H.264 to MPEG-4 Transcoding Using Block Type Information

Jae-Ho Hur and Yung-Lyul Lee

***Abstract*—In this paper, we propose a heterogeneous transcoding method of converting an H.264 video bitstream into an MPEG-4 video bitstream. When the H.264 video bitstream is transformed into the MPEG-4 video bitstream, the conversions between the H.264 block types and the MPEG-4 block types are performed by minimizing the distortion, and 4×4 block-based motion vector mapping is performed. The proposed transcoder runs 5.2 times faster than the cascaded transcoding method, while maintaining the similar PSNR (peak-signal-to-noise ratio).**

***Index Terms*—Transcoding, H.264, MPEG-4, Motion Estimation, Motion Compensation.**

I. Introduction

MULTIMEDIA services for network environments, such as video communications, digital libraries and video adapt- ations, are becoming increasingly prevalent. In multimedia applications, it is often necessary to adapt the bitrates of the video bitstream to the bandwidths of the different kinds of communication channels. Therefore, various video transcoding techniques [1,2] have been developed to convert one comp- ressed bitstream format into another, while guaranteeing the QoS (quality of service). These different transcoding techno- logies are commonly classified according to the techniques which are used to perform the transcoding in the pixel-domain [3] and in the DCT (Discrete Cosine Transform)-domain [4]. So far, homogeneous transcoding methods have been developed to perform conversions from MPEG-2 to MPEG-2 [4] and H.263 to H.263 [5].

In this paper, a pixel-domain transcoding method is proposed for conversion between H.264 (ITU-T video coding standard and ISO/IEC MPEG-4 Part 10 Advanced Video Coding Standard) [6] and MPEG-4 [7], because DCT-domain transcoding cannot be employed in this case, due to the nonlinear loop filtering [8] included in the H.264 standard. In terms of the video quality, the best transcoding method is a cascaded pixel-domain transcoding technique that first decodes the encoded bitstream completely and then re-encodes the decoded video. However, using this method increases the computational complexity, because re-encoding the decoded video requires all of the video coding tools to be utilized. Therefore, a low complexity transcoder is developed which reuses the incoming information contained in each macroblock (MB) in the H.264 decoder. However, in order to take into account the case of small MPEG-4 compatible mobile devices which cannot decode the H.264 bitstream, a transcoder is required to convert the compressed bitstream format before it is transmitted to the MPEG-4 compatible mobile device.

The new transcoding method, which operates by block type conversion from the H.264 blocks to the MPEG-4 blocks is described in Section II, along with its motion vector (MV) mapping technology. The experimental results and analyses of the proposed algorithm are provided in Section III, and our concluding remarks are given in Section IV.

The authors are with the Department of Internet 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 Transcoding Method

As shown in Table I, the H.264 standard has some features which differ from those of the MPEG-4 standard, 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 used for motion estimation (ME) and motion compensation (MC). The H.264 standard performs quarter-pixel ME/MC for the seven variable blocks, as shown in Fig. 1(a). Therefore, there are many different block modes in each MB, including the seven Inter modes, and the Intra16×16, Intra4×4 and SKIP modes, as shown in Fig. 1(a). On the other hand, the MPEG-4 standard performs half-pixel ME/MC for the 16×16 and 8×8 blocks of each MB, with the result that each MB contains the Inter16×16, Inter8×8, Intra and SKIP modes, as shown in Fig. 1(b).

As shown in Fig. 2, in cascaded pixel-domain transcoding, the H.264 bitstream is first decoded by the H.264 decoder, and the decoded video is then re-encoded by the MPEG-4 encoder,

TABLE I
CODING TOOLS OF THE H.264 BP(BASELINE PROFILE) AND THE MPEG-4 SP(SIMPLE PROFILE).

