LOCAL BANDWIDTH CONSTRAINED FAST INVERSE MOTION COMPENSATION FOR DCT-DOMAIN VIDEO TRANSCODING

Shizhong Liu and Alan C. Bovik

Laboratory for Image and Video Engineering, Dept. of Electrical and Computer Engineering,
The University of Texas at Austin, Austin, TX 78712-1084, USA.
Email: {sliu2, bovik}@ece.utexas.edu

ABSTRACT

DCT-based digital video coding standards such as MPEG and H.26x are becoming more widely adopted for multimedia applications. Since the standards differ in their format and syntax, video transcoding, where a pre-coded video bit-stream is converted from one format to another format, is of interest for purposes such as channel bandwidth adaptation and video composition. DCT-domain video transcoding is generally more efficient than spatial domain transcoding. However, since the data is organized block by block in the DCT-domain, inverse motion compensation becomes the bottleneck for DCT-domain methods. In this paper, we propose a novel local bandwidth constrained fast inverse motion compensation algorithm operating in the DCT-domain. Relative to Chang's algorithm [1], the proposed algorithm achieves computational improvement of 25% to 55% without visual degradation. A by-product of the proposed algorithm is a reduction of blocking artifacts in very low bit-rate compressed video sequences.

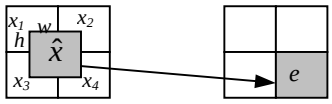
1. INTRODUCTION

Digital video data are becoming widely available as MPEG or H.26x bit-streams. In video communication systems, video transcoding is the key technology to continuously adapt the output channel bandwidth or to convert the video from one format to another format. DCT-domain video transcoding is generally more efficient than spatial domain transcoding because it eliminates the need for complete decompression and compression and subsequent degradation in video quality, etc. [1-4]. However, since data is organized block by block in the DCT-domain, inverse motion compensation becomes the bottleneck for DCT-domain methods [1-4].

The problem of *DCT-domain inverse motion compensation* was studied by Chang and Messerschmitt [1]. The general setup is shown in Fig. 1, where $\hat{x}$ is the current block of interest, x_1, x_2, x_3 and x_4 are the reference blocks from which $\hat{x}$ is derived. According to [1], $\hat{x}$ can be expressed as a superposition of the appropriate windowed and shifted versions of x_1, x_2, x_3 and x_4, i.e.,

$$\hat{x} = \sum_{i=1}^{4} q_{i1} x_i q_{i2} \qquad (1)$$

This research was supported in part by Texas Instruments, Inc. and by Texas Advanced Technology Program.



Intra-coded frame Inter-coded frame

Fig. 1. DCT-domain inverse motion compensation.

where q_{ij}, $i = 1,\ldots,4$, $j = 1, 2$ are sparse 8×8 matrices of zeros and ones that perform windowing and shifting operations. For example, for $i = 1$,

$$q_{11} = \begin{pmatrix} 0 & I_h \\ 0 & 0 \end{pmatrix},\ q_{12} = \begin{pmatrix} 0 & 0 \\ I_w & 0 \end{pmatrix}$$

where I_h and I_w are identity matrices of dimension $h \times h$ and $w \times w$, respectively. h and w are determined by the motion vector of $\hat{x}$. By using the linear, distributive and unitary properties of DCT, we can obtain the following relation in the DCT-domain:

$$\hat{X} = \sum_{i=1}^{4} Q_{i1} X_i Q_{i2} \qquad (2)$$

where $\hat{X}$, $\{X_i\}$ and Q_{ij} are the DCT's of $\hat{x}$, $\{x_i\}$ and q_{ij}, respectively.

