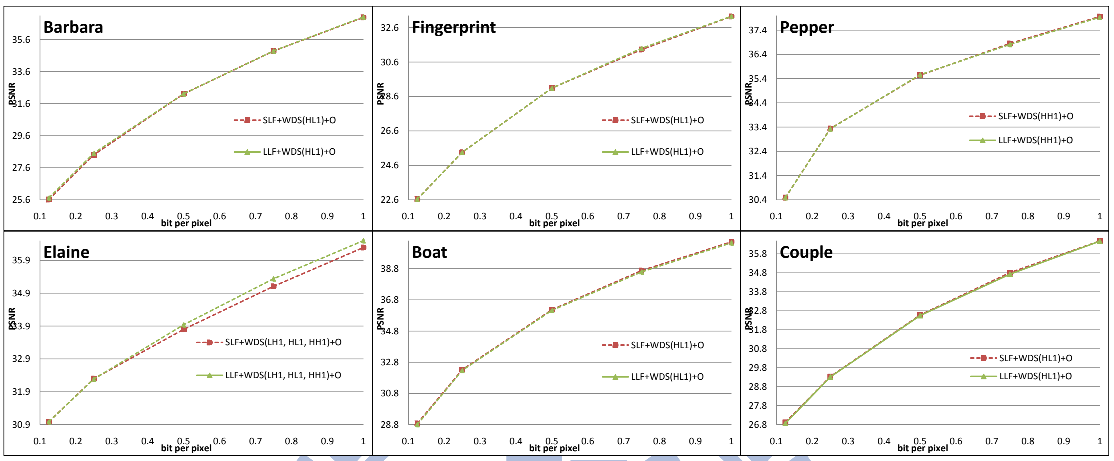


Fig. 6-1. PSNR of the image coding schemes with SLF and LLF (“SLF+WDS+O” and “LLF+WDS+O”)

Table 6-2. Run time of the image coding schemes with SLF and LLF

Scheme	SLF+WDS(HL^1)+O (Barbara, Fingerprint, Boat, Couple, average)	SLF+WDS(HH^1)+O (Pepper)	SLF+WDS(LH^1, HL^1, HH^1)+O (Elaine)
Run Time	4.547 sec	4.550 sec	8.203 sec
Scheme	LLF+WDS(HL^1)+O (Barbara, Fingerprint, Boat, Couple, average)	LLF+WDS(HH^1)+O (Pepper)	LLF+WDS(LH^1, HL^1, HH^1)+O (Elaine)
Run Time	23.031 sec	23.026 sec	62.484 sec

6.1.2. Decision Algorithm

Next, we present the impacts of decision algorithm in terms of PSNR, MSSIM [70] and run time among “SLF+WDS+O”, “SLF+NDS1+O”, and “NDF+NDS2+O” (the 2-D DWT coding scheme). MSSIM represents *mean of structural similarity.* A higher MSSIM implies a better image subjective quality. Fig. 6-2 shows the PSNR of the image coding schemes with and without decision. The image coding scheme with decision (“SLF+WDS+O”) has similar PSNR performance as those without decisions (“SLF+NDS1+O” and “NDF+NDS2+O”). Fig. 6-3 shows the MSSIM of the image coding schemes with and without decision. Our proposed image coding scheme with decision (“SLF+WDS+O”) has similar MSSIM performance as “SLF+NDS1+O” and has better MSSIM than “NDF+NDS2+O”. The visual quality improvement is most obvious on some pictures such as *Elaine*.

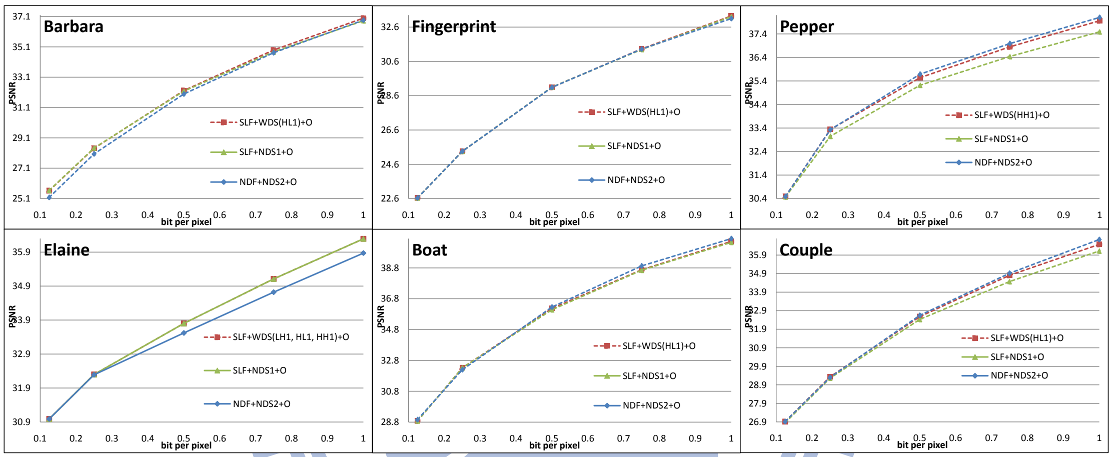


Fig. 6-2. PSNR of the image coding schemes with and without decision (“SLF+WDS+O”, “SLF+NDS1+O”, and “NDF+NDS2+O”).

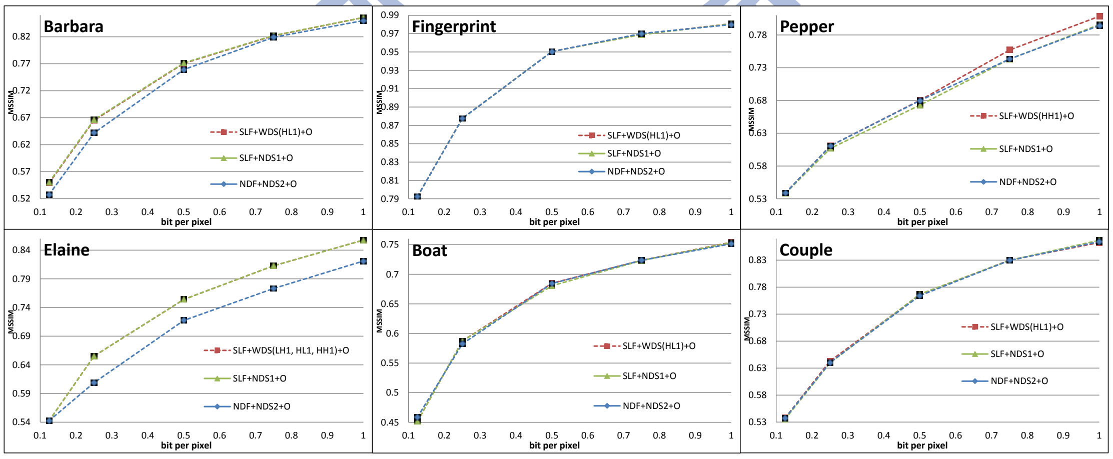


Fig. 6-3. MSSIM of the image coding schemes with and without decision (“SLF+WDS+O”, “SLF+NDS1+O”, and “NDF+NDS2+O”).

Fig. 6-4 shows portions of the original and the reconstructed images of *Barbara* and *Elaine* generated by these three schemes. Noticeably, “SLF+WDS+O” and “SLF+NDS1+O” show more texture details than “NDF+NDS2+O”. Table 6-3 shows the run time of these schemes. “SLF+WDS+O” saves about 50% computational time comparing to “SLF+NDS1+O” but it needs roughly 70% extra computational time comparing to “NDF+NDS2+O”. In brief, the image coding scheme with decision, “SLF+WDS+O,” achieves a good balance between quality and speed.

Table 6-3. Average run time of the image coding schemes with and without decision.

