LOGO INSERTION TRANSCODING FOR H.264/AVC COMPRESSED VIDEO

Di Xu and Panos Nasiopoulos

Department of Electrical and Computer Engineering, University of British Columbia, B.C., Canada
Email: {dixu, panos}@ece.ubc.ca

ABSTRACT

H.264/AVC quickly gains ground in many aspects of video applications, due to its superior coding performance. Inserting a company logo into H.264/AVC compressed video streams has been a highly desirable application in the TV telecasting industry. In this paper, we propose a novel and efficient logo-insertion scheme for H.264/AVC compressed videos. Our proposed scheme overcomes the numerous coding dependencies, and minimizes the changes to the original compressed videos. Experimental results show that our proposed transcoding scheme achieves extraordinary video quality and significantly reduces the bit rate and computational cost. Compared with the cascaded transcoding scheme, our proposed logo-insertion method achieves an average of 1.16 dB PSNR increase, or a 68.6% bit-rate reduction. Our scheme also dramatically reduces the total transcoding time and the motion-estimation time by at least 67.2% and 97.4%, respectively.

***Index Terms*—**Logo insertion, video transcoding, H.264/AVC, compensation, TV broadcasting.

1. INTRODUCTION

Inserting a company logo into compressed video streams is a highly desirable application in the telecasting industry. Adding a logo to videos has been proven to be the most effective way for advertising a telecasting station. A distinct logo assists the audience to easily identify a TV channel, but mainly it can be used as a watermark, helping broadcasters and content providers to protect their ownership rights. For this reason, it is highly desirable to insert the logo at the transmitting end rather than using technologies (such as Flash Video and Microsoft Silverlight which are mainly designed for programming and delivering video on the Internet) that add logos at the receiver end.

The challenge of logo insertion is that a large amount of video content that reaches a TV station is already encoded, and the only straightforward approach to adding a logo is through the cascaded pixel-domain transcoding. This approach first completely decodes the entire bitstream, then adds a logo, and finally re-encodes the new video sequence, as shown in Fig. 1. However, such an approach is not desirable by content providers since they lose control over the original quality, let alone that re-encoding the new stream will always degrade the picture quality (or increase the bit rate). It is, therefore, desirable to design a scheme that adds the logo to the compressed video without significantly affecting the original quality of the content.

A practical way for logo insertion is through efficient transcoding of a small area of the picture where the logo is located, avoiding complete decoding and encoding of the rest of the frame. An efficient transcoding scheme may greatly reduce the computational cost for logo insertion. It also has the potential of resulting in higher video quality than the cascaded transcoding by reducing the re-quantization error resulting from decoding and re-encoding a video stream.

Fig. 1. Flowchart of cascaded logo-insertion transcoding scheme.

Several logo-insertion methods have been proposed for logo insertion in MPEG-2 compressed streams [1]–[3]. Currently, H.264/AVC, which is regarded as the most advanced video coding standard, has been shown to outperform MPEG-2 by 50% and is destined to replace MPEG-2 in every aspect of video communications [4], [5]. For this reason, it is desirable to consider logo insertion for H.264/AVC compressed video, a topic not addressed so far by researchers. The challenges in this case come from the much more advanced coding features supported by H.264/AVC, which were not present in MPEG-2. Such features include intra prediction, deblocking filter, intra 4×4 prediction mode coding, and adaptive entropy coding, all of which make previously developed logo-insertion methods for MPEG-2 not applicable to H.264/AVC compressed videos.

In this paper, we propose a logo-insertion scheme for H.264/AVC pre-encoded video streams. Our proposed scheme overcomes the numerous challenges induced by the H.264/AVC coding standard. Compared with the cascaded transcoding scheme, an average of 1.16 dB PSNR gain, 72.6% computational time reduction, and 97.7% motion-estimation time reduction are achieved by our proposed method.

