Half-Pixel Correction for MPEG-2/H.264 Transcoding*

Soon-young Kwon, Joo-kyong Lee, and Ki-dong Chung

Dept. Of Computer Engineering, Pusan National University, Busan, Korea
{ksy2020, jklee, kdchung}@pusan.ac.kr

Abstract. To improve video quality and coding efficiency, H.264/AVC [1] has adopted several newer coding tools such as a 4×4 integer DCT and a method for calculating half pixels than the previous standards. However, these tools require additional work to transcode video content for pre-coded in the previous standards to H.264/AVC in the DCT domain. In this paper, we propose, as far as we know, the first half-pixel correction method for MPEG-2 to H.264 transcoding in the DCT domain. In the proposed method, an MPEG-2 block is added to the correction block obtained from the difference between half-pixel values of the two standards using a DCT reference frame. Experimental results show that the proposed method achieves better quality than the pixel based transcoding method.

1 Introduction

H.264/AVC, the latest international video coding standard, has been approved by ISO/IEC and by ITU-T. According to [2], H.264 has a higher coding efficiency than any other previous video coding standard such as MPEG-2, H.263, and MPEG-4. Accordingly, it is expected to be in wide use and to replace other standards in the near future. This means that an efficient transcoding method that transcodes the currently widely available video coding standards into the H.264 compressed data format is needed [3]. The existent transcoders can be classified into two types of architectures, PDT (Open-loop Pixel-Domain Transcoder) and DDT (DCT-Domain Transcoder) [4]. Recent researches have focused on DDT approaches to improve computational complexity and to avoid the DCT and IDCT mismatch problem of PDT. However, so far, only PDT architecture has been used to transcode previous standards to H.264. That is, it is difficult to use common tools in transcoding because of the new tools in H.264. Specifically, H.264 uses a 6-tap filter to obtain half-pixel values and a 4×4 integer DCT instead of a 2-tap filter and an 8×8 float DCT. In this paper, we propose an efficient half-pixel correction scheme to support DCT-domain transcoding architecture from MPEG-2 to H.264. A correction scheme is essential for DCT-domain transcoding, because most video coding standards use half-pixel motion vector accuracy, but the half-pixel calculation method used in H.264 differs

* The work was supported by Pusan National University Research Grant.

from those of other standards. Our proposed scheme consists of two parts: CHPD (Correction for Half-Pixel Difference) and DCT_Conv (DCT Conversion). First, in CHPD, the MPEG-2 block is added to the correction block obtained by calculating the difference between the half-pixel values from two standards using a DCT reference frame. Second, the resultant block is passed into DCT_Conv to convert DCT transform between two standards. Both of the phases use matrices stored in memory to reduce computational complexity. DCT-domain H.264 transcoder requires our proposed method. And the proposed method supports previous transcoding technologies in the DCT-domain, such as frame rate control, bitrates reduction and scalability coding.

This paper is organized as follows. In Section 2 we describe the half-pixel operations of previous video standards and H.264. In Section 3 we present the proposed method and experimental results in Section 4. We close the paper with concluding remarks in Section 5.

2 Half-Pixel in Video Coding Standards

MPEG-2, H.263 and MPEG-4 use the same numerical formulas to calculate half-pixel values, however H.264 uses different formulas. The DDT to H.264, that reuses the half-pixel motion accuracy of the input video data, must perform the half-pixel correction. In this paper, we exploit three types of half-pixel MVs (Motion Vectors). They are, half-pixel horizontally, half-pixel vertically, and half-pixel bi-directionally, represented by H-halfpixel, V-halfpixel or HV-halfpixel, respectively. For example, the notations MV (1, 1.5), MV (1.5, 1), MV (1.5, 1.5) would correspond to each of them, respectively.

In MPEG-2, H.263 and MPEG-4, half-pixel values are the average of two or four neighboring pixels [5][6]. Fig. 1 illustrates the sample interpolation for samples b , h and j .

