MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

# Motion Mapping for MPEG-2 to H.264/AVC Transcoding

Jun Xin, Jianjun Li, Anthony Vetro, Huifang Sun

TR2007-085 April 2008

**Abstract**

This paper describes novel motion mapping algorithms aimed for low-complexity MPEG-2 to AVC transcoding. The proposed algorithms efficiently map incoming MPEG-2 motion vectors to outgoing AVC motion vectors regardless of the block sizes that the motion vectors correspond to. Extensive simulation results show that our proposed transcoder incorporating the proposed algorithms achieves very good rate-distortion performance with low complexity. Compared with the cascaded decoder-encoder solution, the proposed approach could achieve similar coding efficiency while significantly reduce the complexity.

*IEEE International Symposium on Circuits and Systems, May 2007*

# Motion Mapping for MPEG-2 to H.264/AVC Transcoding

Jun Xin, Jianjun Li, Anthony Vetro, Huifang Sun
Mitsubishi Electric Research Laboratories
Cambridge, MA, USA
jxin,jli,avetro,hsun@merl.com

Shun-ichi Sekiguchi
Mitsubishi Electric Corporation
Ofuna, Kamakura, Japan
Sekiguchi.Shunichi@eb.MitsubishiElectric.co.jp

***Abstract*— This paper describes novel motion mapping algorithms aimed for low-complexity MPEG-2 to AVC transcoding. The proposed algorithms efficiently map incoming MPEG-2 motion vectors to outgoing AVC motion vectors regardless of the block sizes that the motion vectors correspond to. Extensive simulation results show that our proposed transcoder incorporating the proposed algorithms achieves very good rate-distortion performance with low complexity. Compared with the cascaded decoder-encoder solution, the proposed approach could achieve similar coding efficiency while significantly reduce the complexity.**

## I. Introduction

MPEG-2 [1] has become the primary format for broadcast video after being developed in the early 1990's. The new video coding standard, referred to as H.264/AVC [2] or simply AVC, as will be used in this paper, promises the same quality as MPEG-2 with about half the data rate. Since AVC has been adopted into storage format standards, such as Blu-ray Disc, we expect AVC decoders to appear in consumer video recording systems soon. Certainly, as more high-definition content becomes available and the desire to store more content or record more channels simultaneously increases, long recording mode will become a key feature for future consumer video recorders. To satisfy this need, we have developed novel techniques that convert MPEG-2 broadcast video to the more compact AVC format with low complexity. Complexity is kept low by reusing information contained within the MPEG-2 video stream. At the same time, high quality is maintained.

Straightforward cascading of an MPEG-2 decoder and a stand-alone AVC encoder would form a transcoder; this will be referred to as the *reference transcoder* later on in this paper. The reference transcoder is computationally very complex due to the need to perform motion estimation in the AVC encoder.

It has been well understood that one could reduce the complexity of the reference transcoder by reusing the motion and mode information from the input MPEG-2 video bitstream [3], [4]. However, the reuse of such information in the most cost-effective and useful manner is an open problem.

The transcoder architecture that is targeted for consumer storage applications is shown in Fig. 1. It consists of an MPEG-2 decoder and a simplified AVC encoder. The encoder is "simplified" relative to the encoder in the reference transcoder, since the motion and mode information may be derived based on input MPEG-2 video. In this paper, we focus on the motion mapping algorithm, which is the main obstacle in low-complexity transcoder design. We assume the input MPEG-2 video is coded using frame pictures, and the output is coded using AVC frame pictures with macroblock adaptive frame/field (MBAFF) turned off. However, the proposed method could easily be generalized for other input and output picture formats. In addition, we disable inter prediction for block sizes 8x4, 4x8 and 4x4, although the proposed algorithms can be applied to these modes as well. We think this is a reasonable design for practical applications since block sizes larger than 8x8 are believed to achieve most of the gains promised by variable block size motion compensation.