Scheme	SLF+WDS+O	SLF+NDS1+O	NDF+NDS2+O
Run Time	4.804 sec	8.206 sec	2.688 sec

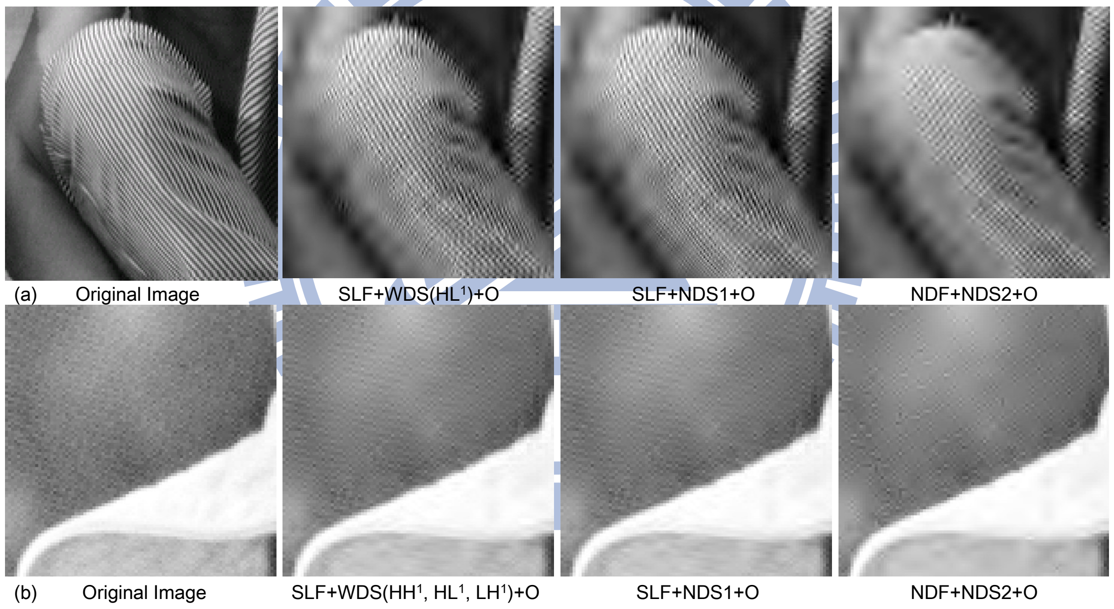


Fig. 6-4. (a) Portions of the original and the reconstructed images of *Barbara* at 0.125bpp. (b) Portions of the original and the reconstructed images of *Elaine* at 0.5bpp.

6.1.3. Proposed ZC Context Tables

Next, we examine the effect of the new ESCOT context tables in terms of PSNR and run time. Table 6-4 shows the PSNR of the image coding schemes with the original and the new ZC context tables when the directional filters are SLF. And Table 6-5 shows the PSNR when the directional filters are LLF. The image coding schemes with the new ZC context tables ("SLF/LLF+WDS+P") have a slightly better PSNR performance than those with the original ZC context table ("SLF/LLF+WDS+O") in all cases. Moreover, Table 6-6 indicates that "SLF/LLF+WDS+P" consumes less computation time than its "SLF/LLF+WDS+O" counterpart in all cases. The context table of "O" considers 26 neighbors in a 3×3×3 cubic but that of "P" considers only 8 neighbors in a 3×3 square. Clearly, "P" uses fewer neighbors and consumes less computation. Thus, our proposed context tables can also speed up slightly the coding process.

Table 6-4. PSNR of the image coding schemes with the original and the new ZC context tables (directional filters = SLF).

Test image	Coding Shceme	0.125 bpp	0.25 bpp	0.5 bpp	0.75 bpp	1.0 bpp
Barbara	SLF+WDS(HL^1)+O	25.62	28.41	32.22	34.89	36.99
	SLF+WDS(HL^1)+P	25.79	28.53	32.33	34.96	37.11
Fingerprint	SLF+WDS(HL^1)+O	22.64	25.36	29.09	31.33	33.25
	SLF+WDS(HL^1)+P	22.64	25.52	29.09	31.33	33.25
Pepper	SLF+WDS(HH^1)+O	30.49	33.34	35.54	36.85	37.96
	SLF+WDS(HH^1)+P	30.6	33.37	35.61	36.82	37.95
Elaine	SLF+WDS(LH^1, HL^1, HH^1)+O	30.99	32.3	33.8	35.11	36.29
	SLF+WDS(LH^1, HL^1, HH^1)+P	31.09	32.31	33.84	35.12	36.37
Boat	SLF+WDS(HL^1)+O	28.88	32.32	36.17	38.68	40.52
	SLF+WDS(HL^1)+P	28.9	32.42	36.26	38.78	40.58
Couple	SLF+WDS(HL^1)+O	26.92	29.33	32.58	34.81	36.48
	SLF+WDS(HL^1)+P	26.92	29.39	32.6	34.85	36.63

Table 6-5. PSNR of the image coding schemes with the original and the new ZC context tables (directional filters = LLF).

Test image	Coding Shceme	0.125 bpp	0.25 bpp	0.5 bpp	0.75 bpp	1.0 bpp
Barbara	LLF+WDS(HL^1)+O	25.72	28.51	32.22	34.89	37.01
	LLF+WDS(HL^1)+P	25.86	28.71	32.41	34.96	37.11
Fingerprint	LLF+WDS(HL^1)+O	22.64	25.36	29.09	31.33	33.25
	LLF+WDS(HL^1)+P	22.64	25.52	29.09	31.41	33.26
Pepper	LLF+WDS(HH^1)+O	30.49	33.33	35.56	36.81	37.93
	LLF+WDS(HH^1)+P	30.6	33.37	35.62	36.9	38.07
Elaine	LLF+WDS(LH^1, HL^1, HH^1)+O	30.99	32.29	33.94	35.34	36.5
	LLF+WDS(LH^1, HL^1, HH^1)+P	31.09	32.33	34	35.38	36.53
Boat	LLF+WDS(HL^1)+O	28.81	32.28	36.13	38.6	40.46
	LLF+WDS(HL^1)+P	28.8	32.39	36.22	38.67	40.58
Couple	LLF+WDS(HL^1)+O	26.87	29.31	32.55	34.73	36.47
	LLF+WDS(HL^1)+P	26.93	29.37	32.56	34.79	36.53

Table 6-6. Run time of the image coding schemes with different ZC context tables.

Scheme	SLF+WDS(HL^1)+O (Barbara, Fingerprint, Boat, Couple, average)	SLF+WDS(HH^1)+O (Pepper)	SLF+WDS(LH^1, HL^1, HH^1)+O (Elaine)
Run Time	4.547 sec	4.550 sec	8.023 sec
Scheme	SLF+WDS(HL^1)+P (Barbara, Fingerprint, Boat, Couple, average)	SLF+WDS(HH^1)+P (Pepper)	SLF+WDS(LH^1, HL^1, HH^1)+P (Elaine)
Run Time	4.203 sec	4.177 sec	7.813 sec
Scheme	LLF+WDS(HL^1)+O (Barbara, Fingerprint, Boat, Couple, average)	LLF+WDS(HH^1)+O (Pepper)	LLF+WDS(LH^1, HL^1, HH^1)+O (Elaine)
Run Time	23.031 sec	23.026 sec	62.484 sec
Scheme	LLF+WDS(HL^1)+P (Barbara, Fingerprint, Boat, Couple, average)	LLF+WDS(HH^1)+P (Pepper)	LLF+WDS(LH^1, HL^1, HH^1)+P (Elaine)
Run Time	22.391 sec	22.386 sec	62.256 sec

6.1.4. Overall Improvement

At last, we compare the performance of the entire image coding scheme for three candidates: “LLF+NDS1+O” (the original WBCT image coding scheme), “NDF+NDS2+O” (the 2-D DWT image coding scheme) and “SLF+WDS+P” (our proposed WBCT image coding scheme). Fig. 6-5 shows the PSNR of these three coding schemes. Generally, our proposed “SLF+WDS+P” has better average PSNR than the “NDF+NDS2+O” and its average PSNR is comparable with that of

"LLF+NDS1+O". Table 6-7 shows their run time. Our proposed scheme "SLF+WDS+P" saves more than 92% computing time than "LLF+NDS1+O" (the original WBCT image coding scheme). On the other hand, it costs 67% extra computing time than "NDF+NDS2+O" (the 2-D DWT image coding scheme). Clearly, our proposed scheme offers a good balance between computational complexity and image visual quality.

Table 6-7. Average run time of the 2-D DWT scheme (NDF+NDS2+O), the original WBCT scheme (LLF+NDS1+O), and the proposed scheme with three new algorithms (SLF+WDS+P).