Fig. 1. Illustration of half-pixel samples. Upper-case letters indicate samples on the full-sample grid, while lower letters indicate samples in between at fractional-sample positions.

$$b = \frac{(G+H+1)}{2}, \quad h = \frac{(G+M+1)}{2} \quad (1)$$

$$j = \frac{(G+H+M+N+2)}{4}$$

$$b_1 = (E - 5F + 20G + 20H - 5I + J) \quad (2)$$

$$h_1 = (A - 5C + 20G + 20M - 5R + T)$$

$$b = \langle b_1 + 16 \rangle \gg 5$$

$$h = \langle h_1 + 16 \rangle \gg 5 \quad (3)$$

$$j_1 = cc - 5dd + 20h_1 + 20m_1 - 5ee + ff$$

$$j = \langle j_1 + 512 \rangle \gg 10 \quad (4)$$

In H.264, half-pixel values are obtained by applying a one-dimensional 6-tap filter horizontally and vertically [1][6]. In Fig. 1, H-halfpixel, b , and V-halfpixel, h , are derived by first calculating the intermediate values b_1 and h_1 , respectively, then applying (2)'s 6-tap filter and obtaining the final values by (3). In the case of, HV-halfpixel, j is obtained by (4) where intermediate values denoted as cc, dd, ee, m_i and ff are obtained in a manner similar to h_1 . In [7], the percentage of half-pixel MVs is 37.03%. MVs with half-pixel are a significant part of all MVs, therefore, the DCT-domain transcoder must have a half-pixel correction filter.

3 Half-Pixel Correction in DCT-Domain

Fig. 2 shows a block diagram of the proposed transcoder architecture for MPEG-2 to H.264/AVC where Q_1 and Q_2 denote MPEG-2 and H.264 quantization and q_1 and q_2 denote Qstep, respectively. This paper focuses on half-pixel correction, therefore, we give the full details of CHPD and DCT_Conv.

Lowercase variables indicate pixel based values, whereas the corresponding uppercase variables are the DCT representation of the lowercase variables.

Fig. 2. Illustration of the proposed transcoder architecture for MPEG-2 to H.264/AVC

3.1 Proposed Schemes

The proposed CHPD consists of 3 phases: the first is IMC (Inverse Motion Compensation), the second is calculating the difference between the half-pixel values from the standards, and the third is adding the correction block to the input block. To obtain a reference frame, IMC uses the DCT reference frame in Frame Store [7][8]. The

Fig. 3. The proposed IMC (Input block is overlapped 9 blocks)

conventional IMC extracts an 8×8 block to obtain MPEG-2 half-pixel values [7], but the proposed IMC extracts a maximum 13×13 block instead because the H.264 filter needs more pixels, for example left 2 pixels, right 3 pixels, upper 2 pixels, and bottom 3 pixels. As we can see in Fig. 1, the half-pixels located on the border of the 8×8 block need more than two or three pixels to calculate HV-halfpixel. Fig. 3 shows that a 13×13 block used to obtain the correction block is overlap by a maximum of 9 blocks.

The calculation, $\hat{b}$, is formalized as

$$\begin{aligned}\hat{b} &= \sum_{i=1}^n e_l \cdot b_i \cdot e_r, \text{ where } n \in S, \quad S = \{2, 3, 4, 6, 9\} \\ &= \underbrace{\begin{pmatrix} b_1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \end{pmatrix}}_{b_1} + \underbrace{\begin{pmatrix} 0 & b_2 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \end{pmatrix}}_{b_2} + \cdots + \underbrace{\begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & b_n \end{pmatrix}}_{b_n}\end{aligned}\quad (5)$$

where n is the number of overlapped blocks and b_i is an 8×8 reference block but the size of e_l , e_r and $\hat{b}$ change according to the half-pixel direction, e_l and e_r are shift matrices and are defined below.

$$e_l = \begin{pmatrix} 0 & I_{l \times l} \\ 0 & 0 \end{pmatrix}_{m' \times n'} \quad \text{or} \quad \begin{pmatrix} 0 & 0 \\ I_{l \times l} & 0 \end{pmatrix}_{m' \times n'}, \quad e_r = \begin{pmatrix} I_{l \times l} & 0 \\ 0 & 0 \end{pmatrix}_{m'' \times n''} \quad \text{or} \quad \begin{pmatrix} 0 & 0 \\ 0 & I_{l \times l} \end{pmatrix}_{m'' \times n''} \quad (6)$$

, where $m', n', m'', n'' \in \{8, 13\}$

For example, the size of $\hat{b}$, e_l and e_r , V-halfpixels are 13×8, 13×8, and 8×8, respectively

Due to the distributive property of the DCT, we can get (7) from (5).

$$\hat{B} = \sum_{i=1}^n E_L \cdot B_i \cdot E_R, \text{ where } n \in S, \quad S = \{2, 3, 4, 6, 9\} \quad (7)$$

Let us find the difference of the half-pixels between the two standards by (8),

$$\begin{cases} C = F_L^S \cdot \hat{B} \gg 5 \\ C = \hat{B} \cdot F_R^S \gg 5 \\ C = F_{HL} \cdot \hat{B} \cdot F_{HR} \gg 10 - F_{ML} \cdot \hat{B} \cdot F_{MR} \gg 2 \end{cases} \quad (8)$$

$$f_{hl} = \begin{pmatrix} 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 \end{pmatrix} \quad f_{ml} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

where f is a matrix of constant values, $F_L^S = F_{HL} - 16 \cdot F_{ML}$, $F_R^S = F_{HR} - 16 \cdot F_{MR}$, $f_{hr} = f_{hl}^T$ and $f_{mr} = f_{ml}^T$.

