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

Jun Xin, Jianjun Li, Anthony Vetro, Shun-ichi Sekiguchi

TR2007-004    November 2007

**Abstract**

This paper describes novel transcoding techniques aimed for low-complexity MPEG-2 to H.264/AVC transcoding. An important application for this type of conversion is efficient storage of broadcast video in consumer devices. The architecture for such a system is presented, which includes novel motion mapping and mode decision algorithms. For the motion mapping, two algorithms are presented. Both efficiently map incoming MPEG-2 motion vectors to outgoing H.264/AVC motion vectors regardless of the block sizes that the motion vectors correspond to. In addition, the algorithm maps motion vectors to different reference pictures, which is useful for picture type conversion and prediction from multiple reference pictures. We also prpose an efficient rate-distortion optimized macroblock coding more decision algorithm, which first evaluates candidate modes based on a simple cost function so that a reduced set of candidate modes is formed, then based on this reduced set, we evaluate the more complex Lagrangian cost calculation to determine the coding mode. 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 coding efficiency is maintained while the complexity is significantly reduced.

*Multimedia Tools and Applications, November 2007*

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

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

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

**Abstract**

This paper describes novel transcoding techniques aimed for low-complexity MPEG-2 to H.264/AVC transcoding. An important application for this type of conversion is efficient storage of broadcast video in consumer devices. The architecture for such a system is presented, which includes novel motion mapping and mode decision algorithms. For the motion mapping, two algorithms are presented. Both efficiently map incoming MPEG-2 motion vectors to outgoing H.264/AVC motion vectors regardless of the block sizes that the motion vectors correspond to. In addition, the algorithm maps motion vectors to different reference pictures, which is useful for picture type conversion and prediction from multiple reference pictures. We also propose an efficient rate-distortion optimised macroblock coding mode decision algorithm, which first evaluates candidate modes based on a simple cost function so that a reduced set of candidate modes is formed, then based on this reduced set, we evaluate the more complex Lagrangian cost calculation to determine the coding mode. 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 coding efficiency is maintained while the complexity is significantly reduced.

*a)* **Keywords***:* — video transcoding, H.264/AVC, MPEG-2, mode decision, motion mapping.

## 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], promises the same quality as MPEG-2 with about half the data rate. Since the H.264/AVC format has been adopted into storage format standards, such as Blu-ray Disc, we expect H.264/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 H.264/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. The diagram of the proposed system is shown in Fig. 1. Since an MPEG-2

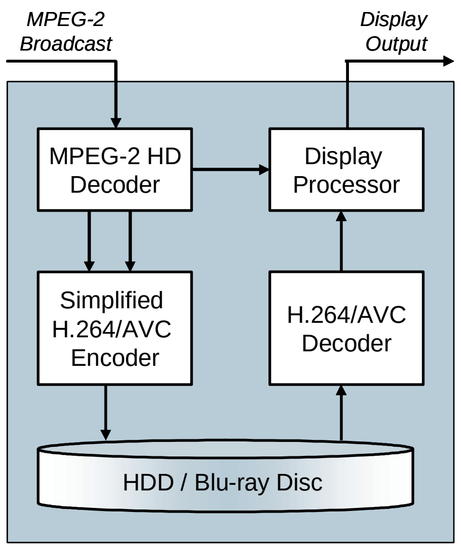


Fig. 1. Storage system using MPEG-2 to H.264/AVC transcoding

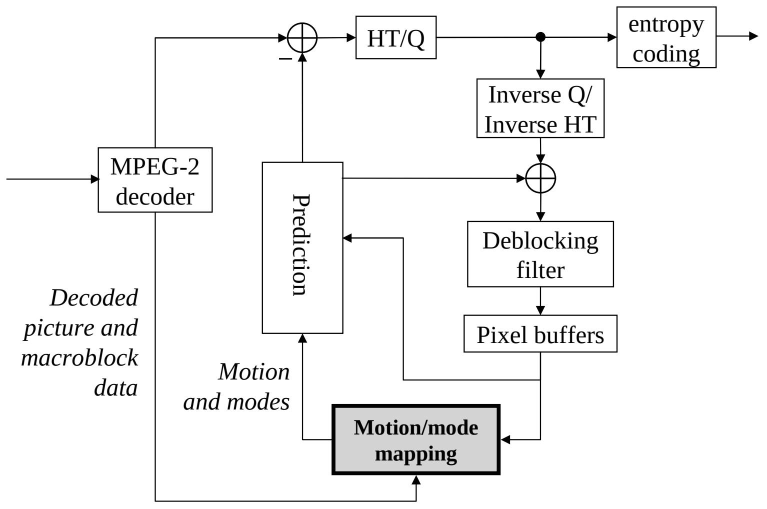


Fig. 2. MPEG-2 to H.264/AVC transcoding architecture with motion and mode mapping.

decoder is present in existing systems, the challenge is integrating the simplified H.264/AVC encoding part of the MPEG-2 to H.264/AVC transcoder into the overall system.

