DIRECTION ALIGNMENT ALGORITHM FOR DIRECTION-ADAPTIVE DISCRETE WAVELET TRASNFORM

Chao-Hsiung Hung[1] *and Hsueh-Ming Hang*[2]

Department of Electronics Engineering, National Chiao-Tung University, Taiwan
[1]morning.ee94g@nctu.edu.tw, [2]hmhang@mail.nctu.edu.tw

ABSTRACT

2-D discrete wavelet transform (2-D DWT) represents non-vertical or non-horizontal edges inefficiently. Direction-adaptive discrete wavelet transform (DA-DWT) solves this problem by filtering along the directions of textures. DA-DWT partitions images into many blocks and finds the most suitable direction of each block. The block direction information needs to be transmitted as side information. A conventional DA-DWT often produces inconsistent directions of neighboring blocks and thus results in large amount of side information. In this paper, we propose a bottom-up direction alignment algorithm to align the block directions in local areas. Our algorithm can reduce 50% or more in side-information bits at the cost of negligible prediction error increase.

Index Terms— Direction-adaptive discrete wavelet transform, direction alignment, image coding.

1. INTRODUCTION

2-D DWT plays an important role in image coding in recent years [1]. It applies two 1-D DWTs separately along the horizontal and vertical directions. However, it does not represent non-horizontal and non-vertical edges well because it produces many small coefficients and spreads the energy into high-pass subbands. Quantizing these coefficients to zero at low bit rates results in Gibbs artifacts along image edges [2].

The so-called direction-adaptive DWT, DA-DWT, [3]-[8] finds the most suitable direction for each image block and thus it provides better coding efficiency than the conventional 2-D DWT. It partitions images into non-overlapping blocks. It then applies the wavelet filter to the block along the candidate directions and calculates the corresponding prediction errors. It finally selects the direction with minimal prediction error as the most suitable direction for the block. Based on this principle, Chang and Girod proposed a DA-DWT with integer pixel accuracy [3]. Ding *et al*. adopted interpolation to achieve quarter pixel accuracy [4]. Liu and Ngan used a weighted function in the lifting scheme [5]. Dong *et al*. proposed a 2-D adaptive interpolate filter for more accurate fractional pixel accuracy [6]. Chang and Girod proposed another DA-DWT based on the quincunx subsampling [7]. Xu and Wu combined different subsampling patterns together and suggested a subsampling-adaptive DA-DWT.

One challenge in the DA-DWT approach is the significant amount of the side information that contains the block partition map and the block directions. To reduce the aforementioned problems, Tanaka *et al*. pre-filter an image before identifying the block directions in [9]. Maleki *et al*. merge blocks of similar directions into *megablocks* [10].

Motivated by these two papers, we propose a bottom-up direction alignment algorithm to align the direction of blocks with similar texture directions. First, we partition images into blocks with the smallest support size and identify each block's direction. Then, to reduce the overhead, we develop a direction adjustment and alignment algorithm, which tries to select consistent directions of nearby blocks. Comparing to the original block direction map without "alignment", we reduce more than 50% side information at the cost of about 1%~2% prediction error increase. Overall, we provide better coding performance than DA-DWT without direction alignment.

The remainder of this paper is organized as follows. We describe the proposed direction alignment algorithm in Section 2. We show the experimental results in Section 3 and conclude this paper by Section 4.

2. DIRECTION ALIGNMENT ALGORITHM

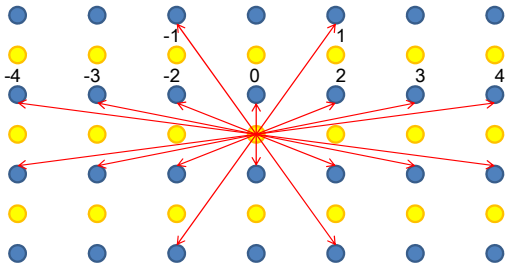

Fig. 1. Candidate directions in [3].

We adopt the 9 wavelet candidate directions in [3] and label them by the numbers from -4 to 4 in Fig. 1. The DA-DWT selects the best direction based on the minimal prediction

error for each block. It often results in different directions of nearby blocks as shown in Fig. 2, which leads to higher side information bits. We thus try to align the directions of neighboring blocks using the Lagrangian cost function. Fig. 3 shows the flow chart of proposed direction alignment algorithm.