In [3], Merhav *et al.*proposed a fast algorithm to compute (2) by factorizing the fixed matrices Q_{ij} into a series of relatively sparse matrices instead of fully pre-computing them. As a result, some of the matrix multiplications can be avoided by using simple addition and permutation operations. Assuncao *et al.* [4] approximate the elements of Q_{ij} to binary numbers with a maximum distortion of $1/32$ so that all multiplications can be implemented by *shifts* and *additions*. They showed that in terms of operations (*shift, add*) required, their algorithm has only 28% of the computational complexity of the method proposed by Merhav *et al.* [3]. While all the methods above adopt 2-D implementation scheme, Acharya *et al.* [2] adopts another implementation scheme, where the problem is decomposed into two separate 1-D problems. They have shown that this decomposition is more efficient than computing the combined operation. Also, any fast algorithm can be easily applied to this separable processing structure. Therefore, we focus on this implementation in our work.

In this paper, we propose a novel algorithm for inverse motion compensation in the DCT-domain. By modeling a natural

image as a 2-D separable Markov Random Field, we estimate the local bandwidth of the block to be reconstructed from the reference blocks. The algorithm can reduce the processing time by avoiding the computations of those DCT-coefficients outside the estimated local bandwidth. Relative to Chang's algorithm [1], the proposed algorithm achieves computational improvement of 25% to 55% without visual degradation. Another advantage of the algorithm is that it can work on top of the fast algorithms proposed in [3, 4] to gain more computational savings. A by-product of the proposed algorithm is a reduction of blocking artifacts in very low bit-rate compressed video sequences.

2. LOCAL BANDWIDTH CONSTRAINED INVERSE MOTION COMPENSATION

2.1. The Basic Idea

As discussed in the last section, all proposed algorithms are based on two operations, i.e., *windowing and shifting*. The *windowing* operation keeps the data inside the window unchanged but zeros all data outside the window. As a result, it usually introduces a steep change at the edge of the window, which means that many high frequencies are possibly introduced by the algorithm. To clarify, let us study the 1-D case. Fig. 2 shows a low bandwidth signal $y(n)$ obtained by summing two functions: $y(n) = y_l(n) + y_r(n)$.

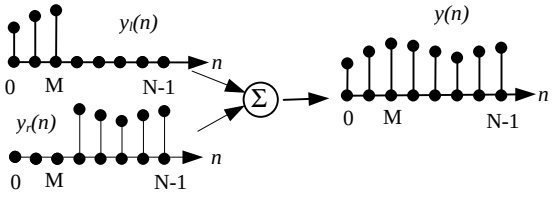


Fig. 2. 1-D windowing operation.

Let $w_l(n)$, $w_r(n)$ be two window functions:

$$w_l(n) = \begin{cases} 1, & 0 \le n \le M \\ 0, & otherwise \end{cases}, \quad w_r(n) = \begin{cases} 1, & M < n < N \\ 0, & otherwise \end{cases}.$$

We can write $y_l(n) = y(n)w_l(n)$ and $y_r(n) = y(n)w_r(n)$. Let $Y(e^{j\omega})$, $Y_l(e^{j\omega})$, $Y_r(e^{j\omega})$, $W_l(e^{j\omega})$ and $W_r(e^{j\omega})$ be the Fourier Transforms of $y(n)$, $y_l(n)$, $y_r(n)$, $w_l(n)$ and $w_r(n)$, respectively. Then the following equations can be obtained:

$$Y_l(e^{j\omega}) = Y(e^{j\omega}) \otimes W_l(e^{j\omega}) \tag{3}$$

$$Y_r(e^{j\omega}) = Y(e^{j\omega}) \otimes W_r(e^{j\omega}) \tag{4}$$

where $\otimes$ denotes convolution of two periodic functions with the limits of integration extending over only one period. Let B, B_l, B_r, B^l_w and B^r_w be the bandwidths of $y(n)$, $y_l(n)$, $y_r(n)$, $w_l(n)$ and $w_r(n)$ respectively. From (3) and (4), we obtain the following inequalities:

$$B_l \ge \max(B, B^l_w) \tag{5}$$

$$B_r \ge \max(B, B^r_w). \tag{6}$$

Let E_l be frequency components beyond B in B_l, and E_r be the frequency components beyond B in B_r. Since $y(n) = y_l(n) + y_r(n)$, the following equation must be satisfied:

$$E_l + E_r = 0. \tag{7}$$

This means that all frequency components beyond B will disappear after summation, implying that there is no need to compute them. So if we can estimate the frequency bandwidth B before constructing $Y(e^{j\omega})$, we need only compute those frequency components inside B when calculating $Y_l(e^{j\omega})$ and $Y_r(e^{j\omega})$. This is the basic idea of our algorithm described below.