Straightforward cascading of an MPEG-2 decoder and a stand-alone H.264/AVC encoder would form a transcoder; this will be referred to as the *reference transcoder* later on in this paper. The reference transcoder is very computationally complex due to the need to perform motion estimation and mode decision in the H.264/AVC encoder.

It is 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. 2. It essentially consists of an MPEG-2 decoder and a simplified H.264/AVC encoder. There is a post-processing unit following the MPEG-2 decoder that may perform artifact removal or resolution scaling if desired. The encoder is ”simplified” relative to the reference transcoder, since the motion and mode information is derived based on input MPEG-2 video. In this paper, we focus on the motion and mode mapping algorithms, which are the main obstacles in low-complexity transcoder design. We assume the input MPEG-2 video is coded using frame pictures, which is the more popular MPEG-2 coding method. The output will be coded using H.264/AVC frame pictures with macroblock adaptive frame/field (MBAFF) turned off. However, the proposed method could easily be generalised for field picture input and frame picture output with MBAFF or field picture output. 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.

In reported works [5], [6] for motion mapping in transcoding, a complete motion estimation algorithm is actually 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, these motion vectors are estimated using pure encoding algorithms without considering incoming MPEG-2 motion vectors. Therefore, such an approach still needs very complicated motion search algorithms. In this paper, we propose very efficient motion mapping algorithms that directly maps incoming MPEG-2 motion vectors to outgoing H.264/AVC motion vectors, regardless of their supporting block sizes. With the proposed algorithms, the need for complex motion search algorithm is completely eliminated. In addition, we propose algorithms to support the mapping to H.264/AVC motion vectors with different reference picture, which may be useful in case of picture type conversion or picture skipping, as well as algorithms to support field to frame motion vector mapping.

For rate-distortion optimised mode decision, the main complexity lies in the Lagrange cost evaluation for all possible coding modes. Existing algorithms typically try to reduce the number of candidate modes actually considered in Lagrange cost evaluation, either based on analysis of pictures for intra mode decision or other heuristics for inter mode decision. In [7], edge direction map is used to reduce the number of candidate intra prediction modes. While in [8], edge vector amplitude is used to reduce the number of candidate inter prediction modes. However, these algorithms may be unreliable at times since the pre-analysis may not correlate well with the rate-distortion based decisions. In [9], it is demonstrated that a low complexity cost function can be used to eliminate unlikely intra_4x4 coding modes. The basic idea is to first rank candidate modes in ascending order based on the low complexity cost function. Then the more complex Lagrange cost function is evaluated only for few best modes, i.e. with the lowest costs. In this paper, we extend the idea to inter-prediction modes. We show that the low complexity cost function correlates well with the Lagrange cost function. It also provides a level of complexity scalability, i.e., we incorporate a low complexity mode decision algorithm if the Lagrange cost calculation is omitted with performance loss that is moderate compared to the full algorithm.

The main contributions of this work, i.e. the proposed motion and mode mapping algorithms, includes the following:

- A novel motion mapping algorithm that directly maps the incoming MPEG-2 motion vectors to outgoing H.264/AVC motion vectors, regardless of their supporting block size, reference picture and field/frame structure.
- An efficient rate-distortion optimised mode decision algorithm that is based on an initial low-complexity ranking of candidate modes.
- Evaluating the effectiveness of various H.264/AVC coding tools in the context of transcoding from MPEG-2 or similar coding formats.

The remainder of this paper is organized as follows. In Section II we describe our proposed motion mapping algorithms, and in Section III we describe the mode decision algorithms. Experimental results are then presented in Section IV to demonstrate the effectiveness of the proposed algorithms. 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 H.264/AVC motion vectors: field/frame mismatch, reference picture mismatch and block size mismatch. Each are discussed further below.

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 H.264/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 H.264/AVC Baseline Profile that does not support B-frames. Under such circumstances, it is quite likely that the target motion vector references a different reference picture from the motion vectors available from the input MPEG-2 video.

The third type of mismatch is block size mismatch. H.264/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 H.264/AVC block motion vector. We first convert the incoming MPEG-2 field motion vectors (if any) to frame motion vectors, and then map them to reference the same reference picture as the target motion vector. Finally,

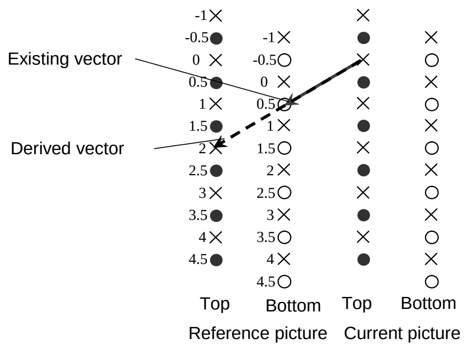


Fig. 3. Field to frame motion vector mapping. The existing input motion is forward motion for top field referencing bottom field. Top field comes first in the video sequence.

we map the resulting motion vectors to the set of target H.264/AVC motion vectors with various supporting block sizes.