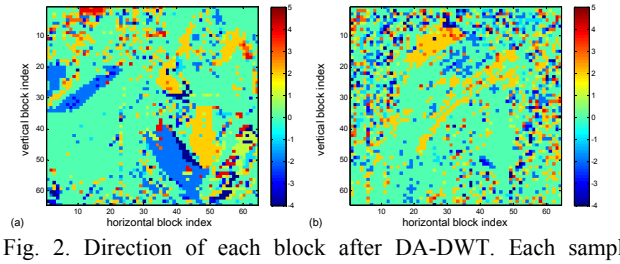

Fig. 2. Direction of each block after DA-DWT. Each sample represents the direction of an 8×8 block. (a) *Barbara* and (b) *Lena*. Indexes "-4"~"4" correspond to the direction index in Fig. 1 while index "5" will be used later.

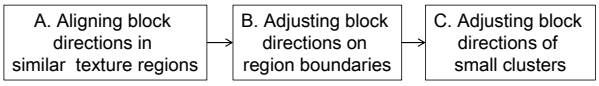

Fig. 3. Flow chart of proposed direction alignment algorithm.

A. Aligning block directions in similar texture regions

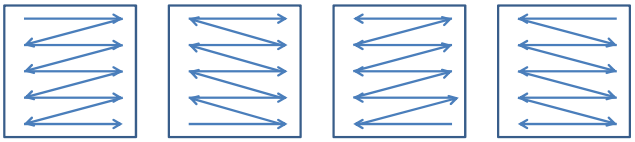

Fig. 4. Four scanning paths for *GB*(*m*,*n*).

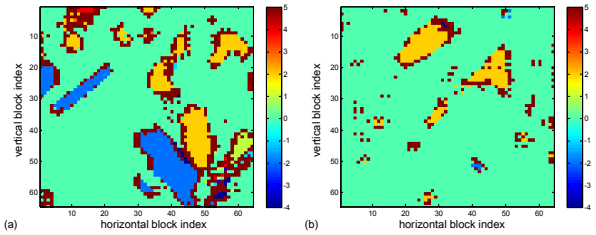

Fig. 5. Results after step A. $UB(i, j)$ is labeled as "5" and $count_d$ is 2. (a) *Barbara* and (b) *Lena*.

We first partition an image into non-overlapping blocks of size $B_H \times B_W$. We assume an image contains $F_H \times F_W$ blocks. Each block $B(i, j)$ has a set of prediction errors $\{D(i, j; d)\}$, each corresponding to a candidate direction d in Fig. 1, $1 \le i \le F_H$, $1 \le j \le F_W$, and $-4 \le d \le 4$.

A Group of Blocks, $GB(m, n)$, is made of 9 blocks: $\{B(i, j), m-1 \le i \le m+1, n-1 \le j \le n+1\}$. (The block number of a $GB(m, n)$ is less than 9 at picture boarders.) The Lagrangian cost function of selecting the same direction d for the entire $GB(m, n)$ is given by (1), in which the first term is the cumulative prediction errors of 9 blocks and R_d is the bits to code the selected direction d. By picking up the minimal cost function value, the best direction $d_{GB}^L(m, n)$ for $GB(m, n)$ is thus obtained and the corresponding cost function is $cost_{GB}^L(m, n)$ (eq(2)). The Lagrangian cost of the original block $B(i, j)$ is defined by (3). The best direction $d_B^L(i, j)$ and the corresponding cost function $cost_B^L(i, j)$ are defined by (4). We compare $cost_{GB}^L(m, n)$ and $cost_B^L(i, j)$, and then decide the direction $d_B(i, j)$ for $B(i, j)$ by (5).