2.2. Local Bandwidth Constrained Inverse Motion Compensation

Generally, neighboring pixels are highly correlated in images. This inter-pixel correlation is often modeled using Markov Random Field (MRF) models [5]. Sikora and Li [6] also assume that the 2-D image random field is separable with identical and stationary correlation along each image dimension and that the simple first order AR(1) Markov model is adopted to model the pixel-to-pixel correlation along image rows and columns. For each image row, the variance-normalized AR(1) 1-D autocorrelation function can be expressed as $R_x = \alpha^{|n|}$ where n describes the distance between two image pixels and α denotes the pixel-to-pixel correlation in the row. α typically takes values between 0.9 to 0.98 [6]. Fig. 3 shows two 1-D eight point adjacent blocks L_1 and L_2 in an image row.

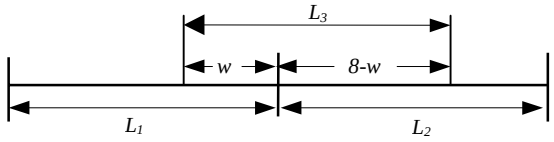


Fig. 3. 1-D block extraction.

According to the above model, L_1 and L_2 should have the same power spectral density function, hence the same bandwidth because they have the same correlation function $R_x = \alpha^{|n|}$ [6]. Similarly, if we want to extract L_3 (shown in Fig. 3) from L_1 and L_2, we can predict that L_3 also has the same bandwidth as L_1 and L_2 based on the model. However, images are usually non-stationary, so the bandwidth of L_1 is often different from that of L_2. To account for this, we take the maximum bandwidth as the estimate for L_3, i.e.,

$$B_3 = \max(B_1, B_2) \tag{8}$$

where B_1, B_2 and B_3 are the bandwidth of L_1, L_2 and L_3, respectively. For example, if the maximum index of the non-zero DCT coefficients (here we use DCT coefficients as the representations of frequency components) is 2 in L_1 and 4 in L_2, we estimate that the maximum index of the non-zero DCT coefficients in L_3 is 4. To extract the DCT coefficients directly from the DCT's of L_1 and L_2, we only need to compute those DCT coefficients with index no greater than 4 in L_3. Since we use separable implementation scheme proposed in [2], the 2-D problem can be converted into two 1-D problems.

3. RESULTS AND DISCUSSIONS

We use the method proposed by Chang and Messerschmitt [1] as the original algorithm. We implement both the original algorithm and our algorithm, and integrate them, for comparison, into our DCT-domain video transcoder as the inverse motion compensation module, respectively. The input of the transcoder is an MPEG-coded video bit-stream. To evaluate the perform-

