MPEG-4 to H.264 Transcoding using Macroblock Statistics

Yung-Ki Lee, Sung-Sun Lee, and Yung-Lyul Lee

Abstract— **In this paper, a temporal resolution reduction transcoding method that transforms an MPEG-4 video bitstream into an H.264 video bitstream is proposed. The block mode statistics and motion vectors in the MPEG-4 bitstream are utilized in the H.264 encoder for block mode conversion and motion vector interpolation methods. The proposed motion vector interpolation methods are developed not to perform brute-force motion estimation again in the H.264. In the experimental results, the proposed methods achieve 3~4 times improvement in computational complexity compared to the cascade pixel-domain transcoding, while the PSNR (peak signal to noise ratio) is degraded with 0.2~0.9dB depending on the bitrates.**

Index Terms— **MPEG-4, H.264, Motion Vector, Motion Estimation, Block Mode**

I. Introduction

Recently multimedia transmission is utilized widely in network environments. Especially, the video occupies high bandwidth in multimedia communications. Therefore, various multimedia compression standards have been established for faster video transmission and better quality. Currently, the H.264 (MPEG-4 AVC (Advanced Video Coding)) codec which has better coding performance than the previous coding standards [1]-[2] was completed [3]. As the number of content representation formats increase, the transcoding methods for video adaptation and digital library have been researched [4]-[5]. Using video transcoding techniques, the format of a pre-coded video can be converted to other formats to adapt to a lower transmission bandwidth or smaller display screen. In this paper, when transforming an MPEG-4 SP (Simple Profile) bitstream into an H.264 BP (Baseline Profile) bitstream by frame rate reduction [6]-[7], a new macroblock (MB) mode conversion based on MB statistics are proposed. Experimental results compare the cascade pixel-domain transcoding to the proposed transcoding method. This paper is organized as follows. Section II describes MB mode conversion from the MPEG-4 block modes to the H.264 block modes. In a reduced frame rate, the proposed MPEG-4 to H.264 transcoding methods by using motion vector interpolations is introduced in section III. Section IV shows the experimental results and the conclusions are provided in section V.

Manuscript received February 15, 2006.

The authors are with the School of Computer Engineering, Sejong University, 98 Kunja-Dong, Kwangjin-Gu, Seoul, 143-747, Korea. Also they are with DMS Lab.

E-mail : yllee@sejong.ac.kr

II. Proposed Block Mode Conversion

As shown in Table 1, the H.264 standard has some improved features such as the 4×4 integer transform, multiple reference frames, universal variable length coding (UVLC) or context-adaptive variable length coding (CAVLC), and the various block types for the quarter-pixels motion estimation (ME) and motion compensation (MC) in comparison to the MPEG-4. The H.264 performs the quarter-pixels ME/MC of the seven variable blocks such as the 16×16, 16×8, 8×16, and 8×8 MC units for each 16×16 MB and 8×4, 4×8 and 4×4 MC units for each 8×8 block. The H.264 encoder part of Fig. 1 performs the variable block size MC explained above. Therefore, there are many block modes in each MB, including the seven Inter modes, Intra16×16, Intra4×4, and SKIP modes, while the MPEG-4 performs the half-pixel ME/MC of the two blocks, namely the 16×16 and 8×8 blocks for each MB, so that the MPEG-4 has the Inter16×16, Inter8×8, Intra, and SKIP modes for each MB. Therefore the block mode conversion from the MPEG-4 MB to the H.264 MB should be estimated for MPEG-4 to H.264 transcoding. From the coding tools and block modes in each MB of Table 1, the H.264 has much higher computational complexity compared to the MPEG-4. However, thanks to the new complex coding tools in the H.264, the H.264 can compress approximately 1.5 times more data than the conventional H.263 or MPEG-4.

As shown in Fig. 1, the best performance of transcoding from an MPEG-4 bitstream to a H.264 bitstream is achieved by a cascade pixel-domain transcoding that first decodes the MPEG-4 video bitstream completely in the MPEG-4 decoder and then re-encodes the reconstructed video in the H.264 encoder. However, it requires high computational complexity

TABLE I
Coding tools of MPEG-4 SP (Simple Profile) and H.264 BP (Baseline Profile)

	MPEG-4	H.264
DCT	8×8 DCT	4×4 Integer Transform
ME/MC Unit	16×16, 8×8	16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4
MC Accuracy	1/2 pel	1/4 pel
VLC Table	Separable Table	Universal VLC, CAVLC
Intra Prediction	AC/DC Prediction (16×16Intra)	Spatial Prediction (16×16 or 4×4Intra)
Loop filter	None	Deblocking Filter