Scheme	SLF+WDS+P	LLF+NDS1+O	NDF+NDS2+O
Run Time	4.499 sec	62.469 sec	2.688 sec

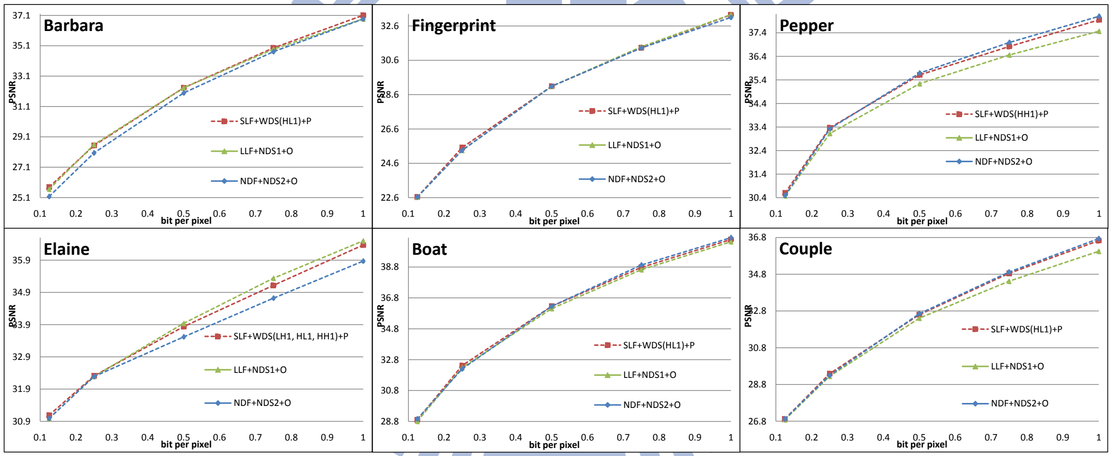


Fig. 6-5. PSNR of the 2-D DWT scheme (NDF+NDS2+O), the original WBCT scheme (LLF+NDS1+O), and the proposed scheme with three new algorithms (SLF+WDS+P).

6.2 Experimental Results of SMDT

We have developed three algorithms for enhancing the coding performance of DA-DWT and SA-DWT. First, the direction alignment algorithm for 2-D DA-DWT aligns the directions of 2-D DA-DWT based on a single subsampling pattern RC.

Then, an extended direction alignment algorithm for 2-D SA-DWT aligns the directions of 2-D SA-DWT based the double subsampling patterns, RC and CR. Finally, the proposed 2-D MSA-DWT improves the compression performance of 2-D SA-DWT on T_Hs by adaptively switching off the second transform performed on B_L.

In this section, we simulate the proposed schemes on images and videos. We first compare the prediction error and the side information of DA-DWT and SA-DWT with and without direction alignment algorithms. Then, we compare the coding performance with and without the direction alignment algorithms. Finally, we compare MSA-DWT and SA-DWT on T_Hs. Six test images are shown in Fig. 6-6 (512×512 image with 256 gray level). In addition, there are six MPEG test videos listed in Table 6-13 (CIF video with I420).

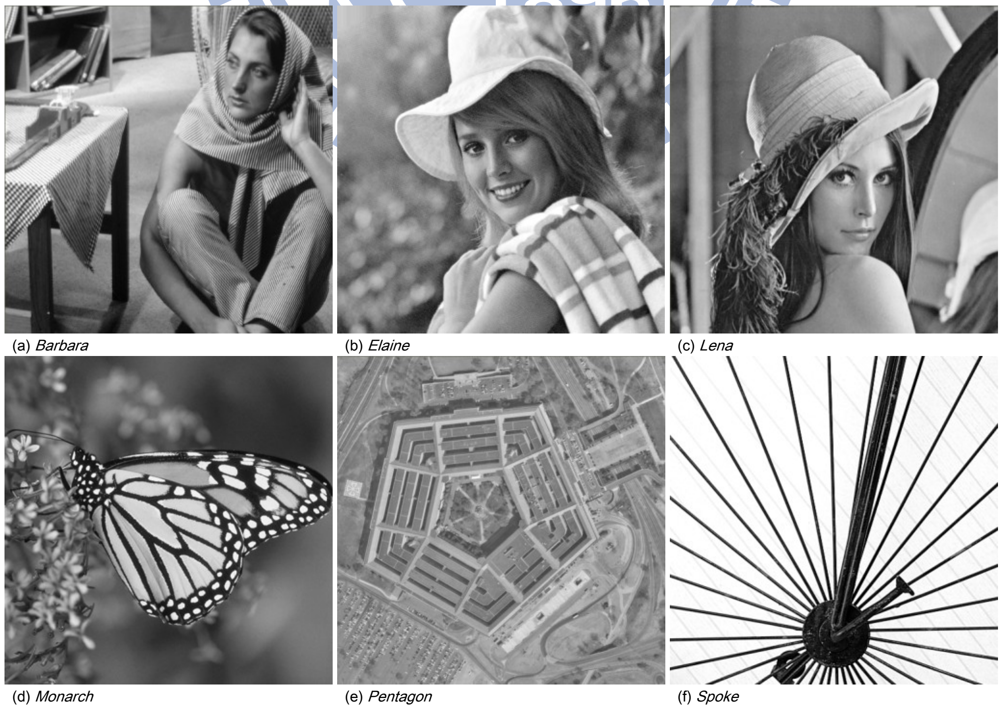


Fig. 6-6. The test images (512×512 image with 256 gray level).

6.2.1. Direction Alignment Algorithm for DA-DWT

We compare the prediction error (sum of absolute coefficients in the high pass subbands) and the side information of DA-DWT *with alignment* (DA-DWT-A) and *without alignment* (DA-DWT). DA-DWT chooses the best direction (with fixed subsampling pattern RC) for each partition block. We compare the side information coded by the quadtree partition (SI-QP) [28][52] and the megablocking partition (SI-MP) [15]. SI-QP includes the side information of quadtree partition and the direction of each block. Except these two pieces of side information, SI-MP includes one more piece of side information used to indicate the megablocking partition of each block. SI-MP codes the direction of each megablock instead of each block. A megablock is composed of many connected quadtree partition blocks of the same direction. Thus, each megablock contains only one direction. Therefore, large megablocks save the total amount of side information.

We adopt the coding method in [52] for encoding the quadtree partition. For a given block, the direction index predictor uses its left, up, and left-up blocks to predict the current block direction. The prediction difference (error) is then coded [28]. In the megablocking partition, we classify blocks into “inner blocks” and “boundary blocks”. Each block has 4 neighboring blocks [55]. The inner block has all its neighboring blocks inside the same megablock. On the other hand, a boundary block has at least one neighboring block from the other megablock [15]. We adopt the coding method in [15] for encoding the megablock partition. We then adopt the run length coding scheme to code these three pieces of side information [15]. Based on our data, we set $\lambda_{A1} = 8$, $\lambda_{A2} = 4$, and $\lambda_{A3} = 4$ for all images.