The remainder of the paper is structured as follows. Section 2 discusses the challenges of H.264/AVC logo-insertion transcoding. Our proposed logo-insertion method is described in Section 3. Section 4 presents the performance evaluation of our method. Conclusions are drawn in Section 5.

2. CHALLENGES OF H.264/AVC LOGO INSERTION

Various coding dependencies are the main challenges of H.264/AVC logo insertion. Compared with MPEG-2, the coding dependencies in H.264/AVC take more advantage of quantity correlation, and accomplish higher coding performance. They, however, become major challenges for logo insertion, since changing the coding of the logo area without affecting the coding of other areas is very difficult, given the existence of coding dependencies. These unique challenges make existing MPEG-2 based logo-insertion schemes not applicable to H.264/AVC compressed videos.

The coding dependencies in H.264/AVC can be classified into the video-content dependencies (i.e., dependencies among pixel values) and data-coding dependencies (i.e., dependencies among coding parameters).

Video-content dependencies are mainly caused by intra and inter predictions. Intra macroblocks (MBs) are coded differently in MPEG-2 and H.264/AVC. In MPEG-2, intra MBs are DCT-transformed without prediction in spatial domain. In H.264/AVC, however, intra prediction is used, which induces dependencies among pixel values in intra slices.

Inter MBs are predictively coded in both MPEG-2 and H.264/AVC. Compared with MPEG-2, the additional features, such as flexible macroblock (MB) partitions for motion compensation, multiple reference frames, and quarter-pixel prediction, in H.264/AVC offer higher prediction accuracy, while causing more content dependencies and complicating logo insertion. Furthermore, the deblocking filter used in H.264/AVC is another way of creating video-content dependencies.

Data-coding dependencies are due to the use of predictive or adaptive parameter-coding schemes. Such coding schemes include the coding of intra 4×4 prediction modes, the coding of motion vectors, the context-adaptive variable length coding (CAVLC), and context-adaptive binary arithmetic coding (CABAC).

Among the aforementioned schemes, encoding intra 4×4 prediction modes and encoding motion vectors are both based on predictive coding. In these cases, some form of prediction residual values rather than the actual parameters (i.e., intra modes or motion vectors) are coded. A change of one parameter is very likely to affect other parameters that are predictively coded based on the changed value. Compensations are a must to change a single parameter only.

Both CAVLC and CABAC are adaptive entropy-coding schemes to further compress quantized integer-transform coefficients. The scheme for coding coefficients of one block depends heavily on the properties of its neighbouring blocks, with CABAC having heavier dependency than CAVLC. For this reason, changing the coding of one block without affecting the others proves extremely challenging.

3. PROPOSED LOGO INSERTION SCHEME

Our proposed logo-insertion transcoding scheme is designed to overcome all aforementioned challenges. Pixel-domain transcoding and transform-domain transcoding are two categories of transcoding approaches. To properly handle the deblocking filter during the logo-insertion process, we need to work in the pixel domain in order to compute the gradient information of pixels, a process required when applying the deblocking filter [6]. For this reason, we chose to use pixel-domain transcoding for our proposed method.

For ease of explanation, in this paper, we refer to the frame region that is covered by a logo as the "logo part", depicted by the diagonally shaded area in Fig. 2. We denote the region that is directly affected by the logo insertion due to video-content dependencies as the "logo-affected part1" and the region directly affected by data-coding dependencies as the "logo-affected part2". These parts are illustrated by the vertically shaded area and horizontally shaded area in Fig. 2. The un-shaded area is called the "logo-unaffected part".

The efficiency of a logo-insertion transcoding scheme depends entirely on minimizing the changes caused on the original video stream. In our proposed logo-insertion method, we aim at quickly stopping change propagations caused by the video-content and data-coding dependencies. The architecture of our proposed scheme is illustrated by the flowchart shown in Fig. 3. We first decode and re-encode the "logo part". Then, we adjust the coding of

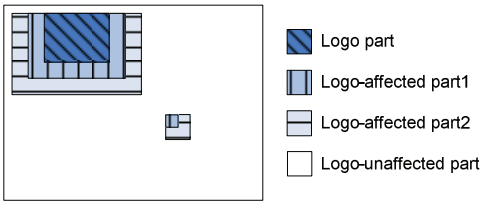

Fig. 2. Slice partition sketch map for our logo-insertion scheme.