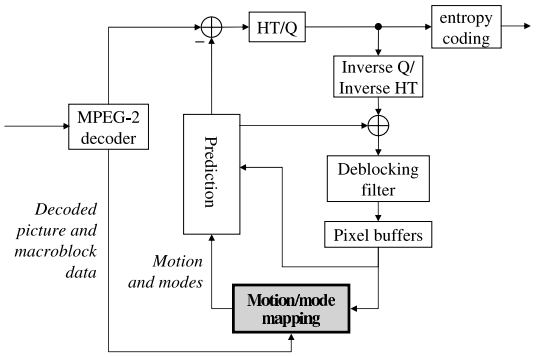


Fig. 1. MPEG-2 to AVC transcoding architecture with motion and mode mapping.

In recent works [5], [6] for motion mapping, a complete motion estimation algorithm is still performed. For inter 16x16 prediction, the motion vectors from incoming MPEG-2 video are used as additional motion vector predictors. For smaller block sizes, e.g. 16x8, 8x16 and 8x8 etc, motion vectors are estimated not directly from incoming motion vectors since MPEG-2 does not have such motion vectors. Instead, they are estimated using pure encoding algorithms without considering incoming motion vectors. Therefore, such an approach still needs complicated motion search algorithms. In this paper, we propose very efficient motion mapping algorithms that directly map incoming MPEG-2 motion vectors to outgoing AVC motion vectors, regardless of their supporting block sizes. Hence, the need for complex motion search algorithm is completely eliminated. In addition, we present algorithms

to that support field to frame motion vector mapping.

The remainder of this paper is organized as follows. In Section II we describe our proposed motion mapping algorithms. Experimental results are then shown in Section III. Finally, concluding remarks are provided.

## II. Motion mapping

The motion mapping algorithm has to solve the following three mismatch problems between MPEG-2 motion vectors and AVC motion vectors: field/frame mismatch, reference picture mismatch and block size mismatch.

The first type of mismatch is frame/field mismatch. In MPEG-2 frame picture coding, each macroblock can be coded with either frame prediction or field prediction. In frame prediction, a macroblock is predicted from a 16x16 block in the reference frame positioned by a motion vector. In field prediction, a macroblock is divided into two 16x8 blocks, one block belonging to the top field, and the other block belonging to the bottom field. Each 16x8 block has a field selection bit that specifies whether the top or the bottom field of the reference frame is used, and a motion vector that points to the 16x8 pixel block in the appropriate reference field. On the other hand, only frame prediction is allowed in H.264/AVC frame picture coding when MBAFF is disabled. Therefore, we need to convert incoming MPEG-2 field motion vectors to frame motion vectors.

Reference picture mismatch arises in cases when a picture type change is required or the output bitstream utilizes prediction from multiple reference picture in AVC. The change in picture type would be required when the incoming MPEG-2 bitstream is coded with B-frames, but the target output is compliant with the AVC Baseline Profile that does not support B-frames. Under such circumstances, it is quite likely that the target motion vector references a different picture from the motion vectors available from the input MPEG-2 video. The basic idea for reference picture mapping is to compose the target motion vector from existing MPEG-2 motion vectors. We refer the readers to the details of the mapping algorithm to our previous work [7] as well as [8], and we will not discuss it in this paper.

The third type of mismatch is block size mismatch. AVC allows the use of various block sizes for inter prediction, while there are only motion vectors based on 16x16 blocks from MPEG-2. This creates the need to map motion vectors corresponding to a given block size to a much wider range of block sizes.

Based on the above discussions, we propose a three-step motion mapping approach that applies to each target AVC block motion vector. We first convert the incoming MPEG-2 field motion vectors to frame motion vectors, and then map them to reference the same picture as the target motion vector. Finally, we map the resulting motion vectors to the set of target AVC motion vectors with various supporting block sizes.

After the above process is finished, we perform motion refinement centered at the mapped motion vector. We first perform an integer refinement with window of $\pm 1$, and then perform half-pel refinement around the best integer motion vector and finally quarter-pel refinement around the best half-pel motion vector.

In what follows, we describe in detail the algorithms for field-to-frame mapping and block size mapping.

### A. Field-to-frame mapping

The following algorithm deal with the case where the incoming MPEG-2 motion vectors for a macroblock are field motion vectors. If a field motion vector refers to a field of the same parity in the reference frame, then it can be directly used as frame motion vector. If a field motion vector refers to a field of the opposite parity , then this motion vector has to be modified as follows.