Coding Tools	MPEG-4	H.264
Transform	8×8 DCT	4×4 Integer DCT
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	Spatial Prediction
Inner loop filter	None	Deblocking Filter

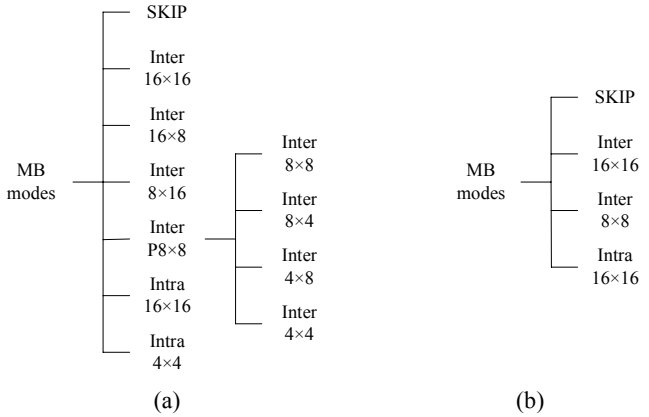

Fig. 1. MB modes in video coding standards: (a) MB modes in the H.264 standard, (b) MB modes in the MPEG-4 standard.

without re-using the information contained in each MB. The cascaded transcoder requires high computational complexity, because it must carry out ME/MC and the decisions for the MB modes again.

A. Block Type Conversion and Motion Vector Mapping

Performing brute-force ME and mode decision for each MB causes a transcoder to have high computational complexity. To reduce this computational complexity, the incoming motion vectors are used for motion vector mapping. In the proposed transcoder, the MPEG-4 encoder utilizes the motion vectors and MB information contained in each MB in the H.264 bitstream, as shown in the dotted lines in Fig. 2, thereby obviating the need to perform RDO (Rate-Distortion Optimization) and brute- force ME/MC. The H.264 standard adopts 1/4 pixel ME/MC, seven Inter-blocks, and Intra16×16, Intra4×4 and SKIP blocks, as shown in Fig. 1(a). Therefore, the block type conversion between the H.264 bitstream and the MPEG-4 bitstream needs to be performed by transforming the MB modes listed in Fig. 1(a) to those listed in Fig. 1(b). As a reference, in the case where there is no residual data, the SKIP mode in the H.264 standard can have the (0,0) motion vector or the PMV (median Predicted Motion Vector), whereas the SKIP mode in the MPEG-4 stand- ard can only have the (0,0) motion vector.

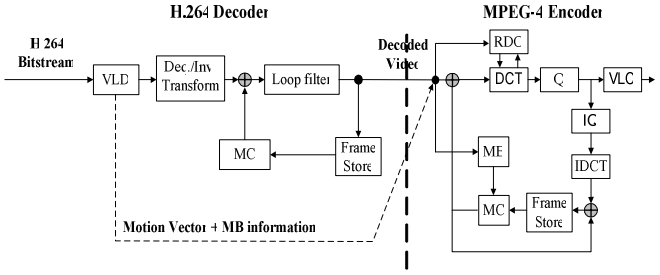

Fig. 2. Cascaded pixel-domain transcoding.

Before developing the proposed transcoder using block mode conversion, we investigated the block mode decision results in the case of an H.264 to MPEG-4 cascade transcoder, and the results are shown in Table II. The InterP8×8 modes, which amount to approximately 37.04% of the data in the H.264 block modes, are converted into the Inter16 × 16 (22.52%), Inter8×8 (12.88%) or SKIP (1.62%) mode in the MPEG-4 block modes, in which the SKIP mode is considered as the subset of the Inter16×16 mode when the motion vector is (0,0) and there is no residual data. The SKIP modes, which amount to approx- imately 24.86% of the data in the H.264 block modes, are mostly converted into the Inter16×16 block mode in the MPEG-4 block modes. The Inter16×16 modes, which amount to approximately 18.77% of the data in the H.264 block modes, are converted into the Inter16 × 16 (13.36%), SKIP (3.99%), INTER8×8 (1.42%) in the MPEG-4 block modes. The Inter- 16 × 8 modes, which amount to approximately 6.81% of the data, in the H.264 block modes are converted into the Inter16×16 (4.61%) or Inter8×8 (1.23%) block modes in the MPEG-4 block modes, and similar results were also observed for the Inter8×16 mode. The results for all of the block mode conver- sions in cascade transcoding are shown in Table II. The proposed transcoder is developed on the basis of the block occurrence probabilities listed in Table II. Fig. 3 describes the process of conversion of the block modes in the H.264 standard into the block modes in the MPEG-4 standard based on the results shown in Table II.