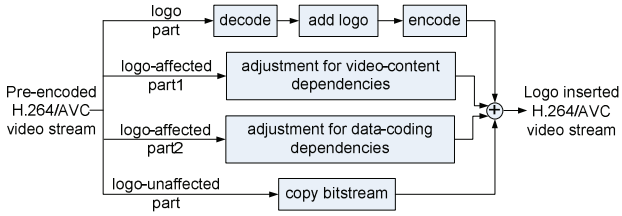

Fig. 3. Flowchart of our proposed logo-insertion scheme.

the "logo-affected part1" and "logo-affected part2" to stop change propagations caused by video-content and data-coding dependencies, respectively. Finally, the bitstream that corresponds to the "logo-unaffected part" is directly copied to the target stream.

For logo insertion at the "logo part", the new video content $P'(x, y, n)$ is usually expressed as a linear combination of the logo value $L(x, y)$ and the original video content $P(x, y, n)$, in the following form:

$$P'(x,y,n) = \alpha \times L(x,y) + (1-\alpha) \times P(x,y,n), \quad (1)$$

where (x, y) is the pixel position, n is the frame index in a video sequence, and α is the transparency factor of the logo. The value α is in the range of $0 < \alpha \leq 1$. In particular, when $\alpha = 1$, the original video content at the "logo part" is completely replaced by the logo, giving rise to an opaque logo overlapping. In our logo-insertion scheme, we re-encode the "logo part" to reflect the video-content changes as shown in (1).

In order to stop the "domino effect" of change propagations caused by various dependencies, the "logo-affected part1" and "logo-affected part2" work as buffer zones in our scheme. If compensations for video-content and data-coding dependencies take place properly within these regions, the change propagation caused by logo insertion can be stopped quickly, leaving the "logo-unaffected part" unaffected. The following two subsections describe in detail our compensation algorithms for the video-content dependencies and data-coding dependencies.

3.1. Adjustment for Video Content Dependencies

In this section, we present our scheme for stopping video content change propagation due to logo insertion. Such propagation is caused by intra prediction, inter prediction, deblocking, and sub-pixel interpolation. The directly affected area is "logo-affected part1". It includes the MBs directly to the right, left, and bottom of the "logo part", which could potentially be affected and start propagating changes by intra prediction, sub-pixel interpolation, and deblocking filters. In addition, the MBs with motion vectors pointing to the "logo part" are affected by inter prediction.

For the "logo-affected part1" in intra slices, we apply lossless coding to ensure that the pixel values remain unchanged in this part. Hence, content change propagation is effectively stopped. For the "logo-affected part1" in inter slices, we choose to re-encode it. Although small requantization errors may occur and possibly propagate to the subsequent inter frames in the same group of

pictures (GOP), the re-encoding scheme turns out to be a good trade-off between video quality and bit rate.

In order to reduce the computational cost of our scheme, we consider using two potentially effective pre-determined motion vectors (i.e., a zero motion vector and the original motion vector that is used in the pre-encoded video stream) instead of completely re-doing motion estimation to re-encode "logo part" and "logo-affected part1" in inter frames.

