FAST INTERFRAME TRANSCODING FROM H.264 TO MPEG-2

Sandro Moiron[1], *Sérgio Faria*[1,2], *Pedro Assunção*[1,2], *Vitor Silva*[1,3], *António Navarro*[1,4]

[1]Instituto de Telecomunicações, Portugal; [2]Instituto Politécnico de Leiria,ESTG, Portugal;
[3]Universidade de Coimbra, DEEC, Portugal; [4]Universidade de Aveiro, DET, Portugal.
{*sandro.moiron, sergio.faria, amado, vitor*}*@co.it.pt, navarro@av.it.pt*

ABSTRACT

This paper deals with conversion from H.264/AVC *(Advanced Video Coding)* coded video into the MPEG-2 format. The proposed approach exploits similarities between the coding techniques used in both standards in order to achieve a computationally efficient method for transcoding interframe coded slices. The conversion process is based on adaptation of both coding mode and motion information embedded in the H.264/AVC video stream, such that subsequent MPEG-2 encoding takes full advantage of the higher computational effort spent on the first encoding step. The proposed transcoding scheme significantly reduces the computational complexity needed for MPEG-2 interframe coding by reusing relevant information from the H.264 bitstream. The simulation results show that computational complexity savings up to 60%, with a marginal objective quality cost, can be achieved in comparison with a cascaded decoder-encoder.

Index Terms— H.264/AVC, MPEG-2, transcoding.

1. INTRODUCTION

The current H.264/AVC standard [1] presents a much better compression performance than its counterparts standards, namely MPEG-2 [2]. However, the MPEG-2 [3] video standard is still the most common video compression format and its widespread use in both professional and user equipment is expected to last for several years ahead, namely in Digital television (DTV), personal video recorders (PVR) and digital versatile disk (DVD).

Due to its higher compression efficiency, H.264 is increasingly gaining acceptance in multimedia applications and services such as High Definition Digital Television (HDTV), Mobile TV (MTV) and the internet. The simultaneous use of diverse coding standards definitely brings interoperability problems because the same type of source material may be available in a format not compatible with the target equipment. Even for some legacy terminal equipment with software upgrade capability, the computational resources needed to encode/decode H.264 might be too high. Furthermore it is not likely that technology migration in both professional and user equipment happens in such a short period of time that problems arising from co-existence of both standards can be ignored. Therefore, transcoding from H.264 to MPEG-2 format is necessary to maintain backward compatibility and ease technology migration.

In the recent past, much research effort has been put to develop efficient transcoding from MPEG-2 into H.264 in order to migrate legacy video content to the new format [4],[5], [6]. In contrast, too little effort has been devoted to the problem of backward compatibility. To the authors knowledge this problem was only recently addressed in [7] and [8]. In the former, some technical problems and research issues are highlighted whereas in the latter the authors only deal with P slices. This paper deals with interframe transcoding for both P and B slices and the proposed coding mode conversion methods are based on minimum residual sub-block information while in [8] only motion vector conversion is addressed.

A trivial implementation of a transcoder is a cascade of a H.264 decoder and an MPEG-2 encoder. However such a scheme completely ignores the H.264 encoding information embedded in the bitstream, which is the result of smart rate-distortion decisions aiming at encoding each block with the highest possible efficiency. By using such a transcoder the H.264 decoded frames have to be fully MPEG-2 encoded as if no previous coding information existed. In order to reduce this unnecessary complexity, the proposed transcoder aims at simplifying the MPEG-2 encoding process by reusing the information contained in the H.264 bitstream. This paper proposes a conversion method for interframe coding modes which is shown to achieve a significant reduction in computational complexity with marginal objective quality reduction, when compared with full recoding.

The paper is organised as follows. Section II describes the proposed coding mode conversion method along with a discussion about possible coding constraints and their influence on transcoding performance. Section III presents a detailed analysis of the simulation results. Finally section IV draws the main conclusions and point out some issues for further performance improvement.