TABLE II
THE BLOCK MODE CONVERSION RESULTS OF THE CASCADED TRANSCODER FROM THE H.264 MB MODES TO THE MPEG-4 MB MODES FOR VARIOUS TEST SEQUENCES.

Modes of the MPEG-4 / Modes of the H.264	INTRA	SKIP	INTER 16×16	INTER 8×8	Total
Inter P8×8 (Fig. 3(a))	0.02%	1.62%	22.52%	12.88%	37.04%
Skip (Fig. 3(b))	0.00%	23.01%	1.73%	0.12%	24.86%
Inter 16×16 (Fig. 3(c))	0.00%	3.99%	13.36%	1.42%	18.77%
Inter 16×8 (Fig. 3(d))	0.01%	0.96%	4.61%	1.23%	6.81%
Inter 8×16 (Fig. 3(e))	0.01%	0.76%	4.30%	1.40%	6.46%
Intra 16×16, Intra 4×4 (Fig. 3(f))	1.33%	0.32%	2.23%	2.17%	6.05%

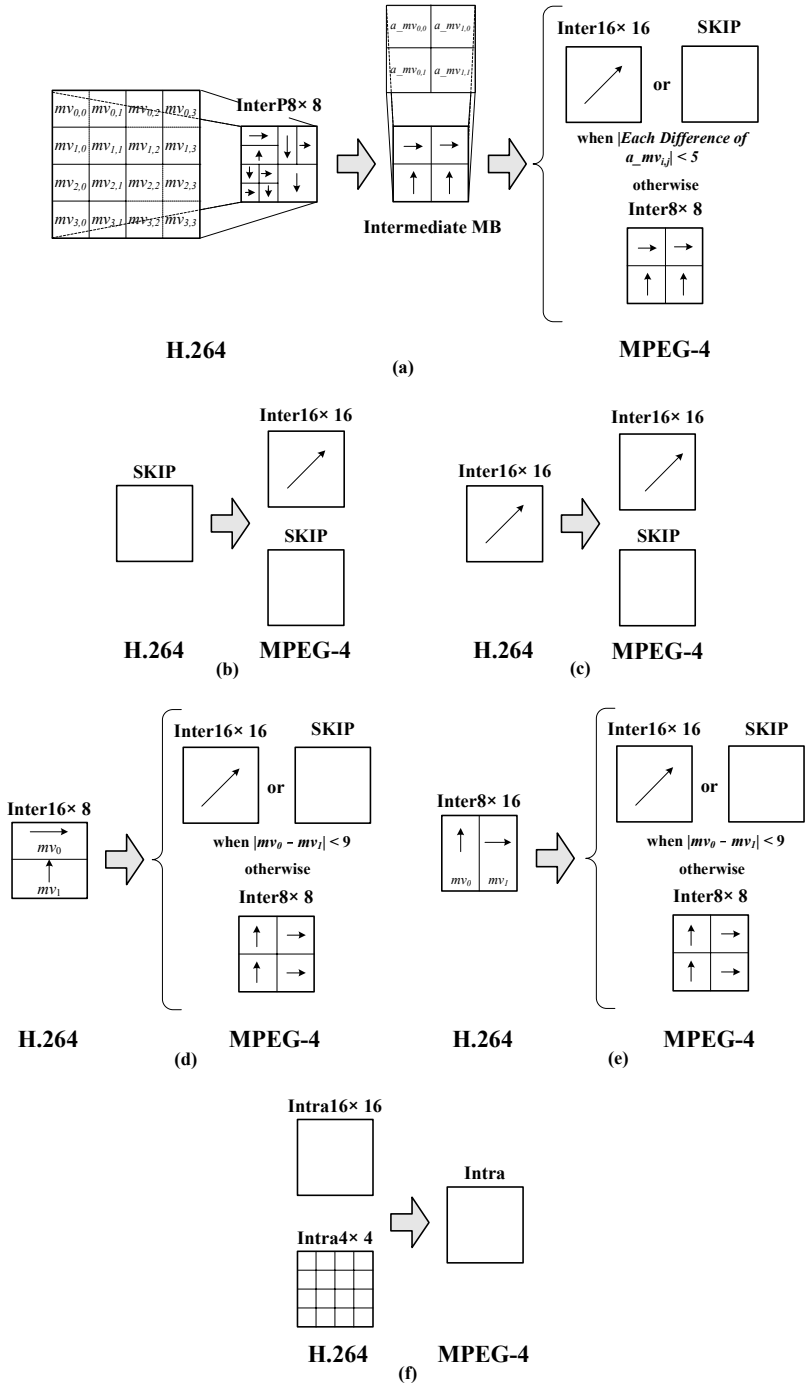

Fig. 3. Block type conversion and motion vector mapping.

1) Fig. 3(a) (H.264 MB mode is InterP8×8):

Since the basic unit of the motion vector in the H.264 standard is the 4×4 block, the 4×4 block-based quarter-pixel motion vectors, $mv_{k,l}$, $k,l=0,1,2,3$, are used to obtain the average Inter8×8 mode quarter-pixel motion vector, $a\_mv_{i,j}$, as shown in eq. (1):

$$a\_mv_{i,j} = \left(\sum_{k=2i}^{2i+1} \sum_{l=2j}^{2j+1} mv_{k,l} + 2 \right) >> 2, \qquad i,j = 0,1 \tag{1}$$

where the subscripts, i and j, denote the vertical and horizontal average motion vector indices of the four 8×8 blocks, and the subscripts k and l denote the vertical and horizontal motion vector indices of the sixteen 4×4 sub-blocks in each MB of the H.264 standard, respectively. Then, the block conversion process is applied such that if the difference values among each a_mvi,j vector are less than 5 and each a_mvi,j vector has the same direction, then the Inter16 × 16 mode is selected, otherwise the Inter8×8 mode is selected. A threshold value of 5 was chosen based on two experiments. The first experiment was performed in three cases. Case I is the conversion of the