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

## *A. Field-to-frame mapping*

The following algorithms deals with the case where the incoming MPEG-2 motion vectors for a macroblock are field motion vectors. If a field motion vector refers to the 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 the field of the opposite parity in the reference frame, then this motion vector has to be modified.

Without loss of generality, we assume in the video sequence that the top field comes first in time. Let us examine the case where the input forward field motion vector is for the top field referencing the bottom reference field. Based on the assumption that motion is linear over time, we can modify the field motion vector to reference the top field by linearly scaling the input motion vector. Also notice there is a half pel vertical displacement between top field and bottom field, as illustrated in Fig. 3, 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:

$$MV_{frame,y} = \frac{2 \times (MV_{field,y} + 0.5) \times (2 \times t_p)}{(2 \times t_p - 1)} \tag{1}$$

$$MV_{frame,x} = \frac{2 \times (MV_{field,x}) \times (2 \times t_p)}{(2 \times t_p - 1)} \tag{2}$$

where $t_p$ is the temporal distance between the current frame and the reference frame. Following the same process, it is straightforward to derive the formula for field-to-frame mapping for the case when a forward motion vector for the bottom field references the top field:

$$MV_{frame,y} = \frac{2 \times (MV_{field,y} - 0.5) \times (2 \times t_p)}{(2 \times t_p + 1)} \tag{3}$$

$$MV_{frame,x} = \frac{2 \times (MV_{field,x}) \times (2 \times t_p)}{(2 \times t_p + 1)} \tag{4}$$

The formulas for field-to-frame mapping for backward motion vectors can be derived in a similar way.

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 of them need to go through the above process. Then, the two resulting motion vectors are averaged to form the final mapped frame motion vector.

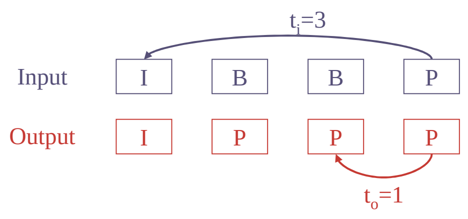


Fig. 4. Reference picture mapping of motion vector: P to P mapping. Target output motion vector references a different picture from the input motion vector.

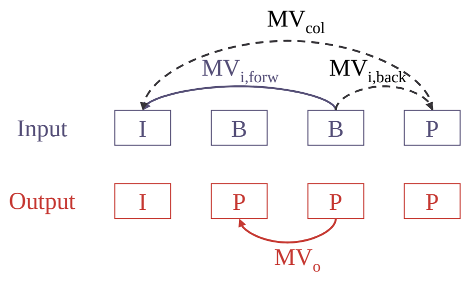


Fig. 5. Reference picture mapping of motion vector: B to P mapping. Target output motion vector references a different picture from the input motion vector.

## B. *Reference picture mapping*

In what follows, we present techniques used to map motion vectors to reference different reference pictures. As an example, we use the case where all non-I pictures are converted to P slices.

First, we consider the mapping of P-frame motion vectors as shown in Fig. 4. We take the first incoming P-frame as an example. In the input video, the P-frame is predicted from its preceding I-frame. Let $t_i$ be the temporal distance between the P-frame and its reference I-frame. In this example, $t_i$=3. Assuming only one temporal reference frame is used in the output video, the set of input motion vectors must be modified to reference the preceding P-frame in the output H.264/AVC video. Denote the temporal distance between the output P-picture and its reference P-picture as $t_o$. In this example, $t_o$=1. Assuming the motion is small and linear during the period of $t_i$ frames, which is typically 100ms or less, we can represent the mapping from the motion vector of the incoming MB to the output MB with:

$$MV_o = (MV_i \div t_i) \times t_o \tag{5}$$

A more complicated case is shown as the second incoming B-frame in Fig. 5, where the output is a P-frame. If a macroblock in the B-frame has a forward motion vector, we use equation (5) to calculate the motion vector for the outgoing P-picture by scaling the incoming forward motion vector. If a macroblock in the B-frame has only a backward motion vector ($MV_{i,back}$), we first add the backward motion vector to the forward motion vector of the collocated future P-frame, i.e., $MV_{i,back} + MV_{i,col}$, and then scale the resulting motion vector according to (5).

With all incoming motion vectors converted to frame motion vectors and referencing the target reference picture, we can derive the target motion vector using the technique described in sub-section II-C.

## *C. Block size mapping*

We present two approaches to map motion vectors with 16x16 block size support to motion vectors with smaller supporting block size that will be used in H.264/AVC coding. For target motion vectors with 16x16 block support, the input motion vectors can be directly used. The algorithms proposed in this section only apply to target motion vector with supporting block size smaller than 16x16, i.e. 16x8, 8x16 and 8x8.