$$L_{GB}(m, n; d) = \sum_{i=n-1}^{m+1} \sum_{j=n-1}^{n+1} D(i, j; d) + \lambda R_d \tag{1}$$

$$d_{GB}^L(m, n) = \arg\min_d \{L_{GB}(m, n; d)\}, cost_{GB}^L(m, n) = L_{GB}(m, n; d_{GB}^L(m, n)) \tag{2}$$

$$L_B(i, j; d) = D(i, j; d) + \lambda R_d \tag{3}$$

$$d_B^L(i, j) = \arg\min_d \{L_B(i, j; d)\}, cost_B^L(i, j) = L_B(i, j; d_B^L(i, j)) \tag{4}$$

$$if\ (cost_B^L(i, j) < cost_{GB}^L(m, n) / \sum_{i=m-1}^{m+1} \sum_{j=n-1}^{n+1} 1) \quad d_B(i, j) = d_B^L(i, j); \tag{5}$$

$$else \quad d_B(i, j) = d_{GB}^L(m, n);\ m-1 \le i \le m+1, n-1 \le j \le n+1$$

When we slide $GB(m, n)$ over an image, its scan order affects the final result. Therefore, we process $GB(m, n)$ in all four scanning paths as shown in Fig. 4. Thus, each $B(i, j)$ has four tentative directions, $d_B^{t1}(i, j)$, $d_B^{t2}(i, j)$, $d_B^{t3}(i, j)$, $d_B^{t4}(i, j)$ corresponding to the four scanning paths. We pick up the major direction if its count is higher than $count_d$. If there is no direction satisfies the preceding condition, it is called *uncertain block*, $UB(i, j)$.

The results of two test images after step A are shown in Fig. 5. It is clear that Step A is able to align block directions in the regions of similar textures. The uncertain blocks appear mostly around object boundaries. Step B is thus proposed.

B. Adjusting block directions on region boundaries

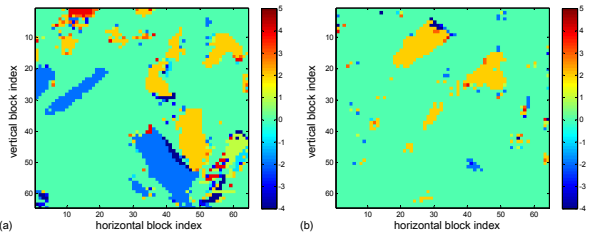

Fig. 6. The directions of $UB(i, j)$ are specified by (4): (a) *Barbara* and (b) *Lena*.

The uncertain blocks $UB(i, j)$ appear mostly on the boundaries between two regions of different texture directions as shown in Fig. 5. The direction of $UB(i, j)$ is first set to the best direction $d_B^L(i, j)$ derived from (4). The result is shown in Fig. 6. Clearly, some of them are not consistent with either of its neighboring regions. We thus adjust their directions as follows.

Each block $B(i, j)$ has four neighboring blocks, $B(i-1, j)$, $B(i+1, j)$, $B(i, j-1)$, and $B(i, j+1)$. Block $B(i, j)$ is said an

isolated block $IB(i, j)$ if $d_B(i, j)$ is different from all its neighboring block directions, $d_B(i\text{-}1, j)$, $d_B(i+1, j)$, $d_B(i, j\text{-}1)$, and $d_B(i, j+1)$. We force the direction candidates in (6) to be these four directions in (7).

$$d_{IB}(i, j) = \arg\min\{D(i, j; d) + \lambda R_d\} \quad (6)$$

$$d_{IB}(i, j) \in \{d_B(i+1, j), d_B(i-1, j), d_B(i, j+1), d_B(i, j-1)\} \quad (7)$$

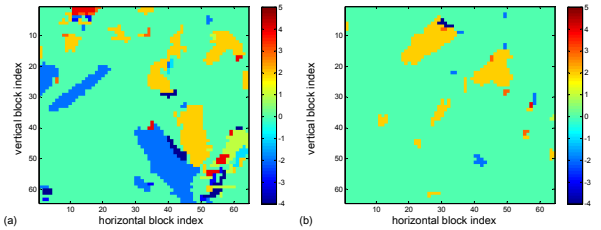

Fig. 7. Results after step B: (a) *Barbara* and (b) *Lena*.

Fig. 7 shows the results after step B and the directions of most $IB(i, j)$ blocks are now aligned with their neighbors. But a few $B(i, j)$ form small clusters and their directions are different from their neighbors. Thus, we add step C.

