Motion Vector Refinement for High-Performance Transcoding

Jeongnam Youn, Ming-Ting Sun, *Fellow, IEEE,* and Chia-Wen Lin

***Abstract*— In transcoding, simply reusing the motion vectors extracted from an incoming video bit stream may not result in the best quality. In this paper, we show that the incoming motion vectors become nonoptimal due to the reconstruction errors. To achieve the best video quality possible, a new motion estimation should be performed in the transcoder. We propose a fast-search adaptive motion vector refinement scheme that is capable of providing video quality comparable to that can be achieved by performing a new full-scale motion estimation but with much less computation. We discuss the case when some incoming frames are dropped for frame-rate conversions, and propose motion vector composition method to compose a motion vector from the incoming motion vectors. The composed motion vector can also be refined using the proposed motion vector refinement scheme to achieve better results.**

I. Introduction

NETWORKED multimedia services, such as teleconferencing, video on demand, and distance learning are emerging. In these applications, it is often needed to adapt the bitrate of the coded video bit streams to the available bandwidth of various channels [1]–[5]. In a heterogeneous network, the bitrate adaptation allows different end-users with different subnetworks to have different quality of service (QoS) based on their available network bandwidths. For some real-time video coding applications, the adaptation of the bitrates can be achieved through the rate-control in the video encoder. However, for many other applications such as video on demand, this can not be done since the video is already compressed at a certain bitrate and stored in the server.

Dynamic bitrate adaptation with limited capability can be achieved using the scalable coding provided in current video coding standards [7], [8]. Scalable coding supports a variety of scaled video qualities with different peak signal-to-noise ratios (PSNR's) (PSNR scalability), frame-rates (temporal scalability), or spatial resolutions (spatial scalability). To achieve different levels of video quality, the video source is first encoded with a low PSNR, low frame-rate, or low spatial resolution to form a base layer. The residual information between the base layer and the original input is then encoded to form one or more enhancement layers. Successful transmission of the base layer results in a video sequence with basic quality. Additional transmissions of the enhancement layers enhance the quality by adding the residual information. Scalable coding in current video coding standards can provide only up to three levels of video quality because of the limitation on the number of enhancement layers. However, many applications require a much finer scaling capability [4].

Converting a previously compressed video bit stream to a lower bitrate through transcoding can provide finer and more dynamic adjustments of the bitrate of the coded video bit stream to meet various channel situations [9]–[15]. One of the simplest architectures for transcoding is open-loop transcoding in which the incoming bitrate is downscaled by modifying the discrete cosine transform (DCT) coefficients. For example, the DCT coefficients can be truncated, requantized, or partially discarded in the optimal sense [9], [15] to achieve the desirable lower bitrate. In the open-loop transcoding, because the transcoding is carried out in the coded domain where complete decoding and reencoding are not required, it is possible to construct a simple and fast transcoder. However, open-loop transcoding can produce "drift" degradations due to the mismatched reconstructed pictures in the front-encoder and the end-decoder, which often results in unacceptable video quality.

Drift-free transcoding is possible by the direct cascade of a decoder and an encoder as shown in Fig. 1. Although this transcoder has higher complexity than the open-loop transcoder, some information extracted from the incoming video bit stream after the decoding can be used to significantly reduce the complexity of the encoder. Thus, the complexity may not be as bad as it appears.

In this paper, we consider the cascaded architecture as our framework for high-performance transcoding. The cascaded transcoder is very flexible and easily extendible to various types of transcoding, such as temporal or spatial resolution conversions. We will investigate techniques which can reduce the complexity while maintaining the same level of video quality.

In transcoding, motion estimation is usually not performed in the transcoder because of its computational complexity. Instead, motion vectors extracted from the incoming bit stream are reused. However, this simple motion-vector reuse scheme may introduce considerable quality degradation in many applications [16], [17]. Although an optimized motion vector can be obtained by a full-scale motion estimation, this is not desirable because of its high computational complexity.

In this paper, we propose a new motion vector refinement scheme for a transcoder for the H.263 [23] encoded bit

Manuscript received September 3, 1998; revised December 15, 1998. The associate editor coordinating the review of this paper and approving it for publication was Dr. M. Reha Civanlar.