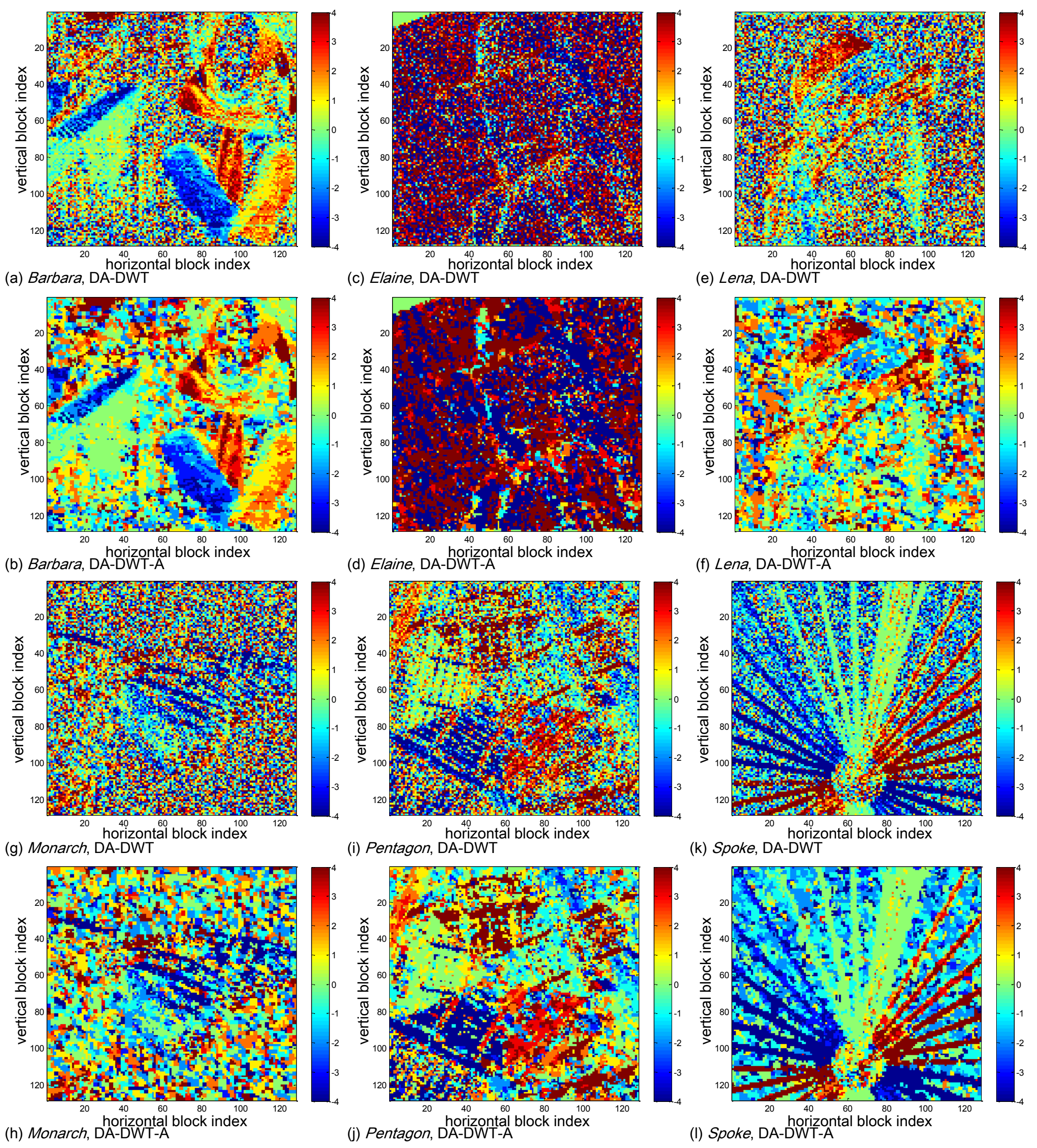


Fig. 6-7. Directions of the first transform after DA-DWT and DA-DWT-A (4×4 block). The direction indexes -4 ~ 4 are identical to d_v (-4 ~ 4) in Fig. 5-1(a).

Fig. 6-7 shows the first directions of several images after DA-DWT and

DA-DWT-A. DA-DWT-A aligns directions and creates large megablocks. Table 6-8 and Table 6-9 show the prediction error and side information of three schemes (DWT, DA-DWT, and DA-DWT-A). The Increment and Decrement are changes (in %) in distortion and side information with the direction alignment algorithm. The proposed alignment algorithm reduces about 38.90% in SI-QP at the cost of 1.76% increment in prediction error in average. DA-DWT creates many isolated blocks and results in large SI-MP in Table 6-9. The proposed alignment algorithm creates large megablocks and saves about 69.10% in SI-MP.

Table 6-8. Prediction errors of the first transform.

	DWT	DA-DWT	DA-DWT-A	Increment
Barbara	512922.314 (100%)	329705.753 (64.280%)	338734.072 (66.040%)	+1.760%
Elaine	672054.724 (100%)	494316.594 (73.553%)	504065.074 (75.004%)	+1.451%
Lena	332228.596 (100%)	272420.695 (81.998%)	283420.955 (85.309%)	+3.311%
Monarch	405322.749 (100%)	301468.907 (73.378%)	310280.483 (76.552%)	+2.174%
Pentagon	591543.922 (100%)	450194.050 (76.105%)	459085.593 (77.608%)	+1.503%
Spoke	907817.169 (100%)	544294.268 (59.956%)	551541.042 (60.755%)	+0.798%

Table 6-9. Side Information in bits of the first transform using two side information coding schemes.

	Quadtree Partition			Megablocking Partition		
	DA-DWT	DA-DWT-A	Decrement	DA-DWT	DA-DWT-A	Decrement
Barbara	23695 (100%)	14192 (59.895%)	-40.105%	29858 (100%)	9057 (30.333%)	-69.667%
Elaine	29409 (100%)	18814 (63.974%)	-36.026%	38096 (100%)	13258 (34.802%)	-65.198%
Lena	27629 (100%)	15973 (57.812%)	-42.188%	34832 (100%)	10616 (30.478%)	-69.522%
Monarch	27665 (100%)	16983 (61.388%)	-38.612%	35379 (100%)	11707 (33.090%)	-66.910%
Pentagon	26039 (100%)	16911 (64.945%)	-35.055%	35112 (100%)	13439 (38.275%)	-61.725%
Spoke	24956 (100%)	14626 (58.607%)	-41.393%	30802 (100%)	9364 (30.401%)	-69.599%

6.2.2. Direction Alignment Algorithm for SA-DWT

We compare the prediction error and the side information of SA-DWT with alignment (SA-DWT-A) and without alignment (SA-DWT). SA-DWT chooses the best direction among d_v and d_h in Fig. 5-1 for each partition block. That is, both direction and subsampling pattern are selected in the first transform. We also implement and compare two side-information coding schemes, SI-QP and SI-MP. For sending the side information of subsampling pattern, we use one more bit for each block (or megablock) in SI-QP (or SI-MP). We also code this subsampling information by run length coding. Again, we set $\lambda_{B2} = 8$, $\lambda_{B3} = 4$, and $\lambda_{B4} = 4$.

Fig. 6-8 shows the first directions of test images after SA-DWT and SA-DWT-A. The alignment algorithm in SA-DWT-A (described in section III) aligns the directions and the subsampling patterns. Table 6-10 and Table 6-11 show the prediction errors and the side information of different schemes. The Increment and Decrement are change (in %) in distortion and side information with direction alignment algorithm. Comparing to DA-DWT, the SA-DWT offers smaller prediction errors at the cost of more side information. The proposed alignment algorithm increases about 1.64% in prediction error but saves about 34.71% in SI-QP (quadtree representation) in average. Similar to DA-DWT, the SA-DWT has larger SI-MP (megablock) than SA-DWT-A due to many isolated blocks. The proposed direction alignment algorithm reduces about 64.33% in SI-MP in average. So far, our implementation of SA-DWT shows some advantages in perdition errors over DA-DWT but has disadvantage in the side information bit rate.