We consider three different MB scenarios for motion compensation: I) MBs in the "logo part" and not predicted from the "logo part", II) MBs in the "logo-affected part1" and predicted from the "logo part", and III) MBs in the "logo part" and also predicted from the "logo part". Since in H.264/AVC, prediction residuals are coded rather than the actual pixel values, here we show the modified residual $R'(x,y,n)$ for Scenarios I, II, and III in (2a)(2b), (3a)(3b), and (4a)(4b), respectively.

$$R'(x,y,n)=\begin{cases} R(x,y,n)+\alpha[L(x,y)-P(x,y,n)], & \text{when } mv'=mv \quad (2a)\\ (1-\alpha)[P(x,y,n)-P(x,y,n-1)], & \text{when } mv'=0 \quad (2b)\end{cases}$$

$$R'(x,y,n)=\begin{cases} R(x,y,n)-\alpha[L(x+\Delta x,y+\Delta y)-P(x+\Delta x,y+\Delta y,n-1)], & \text{when } mv'=mv \quad (3a)\\ R(x,y,n)-P(x,y,n-1)+P(x+\Delta x,y+\Delta y,n-1), & \text{when } mv'=0 \quad (3b)\end{cases}$$

$$R'(x,y,n)=\begin{cases} (1-\alpha)R(x,y,n)+\alpha[L(x,y)-L(x+\Delta x,y+\Delta y)], & \text{when } mv'=mv \quad (4a)\\ (1-\alpha)[P(x,y,n)-P(x,y,n-1)], & \text{when } mv'=0 \quad (4b)\end{cases}$$

where mv and mv' denote the original and modified motion vectors, respectively.

From the above equations, we conclude that when α is small, using the original motion vector as mv' results in an accurate motion estimation, i.e., small modified residual $R'(x,y,n)$. When α is large, using zero motion vector as mv' (for Scenarios I and III) works relatively well.

3.2. Adjustment for Data Coding Dependencies

In H.264/AVC, not only video content is coded based on previously-coded values, but many coding parameters are also predictively or adaptively coded based on their neighboring data. Such coding parameters include intra 4×4 prediction modes, motion vectors, CAVLC, and CABAC. In what follows, we present our proposed compensation method for stopping the change propagation caused by each of the aforementioned coding parameters.

Since the intra 4×4 prediction mode coding, motion-vector coding, and CAVLC scheme use only the neighboring information as coding reference, the potential change propagation caused by such coding can be stopped locally. In our logo-insertion scheme, the propagation is stopped in "logo affected part2", the direct neighborhood of "logo affected part1".

In the process of coding intra 4×4 prediction mode, the actual value of the prediction mode is not directly coded. Instead, a *MostProbableMode* is first predicted according to the availabilities and prediction modes of its left and upper blocks. A flag is sent to signify if the *MostProbableMode* is used as the actual intra 4×4 prediction mode. If the flag is ON, the actual mode is the same as the *MostProbableMode*, and no other information is needed to correctly decode the actual mode. If the flag is OFF, a residual mode is sent to indicate which of the modes other than the *MostProbableMode* is used as the actual prediction mode.

In order to stop the propagation of the intra 4×4 prediction mode changes, for every MB that is directly to the right or on the bottom of the "logo-affected part1", we check if its *MostProbableMode* is changed due to the re-encoding of the "logo part" and the "logo-affected part1". If the *MostProbableMode* is not changed, no coding adjustment is needed. If it is changed, the mode coding needs to be modified to reflect the change. The modification process involves locating the residual mode in the original bitstream, decoding the residual mode, calculating the actual mode, computing and encoding the new residual mode, and putting the new encoded residual mode into the correct position of the transcoded bitstream.

The next step in our algorithm involves an adjustment that stops the propagation of motion-vector changes. During the process involved in coding a motion vector, first a predicted motion vector, *mvp*, is calculated based on some previously-encoded reference motion vectors. Other than some special cases, the predicted motion vector *mvp* is the median of the three reference motion vectors associated with blocks to the left, top, and top right of the current block. After predicting *mvp*, the motion-vector residual *mvr* between the actual motion vector and the predicted vector *mvp* is encoded and transmitted.

During logo insertion, some of the "logo part" and "logo-affected part1" are re-encoded using different motion vectors. For those MBs that are directly to the right, on the bottom, or the bottom left of the "logo part" or "logo-affected part1", their predicted motion vectors may be affected by the re-encoding. In order to keep their actual motion vectors unchanged, coding compensations need to be performed. For a MB in the "logo-affected part2", we first compute a new predicted motion vector *mvp'* based on its neighboring data. Then, we modify its motion-vector residual as $(mvr+mvp-mvp')$ in order to compensate for the change to the predicted motion vector. At last, the new motion-vector residual is re-entropy coded. In so doing, the propagation of motion-vector changes stops at the "logo-affected part2".