*1) Distance weighted average (DWA):* The assumption made by this algorithm is that the motion vector of a rectangular block is same as the motion vector 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.

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 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.

In Fig. 6(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. Note that a macroblock has duplicate labels if it is a candidate macroblock for deriving both $A$ and $B$. 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:

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

where the weight $w_i$ is 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 distances $d_i$ are {5/2, 3/2, 5/2, $\sqrt{17}$/2, 1/2, $\sqrt{17}$/2} assuming an 8 pixel distance is equal to 1. We then normalize these distances to get the respective weights, $w_i$:

$$w_i = \frac{d_i}{\sum_{i=1}^{6} d_i}$$

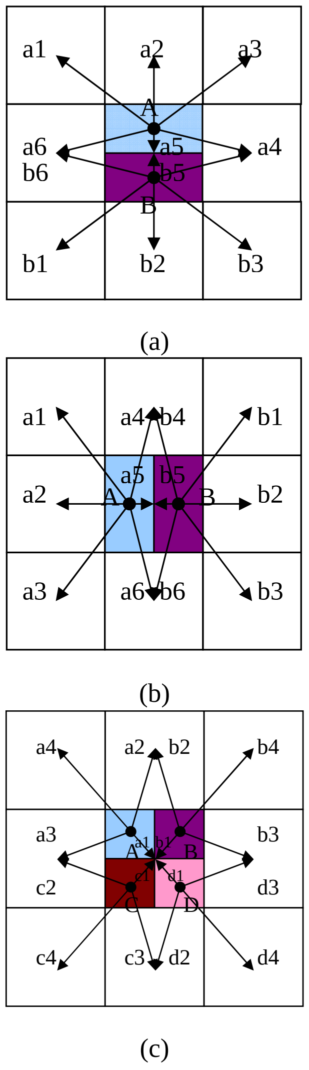


Fig. 6. Deriving motion vectors for various macroblock partitions using distance weighted average: (a) inter_16x8, (b) inter_8x16, (c) inter_8x8.

For this particular case, the weights are calculated to be:

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

The motion vector for macroblock partition $B$ can be calculated in a similar fashion using the motion vectors of $b_i$.

Similarly, candidate macroblocks are illustrated in Fig. 6(b) and (c) for inter_8x16 and inter_8x8 modes respectively, and the motion vector mapping is performed as distance weighted average of candidate motion vectors.

Note that the assumption of this approach is more general than translational block motion model typically 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.

*2) Error-variance weighted average (EWA):* In this algorithm, the output is also derived as a weighted average of the motion vectors of candidate macroblocks. The difference with DWA lies in how the candidate motion vectors are formed as well as how the weights are determined. In the EWA algorithm, a mask is applied to each target H.264/AVC motion vector, i.e., each target H.264/AVC macroblock partition for which a motion vector has to be determined. The weighting masks for 16x16, 16x8, 8x16 and 8x8 macroblock partitions are shown as areas enclosed by dashed lines in Fig. 7. In these figures, each small square is an 8x8 block. The motion vector of each 8x8 block is assumed to be the vector of the macroblock it belongs to. Actual motion vector mapping is performed per weighting mask. As shown in Fig. 8, each 8x8 block and its corresponding motion vector are represented by $B_k$ and $d_k$, respectively. $x_{i,j}$ denotes the location of a pixel in a 8x8 block, and the prediction error is computed using the following equation from the prediction error for each pixel location $e(x_{i,j}, t)$:

$$e_k = \sum_{(i,j)\in B_k} |e(x_{(i,j)}, t)| \tag{7}$$

The variance of the motion vectors within a mask is derived as follows:

$$\sigma = Var(d_i), i \in R \tag{8}$$

where, $R$ is the weighting mask. Finally, the H.264/AVC inter_8x8 motion vector is calculated as:

$$\hat{d} = \frac{\sum_R \varphi_k d_k}{\sum_R \varphi_k} \qquad \varphi_k = \{ \begin{smallmatrix} \frac{A}{e_k} & k\neq 4 \\ \frac{A}{h(\sigma)e_k} & k=4 \end{smallmatrix} \tag{9}$$

In (9), $A$ is a constant value and $h(\sigma)$ specifies a monotonically-decreasing discrete function of $\sigma$, the variance of motion vectors in a mask. As a result of this process, we have the mapped motion vector for each H.264/AVC target macroblock partition.

## III. Mode decision

In H.264/AVC, a macroblock can be coded in one of many possible modes. It is therefore very important to have an effective mode decision algorithm that does not incur significant complexity and leads to good coding efficiency. Without this latter requirements some of coding efficiency benefits of H.264/AVC coding tools may not be fully realized.