2. INTERFRAME TRANSCODING

Although the same block-based coding concept is used in both H.264 and MPEG-2 standards, there are significant differences between them which leads to a complex matching process of macroblock (MB) modes. This section presents the proposed coding mode conversion techniques for the MB types defined in H.264 P and B slices, in particular SKIP, progressive 16×16 as well as those modes used in sub partitioned blocks.

2.1. 16×16 SKIP

Conversion of H.264 skipped MBs into MPEG-2 compatible ones is not a straightforward process because skipped MB have quite different characteristics in the two standards.

In the case of H.264 P slices, skipped MBs correspond to image regions with either no motion (i.e., static) or uniform motion (assuming translational motion) whereas in MPEG-2 only static areas can be encoded as skipped MBs. In both cases no residual information is encoded and the reference frame is always the last one used for such purpose. Since MPEG-2 SKIP MB type only covers a subset of the corresponding H.264 SKIP MB type (i.e. those with no motion), the remaining non-matching modes are converted to either forward or backward predicted. The motion information for these MBs is obtained from previously encoded ones by maintaining the same motion vectors (△MV=0) and then encoding them such as described in the next subsection 2.2.

In the case of B slices, the mode conversion between H.264 and MPEG-2 is different from that used in P slices. The MPEG-2 SKIP MB type in B slices is similar to that of H.264 since both allow the same motion type which simplifies the mode conversion function. However, for both P and B slices MPEG-2 does not allow skipped MBs neither in the first nor in the last MB of a slice. This is particularly relevant in low resolution frames because each MPEG-2 slice must start and end in the same MB row which results in a larger proportion of MBs that cannot be encoded as skipped.

2.2. 16×16 predicted

There are two types of H.264 16×16 MB which can be easily converted into 16x16 predicted MPEG-2 MBs. These are the H.264 16×16 predicted MBs and those H.264 skipped MBs which do not match the MPEG-2 SKIP mode constraints. Despite some similarities between the two standards, the set of H.264 coding tools used in 16x16 prediction modes do not match those used in MPEG-2 counterpart MBs type, as described in the following.

While in MPEG-2 either one or two of the last reference frames are used for predicting the current one, in H.264 up to 16 reference slices can be used for the same purpose. Therefore, in H.264 only those MBs that are predicted from the last two reference frames can be easily converted to MPEG-2 if these references correspond to possible MPEG-2 references.

If H.264 motion vectors point to a reference frame which is not in the same temporal position as the current MPEG-2 reference frames then motion vector rescaling must be performed before MPEG-2 encoding. Figure 1 shows the motion vector conversion process by scaling the original H.264 motion vectors, such that either the last one or two reference frames are used as required by MPEG-2. Assuming that motion between H.264 reference pictures and MPEG-2 ones is uniform, the scaling factor can be obtained by using the temporal distance between them. If both H.264 and MPEG-2 reference frames coincide in the same temporal position then the scaling factor is unitary and the motion vector remains unchanged.

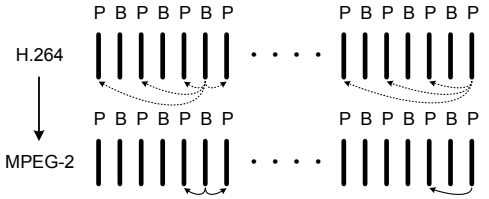

Fig. 1. Motion vector scaling for B and P macroblocks.

If the H.264 coding option defined as "Unrestricted Motion Vectors" is used, then temporal prediction from outside frame boundaries is allowed. In contrast to MPEG-2 where prediction is only permitted from inside frame boundaries. Prediction from outside frame boundaries is particularly useful in sequences where camera panning occurs because reference boundary MBs move towards outside the frame limits. In order to convert MBs using such a coding mode the corresponding H.264 motion vectors are truncated and constrained to point inside the reference frame.