J. Youn and M.-T. Sun are with the Department of Electrical Engineering, Information Processing Laboratory, University of Washington, Seattle, WA 98195 USA (e-mail: jnyoun@ee.washington.edu; sun@ee.washington.edu).

C.-W. Lin is with the Computer and Communication Research Laboratories, ITRI, Hsinchu, Taiwan 310, R.O.C. (e-mail: ljw@n1sun3.ccl.itri.org.tw).

Publisher Item Identifier S 1520-9210(99)01934-3.

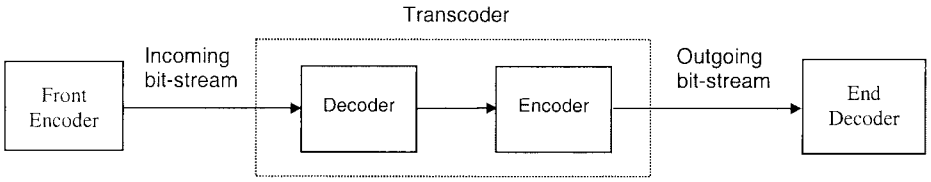


Fig. 1. Transcoding by the cascade of a decoder and an encoder.

streams. We show that the incoming motion vectors become nonoptimal due to the quantization errors. To achieve the best quality possible, new motion estimation should be performed in the transcoder. We propose a fast-search adaptive motion vector refinement scheme that is capable of providing video quality comparable to that which can be achieved by performing a new full-scale motion estimation but requires considerably less computation when compared to full-scale motion estimation. We discuss the case when some incoming frames are dropped for frame-rate conversions and propose a motion vector composition method to compose a motion vector from the incoming motion vectors. The composed motion vector can also be refined using the proposed motion vector refinement scheme to achieve better results. The organization of this paper is as follows. In Section II, the motion estimation in transcoding and the effect of quantization errors on the motion vectors are discussed. In Section III, we introduce the motion vector refinement scheme. Motion vector refinement in the frame-rate conversion is presented in Section IV. An adaptive motion-vector refinement scheme based on the analysis in Section II is proposed in Section V. A fast search algorithm for the motion vector refinement is discussed in Section VI. Complete simulation results are presented in Section VII. Finally, a conclusion is provided in Section VIII.

II. Motion Estimation in Transcoding

Current video compression techniques exploit mainly two types of redundancies in the uncompressed video signal to achieve the desired compression gain [18]. First, preserving only significant DCT coefficients can considerably eliminate the spatial redundancy between pixels within a single frame because of the energy compaction property of the DCT. Furthermore, the motion-compensated predictive coding scheme is used to remove the temporal redundancy between frames. In other words, a motion-compensated block in the previous reconstructed reference frame is subtracted from the current macroblock. The residual signal is encoded using DCT to further remove the spatial redundancy.

To find the motion vector for a macroblock in the current frame, a best matching macroblock is searched within a predefined search window S in the previous reconstructed reference frame as shown in Fig. 2. The motion vector is defined as the displacement of the best matching block from the position of current macroblock.

The motion estimation is performed on the luminance macroblocks and is usually based on the sum of absolute differences (SAD) of the respective pixels. A block with the minimal SAD is considered the best matching block. To obtain

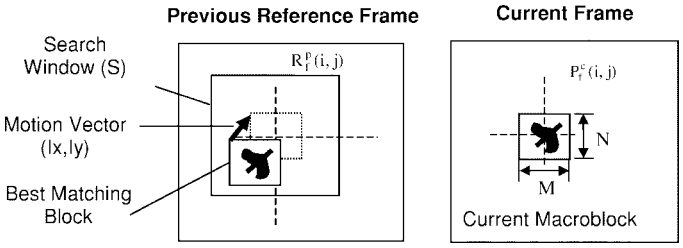


Fig. 2. Motion estimation.

the motion vector for the current macroblock:

$$(Ix, Iy) = \arg \min_{(m,n)\in S} \mathrm{SAD}_f(m, n) \tag{1}$$

$$\mathrm{SAD}_f(m, n) = \sum_{i}^{M} \sum_{j}^{N} \left| P_f^c(i, j) - R_f^p(i+m, j+n) \right| \tag{2}$$

where m and n are the horizontal and vertical components of the displacement of a matching block, $P_f^c(i, j)$ and $R_f^p(i, j)$ represent a pixel in the current frame and in the previous reconstructed reference frame, respectively. The superscript "c" and "p" denotes the "current" and "previous" frame, respectively, and the subscript "f" is used to indicate the "front-encoder" as shown in Fig. 3. In later sections, the subscript "s" will be used to denote the "second-encoder" in the transcoder.