Below we briefly review the rate-distortion optimized (RDO) mode decision algorithm used in JM reference software. The goal of RDO mode decision is to select the best block partitioning as well as inter/intra decision for a macroblock. First, the coding rate and the resulting distortion are computed for all possible macroblock coding modes. Then, the Lagrange cost function is evaluated for each mode $m$:

$$J_1(m) = SSD(m) + \lambda_{mode} \times R(m, Q) \tag{10}$$

where, $SSD(m)$ is the distortion, i.e., the sum of squared difference between the original block, $s$, and the reconstructed block using mode $m$, $\tilde{s}(m)$ and is expressed as,

$$SSD(m) = \left\| s - \tilde{s}(m) \right\|_2{}^2 \tag{11}$$

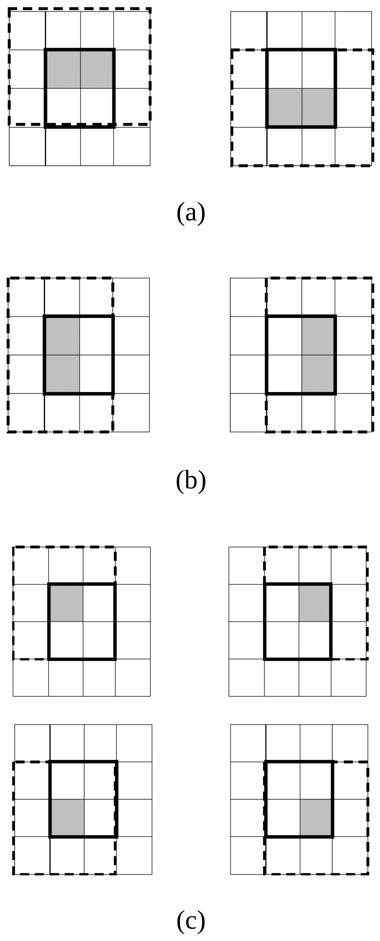


Fig. 7. EWA weighting masks for: (a) 16x8 partitions; (b) 8x16 partitions and (c) 8x8 partitions. Current macroblock is the square area with bold borders. The target macroblock partition is the shaded area. The weighing mask is the area enclosed by dashed lines. Every small block size is 8x8.

where $\|\cdot\|_p$ is the $Lp$-norm. In (10), $R(m,Q)$ is the total number of bits used to code the macroblock, including overhead such as mode and motion vectors, as well as transform coefficients, $\lambda_{mode}$ is the Lagrange multiplier that controls the rate-distortion tradeoff, and $Q$ is the quantizer used for quantization of transform coefficients. As used in the JM software, $\lambda_{mode}$ is set according to the H.264/AVC quantization parameter as follows:

$$\lambda_{mode} = 0.85 \times 2^{QP/3} \tag{12}$$

The optimum mode is the one that minimizes the Lagrange cost function:

$$m^* = argmin_{m \in M}\big(J_1(m)\big) \tag{13}$$

where $M$ is the set of candidate modes. As an example, for P-slices in H.264/AVC, the set of candidate macroblock modes are $\{$skip, inter_16x16, inter_16x8, inter_8x16, inter_8x8, intra_4x4, intra_16x16$\}$.

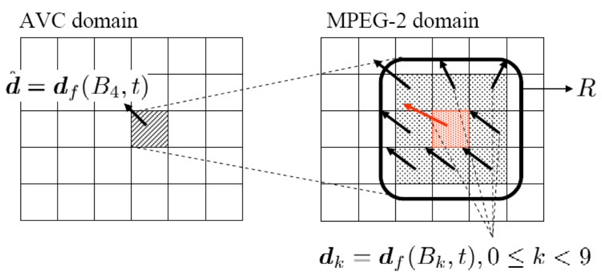


Fig. 8. Error-variance weighted mapping for inter_8x8 block

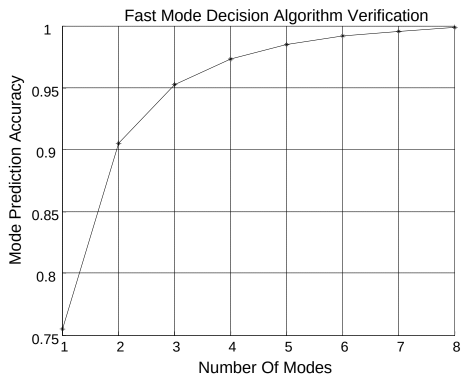


Fig. 9. Number of test modes vs. accuracy

## *A. Ranking based mode decision*

The process for calculating the Lagrange cost needs be performed many times since there are a large number of available modes for coding a macroblock. Therefore, the computation of the rate-distortion optimized coding mode decision is very intensive. This motivates us to develop a more efficient mode decision algorithm.

The basic idea of the proposed approach is to rank all candidate modes using a simpler cost function, and then evaluate the more complex Lagrange rate-distortion costs only for the reduced set of modes determined by the ranking.