ance of our algorithm, we transcode all P and B frames in the incoming bit-stream back to I frames by DCT-domain inverse motion compensation. First, we will investigate the distortion caused by the algorithm by comparing the PSNR of those I frames recovered from P or B frames using both algorithms. Then we will measure the speed of both algorithms to show the speed improvement of our algorithm. To test our algorithm more efficiently, we select four video sequences with intensive motion activities, i.e., "foreman", "coastguard", "mobile" and "stefan". All sequences are CIF resolution with 352 pixels per line and 288 lines. To evaluate the performance of our algorithm at different coding bit-rates, the sequences are encoded at 4 Mb/s and 1 Mb/s respectively. Fig. 4 shows the PSNR result of each reconstructed frame of "coastguard". The average PSNR degradation is 0.12dB at 4 Mb/s and 0.35dB at 1 Mb/s. The results of average PSNR degradation for other sequences are 0.11 dB in "foreman", 0.22 dB in "mobile" and 0.16 dB in "stefan" at 4 Mb/s, and 0.29 dB in "foreman", 0.51 dB in "mobile" and 0.36 dB in "stefan" at 1 Mb/s respectively. The PSNR degradation depends on the images. For example, in the sequence "mobile", the pictures have a lot of strong edges and are very dynamic, hence the AR model of our algorithm is not accurate. As a result, the PSNR of "mobile" degrades more than that of other sequences. Similarly, at low bit-rate, since each block in the frame is independently quantized by a large quantization factor, the correlation between adjacent blocks is reduced. Thus, the local bandwidth estimation based on the AR model may not be accurate enough, which causes more degradation at low bit-rates as shown in the experimental results. However, for both encoding bit-rates, the amount of distortion introduced by the proposed algorithm is hardly visible. A by-product of our algorithm is to reduce the blocking artifacts in images coded at very low bit-rates because the local bandwidth constrained inverse motion compensation also works as a low-pass filter to the reconstructed block. To show this, we encode the gray-level image "Lena" at 0.21 bits/pixel using JPEG. Then we shift the image by 4 pixels in the vertical direction and reconstruct the shifted image using inverse motion compensation. The reconstructed images by the original algorithm and our algorithm are shown in Fig. 5(a) and Fig. 5(b), respectively. We can see the image reconstructed by our algorithm is smoother than that by the original algorithm. Additionally, we apply the blocking artifact measuring method proposed in [7] to both reconstructed images. The measurement results are 6.1 for the image reconstructed by the original algorithm and 3.5 for the image reconstructed by our algorithm, respectively. The results also show that the blocking artifacts in the image reconstructed by our algorithm are much less than that in the image reconstructed by the original algorithm. In general, our algorithm can save more computations as the encoding bit-rate is reduced since the DCT block is sparser. In Tables I, we list the average computation time to recover one P or B frame to an I frame by both algorithms at bit-rates of 4 Mb/s and 1 Mb/s, respectively. The computing time is measured on a Windows NT workstation with 300MHz Pentium II CPU, and 512MB memory. The savings in computation time for reconstructing one P or B frame are 25-30% at 4 Mb/s, and 45-55% at 1Mb/s.

4. CONCLUSIONS

In this paper, we proposed a novel local bandwidth constrained inverse motion compensation algorithm, in which only those DCT coefficients inside the estimated bandwidth are computed. The local bandwidth estimation is based on the assumption that the image can be modeled as two separable AR sequences in the horizontal and vertical directions, respectively. On the average, our algorithm can reduce the computation time by 25-30% at 4 Mb/s, and 45-55% at 1Mb/s, compared to Chang's algorithm. However, the video degradation caused by our algorithm is invisible for both high bit-rate and low bit-rate coded video sequences. Since our algorithm can work on top of the fast algorithm proposed in [3, 4], we could expect further improvement with a combination of our algorithm and those in [3, 4]. A by-product of the proposed algorithm is a reduction of blocking artifacts in very low bit-rate compressed video sequences**.** Some other applications, such as DCT-domain scene-cut detection and DCT-domain feature extraction for video indexing, can also benefit from the fast inverse motion compensation algorithm proposed in this paper.

REFERENCES

[1] S. -F Chang and D. G. Messerschmitt, " Manipulation and Compositing of MC-DCT Compressed Video," *IEEE Journal on Selected areas in Communications*, vol. 13, no.1, Jan. 1995.

[2] S. Acharya and B. Smith, "Compressed Domain Transcoding of MPEG," *Proc. IEEE International Conference on Multimedia Computing and Systems-1998*, pp. 295-304, 1998.

[3] N. Merhav and V. Bhaskaran, " Fast Algorithm for DCT-Domain Image Down-Sampling and for Inverse Motion Compensation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 7, no. 3, pp. 468-476, June 1997.