Fig. 3 details the structure of the cascaded transcoder. The motion estimation block is omitted for simplicity. Since the output bitrate is lower than the input bitrate, the quantization step size in Q2 in the transcoder is usually much coarser than the quantizer step size in Q1 in the front encoder.

In the transcoder, an optimized motion vector for the outgoing bit stream can be obtained by applying the motion estimation such that

$$(Ox, Oy) = \arg \min_{(m,n)\in S} \mathrm{SAD}_s(m, n) \tag{3}$$

$$\mathrm{SAD}_s(m, n) = \sum_{i}^{M} \sum_{j}^{N} |P_s^c(i, j) - R_s^p(i+m, j+n)|. \tag{4}$$

From Fig. 3, since the reconstructed picture in the front-encoder is the same as the current input frame to the second-encoder, $R_f^p(i, j) = P_s^p(i, j)$, and $R_f^c(i, j) = P_s^c(i, j)$. Thus, from (4)

$$\begin{aligned}\mathrm{SAD}_s(m, n) &= \sum_{i} \sum_{j} |P_s^c(i, j) - R_s^p(i+m, j+n)| \\ &= \sum_{i} \sum_{j} |P_f^c(i, j) - R_f^p(i+m, j+n) \\ &\quad + \Delta_f^c(i, j) - \Delta_s^p(i+m, j+n)|\end{aligned} \tag{5}$$

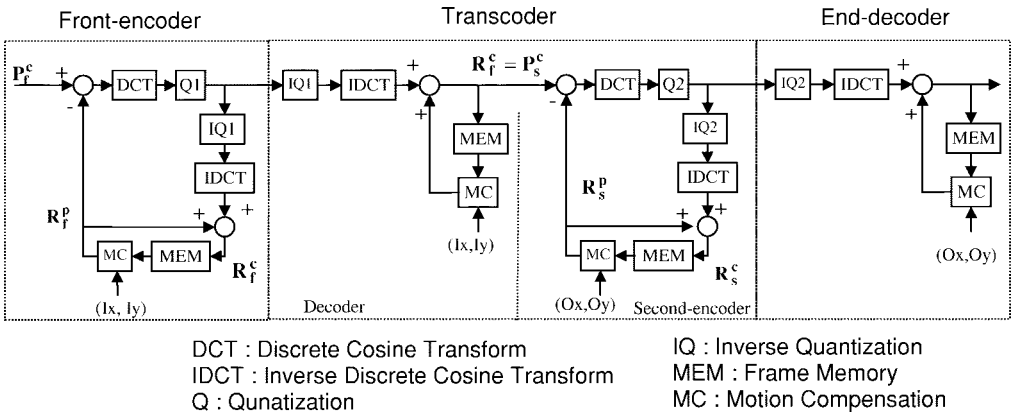


Fig. 3. Structure of a cascaded transcoder.

where $\Delta_f^c(i,j) = R_f^c(i,j) - P_f^c(i,j)$ and $\Delta_s^p(i,j) = R_s^p(i,j) - P_s^p(i,j)$.

In (5), $P_f^c(i,j) - R_f^p(i+m, j+n)$ is the term used to compute the SAD in the front-encoder in (2) to give the incoming motion vector. $\Delta_f^c(i,j)$ represents the reconstruction error of the current frame in the front-encoder due to the quantization Q1, while $\Delta_s^p(i,j)$ represents the reconstruction error of the previous frame in the second-encoder due to the quantization Q2. From (5), when the effect of the term $\Delta_f^c(i,j) - \Delta_s^p(i+m, j+n)$ is negligible, performing a new motion estimation will give the same motion vector as the incoming motion vector (i.e., the incoming motion vector is optimal). However, since in general there is no guarantee that the effect is negligible all the time, there are nonzero probabilities that the quantization errors may cause the incoming motion vector to be nonoptimal [i.e., we can find a better motion vector which minimizes (4)].