The simpler cost function that is used in this current work is the low-complexity cost function used in JM software based on the sum of absolute difference of the Hadamard-transformed prediction residual error signal:

$$SATD(m) = \left\|T(s - \hat{s}(m))\right\|_1 \tag{14}$$

where $\hat{s}(m)$ is the prediction signal using the mode $m$, and $T()$ is the Hadamard-transform operator. The cost function would then be given by,

$$J_2(m) = SATD(m) \tag{15}$$

In this scheme, we first compute the SATD cost $J_2(m)$ for all possible modes. Then, we sort the modes according to their SATD costs in ascending order and put the first $k$ modes in the a test set, denoted as $T$. For the modes in

$T$, we compute the Lagrange cost $J_1$ using (10), and then we finally select the best mode according to the Lagrange cost. The parameter $k$ controls the complexity-quality trade-off.

To verify the correlations between rankings using SATD cost and Lagrange cost, a simple experiment is performed. We collect the two costs for all luma 4x4 blocks in the first frame of 5 CIF test sequences coded with QP=28, and then count the percentage of times when the best mode according to Lagrange cost is in the test set $T$. This is called the mode prediction accuracy. The results are plotted in Fig. III as $k$ vs. mode prediction accuracy. The strong correlation between the two costs is evident in the high accuracies shown. In this work, $k$ is set to be 3.

Note that the above algorithm applies first to decisions of intra_4x4 prediction modes, and then again to the decisions of various inter and intra prediction modes. When applied to intra_4x4 prediction modes, the SATD cost has an additional term compared to (15):

$$J_2(m) = SATD(m) + \lambda_{mode} \times 4 \times (1 - \delta(m = m^+)) \tag{16}$$

where $m^+$ is the most probable mode for the block. The additional term adds a simple rate constraint.

### B. *Transform domain cost calculation*

Transform domain cost calculation is based on our previous work [10], where we found out that the coding distortion based on SSD (11) can be more efficiently calculated in the transform domain without reconstruction of pixels as follows:

$$SSD(m) = \left\| \left(E - \tilde{E}(m) \otimes W_1\right) \otimes W_2 \right\|_2^2 \tag{17}$$

where $E$ and $\tilde{E}$ are the transformed residual signal and reconstructed transform residual signal through inverse scaling and inverse transform. $\otimes$ is the operator for scalar multiplication or element-wise multiplication. $W_1$ and $W_2$ are weighting matrices to compensate for the different norms of H.264/AVC transform and quantization design, and are given as:

$$W_1 = \frac{1}{64}\begin{pmatrix} 16 & 20 & 16 & 20 \\ 20 & 25 & 20 & 25 \\ 16 & 20 & 16 & 20 \\ 20 & 25 & 20 & 25 \end{pmatrix}$$

$$W_2 = \begin{pmatrix} \frac{1}{4} & \frac{1}{\sqrt{40}} & \frac{1}{4} & \frac{1}{\sqrt{40}} \\ \frac{1}{\sqrt{40}} & \frac{1}{10} & \frac{1}{\sqrt{40}} & \frac{1}{10} \\ \frac{1}{4} & \frac{1}{\sqrt{40}} & \frac{1}{4} & \frac{1}{\sqrt{40}} \\ \frac{1}{\sqrt{40}} & \frac{1}{10} & \frac{1}{\sqrt{40}} & \frac{1}{10} \end{pmatrix}$$

When we can compute the distortion in transform domain, there is not need to perform the inverse transformation and reconstruction of pixels. Therefore complexity for mode decision is reduced. This technique is used in calculation the Lagrange rate-distortion costs in the intra_4x4 mode decision process. For more details about the technique, please refer to [10].

## IV. Simulation Results

We use two interlaced sequences in the simulations: HarborScene and StreetCar. Both of them have resolution 1920x1080 with frame rate 30 frames/s, and are 15 seconds long (450 frames). They are encoded using the MPEG-2 reference software [11] 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. The search ranges are set to be 85, 170, 255 for temporal prediction distance of 1, 2 and 3 frames, respectively.

As a benchmark, we compare the performance of the proposed transcoding algorithms to the reference transcoder, which is actually a cascaded MPEG-2 decoder and an H.264/AVC encoder, where the H.264/AVC encoder encodes the pictures reconstructed by the MPEG-2 decoder. The H.264/AVC encoder we use is the JM10.2 reference software [12]. Inter predictions using block sizes 4x8, 8x4 and 4x4 are disabled to make fair comparisons.

In all simulations, I-frames are always transcoded to I-pictures. The QP values are chosen for the H.264/AVC quantization such that the output bit rate is around 10Mbps, which is the target bit rate of interest for consumer storage applications. UVLC is the H.264/AVC entropy coding method.

The relative complexities for proposed transcoder algorithms are shown in Fig. 13. The complexity numbers are measured as the average CPU time consumed by the transcoder for all QP values for the StreetCar sequence. Only StreetCar results are shown since the results are similar for other sequences. Complexity analysis will be provided in further discussions below.