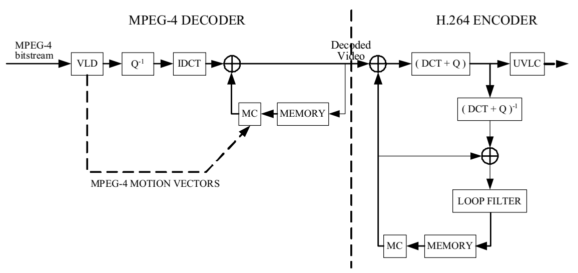


Fig. 1. Cascade MPEG-4 to H.264 transcoding

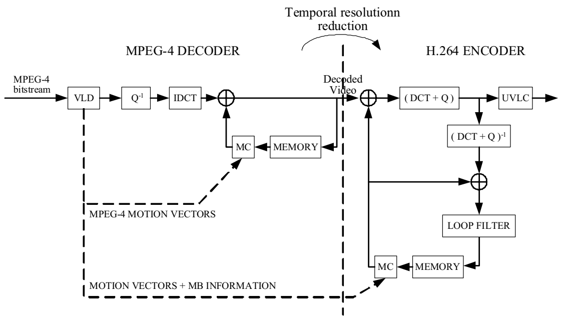


Fig. 2. Proposed MPEG-4 to H.264 transcoding to lower temporal resolution

because of performing ME/MC and block mode decision for every MB in the H.264. This is not suitable at a real time situation, so that we propose to re-use the incoming MPEG-4 motion vectors and block modes for the outgoing H.264 video bitstream. Fig. 2 shows the proposed temporal resolution reduction transcoding that reduces the computational complexity by using the MPEG-4 motion vector interpolation method and re-using the MPEG-4 block modes information in each MB. The proposed MPEG-4 to H.264 transcoding method can be used to reduce the bitrate requirements imposed by a network and to satisfy processing limitation imposed by a H.264 terminal. In order to reuse the MPEG-4 block mode in the H.264 encoder, the analysis of the block mode conversion ratio of MPEG-4 block mode to H.264 block mode is performed in the cascade transcoder for various test sequences in QCIF and CIF resolutions as shown in Table 2 and 3, respectively. The Inter16× 16 block mode in the MPEG-4 is converted to the Inter16× 16, Inter16× 8, Inter8× 16 or Inter8× 8 in the H.264 with the probability of 42.2%, 12.4%, 13.6% and 9.6% in Table 2 and with the probability of 43.6%, 13.5%, 12.8% and 14.4%, respectively, in Table 3. The SKIP mode in MPEG-4 is converted to SKIP or Inter16× 16 mode in the H.264 with the probability of 90.9% and 5.8% in Table 2 and with the probability of 89.4% and 7.2% in Table 3, respectively. To calculate the MB conversion statistics, the target bitrates are set to 130 ~250 kbps in QCIF resolution and 200 to 384 kbps.

The proposed MPEG-4 to H.264 block mode conversion method in Fig. 3 is decided by the analysis of the probability of frequently occurring H.264 block modes for a given MPEG-4 block mode from Table 2 and 3. In Table 2 and 3, the frequently occurring block modes are shown in dark gray.

Inter8× 16, Inter8× 8 or Skip block mode in H.264 by finally calculating the rate-distortion optimization (RDO) module [8] that selects the optimum block mode among the five blocks in consideration of both the minimum mean-square error and the minimum bit allocations for the Inter16× 16 block mode in the MPEG-4 in Fig. 3(a). If the RDO is calculated for all H.264 block modes such as the seven Inter, Intra16× 16, Intra4× 4, and SKIP block modes, it requires high computational complexity so that the possible block mode conversion is set as Fig. 3. The Inter8× 8 block mode in the MPEG-4 is converted to the Inter8× 8, Inter8× 4, Inter4× 8, Inter4× 4, Inter16× 16, Inter16× 8 or Inter8× 16 block mode in H.264 by using RDO among the seven block modes as shown in Fig. 3(b).

The SKIP mode in the MPEG-4 is converted to the SKIP, Inter16× 16 block mode in the H.264 by using the RDO among the two block modes as shown in Fig. 3(c). Finally, in Fig. 3(d), the Intra16× 16 block mode in the MPEG-4 is converted to the Intra16× 16 or Intra4× 4 block mode in the H.264 by using the RDO. Also, the Intra4× 4 mode in the H.264 that calculates the nine spatial directional prediction modes and the Intra16× 16

TABLE II
THE MB CONVERSION PERCENT OF MPEG-4 MB TYPES TO H.264 MB TYPE IN CASCADE TRANSCODING FOR VARIOUS QCIF SEQUENCES, IN WHICH THE TARGET BITRATES ARE APPROXIMATELY 130~250KBPS.