III. Motion Vector Refinement (MVR)

Although the optimized motion vector can be obtained by a new motion estimation, it is not desirable because of its high computational complexity. The reuse of the incoming motion vectors has been widely accepted because it was generally thought to be almost as good as performing a new full-scale motion estimation and was assumed in many transcoder architectures [10]–[12]. However, as discussed in the previous section, simply reusing the incoming motion vectors is not optimal. Our simulation results (which will be presented in the later sections) show that its performance may be considerably worse than that can be achieved with a new motion estimation.

In the analysis in the previous section, we showed that the differential reconstruction error causes incoming motion vectors to deviate from optimal values. In most macroblocks the deviation is within a small range and the position of the optimal motion vector will be near that of the incoming motion vector. Therefore, the optimal motion vector can be easily obtained by refining the incoming motion vector within a small range as opposed to applying a full-scale motion estimation [16], [17].

For the refinement of incoming motion vectors, we define a base motion vector (Bx, By) as the motion vector obtained from the incoming bit stream. A delta motion vector (Dx, Dy) can be estimated within a new search window S^R, around the point indicated by the base motion vector:

$$(Dx, Dy) = \arg \min_{(m,n)\in S^R} \text{SAD}_R \tag{6}$$

$$\text{SAD}_R = \sum_i \sum_j |P_s^c(i,j) - R_s^p(i+Bx+m, j+By+n)|. \tag{7}$$

The new search window S^R can be set much smaller than the full-scale window S (e.g., a search range of ± 2 pixels instead of ± 15 pixels or larger) and still produce almost the same video quality as the full-scale motion estimation.

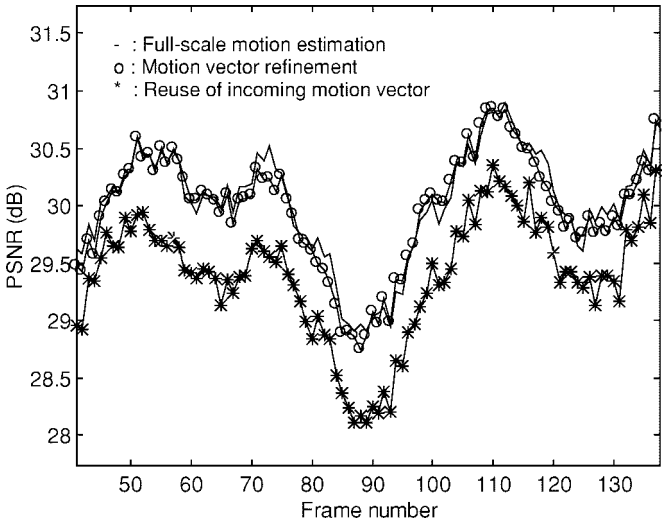


Fig. 4. Performance of motion vector refinement ("Carphone" of QCIF format, 300 frames). Incoming bit stream at 128 Kb/s was transcoded to 32 Kb/s with 30 frames/s. The search range for full-scale motion estimation is ± 15 pixels, and the search range for motion vector refinement is ± 2 pixels throughout the paper.

Fig. 4 shows the performance of motion vector refinement. The quality degradation introduced by reusing incoming motion vectors is about 0.45 dB on average compared with the application of a new full-scale full-search motion estimation. However, the refinement of the incoming motion vectors using a small search window (e.g., search range of ± 2 pixels) increases the performance close to that of the full-scale full-search motion estimation. Detailed simulation environment

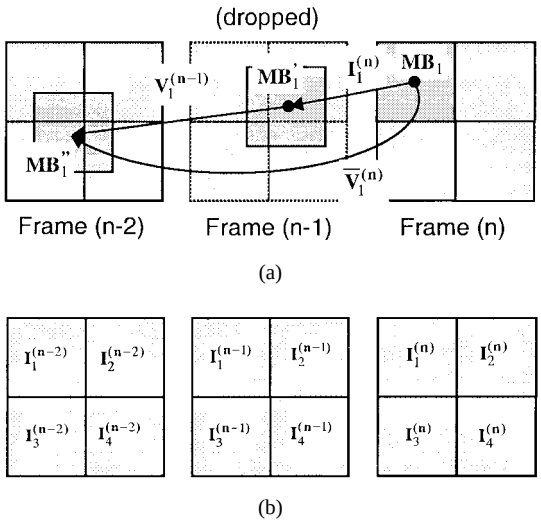


Fig. 5. Backward motion vector interpolation. (a) Tracking macroblocks in backward order and (b) incoming motion vectors.

and coding parameters used in the simulations are described in Section VII.