### A. *Motion mapping evaluation*

The simulation results reported in this subsection are aimed at evaluating the performance of the DWA and EWA motion mapping algorithms. We simulate two transcoders, one using the motion mapping process described in Section II with DWA, and the other one with EWA. We compare the performance of the two transcoders to that of the reference transcoder. To demonstrate the performance of the complete mapping algorithm, we convert incoming P- and B-frames to P-pictures of H.264/AVC output. This effectively simulates the case in which compliance to the H.264/AVC Baseline Profile is desired.

The results are shown in Fig. 10. It is clear from these plots that the proposed mapping algorithms (both DWA and EWA) achieve comparable rate-distortion performance to the reference transcoder. At 10 Mbps rate point, the performance loss relative to the reference transcoder in terms of PSNR is less than 0.4dB for HarborScene and about 0.15dB for StreetCar. Compared to the reference transcoder using exhaustive search, the complexity saving is more than 95%. The complexity is measured using consumed CPU time, and the computational saving is similar for both sequences and methods. Both DWA and EWA have almost the same PSNR performance (hard to tell the difference from the rate-distortion plot), therefore in all remaining simulations we will use DWA as the mapping algorithm. In terms of computational complexity, DWA is slightly simpler than EWA. However, the difference is

[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.

fairly small considering the overall trancoder complexity. For this reason, in the complexity plot Fig. 13, only DWA complexity is shown.

## *B. Mode decision evaluation*

The simulation results reported in this subsection evaluate the performance of the proposed ranking-based mode decision algorithm. We simulate two transcoders with RDO turned on. One uses the proposed ranking-based RDO algorithm, while the other one uses an exhaustive RDO algorithm. Both use the DWA mapping algorithm. As additional points of comparison, we also plot the performance of the DWA mapping with RDO off (DWA) and the reference transcoder with RDO on (REF+RDO).

As shown in Fig. 11, the performance of the simplified ranking-based mode decision (DWA+ranking) is very close to the RDO mode decision (DWA+RDO), with negligible PSNR loss of less than 0.1dB. The simplified mode decision saves close to 50% of the computation compared to the exhaustive RDO mode decision. We also observe that the simplified RDO mode decision adds about 50% complexity relative to DWA without RDO, while achives a small PSNR improvement. Consistent to the previous set of simulations, small ferformance gaps can be observed relative to the reference transcoder with RDO.

## *C. Impact of B-slices*

In the last set of simulations, we examine the performance changes when B-slices are introduced as part of the output. In Fig. 12, performance of proposed transcoder using DWA and B slices with RDO both on (DWA_IPB+RDO) and off (DWA_IPB) are shown. Again, the performance of DWA without RDO or B slices are shown as reference. We also plot the performance of reference transcoder using B slices and RDO (REF_IPB+RDO) for comparison. The interesting point worth noting is that introducing B-slices actually improves the compression efficiency fairly significantly while not significantly increasing the complexity. The reason seems to be that when B-slices are introduced, the number of frames for which sub-pel interpolation is needed is reduced by 2/3. Although transcoding B-slices would need more complexity for motion compensation and motion mapping, the complexity saving from sub-pel interpolation offsets much of the increase. This suggests B-slices could be a more cost-effective tool for the transcoder design than the RDO tool. We can also see from the plots that the proposed transcoder achieves comparable quality to the reference transcoder.

## V. Concluding Remarks

We presented motion mapping algorithms that can efficiently map incoming MPEG-2 motion vectors to outgoing H.264/AVC motion vectors, even when they have different block size support and different reference pictures. We then presented an efficient rate-distortion optimized macroblock coding mode decision algorithm, where we first evaluate candidate modes based on a simple cost function so that a reduced set of candidate modes is formed, then we evaluate the more complex Lagrangian cost calculation only for the reduced set of modes. Extensive 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 maintained while the complexity is significantly reduced. We also compared the transcoding performance with and without frame type conversion.

With the above transcoding algorithms, our current transcoder's complexity is dominated by sub-pel interpolation and motion refinement. Also, mode decision still has considerable complexity. Therefore, as future work, we will explore even more efficient transcoder design along these directions.

## 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.*, 20(2):18–29, March 2003.

[4] J. Xin, C.-W. Lin, and M.-T. Sun. Digital video transcoding. *Proc. IEEE*, 93(1):84–97, January 2005.

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

[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] F. Pan, X. Lin, R. Susanto, K.P. Lim, Z.G. Li, G.N. Feng, D.J. Wu, and S. Wu. JVT-G013: Fast mode decision for intra prediction, 2003.