H.264 (output) / MPEG-4 (input) (%)	INTER MODE							INTRA MODE		SKIP	TOTAL
	16× 16	16× 8	8× 16	P8× 8				16× 16	4× 4		
				8× 8	8× 4	4× 8	4× 4				
INTER16× 16	42.2%	12.4%	13.6%	9.6%	1.7%	2.1%	1.2%	1.1%	0.8%	15.5%	100%
INTER8× 8	8%	11.8%	15.8%	28.6%	5.3%	6.7%	9.5%	0.7%	13.5%	0.1%	100%
SKIP	5.8%	1.1%	1.1%	0.4%	0.1%	0.1%	0%	0.5%	0%	90.9%	100%
INTRA16× 16	0%	0%	0%	2.6%	0%	0%	7.9%	21.1%	68.4%	0%	100%

TABLE III
THE MB CONVERSION PERCENT OF MPEG-4 MB TYPES TO H.264 MB TYPE IN CASCADE TRANSCODING FOR VARIOUS CIF SEQUENCES. , IN WHICH THE TARGET BITRATES ARE APPROXIMATELY 200~384KBPS.

H.264 (output) / MPEG-4 (input) (%)	INTER MODE							INTRA MODE		SKIP	TOTAL
	16× 16	16× 8	8× 16	P8× 8				16× 16	4× 4		
				8× 8	8× 4	4× 8	4× 4				
INTER16× 16	43.6%	13.5%	12.8%	14.4%	4.3%	4.4%	2%	0.8%	0.5%	3.7%	100%
INTER8× 8	15.3%	13.5%	12.8%	27.5%	8.6%	8.7%	8.5%	0.4%	4.7%	0.1%	100%
SKIP	7.2%	0.9%	0.8%	0.2%	0%	0%	0%	1.5%	0%	89.4%	100%
INTRA16× 16	5.9%	3.5%	3.5%	8.2%	2.4%	1.2%	14.1%	8.2%	52.9%	0%	100%

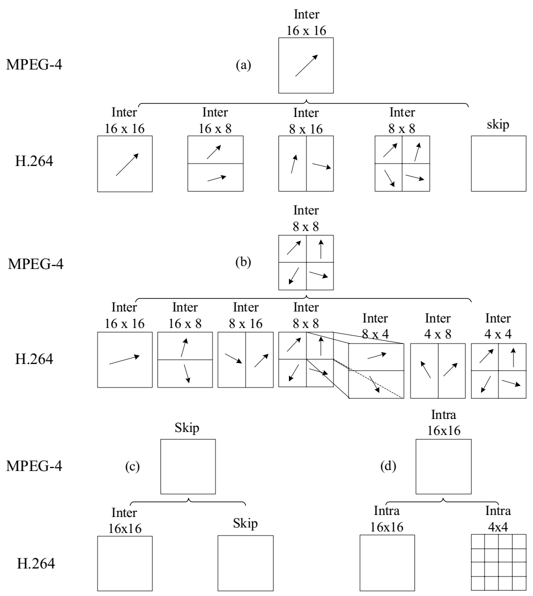


Fig. 3. Proposed MPEG-4 to H.264 block mode conversion

mode in the H.264 that calculates four spatial directional prediction modes need high computational complexity to perform the RDO. To select an optimum mode for the Intra4× 4 for each prediction, the RDO for estimating nine Intra prediction directions in each 4 × 4 Intra block mode is simplified by calculating only mean square error (MSE) of every Intra prediction direction instead of calculating rates and distortion of those prediction modes. The Intra16× 16 is dealt with in the similar way as the Intra4× 4. The block mode conversion from the MPEG-4 block modes to the H.264 block modes in Fig. 3 was experimentally decided to avoid unnecessary RDO calculations and to select efficient Intra prediction modes fast for all Intra block modes in the H.264 standard.