IV. Motion Vector Refinement in Frame-Rate Conversion

To transport video over low bandwidth channels, such as the public switched telephone network (PSTN) or wireless network, a high transcoding ratio is required. However, the high transcoding ratio may result in unacceptable picture quality when the video is transcoded with the full frame-rate or full spatial resolution. For example, in a wireless network, which normally has less than 20 Kb/s bandwidth, the quality degradation due to the low bitrate is significant at 25 or 30 frames/s). Frame-rate reduction is often used as an efficient scheme to allocate more bits to the remaining frames, so that acceptable quality for each frame can be maintained. In addition, the frame-rate conversion is also needed when an end-system supports only a lower frame-rate. In this section, we discuss motion vector refinement for transcoding involving the frame-rate conversion.

A. Base Motion Vector Composition

When some incoming frames are dropped for the frame-rate conversion, the incoming motion vectors are not valid because they point to the dropped frames that do not exist in the transcoded bit stream.

In Fig. 5, a situation where one frame is dropped is illustrated. In the figure, we assume that MB_1' represents the best matching block to MB_1, and MB_1'' represents the best matching block to MB_1'. Since frame $(n-1)$ is dropped, for MB_1, we need to find a motion vector pointing to a block in frame $(n-2)$ which matches well with MB_1. One possible way to generate such a motion vector without performing motion estimation is to use the vector sum of $I_1(n)$ and $V_1(n-1)$. In practice, however, since MB_1' is not on a macroblock-boundary, $V_1(n-1)$ is not available from the incoming bit stream. It is possible to use the bilinear interpolation from the motion vectors $\{I_1^{(n-1)}\ I_2^{(n-1)}\ I_3^{(n-1)}\ I_4^{(n-1)}\}$ of the four neighboring macroblocks with MB_1' to come up with an approximation of $V_1(n-1)$ [6], [17]. However, the bilinear interpolation of motion vectors has several drawbacks for temporal transcoding. First, for consecutively dropped frames, the interpolation has to be processed in the backward order starting from the last dropped frame to the first dropped frame. This backward processing requires all motion vectors of the dropped frames to be stored, which requires much extra memory. Another drawback of the bilinear interpolation is the inaccuracy of the resultant motion vector. In spite of the weighting of each neighboring motion vector based on the overlapping segments, unreliable motion vectors can be produced because the area covered by the four macroblocks may be too divergent and too large to be described by a single motion vector.

In this paper, we propose a forward dominant vector selection (FDVS) method. The proposed method selects one dominant motion vector from the four neighboring macroblocks. A dominant motion vector is defined as the motion vector carried by a dominant macroblock. The dominant macroblock is a macroblock that has the largest overlapping segment with the block pointed by the incoming motion vector. For example, for block MB_1' in Fig. 5

$$V_1^{(n-1)} = I_2^{(n-1)}. \tag{8}$$

Our simulation results show that FDVS can achieve higher performance with less computation than the bilinear interpolation. Another advantage of FDVS over the bilinear interpolation scheme is that when multiple frames are dropped, it can be processed in the forward order, eliminating the multiple memories needed to store the incoming motion vectors of all the dropped frames. Fig. 6 shows a case when two frames are dropped. When the frame $(n-2)$ is dropped, we store its motion vectors in a table. The stored motion vectors will be used to compose motion vectors at the next frame-dropping. That is, when the frame $(n-1)$ is dropped, the FDVS searches a dominant macroblock for each current macroblock. For example, the first macroblock in frame $(n-2)$ becomes the dominant macroblock of the second macroblock in frame $(n-1)$. The dominant motion vector is selected from the table at the location of the first macroblock, and then added to the current incoming motion vector corresponding to the current second macroblock. Then the table is updated with the new composed value. In Fig. 6, the resultant motion vector for the second macroblock at frame $(n-1)$ will be $I_1^{(n-2)} + I_2^{(n-1)}$. When the frame (n) is processed, the composed motion vector for the first macroblock at frame (n) will be set at $[I_1^{(n-2)} + I_2^{(n-1)}] + I_1^{(n)}$ because the stored value in the table for the dominant block pointed by $I_1^{(n)}$ will be the dominant motion vector of MB_1'. Using this scheme, only one table is needed for all the dropped frames.