Equation (8) is divided into three parts such as, V-halfpixel, H-halfpixel and HV-halfpixel according to half-pixel direction,

To reduce complexity, we use (9) instead of (7), and (8) for V-halfpixel and H-halfpixel because E_L , E_R , F_L^S , F_R^S are matrices of constant values

$$\begin{aligned} C &= \sum_{i=1}^n W_L \cdot B_i, \text{ where } n \in S, \quad S = \{2, 3, 4, 6, 9\} \\ C &= \sum_{i=1}^n B_i \cdot W_R, \text{ where } n \in S, \quad S = \{2, 3, 4, 6, 9\} \end{aligned} \quad (9)$$

$$W_L = F_L^S \cdot E_L \gg 5, \quad W_R = E_R \cdot F_R^S \gg 5$$

Then correction block (C) is added to input block ($\tilde{B}$) as follows:

$$N = \tilde{B} - C \quad (10)$$

At last, we derive a new residual block from (10) in the DCT-domain and then the new residual block (N) is used as an input into DCT_Conv to adjust the transform between the two standards.

3.2 DCT Conversion

H.264 adopts a 4×4 integer transform instead of the 8×8 float transform used in MPEG-2, H.263 and MPEG-4, so the transcoder must convert an 8×8 float DCT block into four 4×4 integer DCT blocks as shown in Fig. 4. DCT Conversion is performed once by the matrix operation defined as:

$$\begin{aligned} \hat{N}_i &= h_i \cdot n_i \cdot h_i^T \quad (1 \leq i \leq 4) \\ &= h_i \cdot (l_i \cdot n \cdot r_i) \cdot h_i^T \\ &= (h_i \cdot l_i) \cdot n \cdot (r_i \cdot h_i^T) \\ &= (h_i \cdot l_i) \cdot (h_f^T \cdot N \cdot h_f) \cdot (r_i \cdot h_i^T) \\ &= (h_i \cdot l_i \cdot h_f^T) \cdot N \cdot (h_f \cdot r_i \cdot h_i^T) \\ &= u_i \cdot N \cdot u_r \end{aligned} \quad (11)$$

Fig. 4. Extracting 4×4 sub-blocks from an 8×8 block in the transcoder. N and $\hat{N}_i$ denote an 8×8 float DCT block, and 4×4 integer DCT sub-block, respectively

where $h_l \cdot l_i \cdot h_F^T (= u_l)$, $h_F \cdot r_i \cdot h_l^T (= u_r)$, h_l and h_F denote the 4×4 integer transform matrix, and 8×8 float transform matrix, respectively. 4×8 matrix l_i and 8×4 r_i are defined in (12), where $I_{4 \times 4}$ is an identity matrix of size 4. Refer to [9] for a more detailed explanations.

$$\begin{aligned} l_0 = l_1 &= \begin{bmatrix} I_{4 \times 4} & 0 \end{bmatrix}_{4 \times 8}, \quad l_2 = l_3 = \begin{bmatrix} 0 & I_{4 \times 4} \end{bmatrix}_{4 \times 8} \\ r_0 = r_2 &= \begin{pmatrix} I_{4 \times 4} \\ 0 \end{pmatrix}_{8 \times 4}, \quad r_1 = r_3 = \begin{pmatrix} 0 \\ I_{4 \times 4} \end{pmatrix}_{8 \times 4} \end{aligned} \quad (12)$$

4 Experimental Results

In this section, we compare the performance of our proposed scheme with that of the PDT as shown in Fig. 5, because DDT that performs half-pixel correction does not exist currently. To precisely simulate different transcoding processes, we have implemented the architecture shown in Fig. 2 by modifying the MPEG-2 TM5 [10] and by partially adopting the H.264 JM8 [11]. Test video sequences are constrained by a GOP structure of (N = 15, M = 1) and at a constant frame rate of 30.