MB mode including the InterP8×8 blocks into the Inter16×16 mode of the MPEG-4 block mode, Case II is the conversion of the MB mode including the InterP8×8 blocks into the Inter8×8 mode of the MPEG-4 block mode, and Case III is the conversion of the MB mode including the InterP8×8 blocks into the Inter16×16 or Inter8×8 mode (alternative method) depending on the difference values between the *a_mvi,j* vectors, as shown in Fig. 4. Fig. 4 shows that Case I provides a better result than both Case II and Case III at low bitrates in terms of the PSNR, Case II provides a better result than Case I and Case III at high bitrates, and Case III (alternative method) using the threshold T=5 which depends on the difference values between each *a_mvi,j* vector provide intermediate results at all bitrates.

We selected the alternative method, since our target bitrates are approximately 130 ~ 180 kbps for a mobile device. However, Case I and Case II can also be applied, depending on the application bandwidth. After applying the alternative method using various thresholds in the second experiment, we found that the best result was obtained when T is set to 5.

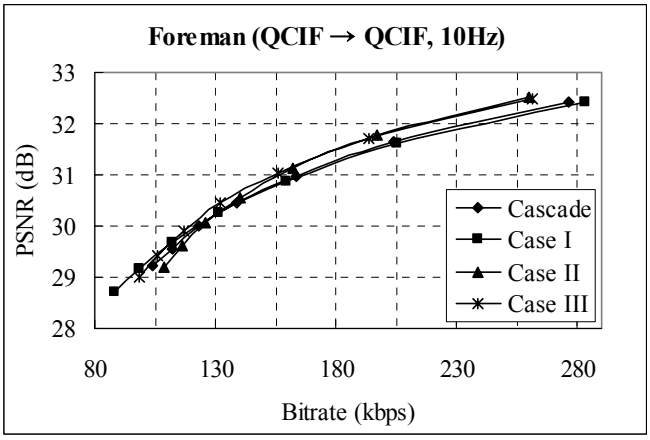

Fig. 4. PSNR comparisons when case I (all 16×16), case II (all 8×8), and case III (alternative method) are applied for various bitrates.