During the composition process, intracoded macroblocks may exist in the dropped frames. Since intracoded macroblock does not carry motion vector, we assume a zero motion vector for this macroblock to continue the composition process. After finishing the composition, the macroblock coding mode is recomputed.

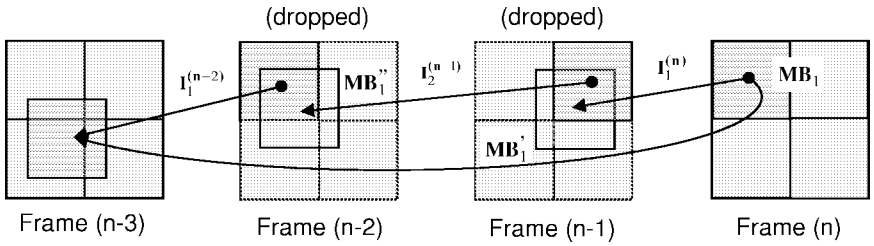
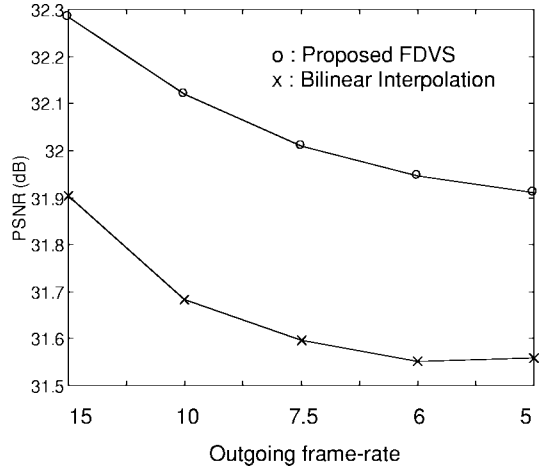
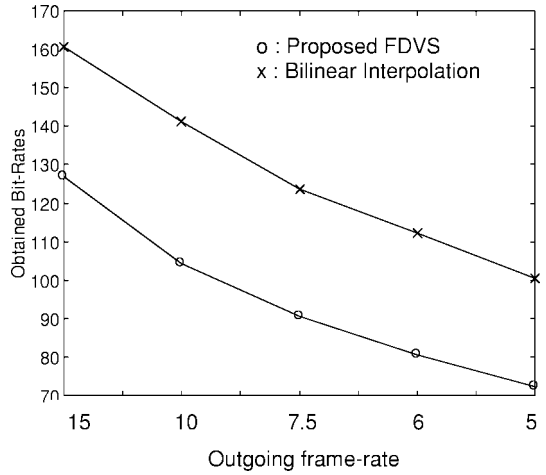


Fig. 6. Forward dominant vector selection (FDVS) composition scheme.

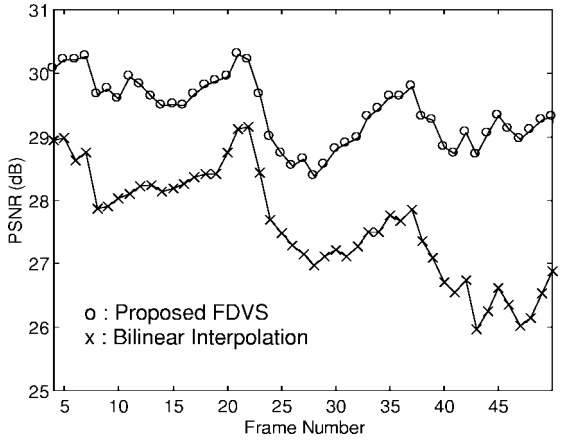


(a)