C. Adjusting block directions of small clusters

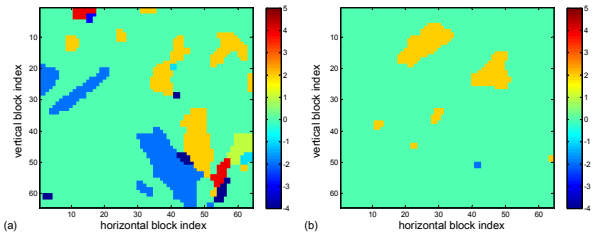

Fig. 8. Results after step C and $C_{iteration}$=4: (a) *Barbara* and (b) *Lena*.

A connected region is called a *small* cluster, if this region is long and narrow (with one-block width). If $B(i, j)$ belongs to a small cluster, its direction is re-selected by using (7) but the candidate directions are limited to the other three neighboring directions that are different from the cluster direction. This alignment operation may result in some new $IB(i, j)$ and we can adjust them using (7) again. We iterate the above procedure for $C_{iteration}$ times.

Fig. 8 shows the results after step C and all small clusters seem to disappear. The aligned regions in Fig. 8 have more rectangular shape than those in Fig. 7. The rectangle shape helps in reducing the side information. We adopt the coding scheme in [10] to merge blocks into megablocks and code the megablock directions.

Step B adjusts the directions of blocks around boundary and step C adjusts those of small clusters. Images with low-resolution may contain complex textures inside a small region. Aligning directions of the small regions in this case may increase prediction error. Thus, applying step B and step C to low-resolution images is less desirable.

3. EXPERIMENTAL RESULTS

We first look at the differences before and after direction alignment in prediction error and side information. We calculate the prediction error of images after 2-D DWT, DA-DWT [3], and DA-DWT with direction alignment (DA-DWT-A). We code the side information of DA-DWT and DA-DWT-A using [4]. Essentially, we use quadtree structure to represent the direction map.

TABLE I
PREDICTION ERRORS OF DIFFERENT SCHEMES

	2-D DWT	DA-DWT	DA-DWT-A
Barbara	1246982.949 (100%)	729310.350 (58.486%)	738823.981 (59.249%)
Elaine	933293.498 (100%)	821529.403 (88.025%)	833158.899 (89.271%)
Lena	582066.474 (100%)	545494.030 (93.717%)	549668.189 (94.434%)
Monarch	621922.367 (100%)	560337.203 (90.098%)	576912.779 (92.763%)
Pentagon	938440.040 (100%)	808399.145 (86.143%)	831827.392 (88.639%)
Spoke	1506005.112 (100%)	787627.590 (52.299%)	805417.063 (53.480%)

TABLE II
SIDE INFORMATION OF DIFFERENT SCHEMES

	DA-DWT	DA-DWT-A
Barbara	17602 (100%)	8891 (50.511%)
Elaine	25078 (100%)	11631 (46.379%)
Lena	21838 (100%)	8271 (37.873%)
Monarch	20964 (100%)	5166 (24.644%)
Pentagon	14541 (100%)	4729 (32.520%)
Spoke	15241 (100%)	8007 (52.536%)

The prediction error produced by 2-D DWT is the base for comparison, i.e., it is 100%, and the prediction error percentages of the other schemes are thus calculated in TABLE I. It shows that the DA-DWT-A increases 1%~2% in prediction error comparing to DA-DWT. For side information coding, we use quadtree and arithmetic coder to code quadtree block maps and prediction error of direction index [4]. The number of bits needed to encode the DA-DWT side information is set to 100% and the DA-DWT-A side information is compared against it in TABLE II. It shows that DA-DWT-A reduces more than 50% side information bits.

We compare the coding performance among JPEG2000 (2-D DWT+EBCOT), DA-DWT and DA-DWT-A. For all coding scheme, we adopt CDF 9-7 wavelet filters [11][12] and EBCOT [13] for wavelet transform and coefficient coding. We code the side information of DA-

DWT by the scheme in [4]. For the side information of DA-DWT-A, we adopt that in [10], as said earlier.