2) Fig. 3(b) (H.264 MB mode is SKIP):

The SKIP mode of the H.264 bitstream can be converted into the SKIP mode or Inter16×16 mode of the MPEG-4 block modes based on Table II. First of all, every SKIP mode in the H.264 bitstream is changed into an Inter16×16 mode in the motion vector refinement process, since the SKIP mode of the MPEG-4 standard is one of the Inter16×16 modes when the motion vector is (0,0) and CBP is zero. Then, the Inter16×16 mode is converted into the SKIP mode, only if both the motion vector of the Inter16×16 mode resulting from the motion vector refinement process is zero and CBP is set to zero. Otherwise, the mode is set to the Inter16×16 mode of the MPEG-4 block mode. The motion vector refinement process will be explained in Section II.B.

3) Fig. 3(c) (H.264 MB mode is Inter16×16):

The Inter16×16 mode can be converted into the Inter16×16 mode or the SKIP mode, in accordance with the same rule as that described above for the SKIP mode conversion.

4) Fig. 3(d) and (e) (H.264 MB mode is Inter16×8 or Inter-8×16):

The two H.264 MB modes, Inter8×16 and Inter16×8, from Table II can be converted into the Inter16×16, Inter8×8 or the SKIP mode. First, the difference between the two motion vectors is calculated in quarter-pixel units, i.e. $|mv_0\text{-}mv_1|$. If the difference between the motion vectors is smaller than 9, and mv_0 and mv_1 have the same direction, the Inter16×16 block mode is selected, otherwise the Inter8×8 block mode is selected. The threshold value of 9 was obtained in a similar way to that described in Fig. 4.

5) Fig. 3(f) (H.264 MB mode is Intra16×16 or Intra4×4):

The Intra16×16 and Intra4×4 block modes of the H.264 bitstream can be converted into the Intra16×16, Inter16×16, or Inter8×8 mode of the MPEG-4 bitstream in the cascaded transcoding process, as shown in Table II. However, in order to reduce the computational complexity, the Intra16×16 and Intra4×4 block modes of the H.264 bitstream are directly converted into the Intra16×16 mode of the MPEG-4 bitstream. This direct conversion does not affect the PSNR value, since this kind of Intra MB only amounts to 1.36% of the MPEG-4 bitstream.

B. Motion Vector Refinement

In order to improve the coding efficiency, which can be degraded by the motion vector mapping calculations corresponding to eq. (1), motion vector refinement is performed. First of all, the quarter-pixel motion vector is 1/4 scaled to the integer-pixel motion vector. A search for the ±1 integer-pixel motion vector is performed, and then a half-pixel motion search involving the eight-neighbor half-pixels immediately surrounding the integer-pixel motion vector is performed to find the best half-pixel motion vector. To determine the optimum search window of the integer motion vector to use for the motion vector refinement, various search ranges in integer units were experi- mented with. When a motion vector refinement search window of ±1 was applied, the PSNR values of this search window were almost saturated in comparison with those of a higher motion vector refinement search window.

III. Simulation Results

The proposed transcoding method was implemented using the H.264 JM 7.3 decoder and the MPEG-4 MoMuSys-FDIS-V1.0 encoder. In order to evaluate the performance of the proposed transcoder, the PSNR values, bitrates and computation times were analyzed for the "Foreman" sequence of the QCIF (Quarter Common Intermediate Format), the "News" sequence, "Paris" sequence and "Coast" sequence of the same format, all of which have 300 frames. The H.264 frame rate was set to 10 frames per second (fps) for the evaluation. For the experiments, each sequence was compressed with the "I, P, P, P,…" scheme, i.e. the first frame is an INTRA frame and the other frames are all INTER frames without any B frames being included. The computation times of the proposed transcoding method were compared with those of the cascaded transcoding method, as shown in Fig. 5. The average speed of the proposed transcoder is 5.2 times higher than that of the cascaded