Fig. 5. The PDT (Open-loop Pixel-Domain Transcoder) architecture

Fig. 6 shows the average PSNRs of the frames ranging from 1 to 100 for the Football, and the Foreman sequences. In these figures, we can see that, as a whole, the proposed scheme performs better than that of the PDT. Table 1 denotes the average PSNR of the first 100 frames for Football and Foreman, and it shows that the proposed method outperforms PDT by 0.22 dB. As we can see in Fig. 5, PDT has two MCs, MC_1 and MC_2 , which calculate MPEG-2, and H.264 half-pixel values, which lead to rounding errors. The proposed method reduces rounding errors because it directly calculates differences using matrix operations, and it avoids the DCT and IDCT mismatch problem of PDT. These are the reasons the proposed method outperforms PDT. Now, let's examine the real images as reconstructed by both transcoders. Fig. 7 compares in subjective quality. For example, the right person's leg of (a) shows better quality than that of (b).

Table 1. Illustration of the average PSNRs with 100 frames

	PDT		Proposed(CHPD)	
	$q_1 = 5 \ q_2 = 10$	$q_1 = 5 \ q_2 = 15$	$q_1 = 5 \ q_2 = 10$	$q_1 = 5 \ q_2 = 15$
Football	29.4	28.3	29.6	28.6
Foreman	31.1	29.5	31.3	29.7

Table 2 shows transcoding times for both methods without VLD, Q_1 , Q_2 and VLC. The table shows that sequences with little activity, such as Claire, News, and, Akiyo, take a shorter time to transcoder than they did in PDT, whereas the sequences with high activity, such as Foreman takes a longer time than that of PDT due to high computational matrix operations. Accordingly, the proposed method is more appropriate for transcoding video already compressed by MPEG-2 to high quality video, H.264, offline, however, it will be possible to transcode in real time in the near future.

Fig. 6. The PSNR performance for the sequence from frame 1 to 100. (a) For Football where q_1 and q_2 are set at 10 and 15, respectively. (b) For Foreman where q_1 and q_2 are set at 10.

Fig. 7. The real images for the sample video sequences at q_1 and $q_2 = 10$: Football (a)(b); Car-phone (c)(d); The PDT images are arranged in the first row, and the proposed method images are in the second row.

Table 2. Transcoding time of both methods (ms)

	Claire	News	Akiyo	Foreman
PDT	32.91	33.80	25.13	148.73
Proposed	30.15	32.45	24.81	206.94

5 Conclusions

In this paper, we proposed a half-pixel correction method for DCT-domain MPEG-2 to H.264 transcoding. The proposed method consists of CHPD and DCT_Conv. The former adds the input block to a correction block that contains the differences between two standards. The latter converts the 8×8 float DCT block into four 4×4 integer DCT blocks. The matrices are stored in memory to reduce computational complexity because they contain fixed constant values. The proposed method improves the video quality by reducing the MC phase. As far as we know, our method is the first correction scheme for the DCT-domain H.264 transcoding architecture. Our next research is to reduce the computational complexity for high activity video sequences.

Reference

1. T.Wiegand, G.J.Sullivan, G. Bjontegaard, and A.Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 560-576, July 2003
2. J. Ostermann, J.Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video Coding with H.264/AVC: Tools, Performance, and Complexity", *IEEE Circuits Syst. Magazine*, vol.4, no.1, pp.7-28, Apr. 2004.
3. Hari Kalva. "Issues in H.264/MPEG-2 Video Transcoding". Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE, 5-8 Jan. 2004
4. A. Vetro, C. Christopoulos, and H. Sun, "Video Transcoding Architectures and Techniques: An Overview," *IEEE Signal Processing Magazine*, vol. 20,no.2,p.18-29, March2003
5. ISO/IEC 13818-2:1995(E) pp.83~100
6. Iain E.G. Richardson, "H.264 and MPEG-4 Video Compression" WILEY , 2003
7. G.Cao, Z.Lei, J.Li, N.D.Georganas, Z.Zhu. "A Novel DCT Domain Transcoder for Transcoding Video Streams with Half-pixel Motion Vectors" *Real-Time Imaging (Elsevier Science)* Vol.10 Issue 5, Oc 2004, pp 331-337
8. T. Shanableh and M. Ghanbari, "Hybrid DCT/pixel domain architecture for heterogeneous video transcoding" *Signal processing: Image Communication*, vol.18, pp.601-620, 2003
9. Quantization/DCT Conversion Scheme for DCT-domain MPEG-2 to H.264/AVC Transcoding, *IEICE Trans. Communication*, July, 2005 accepted
10. <http://diml.yonsei.ac.kr/~wizard97/mpeg2/mpeg2v12.zip>
11. http://bs.hhi.de/~suehring/ttml/download/old_jm/jm82.zip