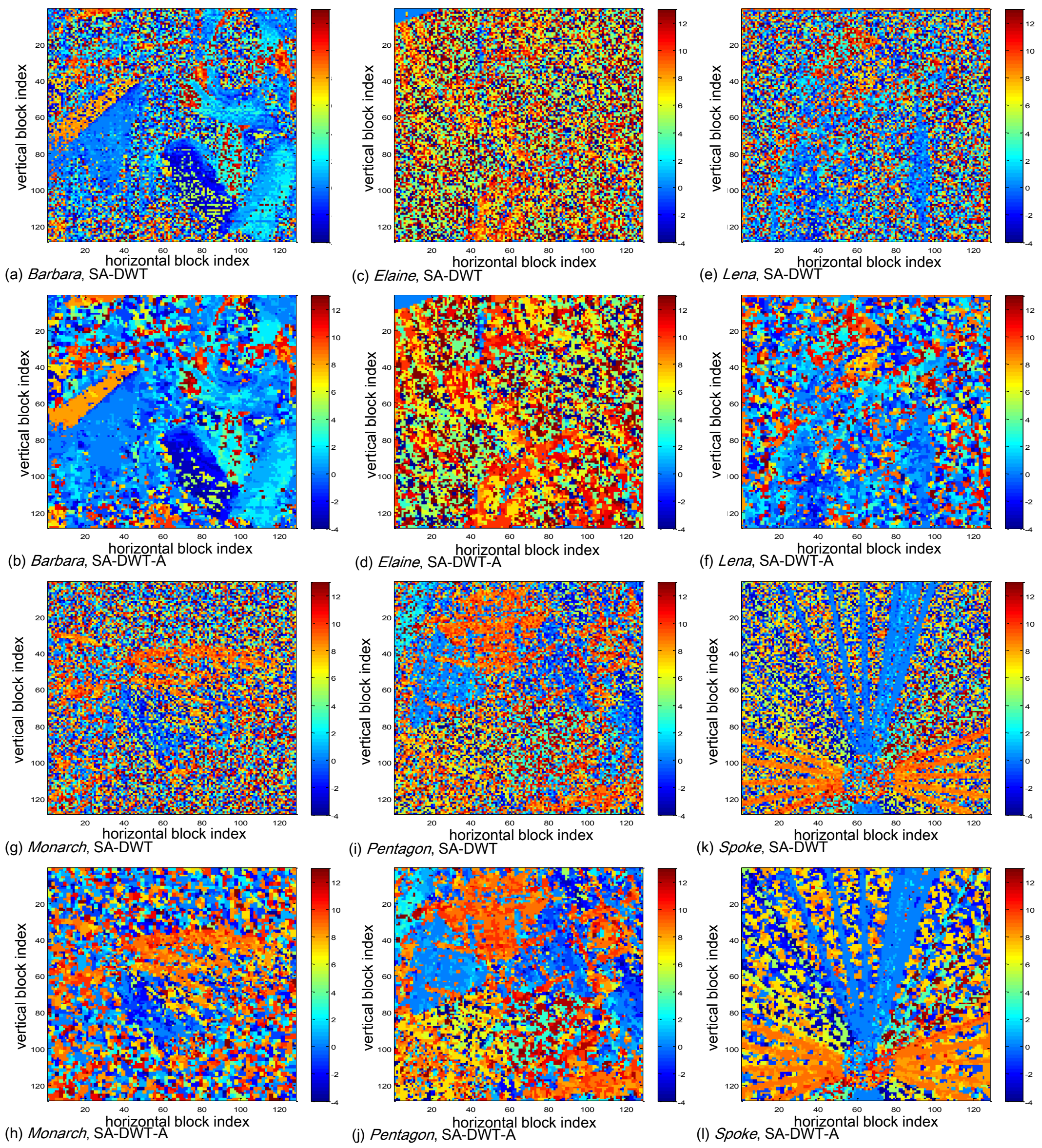


Fig. 6-8. Directions of the first transform after SA-DWT and SA-DWT-A (4×4 block). The direction indexes -4 ~ 4 correspond to d_v' (-4 ~ 4) and 5 ~ 13 correspond to d_h (-4 ~ 4).

Table 6-10. Prediction errors of the first transform.

	DWT	SA-DWT	SA-DWT-A	Increment
Barbara	512922.314 (100%)	327553.509 (63.860%)	327678.521 (63.885%)	+0.024%
Elaine	672054.724 (100%)	430522.407 (72.659%)	445250.739 (75.145%)	+2.486%
Lena	332228.596 (100%)	246194.604 (74.104%)	258235.258 (77.728%)	+3.624%
Monarch	405322.749 (100%)	229335.836 (61.360%)	239349.809 (64.039%)	+2.679%
Pentagon	591543.922 (100%)	354943.377 (60.003%)	363918.102 (61.520%)	+1.517%
Spoke	907817.169 (100%)	343166.137 (37.801%)	338832.988 (37.324%)	-0.477%

Table 6-11. Side information in bits of the first transform using two side information coding schemes.

	Quadtree Partition			Megablocking Partition		
	SA-DWT	SA-DWT-A	Decrement	SA-DWT	SA-DWT-A	Decrement
Barbara	30594 (100%)	18302 (59.822%)	-40.178%	37875 (100%)	11021 (29.098%)	-70.902%
Elaine	36684 (100%)	25352 (69.109%)	-30.891%	44915 (100%)	20871 (46.468%)	-53.532%
Lena	34719 (100%)	23142 (66.655%)	-33.345%	41782 (100%)	14968 (35.824%)	-64.716%
Monarch	34423 (100%)	22220 (64.550%)	-34.450%	41795 (100%)	13621 (32.590%)	-67.410%
Pentagon	31077 (100%)	21458 (69.048%)	-30.952%	40215 (100%)	15404 (38.304%)	-61.696%
Spoke	31795 (100%)	19547 (61.478%)	-38.520%	36925 (100%)	11927 (32.301%)	-67.699%

6.2.3. Image Coding

We compare the coding performance of three wavelet schemes. Scheme 1 is three level 2-D DWT (2-D DWT × 3). Scheme 2 is one level 2-D SA-DWT together with two level 2-D DA-DWT (2-D SA-DWT × 1 + 2-D DA-DWT × 2). Scheme 3 is one level 2-D SA-DWT-A together with two level 2-D DA-DWT-A (2-D SA-DWT-A × 1 + 2-D DA-DWT-A × 2). In the first level transform of last two schemes, we found that SA-DWT is able to compact more energy into the low-pass subband.

The SA-DWT adopts the phase-completion process to implement the lifting scheme in the transition between two different neighboring subsampling patterns [32].

It estimates the missing pixels from the neighboring pixels with the same subsampling phase. This may lead to mismatch and boundary effect problem and thus reduce the coding performance [29]. Therefore, when the mismatch problem becomes more serious at lower resolutions (higher transform levels), we adopt DA-DWT for the second and the third level transforms. We use variable block sizes from 4×4 to 128×128 to partition images. We set $\lambda_{A1} = 8$, $\lambda_{A2} = 4$, $\lambda_{A3} = 4$, $\lambda_{B2} = 8$, $\lambda_{B3} = 4$, and $\lambda_{B4} = 4$ for the direction alignment algorithm at different levels. We set $\lambda_t = 12$ for the quadtree combination. We code the transformed coefficients by EBCOT [35] and the side information by SI-MP (megablock). Table 6-12 shows the coding results.

Table 6-12. PSNR of different coding schemes.

Test Image	Transform Schemes	0.125 bpp	0.25 bpp	0.5 bpp	0.75 bpp	1.0 bpp
Barbara	2-D DWT × 3	25.26	28.25	32.10	34.77	36.98
	2-D SA-DWT × 1 + 2-D DA-DWT × 2	26.38	29.68	33.58	35.72	37.82
	2-D SA-DWT-A × 1 + 2-D DA-DWT-A × 2	26.64	29.91	33.64	35.85	37.86
Elaine	2-D DWT × 3	31.01	32.25	33.55	34.70	35.87
	2-D SA-DWT × 1 + 2-D DA-DWT × 2	30.84	32.43	33.69	34.88	36.03
	2-D SA-DWT-A × 1 + 2-D DA-DWT-A × 2	31.20	32.39	33.73	34.95	36.07
Lena	2-D DWT × 3	30.66	33.82	37.00	38.79	40.05
	2-D SA-DWT × 1 + 2-D DA-DWT × 2	30.84	34.33	37.37	38.97	40.06
	2-D SA-DWT-A × 1 + 2-D DA-DWT-A × 2	31.17	34.41	37.42	39.00	40.17
Monarch	2-D DWT × 3	27.09	30.35	35.55	39.08	41.69
	2-D SA-DWT × 1 + 2-D DA-DWT × 2	26.89	30.93	35.80	39.23	41.61
	2-D SA-DWT-A × 1 + 2-D DA-DWT-A × 2	27.18	31.03	35.92	39.27	41.67
Pentagon	2-D DWT × 3	26.95	28.66	31.36	33.39	35.07
	2-D SA-DWT × 1 + 2-D DA-DWT × 2	26.73	28.97	31.78	33.95	35.54
	2-D SA-DWT-A × 1 + 2-D DA-DWT-A × 2	27.06	29.17	31.85	33.97	35.63
Spoke	2-D DWT × 3	20.57	23.62	28.89	32.36	35.14
	2-D SA-DWT × 1 + 2-D DA-DWT × 2	22.26	27.02	31.90	35.04	37.07
	2-D SA-DWT-A × 1 + 2-D DA-DWT-A × 2	22.72	27.35	32.12	35.14	37.14

Table 6-12 shows the PSNR of three coding schemes. "bpp" means bit per pixel in bit rate. "2-D SA-DWT × 1 + 2-D DA-DWT × 2" sometimes has the lowest PSNR particularly at low bit rates due its huge side information. "2-D SA-DWT-A × 1 + 2-D

DA-DWT-A × 2" needs fewer side information bits and outperforms "3 DWT" at all bit rates. "2-D SA-DWT-A × 1 + 2-D DA-DWT-A × 2" also outperforms "2-D SA-DWT × 1 + 2-D DA-DWT × 2", especially at low bit rates. The PSNR gain is about 0.3dB~0.5dB at low bit rates.

6.2.4. Video Coding

As said in Section I, we adopted the interfarme wavelet structure [7] for our video codec, in which MCTF [8]-[13] decomposes video frames into T_Ls and T_Hs. As discussed in section IV, because T_Ls and T_Hs have different signal characteristics [43], we apply 2-D DA-DWT to T_Ls and apply 2-D MSA-DWT to T_Hs.