H.264 prediction may use motion vectors with an accuracy of quarter pixel while in MPEG-2 the maximum motion accuracy is half pixel. Therefore, motion vector conversion also implies accuracy conversion if quarter pixel is used in H.264. Since computation of MPEG-2 motion vectors may involve both scaling and pixel accuracy conversion, it is essential to use rounding instead of truncation in order to achieve an efficient prediction.

2.3. MB Sub-partitions

The MB sub-partition scheme used in H.264 is a major source of complexity in transcoding for MPEG-2. This feature improves the prediction performance by splitting the 16×16 MB into 3 possible partition sizes: (16×8, 8×16 and 8×8). Each one of these can use a different reference image. The 8×8 partition can be further split into 3 more sub-partition types, such as (8×4, 4×8 and 4×4). This allows to find better predictions for the set of pixels belonging to each sub-partition. Sub-partitions introduce significant complexity in the MB conversion process, because the number of sub-blocks with different sizes and multiple reference frames combinations greatly increase. This is even more complex in B slices because it is possible to use several prediction modes from different reference frames (e.g., different partitions of the same MB may use forward, backward or bidirectional prediction). This leads to a greater mismatch between H.264 and MPEG-2, because the number of prediction possibilities in MPEG-2 in much smaller.

The full method used for MB conversion and motion vector calculation is based on the H.264 coded information included in the video stream. The MPEG-2 motion vector is obtained after computing the sum of squared differences of the whole MB using each sub-partition motion vector. The selected motion vector is the one which produces the minimum sum of squared differences.

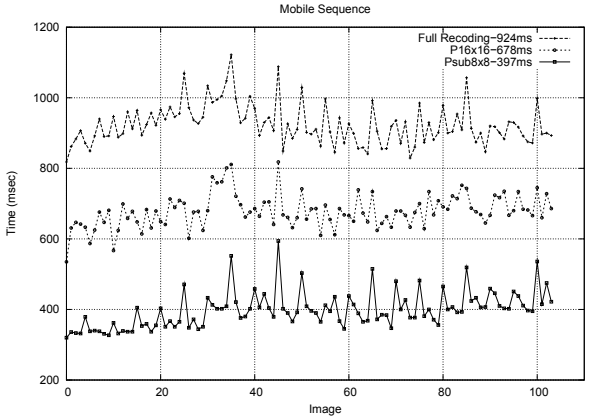

Fig. 2. Computational complexity results for P slices

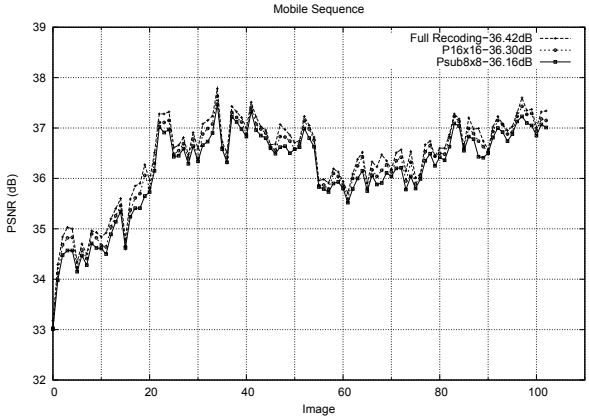

Fig. 3. Transcoded image quality for P slices

3. EXPERIMENTAL RESULTS

Three distinct test sequences with 250 frames and resolution 720×576@25 Hz were used for simulation and performance evaluation (Mobile, Stockholm and Shields). They are characterised by different motion activity and different spatial detail. The GOP size was 12 and its structure 'IBPBPBP'. Five reference frames were used and the bitrate was 5Mbps for both the H.264 and MPEG-2 streams. The transcoded sequence keeps the same GOP structure using the Main Profile. The transcoders are based on the H.264/AVC JM10.2 decoder and the MPEG-2 v1.2 encoder. A linux system running on a 3GHz processor and 1.5GB of RAM memory was used in the simulations. The experiments were intended to evaluate the performance of the proposed transcoding scheme by comparing the computational complexity and the objective video quality with the reference cascaded transcoder (i.e., full H.264 to MPEG-2 recoding). The results are analysed for P and B slices and do not include transcoding of intra slices. The intra mode was partially addressed by the authors in [9].