[4] Pedro A. A. Assuncao and M. Ghanbari, "A Frequency-Domain Video Transcoder for Dynamic Bit-Rate Reduction of MPEG-2 Bit Streams," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 8, no. 8, pp. 953-967, Dec. 1998.

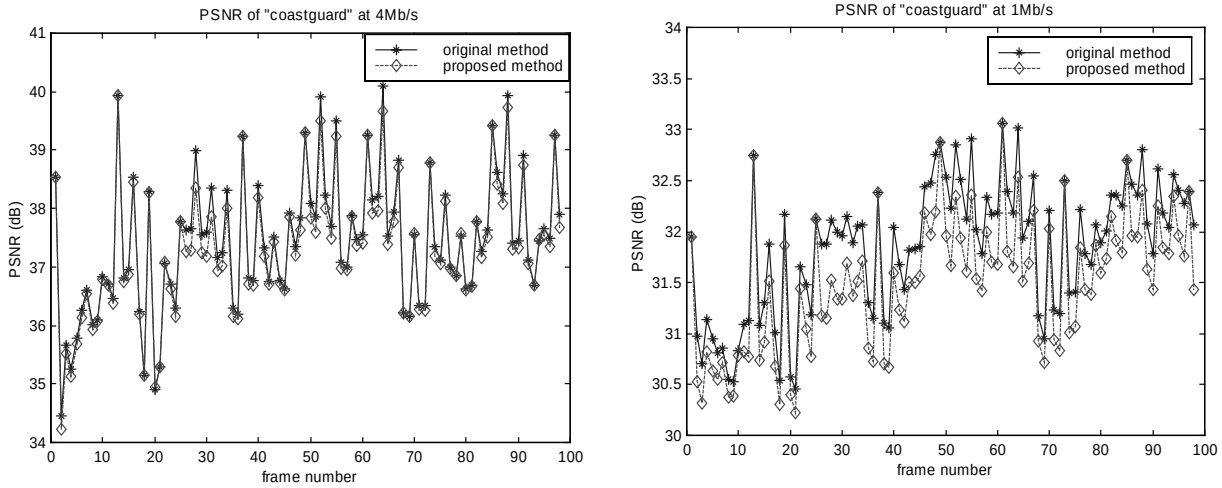
[5] M. Bhatt and U. Desai, "Robust image restoration algorithm using Markov random field model," *CGVIP: Graphical Models and Image Processing*, vol. 56, pp. 61--74, 1994.

[6] T. Sikora and H. Li, "Optimal Block-Overlapping Synthesis Transforms for Coding Images and Video at Very Low Bitrates," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, no. 2, pp. 157-167, Apr. 1996.

[7] Z. Wang and A. C. Bovik, "Blind Measurement of Blocking Artifacts in Images," *ICIP-2000*, Vancouver, Canada, pp. 981-984.

Table I Time to convert a P or B frame to an I frame at bit-rate of 4 Mb/s and 1 Mb/s (Unit: second)

Video sequence	The original algorithm				The proposed algorithm			
	4 Mb/s		1 Mb/s		4 Mb/s		1 Mb/s	
	P frame	B frame	P frame	B frame	P frame	B frame	P frame	B frame
“foreman”	0.3137	0.4738	0.2512	0.3987	0.2387	0.3644	0.1324	0.2152
“coastguard”	0.2374	0.3417	0.1912	0.3099	0.1700	0.2464	0.0937	0.1490
“mobile”	0.3487	0.4136	0.2983	0.3686	0.2513	0.3113	0.1550	0.2061
“stefan”	0.2057	0.3667	0.1636	0.2941	0.1370	0.2484	0.0743	0.1408



(a) (b)

Fig. 4. PSNR results of each reconstructed frame in “coastguard”.

(a) (b)

Fig. 5. (a) reconstructed image “Lena” by the original algorithm;
(b) reconstructed image “Lena” by the proposed algorithm.