Without loss of generality, we assume that the top field comes first in time in the video sequence. Let us examine the case where the input forward motion vector is for the top field referencing the bottom reference field. Based on the assumption that motion is linear over a short period of time, we can modify the field motion vector to reference the top field by scaling the input motion vector. Also notice there is a half pel vertical displacement between top field and bottom field, as illustrated in Fig. 2, where the temporal distance between the current frame and the reference frame is one frame. In the figure, the input vertical field motion is 0.5, and the output field motion is 2 in field pixel units, and therefore the frame motion is 4 in frame pixel units. For the case of forward motion vectors where the top field references the bottom field, the general formula for field-to-frame mapping is:

$$\begin{aligned} MV_{frame,y} &= \frac{2 \times (MV_{field,y} + 0.5) \times (2 \times t_p)}{(2 \times t_p - 1)} \\ MV_{frame,x} &= \frac{2 \times (MV_{field,x}) \times (2 \times t_p)}{(2 \times t_p - 1)} \end{aligned} \quad (1)$$

where $t_p$ is the temporal distance between the current frame and the reference frame. Following the same process, it is straightforward to derive formula for field-to-frame mapping for other cases of field/frame mismatches.

When a macroblock is coded as a field macroblock, it has two field motion vectors, one for top field, and the other one for bottom field. Both need to go through the above process. Then, the two resulting motion vectors are averaged to form the final mapped frame motion vector.

### B. Block size mapping

For target 16x16 motion vectors, the input motion vectors are directly used. The proposed algorithms apply to target motion vectors with supporting block sizes smaller than 16x16, i.e. 16x8, 8x16 and 8x8.

This algorithm has an assumption: the motion vector of a rectangular block is same as that of its geometric center. Consequently, the input to the block size mapping becomes the motion vector of the incoming macroblocks' geometric center, and the output becomes the motion vector of the target block's geometric center. Note that the assumption of this approach is more general than translational block motion model typically

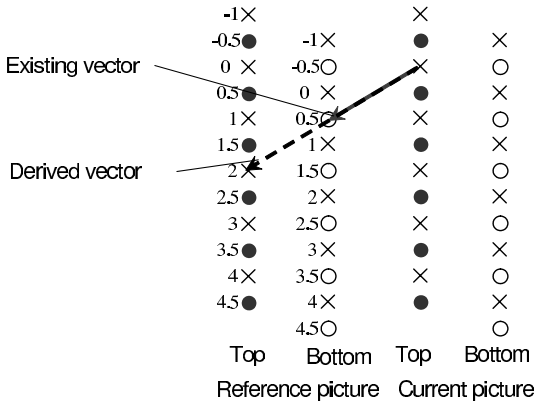


Fig. 2. Field to frame motion vector mapping. The existing input motion is forward motion for top field referencing bottom field. Top field comes first in the video sequence.

assumed in related works. Even when there are motions like zoom-in or zoom-out, the motion vector of a rectangular block can be considered to be approximately same as the motion vector of its geometric center.

In this algorithm, the output is derived as a weighted average of the motion vectors of candidate macroblocks. The candidate macroblocks include the current macroblock and those immediately adjacent to the current target block. The weight of an input motion vector is inversely proportional to the distance between its associated macroblock's geometric center to the target block's geometric center. Therefore, this algorithm is called Distance Weighted Average (DWA).

In Fig. 3(a), for target macroblock partitions $A$ and $B$ for inter_16x8 mode, the candidate macroblocks are labelled with $a_1$ through $a_6$, and with $b_1$ through $b_6$, respectively. A macroblock has duplicate labels if it is a candidate for deriving both $MV(A)$ and $MV(B)$. The geometric centers of each candidate macroblock and target macroblock partitions are also indicated in the figure. Based on the notations given in the figures, the target motion vectors for $A$ are computed as:

$$MV(A) = \frac{\sum_{i=1}^{6} w_i \times MV(a_i)}{\sum_{i=1}^{6} w_i} \qquad (2)$$

where the weight $w_i$ is inversely proportional to the distance between the geometric center of the candidate macroblock $a_i$ and that of target macroblock partition $A$. In this case, the values of $w_i$ are given as follows:

$$\{w_i\} = \{0.0902, 0.1503, 0.0902, 0.1093, 0.4508, 0.1093\}$$

Similarly the motion vector for macroblock partition $B$ can be calculated using motion vectors of $b_i$.