Figure 2, shows the computational complexity gain for P slices of the Mobile sequence measured in processing time. The three lines (Full Recoding, P16×16 and Psub8x8) correspond to the full cascaded recoding, fast transcoding of 16×16 MBs and fast transcoding of all MB prediction modes, respectively. Each graphic label contains the respective average frame computation time. As it can be seen, fast transcoding of 16×16 MBs, (P16×16) achieves near 27% of processing time reduction when compared with full recoding. This gain is obtained with a marginal loss of 0.12 dB in objective quality, as shown in Figure 3. When fast transcoding is applied to all MB modes of the P slices, the time reduction rises up to 57% with an objective quality loss of 0.26 dB.

As referred to in section 2.3, transcoding of B slices is much more complex than P slices. Therefore considerably different results are expected in this case. Figure 4 shows the computational complexity gain obtained in the case of B slices. As it can be seen in the figure, the complexity reduction achieved by the B slice is substantially different. In the

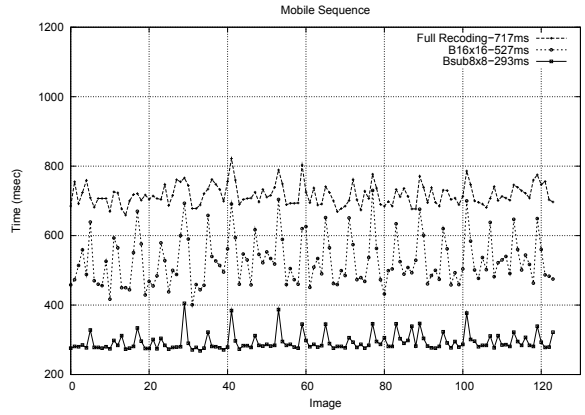

Fig. 4. Computational complexity results for B slices

case of 16×16 MB fast transcoding achieves a complexity reduction of 26%, with an objective quality loss around 0.22 dB, as illustrated in Figure 5.

Fast transcoding of all MBs gives a complexity gain of about 60%, with a quality loss of 0.33 dB. The quality loss is slightly larger in this case than in P slices. However since B slices are not used as references in MPEG-2, the error does not propagate through the sequence. Note that only B slices were taken into account in this case and the average quality loss for the whole sequence is lower, as shown in Table 1.

In Figure 6 the global computational complexity reduction achieved by the proposed fast transcoding in both B and P slices is shown. The reduction of computational complexity is 58% on average, for the Mobile sequence with 0.29 dB of quality loss. As it can be seen in Table 1, a better performance is obtained for sequences Stockholm and Shields, where larger complexity gains are achieved with smaller quality losses. This increase in the performance is mainly due to the fact that the coding complexity of these two sequences is lower. The objective quality loss obtained with the proposed fast transcoding scheme in comparison with full recod-