We adopt four-level MCTF for temporal transform and it generates 1 T_L subband and 15 T_H subbands (residuals) from 16 video frames [45]. Then, two-level spatial transforms are applied to each temporal subbands (residuals). We design four coding schemes using different combinations of spatial transforms. Scheme 1 applies two-level 2-D DWT to all residuals (T_L(2-D DWT × 2), T_H(2-D DWT × 2)). Scheme 2 applies one-level 2-D SA-DWT together with one-level 2-D DA-DWT to T_Ls and two-level DWT to T_Hs (T_L(2-D SA-DWT × 1 + 2-D DA-DWT × 1), T_H(2-D DWT × 2)). For similar reasons as discussed earlier, we use 2-D DA-DWT instead of 2-D SA-DWT for the second-level transform to avoid the mismatch problem on T_Ls. Scheme 3 use the two-level 2-D SA-DWT on T_Hs (T_L(2-D SA-DWT × 1 + 2-D DA-DWT × 1), T_H(2-D SA-DWT × 2)). Scheme 4 is similar to Scheme 3 except that it adopts the two-level MSA-DWT on T_Hs (T_L(2-D SA-DWT × 1 + 2-D DA-DWT × 1), T_H(2-D MSA-DWT × 2)). T_H subband usually composes of uniform small-coefficient smooth regions and large-coefficient (prediction error) edge regions [43]. The small-coefficient regions help in reducing the mismatch problem in 2-D SA-DWT because they are close to zero. Thus, in Schemes 3 and 4, we employ 2-D SA-DWT and 2-D MSA-DWT for the second-level spatial transform to reduce coefficients spreading.

For 2-D DA-DWT, 2-D SA-DWT and 2-D MSA-DWT, we partition the images into blocks with size 4×4 and choose the best direction for each block. We assume the side information of these spatial transforms are all 0. The test video sequences are all CIF format, I420, and 30 fps. We compress 32 frames of different test video sequences and code the transform coefficients by 3-D ESCOT [36]. Table 6-13 shows the coding results of different test video sequences. "kpbs" means 1024 bits (kilobits) per second in bit rate.

MCTF concentrates most energy into T_Ls. Scheme 2 compresses T_Ls well and often outperforms Scheme 1 significantly in Table 6-13. In MCTF, T_Ls and T_Hs compose of “the same” and “different” components of video sequences. For video sequences with fast moving objects, such as *Stefan*, T_Ls contain less “the same” components. The coding gain of scheme 2 is about 0.1dB for *Stefan*.

As discussed in section VI, in the second transform, 2-D DA-DWT and 2-D SA-DWT could spread out coefficients in T_Hs. The 2-D MSA-DWT can switch off the second transform and thus provides better coding performance than 2-D SA-DWT on T_Hs in Table 6-14.

Table 6-13. PSNR of different coding schemes on T_Ls.

Test Video Sequence	Spatial Transform Schemes	128 kbps	256 kbps	512 kbps	1024 kbps	2048 kbps
Akiyo	T_L(2-D DWT × 2), T_H(2-D DWT × 2)	41.11	44.43	47.28	50.33	53.60
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D DWT × 2)	41.81	44.85	47.48	50.63	53.83
Bus	T_L(2-D DWT × 2), T_H(2-D DWT × 2)	26.37	29.78	32.62	35.74	39.33
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D DWT × 2)	26.61	30.01	32.86	35.78	39.34
Foreman	T_L(2-D DWT × 2), T_H(2-D DWT × 2)	33.64	36.61	39.29	41.76	44.25
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D DWT × 2)	34.33	36.91	39.39	41.93	44.41
Mobile	T_L(2-D DWT × 2), T_H(2-D DWT × 2)	24.53	27.77	31.03	34.07	37.63
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D DWT × 2)	25.22	28.23	31.24	34.09	37.56
News	T_L(2-D DWT × 2), T_H(2-D DWT × 2)	35.33	39.08	43.10	46.70	50.14
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D DWT × 2)	35.50	39.29	43.12	46.86	50.34
Stefan	T_L(2-D DWT × 2), T_H(2-D DWT × 2)	25.56	29.33	32.78	35.77	39.02
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D DWT × 2)	25.73	29.45	32.85	35.71	39.00

Table 6-14. PSNR of different coding schemes on T_Hs.

Test Video Sequence	Spatial Transform Schemes	128 kbps	256 kbps	512 kbps	1024 kbps	2048 kbps
Akiyo	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D DWT × 2)	41.81	44.85	47.48	50.63	53.83
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D SA-DWT × 2)	41.84	44.88	47.54	50.65	53.80
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D MSA-DWT × 2)	41.87	45.08	47.61	50.74	53.82
Bus	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D DWT × 2)	26.61	30.01	32.86	35.78	39.34
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D SA-DWT × 2)	26.61	30.02	32.85	35.72	39.32
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D MSA-DWT × 2)	26.64	30.13	33.10	35.90	39.38
Foreman	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D DWT × 2)	34.33	36.91	39.39	41.93	44.41
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D SA-DWT × 2)	34.40	37.01	39.54	42.01	44.39
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D MSA-DWT × 2)	34.45	37.15	39.65	42.11	44.46
Mobile	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D DWT × 2)	25.22	28.23	31.24	34.09	37.56
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D SA-DWT × 2)	25.25	28.20	31.21	34.02	37.47
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D MSA-DWT × 2)	25.26	28.24	31.33	34.27	37.67
News	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D DWT × 2)	35.50	39.29	43.12	46.86	50.34
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D SA-DWT × 2)	35.59	39.41	43.31	46.80	50.34
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D MSA-DWT × 2)	35.64	39.46	43.42	46.87	50.37
Stefan	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D DWT × 2)	25.73	29.45	32.85	35.71	39.00
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D SA-DWT × 2)	25.70	29.50	32.88	35.60	39.03
	T_L(2-D SA-DWT × 1+ 2-D DA-DWT × 1), T_H(2-D MSA-DWT × 2)	25.77	29.61	33.02	35.75	39.15

Chapter 7 Conclusions

In this thesis, we study and improve two types of popular directional wavelet-based image and video coding schemes. We propose three enhanced algorithms to improve the coding performance of *wavelet-based contourlet transform* (WBCT) on image coding. We propose another three enhanced algorithms to improve the coding performance of *direction-adaptive discrete wavelet transform* (DA-DWT) on images and video coding.

The WBCT-based image coding approach is explored in this thesis. We propose three components to enhance its performance. First, we design a short-length filters (SLF) to speed up the filtering process. It provides similar coding performance but requires only 10% of computational complexity of the original long-length filters (LLF). Second, we construct a mean-shift-based decision process to decide if a higher subband (HH^1, HL^1, or LH^1) is appropriate for directional decomposition. Threshold values are carefully selected to identify the energy peaks in each candidate subband. Finally, we design new zero-coding (ZC) context tables for ESCOT because the coefficients produced by directional decomposition have different statistical characteristics among near-by coefficients. Compared with the conventional 2-D DWT coding scheme, our scheme provides better visual quality with a moderate additional computational cost. Compared with the original WBCT coding scheme, the proposed coding scheme provides comparable image quality (PSNR and MSSIM) but with significantly less computing time.

We further study the DA-DWT approach. We propose three algorithms to enhance the coding performance of 2-D DA-DWT and 2-D SA-DWT. We first propose a direction alignment algorithm to reduce the side information of 2-D DA-DWT. We then extend the direction alignment algorithms to reduce the side information of 2-D SA-DWT. This extension requires quite a bit of extra work to reduce complexity in the selection process. The proposed alignment algorithms save a large amount of side information at the cost of small increment in prediction error. Overall, it also improves the coding performance on still images. To encode the temporal high-pass bands (T_H) more efficiently, we propose an adaptive switching algorithm that turns off the second transform in 2-D SA-DWT. This so-called 2-D MSA-DWT provides better coding efficiency than 2-D SA-DWT and 2-D DWT on T_Hs.

References

[1] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using vector quantization in the wavelet transform domain," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 1990, Albuquerque, NM, USA, vol. 4, pp. 2297–2300, Apr. 1990.

[2] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Process.*, vol. 1, no. 2, pp. 205–220, Apr. 1992.

[3] A. S. Lewis and G. Knowles, "Image compression using the 2-D wavelet transform," *IEEE Trans. Image Process.*, vol. 1, no. 2, pp. 244–250, Apr. 1992.

[4] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell..*, vol. 11, no. 7, pp. 674–69., July 1989.

[5] M. Vetterli, "Wavelets, approximation and compression," *IEEE Signal Proc. Mag.*, vol. 18, no. 5, pp. 59–73, Sep. 2001.

[6] D. Taubman, and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards, and Practice.* Norwell, MA: Kluwer, 2002.