Candidate macroblocks are illustrated in Fig. 3(b) and (c) for inter_8x16 and inter_8x8 modes respectively, and it is straightforward to perform similar motion vector mapping for these two cases.

### C. Complexity analysis

Each field-to-frame mapping, from (1), needs 1 addition, 2 multiplications and 2 divisions. For the worst case, an incoming B macroblock has 2 forward and 2 backward motion vectors, and all need field-to-frame mapping, then 4 additions, 8 multiplications, and 8 divisions are needed. For block size mapping operation, e.g. (2), it is straightforward to use fixed point algorithm. With such implemenation, for each target 16x8/8x16 motion vector, 8 multiplications, 12 additions and 2 shifts are needed, while for each target 8x8 motion vector, 6 multiplications, 8 additions and 2 shifts are needed. Compared to even a single block matching operation, the above mapping operations are much simpler. Therfore, the complexity of the proposed three step mapping process is dominated by the refinement step. Consider that the integer refinement only requires 9 block matching operations for each block size, and it is much less complex than any full or fast motion estimation algorithm in the literature. It follows that efficient sub-pel motion estimation and interpolation algorithms would be critical to the transcoder design, which will be our future work.

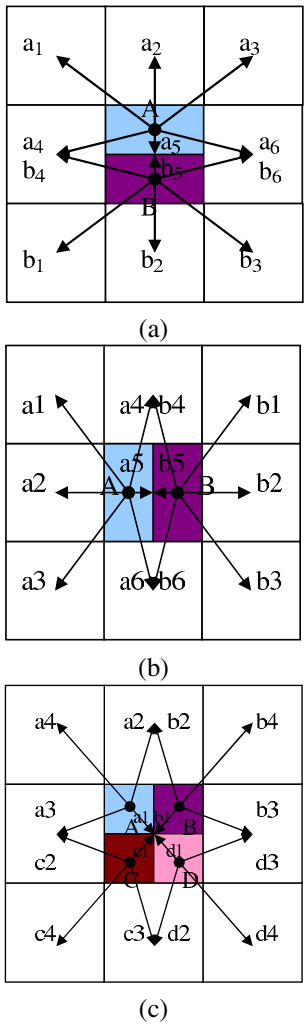


Fig. 3. Deriving motion vectors for various macroblock partitions using distance weighted average: (a) inter_16x8, (b) inter_8x16, (c) inter_8x8.

## III. Simulation Results

We use two interlaced sequences in the simulations: HarbourScene and StreetCar. Both have resolution 1920x1080 with frame rate 30 frames/s, and are 15 seconds (450 frames) long. They are encoded using the MPEG-2 reference software [9] at 30Mbps[1] and are used as input to the transcoder. The group of picture (GOP) size for MPEG-2 encoding is N=30 and two B-frames are coded between every consecutive pair of anchor frames, i.e. M=3.

We simulate two transcoders. In the first transcoder ("DWA"), I-frames are transcoded to I-pictures, and P and B frames are transcoded to P-pictures. This effectively simulates the case in which compliance to the AVC Baseline Profile is desired. In the second transcoder ("DWA_IPB"), P and B frames are transcoded to P and B frames respectively, i.e. picture types are kept intact. The QP values are chosen such that the output bit rate is around 10Mbps, which is the target bit rate of interest for consumer storage applications.

We compare the performance of the proposed two transcoders to the reference transcoders: "Reference" and "Ref_IPB", where the AVC encoder is the JM10.2 reference software [10] with UVLC coding and RDO disabled. Inter predictions using block sizes 4x8, 8x4 and 4x4 are disabled to make fair comparisons. The difference between "Reference" and "Ref_IPB" is that "Ref_IPB" keeps the same GOP structure while "Reference" does not encode B pictures.