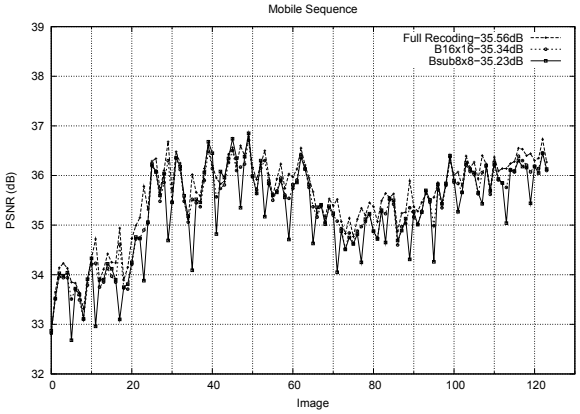

Fig. 5. Transcoded image quality for B slices

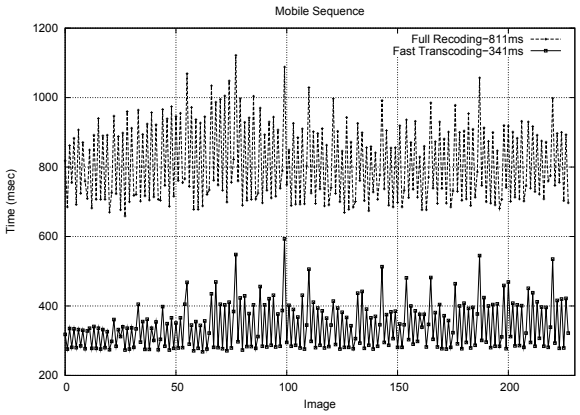

Fig. 6. Computational complexity comparison for P/B slices

ing is expected to be improved by the on-going work in motion vector refinement combined with R-D optimisation partially based on information from H.264 optimised streams.

4. CONCLUSIONS

The transcoding method described in this paper is focused on the computational complexity reduction of a video transcoder between H.264 and MPEG-2 standards. Particularly, inter-frame coding similarities are exploited between both standards, in order to re-encode the various MBs modes by adapting relevant information embedded in the H.264 bitstream. The simulation results show a computational complexity reduction up to 60%, with small quality loss, when comparing with the full recoding scheme.

In order to achieve a better performance for the proposed transcoding method, future work will further exploit H.264 embedded coding information. This will be focussed on the sub-partition block information and motion vector refinement in a small search window.

Table 1. Transcoder results for various sequences.

	PSNR (dB)	Diff (dB)	Complexity (ms)	Gain (%)
Mobile	35.96	-0.29	811	58
Stockholm	35.94	-0.30	755	60
Shields	34.20	-0.24	792	62

5. ACKNOWLEDGEMENT

This work was sponsored by IT/LA H2M project from Instituto de Telecomunicações.

6. REFERENCES

[1] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 / ISO/IEC 14 496-10 AVC)*, March 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 and Systems Magazine*, pp. 7–28, 1st Quarter 2004.

[3] ITU-T, *Recommendation H.262, Information Technology - Generic Coding of Moving Pictures and Associated Audio Information: Video*, Feb. 2000.

[4] J. Xin, A. Vetro, S. Sekiguchi, and K. Sugimoto, "MPEG-2 to H.264/AVC transcoding for efficient storage of broadcast video bitstreams," in *Proc. of the International Conference on Consumer Electronics*, Jan. 2006, pp. 417–418.

[5] L. Yang, X. Song, C. Hou, and J. Dai, "A scheme for MPEG-2 to H.264 transcoding," in *Proc. of the Canadian Conference on Electrical and Computer Engineering*, May 2006, pp. 310–313.

[6] T. Qian, J. Sun, D. Li, X. Yang, and J. Wang, "Transform domain transcoding from mpeg-2 to h.264 with interpolation drift-error compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 4, pp. 523–534, April 2006.

[7] Hari Kalva, "Issues in H.264/MPEG-2 video transcoding," *Computer Science and Engineering*, 2004.

[8] L. Yang, X. Song, C. Hou, and J. Dai, "H.264 MPEG-2 transcoding based on personal video recorder platform," in *Proc. of the Ninth International Symposium on Consumer Electronics*, June 2005, pp. 438–440.

[9] Ricardo Marques, Sérgio Faria, Pedro Assuncao, Vitor Silva, and António Navarro, "Fast conversion of H.264/AVC integer transform coefficients into dct coefficients," *SIGMAP*, pp. 5–8, Aug. 2006.