III. Motion Vector Interpolation in Reduced Frame Rates

The cascade transcoder requires high computational complexity because it must carry out ME/MC and MB mode decision (RDO) for each MB again. To reduce the computational complexity more, the proposed interpolation

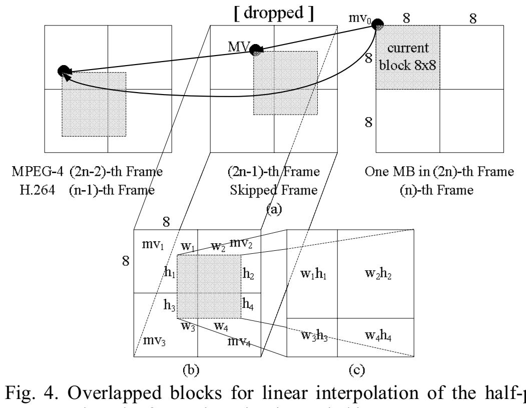


Fig. 4. Overlapped blocks for linear interpolation of the half-pixel motion vectors when the frame dorpping is needed in MPEG-4 to H.264 transcoding

methods for the motion vectors to reuse the MPEG-4 motion vectors in the H.264 are introduced. When the frame rate is reduced to a half for the bitrate reduction in network environments, the frame dropping is needed in transcoder. In Fig. 4(a), the n-th frame in H.264 corresponds to the (2n)-th frame of MPEG-4 and the (2n-1)-th frame of MPEG-4 is dropped. In order to find the motion vectors of the (n-1)-th frame in the H.264, the motion vector interpolation, MV, for the dropped (2n-1)-th frame in MPEG-4 should be performed.

A. Linear Interpolation (LI) of Motion Vectors

For motion estimation, the MPEG-4 standard has two Inter block modes, namely the 16× 16 and 8× 8 block mode for each MB. The motion vector of the 16× 16 block can be expressed as the motion vectors of the four 8× 8 blocks in the 16× 16 block. Therefore the motion vectors in all frames can be expressed as the motion vectors of 8× 8 block unit. Fig. 4(b) shows the overlapped blocks for linear interpolation of the motion vectors when frame dropping is needed in MPEG-4 to H.264 transcoding. The half-pixel motion vectors, mv_i, i=1,2,3,4, of the MPEG-4 in Fig. 4(b) are used for the motion vector interpolation process. If the interpolated integer motion vector in the (2n-1)-th frame in the MPEG is MV, the final integer motion vector in the (n-1)-th frame of the H.264 can be calculated by $((mv_0+1)/2+MV)$ as shown in Fig. 4(a). *MV* is calculated in Fig. 4 as follows:

$$MV = \frac{\sum_{i=1}^{4}(w_i \times h_i \times \frac{(mv_i+1)}{2})}{\sum_{i=1}^{4}(w_i \times h_i)} \tag{1}$$

where mv_i is a half-pixel motion vector of the current 8× 8 block in the (2n-1)-th frame of the MPEG-4. In eq. (1), w_i and h_i is the overlapped block width and height, and *MV* is the integer motion vector of the overlapped blocks. Fig. 4(b) and (c) show the linear interpolation process.

IV. Experimental Results

The experiments were conducted by using the MoMuSys decoder that supports the MPEG-4 SP and the JM73 (Joint Model) encoder that supports the H.264 (MPEG-4 Part10 AVC) BP. All experiments were carried out on a Pentium-IV 2.66 GHz, using several QCIF (Quarter Common Intermediate Format, 176× 144) and CIF (Common Intermediate Format, 352× 288) sequences. Each image sequence has 300 frames which were encoded in the MPEG-4 standard of 30Hz frame rates. All video tools for the H.264 BP encoder were used for MPEG-4 to H.264 transcoding. Only the first frame was encoded as an INTRA frame (I-frame), and the others frames were encoded as the INTER frames (P-frames) without the B frame for the transcoded bitstream. An MPEG-4 bitstream of 30 Hz was transcoded into an H.264 bitstream of 15 Hz. In order to improve coding efficiency due to incomplete motion vector interpolations in eq. (1), motion vector refinement is performed.

In order to decide the search window size of integer motion vector refinement, the bitstreams of several QCIF and CIF

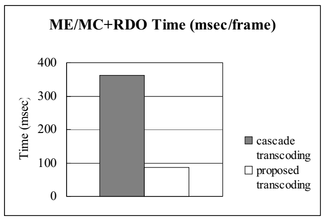
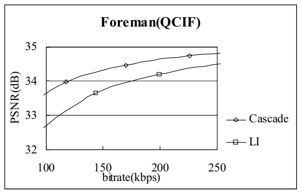


Fig. 5. Processing time comparison of the ME/MC+block mode decision parts of the proposed method to the cascade transcoding in average.

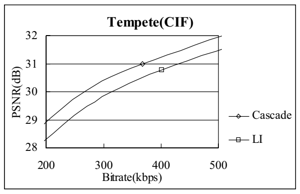
resolutions encoded in MPEG-4 are cascade-transcoded with increasing the motion vector window sizes by ±1, ±2, ±3, and so on. We set the search widow size to 1 from the simple experiments empirically.