[7] R. Xiong, X. Ji, D. Zhang, and J. Xu, "Vidwav wavelet video coding specifications," ISO/IEC JTC1/SC29/WG11 MPEG, M12339, 2005.

[8] J. R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Trans. Image Process.*, vol. 3, no. 9, pp. 559–571, Sep. 1994.

[9] J.-R. Ohm, M. van der Schaar, and J. W. Woods, "Interframe wavelet coding - motion picture representation for universal scalability," *Signal Processing: Image Communications*, vol. 19, no. 9, pp. 877–908, Oct. 2004.

[10] B. Pesquet-Popescu and V. Bottreau, "Three-dimensional lifting schemes for motion compensated video compression," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1793–1796, Salt Lake City, Utah, USA, May 2001.

[11] L. Luo, F. Wu, S. Li, Z. Xiong, and Z. Zhuang, "Advanced motion threading for 3-D wavelet video coding," *Signal Processing: Image Communication.*, vol. 19, no. 7, pp. 601–616, Aug. 2004.

[12] A. Secker and D. Taubman, "Lifting based invertible motion adaptive transform (LIMAT) framework for highly scalable video compression," *IEEE Trans. Image Process.*, vol. 12, no. 12, pp. 1530–1542, Dec. 2003.

[13] R. Xiong, J. Xu, F. Wu, and S. Li, "Barbell-lifting based 3-D wavelet coding scheme," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1256–1269, Sep. 2007.

[14] M. N. Do, "Directional Multiresolution Image Representations," Ph.D. Dissertation, Swiss Fed. Inst. Technol., Lausanne, Switzerland, Nov. 2001.

[15] A. Maleki, B. Rajaei, and H. R. Pourreza, "Rate-distortion analysis of directional wavelets," *IEEE Trans. Image Process.*, vol. 21, no. 2, pp. 588–600, Feb. 2012.

[16] D. Taubman and A. Zakhor, "Orientation adaptive subband coding of images," *IEEE Trans.*

Image Process., vol. 3, no. 4, pp. 421–437, Apr. 1994.

[17] M. N. Do, and M. Vetterli, "The contourlet transform: an efficient directional decomposition multiresolution image representation," *IEEE Trans. Image Processing.* vol. 14, no. 12, pp. 2091–2106, Dec. 2005.

[18] Y. Lu and M. N. Do, "The finer directional wavelet transform," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Philadelphia, PA, USA, March 2005.

[19] T. T. Nguyen and S. Oraintara, "A class of multiresolution directional filter bank," *IEEE Trans. Signal Process*, vol. 55, no. 3, pp. 949–961, Mar. 2007.

[20] T. T. Nguyen and S. Oraintara, "Multiresolution directional filterbanks: theory, design, and applications," *IEEE Trans. Signal Process*, vol. 53, no. 10, pp. 3895–3905, Oct. 2005.

[21] I. W. Selesnick, R. G. Baraniuk, and N. G. Kingsbury, "The dual-tree complex wavelet transform," *IEEE Signal Process. Mag.*, vol. 22, no. 6, pp.123–151, Nov. 2005.

[22] R. Eslami and H. Radha, "Wavelet-based contourlet transform and its application to image coding," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, vol. 5, pp. 3189–3192, Singapore, Oct. 2004.

[23] R. Eslami and H. Radha, "A new family of nonredundant transforms using hybrid wavelets and directional filter banks," *IEEE Trans. Image Processing*, vol. 16, no. 4, pp. 1152–1167, Apr. 2007.

[24] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Trans. Commun.*, vol. 31, no. 4, pp. 532–540, April 1983.

[25] R. H. Bamberger and M. J. T. Smith, "A filter bank for the directional decomposition of images: Theory and design," *IEEE Trans. Signal Process.*, vol. 40, no. 4, pp. 882–893, Apr. 1992.

[26] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, no. 3, pp. 247–269, 1998.

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

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

[29] Y. Liu and K. N. Ngan, "Weighted adaptive lifting-based wavelet transform for image coding," *IEEE Trans. Image Processing,* vol. 17, no. 4, pp. 500–511, Apr. 2008.

[30] W. Dong, G. Shi, and J. Xu, "Adaptive nonseparable interpolation for image compression with directional wavelet transform," *IEEE Signal Processing Letters*, vol. 15, pp. 233–236, 2008.

[31] C. -L. Chang, A. Maleki, and B. Girod, "Adaptive wavelet transform for image compression via directional quincunx lifting," in *Proc. IEEE Workshop Multimedia Signal Processing*, Shanghai, China, Oct. 2005.

[32] J. Xu and F. Wu, "Subsampling-adaptive directional wavelet transform for image coding," in *IEEE Data Compression Conference*, pp. 89–98. Mar. 2010.

[33] A. Said and W. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, June 1996.

[34] J. M. Shapiro, “An embedded hierarchical image coder using zerotrees of wavelet coefficients,” in *IEEE Data Compression Conf.*, Snowbird, UT, 1993, pp. 214–223.

[35] D. Taubman, “High performance scalable image compression with EBCOT,” *IEEE Trans. Image Processing,* vol. 9, no. 7, pp. 1158–1170, Jul. 2000.

[36] J. Xu, Z. Xiong, S. Li, Y, Zhang, “Three-dimensional embedded subband coding with optimized truncation (3D ESCOT)”, *Applied and Computational Harmonic Analysis: Special Issue on Wavelet Applications in Engineering*, vol. 10, no. 3, pp. 290–315, May 2001.

[37] C. Tian and S. S. Hemami, “An embedded image coding system based on tarp filter with classification,” in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Process*, vol. 3, pp. 49–52, Montreal, QC, Canada, May 2004.

[38] X. Wu, “High-order context modeling and embedded conditional entropy coding of wavelet coefficients for image compression,” in *Proc. 31st Asilomar Conf. Signals, Systems, Computers*, pp. 1378–1382, Nov. 1997.

[39] S.-T. Hsiang and J. W. Woods, “Embedded image coding using zeroblocks of subband/wavelet coefficients and context modeling,” in *IEEE Int. Conf. Circuits and Systems (ISCAS)*, vol. 3, pp. 662–665, May 2000,

[40] C.-H. Hung and H.-M. Hang, “Image coding using short wavelet-based contourlet transform,” *IEEE International Conference on Image Proc.(ICIP),* San Diego, CA, USA, Oct. 2008.

[41] C.-H. Hung and H.-M. Hang, “Decision-directed adaptive wavelet image coding with directional decomposition,” *IEEE International Symposium on Circuits and Systems (ISCAS),* Taipei, Taiwan, May, 2009.

[42] Y. Tanaka, M. Hasegawa, S. Kato, M. Ikehara, and T.Q. Nguyen, “Adaptive Directional Wavelet Transform Based on Directional Prefiltering,” *IEEE Trans. Image Processing.*, vol. 19, no. 4, pp. 934–945, April. 2010.

[43] F. Kamisli and J. S. Lim, "1-D transforms for the motion compensation residual," *IEEE Trans. Image Processing.*, vol.20, no.4, pp.1036–1046, Apr. 2011.

[44] F. Kamisli and J. S. Lim, "Directional wavelet transforms for prediction residuals in video coding," *IEEE International Conference on Image Proc.(ICIP),* Cairo, Egypt, Nov. 2009.

[45] C.-H. Hung and H.-M. Hang, “Scalable video coding using adaptive directional lifting-based wavelet transform,” in *Asian-Pacific Signal and Inform. Proc. Association Annual Summit and Conf. (APSIPA ASC),* Xi’an, China, Oct. 2011.

[46] C.-H. Hung and H.-M. Hang, “Direction alignment algorithm for direction-adaptive discrete wavelet transform,” in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process. (ICASSP)*, Kyoto, Japan, Mar. 2012.

[47] A. Cohen, I. Daubechies, and J.-C. Feauveau, “Biorthogonal bases of compactly supported wavelets,” *Commun. Pure Appl. Math.*, vol. 45, pp. 485–560, 1992.

[48] R. Eslami and H. Radha, “On low bit-rate coding using the contourlet transform,” in *Proc. Asilomar Conf. Signals, Systems, and Computers*, pp. 1524–1528, Pacific Grove, CA, Nov. 2003.

[49] S Parrilli, L Verdoliva, G Poggi, "A SPIHT-LIKE image coder based on the contourlet transform," *IEEE International Conference on Image Proc.(ICIP),* San Diego, CA, USA, Oct. 2008.