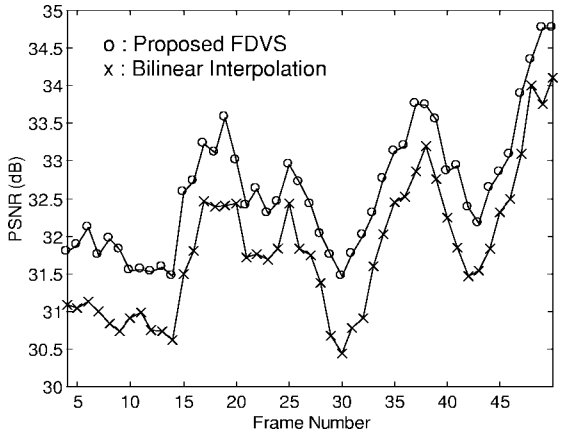


(b)

Fig. 7. Performance comparison of the proposed FDVS and the bilinear interpolation motion vector composition. "Foreman" was encoded at 30 frames/s using a quantization parameter of 7 over 300 frames, and then transcoded at different frame-rates using the same quantization parameter of 7. (a) Quality comparison and (b) generated bitrate.



(a)



(b)

Fig. 8. Performance comparison of motion vector composition methods (constant bitrate). Incoming bit streams at 128 Kb/s with 30 frames/s (300 frames) were transcoded to 50 Kb/s with 10 frames/s (100 frames). (a) Foreman sequence and (b) carphone sequence.

The performances of the bilinear interpolation method and the proposed FDVS method are compared in Figs. 7 and 8 using a public domain software [24]. As shown in Fig. 7(a), the proposed FDVS outperforms the bilinear interpolation method at various outgoing frame-rates. Also, the bilinear interpolation produces much higher bitrates than the FDVS as shown in Fig. 7(b).

Fig. 8 and Table I show another simulation results of two test sequences, "foreman" and "carphone." The performance of the proposed FDVS method is about 1.7 dB (foreman) and 0.8 dB (carphone) better than the bilinear interpolation. It should be noted that a composed motion vector might not be optimal because each dominant motion vector is an approximated value. Furthermore, the composed motion vector may have degraded performance due to the effect of reconstruction errors when a coarser quantization step size is applied during the transcoding. Therefore, the composed motion vector also needs to be refined to improve the performance.

B. Motion Vector Refinement for the Composed Motion Vector

If the k frames from $n-k$ to $n-1$ are dropped during transcoding, the base motion vector can be composed by

TABLE I
PERFORMANCE OBTAINED USING DIFFERENT MOTION VECTOR COMPOSITION METHODS. INCOMING BIT-STREAM OF 128 Kb/s AND 30 frames/s WAS TRANSCODED INTO 50 Kb/s AND 10 frames/s

Test sequence	Composition method	Average PSNR
Foreman	FDVS	29.5 dB
	Bilinear interpolation	27.8 dB
Carphone	FDVS	32.7 dB
	Bilinear interpolation	31.9 dB

applying the FDVS. The resultant base motion vector will be

$$(Bx, By)_n = \left(\sum_{d=k}^{1}(Vx)_{n-d} + (Ix)_n, \sum_{d=k}^{1}(Vy)_{n-d} + (Iy)_n\right) \tag{9}$$

where $(Vx, Vy)_{n-d}$ is the dominant motion vector at the frame $(n-d)$, and $(Ix, Iy)_n$ is the incoming motion vector of the frame (n).

The delta motion vector is estimated within a new search area around the base motion vector as in the case of nonframe-dropping:

$$(Dx, Dy) = \arg \min_{(m,n)\in S^D} \text{SAD}_D \tag{10}$$

$$\text{SAD}_D(m, n) = \sum_i \sum_j |P_s^n(i, j) - R_s^{n-k-1}(i + Bx + m, j + By + n)| \tag{11}$$

where the previously reconstructed reference frame for (Dx, Dy) is set to the frame $(n-k-1)$, the frame before the first dropped frame. Our simulation results show that the new search area S^D can be as small as S^R, the small search area used in motion vector refinement without frame-dropping. Note that the macroblock coding mode is recomputed based on the refined motion vector, not the composed motion vector.