The complexity of the cascade transcoder is compared to that of the proposed transcoder in Fig. 5. The computational complexity reduction of the proposed method comes from the ME/MC and block mode decision with RDO. The average processing times of the ME/MC and block mode decision between two transcoders are shown in Fig. 5.

The bitrate-PSNR curves of the cascade pixel-domain transcoding and the proposed transcoding methods using LI method with the proposed block mode conversion are compared for the "Foreman" QCIF sequence and "Tempete" CIF sequence in Fig. 6(a) to (b), where the horizontal axis represents the bitrate of transcoder and vertical axis represents the PSNR of motion vector interpolation method in comparison to the cascade pixel-domain transcoding. The proposed method for the QCIF sequence shows similar performance. When the proposed method is compared to the cascade pixel-domain transcoding, PSNR is dropped 0.2dB at near 250 kbps and maximum 0.9dB at near 130 kbps. In the CIF sequence, we obtained 0.4 ~ 0.5 dB PSNR loss compared with the cascaded transcoding. Fig. 7 shows the execution time of transcoding method on two size sequences where the horizontal axis represents the quantization parameter (QP) of the H.264 encoder and the vertical axis represents the total transcoder execution time. The proposed method is 3-4.1 times faster than the cascade transcoding in QCIF sequences and 3.3-4.3 times faster than that in CIF sequences. The proposed method can be considered as good interpolation method in the viewpoint of hardware implementation. Since one of important factors in the hardware implementation is the worst case complexity that provides the fixed maximum computational complexity, we believe that the hardware engineers will be favor of the proposed method. In conclusion, the proposed method that can be easily implemented on hardware can be considered as the best interpolation method in the aspect of hardware implementation.

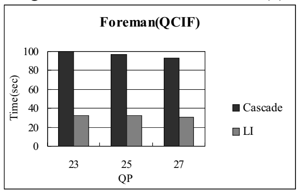


(a)

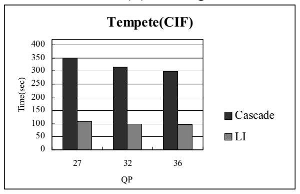


(b)

Fig. 6. Rate-Distortion curves: (a) "Foreman" and (b) "Tempete"



(a)



(b)

Fig. 7. Execution time comparison of the cascade pixel-domain transcoding to the proposed transcodings: (a) "Foreman" and (b) "Mobile&Calendar"

V. Conclusions

In this paper, the transcoding methods that transform MPEG-4 bitstream to H.264 bitstream are proposed by using the MPEG-4 to the H.264 block mode conversion statistics and the motion vector interpolation methods. Although the proposed methods result in small degradation in PSNR, the proposed methods requiring low complexity can be applied to MPEG-4 to H.264 transcoder for video adaptation applications.

References

[1] "Generic coding of Moving Pictures and Associated Audio Information: Video", ISO/IEC 13818-2. 2000.

[2] MPEG-4 Video Group, "MPEG-4 Video Verification Model Version 17.0", ISO/IEC/JTC1/SC29/WG11 N3515, Jul. 2000.

[3] "Joint Final Committee Draft (JFCD) of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)", Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Jun. 2003.

[4] Vetro, A.; Christopoulos, C.; Huifang Sun, "Video Transcoding Architecture and Techniques: An Overview", IEEE Signal Processing Magazine, vol.20, Issue: 2, Mar. 2003.

[5] J.L. Wu, S.J. Huang, Y.M. Huang, C.T. Hsu, and J.Shiu, "An efficient JPEG to MPEG-1 transcoding algorithm", IEEE Trans. Consumer Electron., Vol.42, Issue: 3, pp.447-457, Aug. 1996.

[6] Mei-Juan Chen, Ming-Chung Chu, and Chih-Wei Pan, "Efficient Motion Estimation Algorithm for Reduced Frame-Rate Video Transcoder", IEEE Trans. CSVT Vol.12, no.4, pp.269-275, Apr. 2002.

[7] Jeongnam Youn, Ming-Ting Sun, Chia-Wen Lin, "Motion Vector Refinement for High-Performance Transcoding", IEEE Trnas. Multimedia, Vol.1, no.1, pp.30-40, Mar. 1999.

[8] Gary J. Sullivan and Thomas Wiegand, "Rate-Distortion Optimization for video Compression", IEEE SIGNAL PROCESSING MAGAZINE, pp 74-90, Nov. 1998.