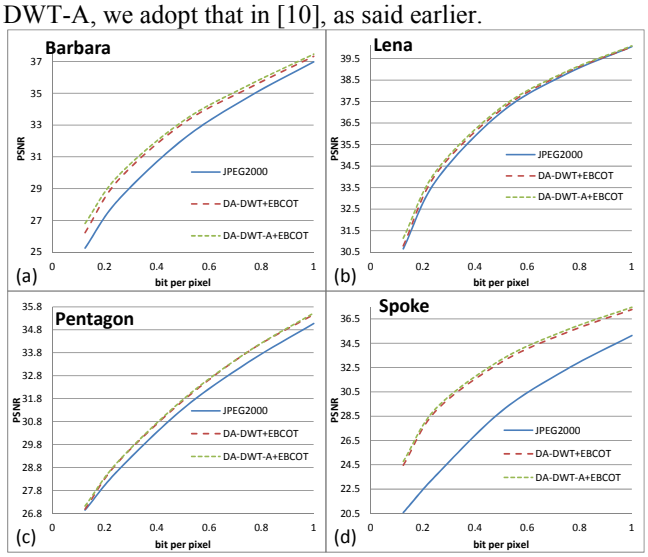

Fig. 9. PSNR of different coding schemes.

TABLE III
COMPARISON BETWEEN DIFFERENT CODING SCHEMES

Coding Scheme	DA-DWT-A+EBCOT compared to DA-DWT+EBCOT		DA-DWT-A+EBCOT compared to JPEG2000	
	BD-PSNR(dB)	BD-BR(%)	BD-PSNR(dB)	BD-BR(%)
Barbara	0.253	-4.569	1.261	-20.151
Elaine	0.028	-1.674	0.149	-6.375
Lena	0.136	-2.945	0.316	-6.851
Monarch	0.060	-0.992	0.362	-4.541
Pentagon	0.059	-1.687	0.381	-8.642
Spoke	0.218	-3.466	4.485	-47.597

Fig. 9 shows the PSNR of several test images. DA-DWT-A+EBCOT provides better results than DA-DWT+EBCOT, particularly at low bit rates because of its lower side information. Clearly, both DA-DWT+EBCOT and DA-DWT-A+EBCOT outperform JPEG2000.

We also use the Bjontegaard delta bit rate (BD-BR) measure and the Bjontegaard delta PSNR (BD-PSNR) measure [14] to compare the coding performance. TABLE III shows that DA-DWT-A+EBCOT provides the best coding performance.

4. CONCLUSIONS

In this paper, we propose a bottom-up direction alignment algorithm with three steps. Step A aligns the directions of textures with similar directions. Step B and Step C adjust the directions of boundary blocks and small clusters to match their large neighboring clusters. The proposed alignment algorithm thus produces only a few dominate direction regions for the entire picture. It decreases 50% or higher side information with about 1%~2% increase in prediction error. Experimental results show that this algorithm also provides better overall coding performance than JPEG200 and the original DA-DWT without direction alignment.

5. ACKNOWLEDGEMENT

This work was supported in part by the NSC, Taiwan under Grant 98-2221-E-009 -076.

6. REFERENCES

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

[2] D. Taubman and A. Zakhor, "Orientation adaptive subband coding of images," *IEEE Trans. Image Process.*, vol. 3, no. 4, pp. 421–437, Apr. 1994.

[3] 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.

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

[5] Y. Liu and K. N. Ngan, "Weighted adaptive lifting-based wavelet transform for image coding," *IEEE Trans. Image Processing,* pp. 500-511, Apr. 2008.

[6] 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.

[7] 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.

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

[9] 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.

[10] A. Maleki, B. Rajaei, and H. R. Pourreza, "Rate-distortion improvement of directional wavelets by megablocking," in *Proc. ICASSP*, May 2011, pp. 801–804.

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

[12] I. Daubechies, W. Sweldens, "Factoring wavelet and subband transforms into lifting steps", Bell Laboratories, Lucent Technologies, 1996.

[13] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Processing,* vol. 9, no. 7, pp. 1158-1170, Jul. 2000.

[14] G. Bjontegaard, *Calculation of average PSNR differences between RD curves*, document VCEG-M33, ITU-T SG16 Q.6 VCEG, Apr. 2001.