The results are shown in Fig. 4. First let us examine "DWA" and "Reference", where there is no B picture output. It is clear from these plots that the proposed mapping algorithms (DWA) achieves comparable performance to the reference transcoder in terms of coding efficiency. At 10 Mbps rate point, the performance loss in terms of PSNR is less than 0.4dB for the case of HarbourScene and about 0.15dB in the case of StreetCar. Compared to the reference transcoder using exhaustive search, the complexity saving is around 95% which is measured using consumed CPU time. The computational saving is similar for both sequences. Comparing "DWA_IPB" and "Ref_IPB" also suggests that the proposed DWA algorithm is cost-effective, as "DWA_IPB" achieves comparable performance to "Ref_IPB". The RD performance gap and complexity saving are similar to the case of "DWA" and "Reference". It can also be observed that supporting B pictures improves the RD performance considerably. The complexity increase cause by adding B pictures, as measured by consumed CPU time, is approximately 40%.

## IV. Concluding Remarks

We presented motion mapping algorithms that can efficiently map incoming MPEG-2 motion vectors to outgoing AVC motion vectors, even when they have different block size support and different reference pictures. Simulation results show that our proposed transcoder incorporating the proposed algorithms could achieve good rate-distortion performance with low complexity. Compared with the cascaded decoder-encoder reference, the RD performance is comparable while the complexity is significantly reduced. Our preliminary complexity analysis show that two major complexity intensive operations in our current transcoder are motion refinement and sub-pel interpolation. We are working on further measures to reduce the complexity of these components.

[1]The input rate of 30Mbps was chosen since the MPEG-2 reference software is not an optimized encoder and the quality at 30Mbps was found to be visually comparable to typical broadcast content.

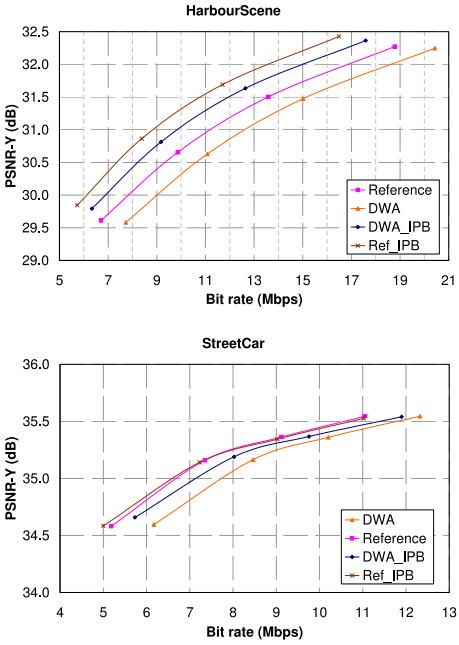


Fig. 4. Motion mapping performance comparison for HarbourScene and StreetCar sequences: proposed mapping with DWA and reference transcoder.

## References


[1] "ISO/IEC 13818-2: Information technology - Generic coding of moving pictures and associated audio information: Video." 2000.

[2] "ITU-T Rec. H.264 — ISO/IEC 14496-10: Advanced Video Coding," 2003.

[3] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," *IEEE Signal Processing Mag.*, vol. 20, no. 2, pp. 18–29, Mar. 2003.

[4] J. Xin, C.-W. Lin, and M.-T. Sun, "Digital video transcoding," *Proc. IEEE*, vol. 93, no. 1, pp. 84–97, Jan. 2005.

[5] Z. Zhou, S. Sun, S. Lei, and M. Sun, "Motion information and coding mode reuse for MPEG-2 to H.264 transcoding," in *IEEE Int. Symposium on Circuits and Systems*, 2005, pp. 1230–1233.

[6] X. Lu, A. Tourapis, P. Yin, and J. Boyce, "Fast mode decision and motion estimation for H.264 with a focus on MPEG-2/H.264 transcoding," in *IEEE Int. Symposium on Circuits and Systems*, 2005.

[7] J. Xin, A. Vetro, S. Sekiguchi, and K. Sugimoto, "Motion and mode mapping for MPEG-2 to H.264/AVC," in *IEEE Int. Conf. Multimedia & Expo*, 2006.

[8] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding to lower spatio-temporalresolutions and different encoding formats," *IEEE Trans. on Multimedia*, vol. 2, no. 2, pp. 101–110, 2000.

[9] "MPEG-2 encoder/decoder v1.2," 1996, by MPEG Software Simulation Group, available online at http://www.mpeg.org/MPEG/MSSG.

[10] "H.264/AVC reference software JM10.2," 2006, available online at http://bs.hhi.de/ suehring/tml/download/.