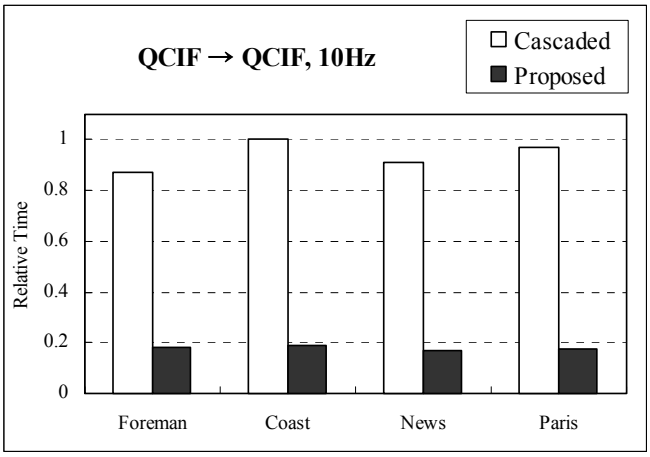

Fig. 5. Transcoding time comparisons between the cascaded and proposed transcoding methods, in which a relative time comparison is performed for various QCIF sequences.

transcoding method. Quantization values of 30 and 12 were used in the H.264 and MPEG-4 bitstreams, respectively, for our target bitrates.

Fig. 6 shows the plots of the PSNR-bitrate for the cascaded transcoding method vs. that of the proposed transcoding method. The PSNR of the proposed method is almost the same as that of the cascaded transcoding method for the "News" and "Paris" sequences, and is 0.2 dB higher than that of the of the cascaded transcoding method in the case of the "Foreman" sequence. However, the PSNR of the proposed method is 0.3 dB lower than that of the cascade transcoding method in the case of the low bitrate "Coast" sequence. According to our experiment results, the performance of the proposed transcoding method is almost identical to that of the cascaded transcoding method.

IV. Conclusion

We developed a low-complexity H.264 to MPEG-4 video transcoder by re-using the 4×4 block based-H.264 motion vectors and utilizing H.264 MB information. Through the results of the simulation, we showed that the proposed transcoding method is able to reduce the computational complexity without degrading the video quality. Therefore, the proposed transcoding method can be used in practical multimedia applications, such as digital libraries and video adaptations, in which the users only have an MPEG-4 compatible terminal.

References

[1] J. Jeongnam Youn, M.T. Sun, and C.W. Lin, "Motion Vector Refinement for High Performance Transcoding," *IEEE Trans. Multimedia*, vol.1, no.1, pp.30~40, March 1999.

[2] A. Vetro, C. Christopoulos, and H. Sun, "Video Transcoding Architectures and Techniques: An Overview," *IEEE Signal Processing Magazine*, March 2003. pp18-29.

[3] P. Yin, M. Wu, and B. Lui and H. Sun, "Drift compensation for reduced spatial resolution transcoding," *IEEE Trans. Circuits Syst. Video Technol*., vol. 12, pp.1009-1020, Nov. 2002.