[50] T. Wiegand and B. Girod, "Lagrange multiplier selection in hybrid video coder control," in *EEE International Conference on Image Proc.(ICIP).*, vol. 3, pp. 542–545, Thessaloniki, Greece, Oct. 2001.

[51] W. Ding, F. Wu, and S. Li, "Lifting-based wavelet transform with directionally spatial prediction," in *Picture Coding Symp*. (PCS), vol. 62, pp. 291–294, San Francisco, CA, Dec. 2004.

[52] G. J. Sillivan and R. L. Baker, "Efficient Quadtree Coding of Images and Video," *IEEE Trans. Image Processing*, vol. 3, no. 3, pp. 327–331, May 1994.

[53] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 688–703, July 2003.

[54] T. Xu, C.-L. Chang, and B. Girod, "Scalable direction representation for image compression with direction-adaptive discrete wavelet transform," in *Proc. Visual Communication and Image Processing*, San Jose, CA, Jan. 2007.

[55] R. Shukla, P. L. Dragotti, M. N. Do, and M. Vetterli, "Rate-Distortion Optimizes Tree-Structured Compression Algorithm for Piecewise Polynomial," *IEEE Trans. Image Processing.,* vol. 14, no. 3, pp. 343–359, Mar. 2005.

[56] X. Peng, J. Xu, and F. Wu, "Directional filtering transform for image/intra-frame compression," *IEEE Trans. Image Processing*, vol. 19, no. 11, pp. 2935–2946, Nov. 2010.

[57] A. Gouze, M. Antonini, M. Barlaud and B. Macq, "Design of signal-adapted multidimensional lifting scheme for lossy coding," *IEEE Trans. on Image Processing,* vol. 13, no. 12, pp. 1589–1603, Dec 2004.

[58] M. Vetterli. Multidimensional subband coding: Some theory and algorithms. *Signal Proc.*, vol. 6, no. 2, pp. 97–112, Apr. 1984.

[59] S.-M. Phoong, C. W. Kim, P. P. Vaidyanathan, and R. Ansari, "A new class of two-channel biorthogonal filter banks and wavelet bases," *IEEE Trans. Signal Process*., vol. 43, no. 3, pp. 649–665, Mar. 1995.

[60] A. V. Oppenheim R. W. Schaffer *Discrete-Time Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1989.

[61] Y. P. Lin and P. P. Vaidyanathan, "Theory and design of two-dimensional filter banks: a review," *Multidimensional System and Signal Proc.*, vol. 7, no. 3-4, pp. 263–330, July 1996.

[62] Y. Cheng, "Mean Shift, Mode Seeking, and Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, Aug. 1995.

[63] D. Comaniciu and P. Meer, "Mean shift: A Robust Approach toward Feature Space Analysis," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 24, no. 5, pp. 603–619, May 2002.

[64] D. Comaniciu and P. Meer, "Mean Shift Analysis and Applications," *Proc. IEEE Int'l Conf. Computer Vision*, pp. 1197–1203, 1999.

[65] P. S. Chen and J. W. Woods, "Improved MC-EZBC with quarter-pixel motion vectors", ISO/IEC

JTC1/SC29/WG11 doc. No. m8366, FairFax, VA, May 2002

[66] A. Alecu, A. Munteanu, A. Pizurica, J. Cornelis, and P. Schelkens, “On hybrid directional transform-based intra-band image coding,” in Advanced Concepts for Intelligent Vision Systems, Delft, Netherlands, 2007.

[67] S.-M. Phoong, C. W. Kim, P. P. Vaidyanathan, and R. Ansari, “A new class of two-channel biorthogonal filter banks and wavelet bases,” *IEEE Trans. Signal Process*., vol. 43, no. 3, pp. 649–665, Mar. 1995.

[68] Z. Liu and L. Karam, “Mutual information-based analysis of JPEG2000 contexts,” *IEEE Trans. Image Processing*, vol. 14, no. 4, pp. 411.422, Apr. 2005.

[69] T. T. Nguyen and S. Oraintara, “On the aliasing effect of the contourlet filter banks,” *IEEE International Symposium on Circuits and Systems (ISCAS)*, Island of Kos, Greece, May 2006.

[70] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error measurement to structural similarity,” *IEEE Trans. Image Process*., vol. 13, no. 4, pp. 600–612, Apr. 2004.

Personal Resume

Contact Information
Name: Chao-Hsiung Hung (洪朝雄) EMAIL: hongmorning@gmail.com Address: Department of Electronics Engineering National Chiao Tung University 1001 Ta-Hsueh Road Hsinchu, Taiwan 30050, R.O.C. Lab: Communication Electronics & Signal Processing Laboratory

Education		
Ph. D.	University: National Chiao Tung University Department: Department of Electronics Engineering & Institute of Electronics Dissertation: Directional Wavelet-based Image and Video Coding Advisor: Prof. Hsueh-Ming Hang	Sep. 2005 ~ Sep. 2012
M. S.	University: National Chiao Tung University Department: Department of Electronics Engineering & Institute of Electronics Dissertation: HVS-based Rate Control Algorithm for Interframe Wavelet Video Coding Advisor: Prof. Hsueh-Ming Hang	Sep. 2003 ~ Jun. 2005
B. S.	University: National Chiao Tung University & Institute of Electronics Department: Department of Electronics Engineering	Sep. 1999 ~ Jun. 2003

Academic Activity
1. Journal Reviewer, *IEEE Trans. On Image Processing*. 2. IEEE Student Membership. 3. Visiting Scholar, Coordinated Science Laboratory (CSL), University of Illinois at Urbana- Champaign (UIUC), U.S., Jul. 2007. 4. Visiting Scholar, Coordinated Science Laboratory (CSL), University of Illinois at Urbana- Champaign (UIUC), U.S., Jul. 2008.

Teaching Assistant
1. Signal and Systems (Sep. 1999 ~ Jan. 2000，Feb. 2007 ~ Jun. 2007).
2. Source Coding (Sep. 2005 ~ Jan. 2006).
3. Digital Communication (Feb. 2011 ~ Jun. 2011, Feb. 2012 ~ Jun. 2012).
4. Channel Coding (Sep. 2011 年 ~ Jan. 2012).
5. LEE and MTI Science and Technology Forum (Feb. 2005 ~ Jan. 2009).

Working Experience
1. Project consultant, Ambarella Co., Hsinchu, Sep. 2007 ~ Feb. 2009 (Part-time).

Honor & Award
1. Ph.D. student scholarship, 2005-2007, awarded by Dept. of E. E., NCTU.

Publication Papers

Journal Papers

1. C. –H. Hung and H. –M. Hang, “An Image Coding Scheme Using Decision-directed Wavelet-based Contourlet Transform,” *Journal of Visual Communications and Image Representation*, vol. 23, no. 7, pp. 1128–1143, Oct. 2012.
2. C. –H. Hung and H. –M. Hang, "Direction-Aligned Adaptive Wavelet Transform with Applications to Image and Video Coding,” submitted to *IEEE Trans. Image Process*.

Conference Paper

1. C. –H. Hung and H. –M. Hang, “HVS-Based Rate Control Algorithm for Wavelet Image Coding,” *IPPR Conference on Computer Vision, Graphics and Image Processing (CVGIP)* , Taipei, Taiwan, Aug. 2005.
2. C. –H. Hung and H. –M. Hang, “Image Coding Using Short Wavelet-based Contourlet Transform,” *IEEE International Conference on Image Proc.(ICIP),* San Diego, CA, USA, Oct. 2008.

3. C. –H. Hung and H. –M. Hang, “Decision-directed Adaptive Wavelet Image Coding with Directional Decomposition,” *IEEE International Symposium on Circuits and Systems (ISCAS),* Taipei, Taiwan, May, 2009.

4. C. –H. Hung and H. –M. Hang, “Decision-Directed Adaptive Wavelet-based Contourlet Transform Coding Based on Mean Shift,” *IPPR Conference on Computer Vision, Graphics and Image Processing (CVGIP)* , Taichung, Taiwan, Aug. 2009.

5. C. –H. Hung and H. –M. Hang, “Scalable Video Coding Using Adaptive Directional Lifting-Based Wavelet Transform,” *IEEE Asian-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Xi’an, China, 2011.

6. C. –H. Hung and H. –M. Hang, “Direction Alignment Algorithm for Directional-Adaptive Discrete Wavelet Transform,” *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Kyoto, Japan, March, 2012.

Patent

1. 洪朝雄，杭學鳴，蔣迪豪，“基於移動平均判斷方向性濾波器之適用性方法”，中華民國 (審核中)。