The next step involves entropy-coding dependencies. In the case of CAVLC, the number of nonzero transform coefficients is coded using a look-up table. The choice of look-up tables depends on the number of nonzero coefficients in neighboring blocks. In our scheme, for each MB in the "logo-affected part2" and also directly to the right or on the bottom of the "logo-affected part1", we check if its predicted number of nonzero transform coefficients is changed. If this number is unchanged, the corresponding bitstream remains the same. Otherwise, we may need to re-do the entropy coding according to the newly predicted number of nonzero transform coefficients and the corresponding look-up table.

In the case of CABAC, probability models used in the entropy coding of transform coefficients are selected based on local statistics of already encoded information. This highly adaptive entropy-coding scheme imposes extremely strong dependency among neighboring blocks. For this reason, the addition of a logo causes changes that make it almost impossible to avoid repeating the entropy coding step for all the blocks that are coded after the "logo part" in each slice. Thus, our scheme performs CABAC entropy coding for all MBs that are encoded after the "logo part".

4. EXPERIMENTAL RESULTS

Our proposed transcoding scheme was implemented based on the H.264/AVC reference software JM13.2 [7]. Several representative

and commonly used YUV video sequences of size 640×480 were employed in our experiments. All video sequences were set to 30 frames per second, with an "I,B,B,P,B,B,P, …"GOP structure and GOP length of 15. A total number of 300 frames were used for each sequence. The input bitstreams were generated using quantization parameter equal to 28. The logo used is the "AChannel" with a 68×70 pixel size and 0.6 transparency factor.

Very limited bit-rate control is provided by our proposed logo-insertion method. This is because the majority of the pre-encoded bitstream is either directly copied to the targeting transcoded bitstream or it is slightly adjusted. Only a small portion of a video is re-encoded, where the re-encoding quantization parameter is the main bit-rate control factor. Furthermore, in order to get a perceptually uniform video quality, we use the same quantization parameter as that used in the pre-encoded stream. In so doing, no bit-rate control needs to be provided by our method.

We compare the rate distortion between our proposed method and the cascaded approach in Fig. 4. The rate-distortion curve for the cascaded transcoding method was generated by using different quantization parameters at the re-encoding stage. The PSNR values were computed using the original uncompressed YUV file with the logo added as a reference. All PSNR values are averages over 300 frames. From Fig. 4, we observe that for each of the three sequences and for the same bit rate, our proposed transcoding scheme outperformed the cascaded transcoder by at least 1.13 dB in terms of PSNR. An average of 1.16 dB PSNR increase is achieved. We also observe that, at the same PSNR, our proposed transcoding method yields bit savings that range from 66.8% to 70.8% compared with the cascaded method. Although the lossless coding we proposed consumes 12.32% to 17.95% more intra-frame bit rate than the cascaded approach, the resulting less re-quantization errors yield better picture quality for the transcoded videos. Note that the highest resulting PSNR for both methods is determined by the quality of the pre-encoded video.

Fig. 5 presents the average relative computational time that our method and the cascade method used for different test sequences. These results indicate that our proposed transcoder reduces the total transcoding time by at least 67.2% (i.e., 1 – 32.8%) compared to the time used by the cascaded transcoder. In addition, the motion-estimation (ME) time used by our proposed transcoder was only 2.1% to 2.6% that of the cascaded transcoder, an exciting fact given that motion estimation consumes a large portion of the total encoding time in H.264/AVC. In our experiments, a fast search rather than a full search was used in both transcoding schemes, which reduces the weigh in computation that motion estimation costs. We expect our proposed scheme to achieve even more computational time reduction when a full search is used.