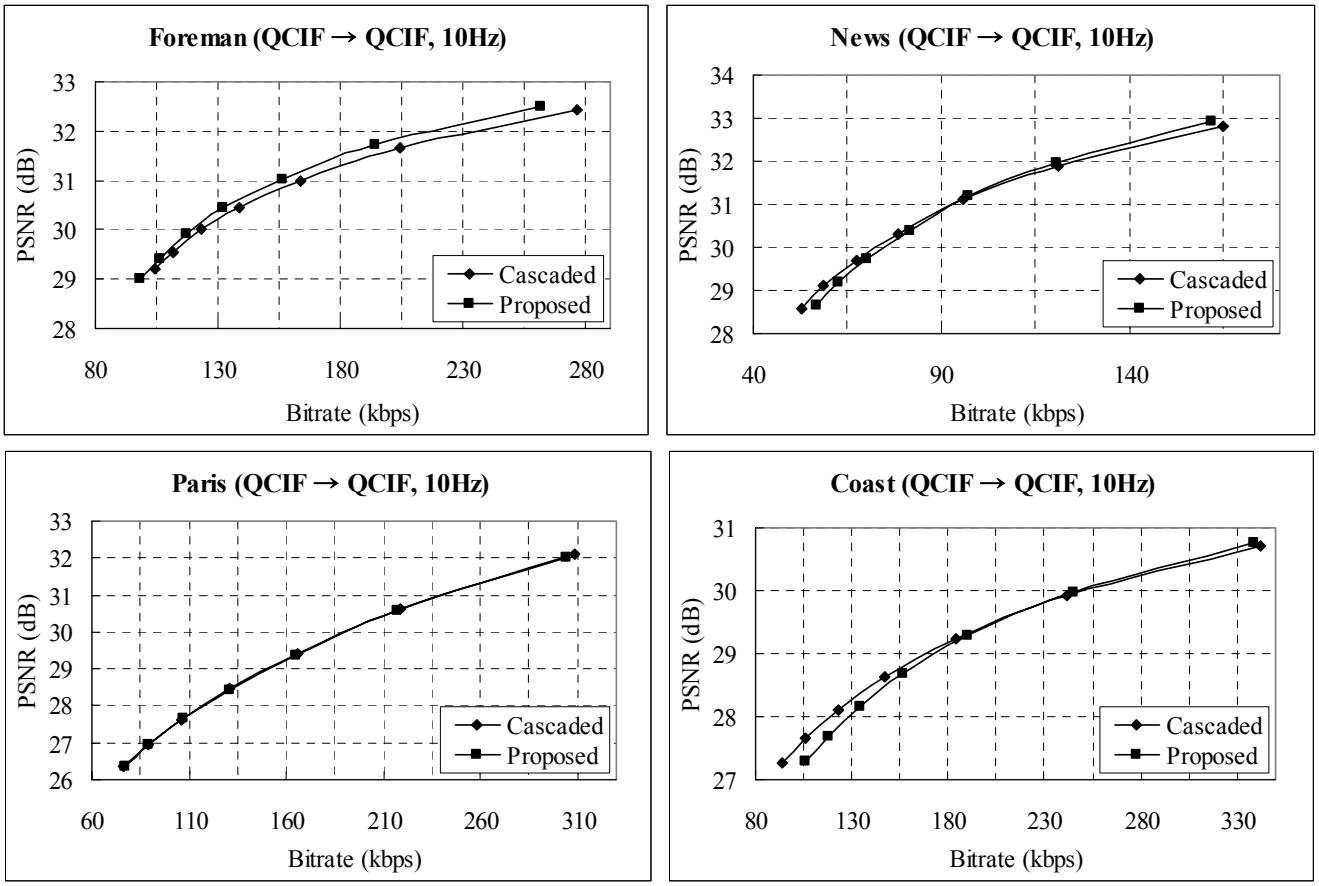

Fig. 6. Comparisons of the PSNR between the cascaded and proposed transcoding methods.

[4] P. Assuncao and M. Ghanbari, "A frequency Domain video transcoder for dynamic bit-rate reduction of MPEG-2 bitstreams," *IEEE Trans. Circuits Syst. Video Technol*., vol. 8, pp.953~967,Dec. 1998.

[5] H. Sun, W. Kwok, and J. W. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Trans. Circuits Syst. Video Technol*., vol.6, pp. 191-199, Apr. 1996.

[6] Tomas Wiegand, Joint Final Committee Draft(JFCD) of joint Vide specification(ITU-T Rec. H.264 |ISO/ICE 1449-10 AVC), JVT-G050, March 2003.

[7] Weiping Li, Jens-Rainer Ohm, Mihaela van der Schaar, Hong Jiang, Shipeng Li, "MPEG-4 Video Verification version 17.0", ISO|IEC JTC1|SC29|WG11 N3515, July 2000.

[8] Peter List, Anthony Joch, Jani Lainema, Gisle Bjøntegaard, and Marta Karczewicz, "Adaptive Deblocking Filter," *IEEE Trans. Circuits Syst. Video Technol*., vol.13, no.7, pp. 614-619, Jul. 2003.