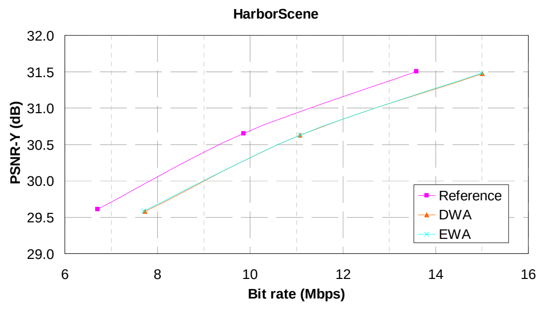
[8] K.P. Lim, S. Wu, D.J. Wu, S. Rahardja, X. Lin, F. Pan, and Z.G. Li. JVT-I020: Fast inter mode selection, 2003.

[9] Y. Su, J. Xin, A. Vetro, and H. Sun. Efficient MPEG-2 to H.264/AVC intra transcoding in transform-domain. In *IEEE Int. Symposium on Circuits and Systems*, 2005.

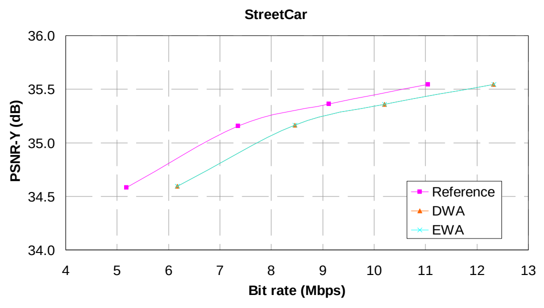
[10] J. Xin, A. Vetro, and H. Sun. Efficient macroblock coding mode decision for H.264/AVC video coding. In *Picture Coding Symposium*, 2004.

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

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

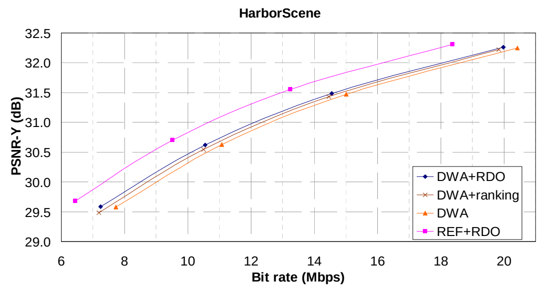


(a)

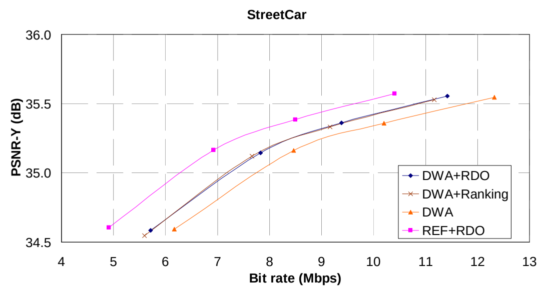


(b)

Fig. 10. RD performance comparison for motion mapping algorithms: (a) HaborScene; (b) StreetCar.

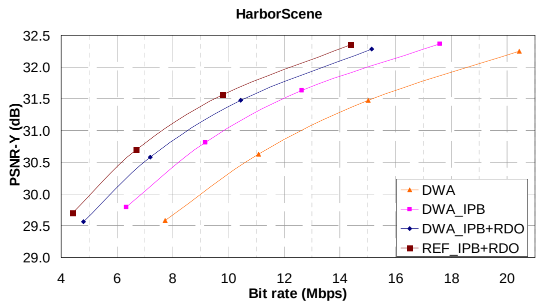


(a)

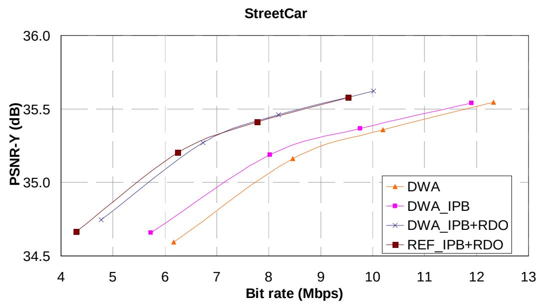


(b)

Fig. 11. RD performance comparison for rate-distortion optimization algorithms: (a) HarborScene; (b) StreetCar.



(a)



(b)

Fig. 12. RD performance comparison for motion mapping algorithms with B slices output: (a) HarborScene; (b) StreetCar.

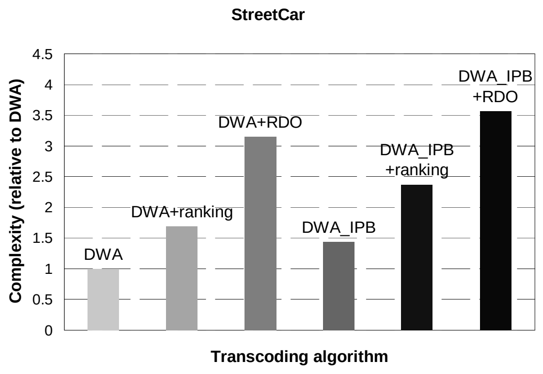


Fig. 13. Complexity comparison for proposed transcoding algorithms for StreetCar sequence. Complexities are relative to the complexity of DWA algorithm with RDO off.