Subjective video quality was also assessed in our experiments. Results showed that, for the same bit rate, our proposed transcoder gave better or comparable reconstructed-video quality to that obtained by the cascaded transcoder. The cascaded transcoder loses more high-frequency component during the re-encoding process, resulting in smoothed frames, while our proposed scheme preserves more video details.

Note that we did not compare our results with another transcoding method, since we are not aware of any other efficient logo-insertion method for H.264/AVC pre-encoded video. We did not compare with the cascaded transcoding approach that re-uses the original prediction modes and motion vectors in the re-encoding process, since it causes even worse video quality distortion than the cascaded transcoding method that we present herein, and its computational cost is still higher than our proposed method.

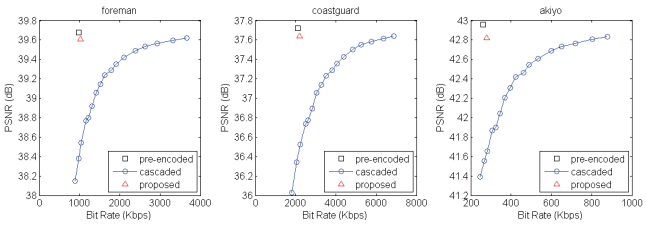

Fig. 4. Rate-distortion comparison.

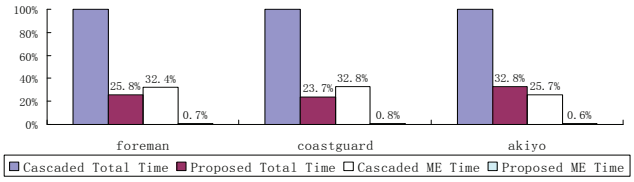

Fig. 5. Comparison of relative transcoding time.

5. CONCLUSIONS

We proposed an efficient logo-insertion scheme for H.264/AVC compressed video streams. Our approach is attractive for pre-encoded videos, which content providers prohibit telecasting station to completely decode and re-encode due to changes that such process may cause. Our scheme overcomes many video content and data-coding dependencies, and minimizes the changes to the original video streams. Experimental results show that our proposed scheme outperforms the cascaded transcoding method by an average of 1.16 dB PSNR increase, or a 68.6% bit-rate reduction. Furthermore, our proposed scheme significantly reduces the total transcoding time by at least 67.2%, and the motion-estimation time by at least 97.4%, compared with the cascaded-transcoding method. The above results are very encouraging, given that our method has exceptionally performance in both video quality and computational complexity.

REFERENCES

[1] Y. Liu, G. Li, Q. Tang, and J. Guo, "DCT domain logo insertion of MPEG2 transcoding," in *Proc. IEEE Canadian Conference on Electrical and Computer Engineering*, vol.2, May 2003, pp. 1219- 1222.

[2] S. Xiao, L. Lu, J. L. Kouloheris, and C. A. Gonzales, "Low-cost and efficient logo insertion scheme in MPEG video transcoding," in *Proc. of SPIE, Visual Communications and Image Processing*, vol. 4617, Jan. 2002, pp. 172-179.

[3] K. Panusopone, X. Chen, and F. Ling, "Logo insertion in MPEG transcoder," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City , USA, vol.2, May 2001, pp. 981-984.

[4] "Information technology – generic coding of moving pictures and associated audio information: video," Int. Telecommun. Union-Telecommun. (ITU-T) Recommendation H.262, International Standard 13818-2, 2nd ed., Feb. 2000.

[5] "Advanced video coding for generic audiovisual services," Int. Telecommun. Union-Telecommun. (ITU-T) Recommendation H.264, International Standard 14496-10, Mar. 2005.

[6] J.-H. Hur and Y.-L. Lee, "H.264 to MPEG-4 resolution reduction transcoding," in *Proc. IEEE Region 10 Conference*, TENCON, Nov. 2005, pp. 1-6.

[7] JM 13.2. H.264/AVC reference software from http://iphome.hhi.de/suehring/tml/download/.