Figs. 9 and 10 shows the performance of motion vector refinement in frame-rate conversion. In the simulation, ± 2 pixels were used for S^D. As shown in Fig. 9, the quality degradation introduced by using only the base motion vectors was significant, about 0.7 dB on average, when compared with the application of full-scale motion estimation. However, the refinement of the base motion vectors increases the performance almost to the level of the full-scale full-search motion estimation.

V. ADAPTIVE MOTION VECTOR REFINEMENT

In the motion vector refinement discussed in the previous section, all incoming motion vectors are refined. However, there are cases where incoming motion vectors can produce near optimal results. Fig. 11 shows a performance comparison for the two schemes: one reusing incoming motion vectors and the other applying full-scale motion estimation. The quality obtained by reusing the incoming motion vectors is indicated by the average PSNR degradations compared to that obtained by the full-scale full-search motion estimation. The quality degradation becomes significant when the outgoing

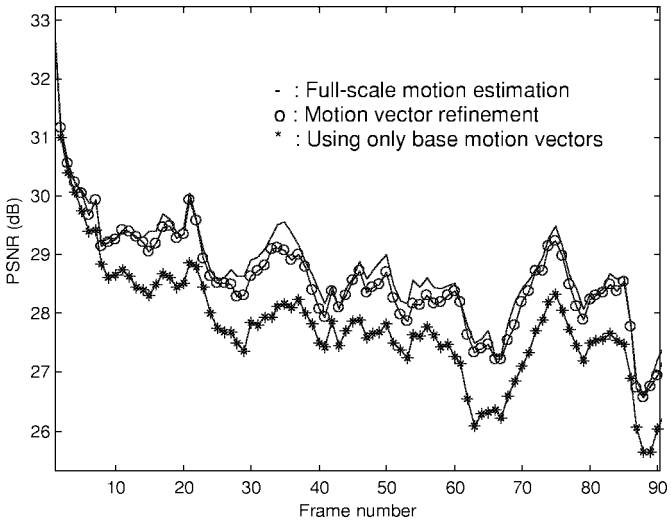


Fig. 9. Performance of motion vector refinement with frame-rate conversion ("Foreman"). Incoming bit stream at 128 Kb/s with 30 Kb/s (300 frames) was transcoded to 32 Kb/s with 10 frames/s (search range of ± 2 pixels applied).

(a)

(b)

Fig. 10. Subjective quality of the picture with the worst PSNR drop (frame number 34). Same coding parameters as in Fig. 9 were used). (a) Full-scale ME (29.29 dB) and (b) FDVS with MVR (28.90 dB).

quantization step size ($2\times$ quantization parameter) is much coarser than the incoming quantization step size. However, when the outgoing quantization step size is very close to the incoming quantization step size, the new full-scale full-search motion estimation does not produce significantly better quality. In the figure, at 30 frames/s and when the same quantization

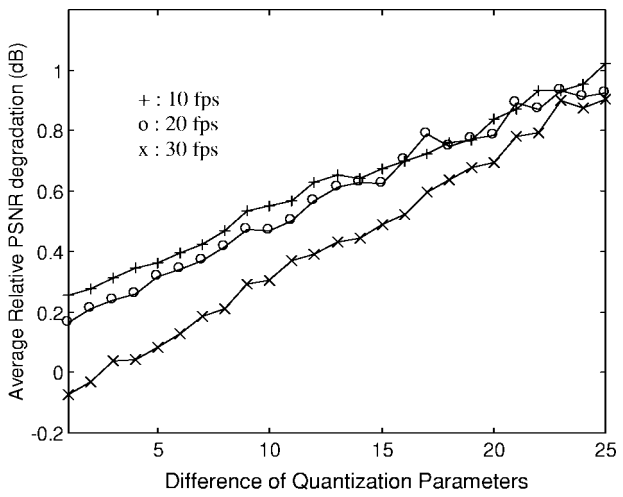


Fig. 11. Quality degradation when base motion vectors are used. "Foreman" of 300 frames was encoded using the quantization parameter of 5, and then transcoded to lower bitrate using coarser quantization parameters. The PSNR degradation means the difference PSNR between the full-scale motion estimation and the use of the base motion vectors.

step size is applied, the performance obtained by using new motion vectors is actually slightly worse than that obtained by using the incoming motion vectors. This is because when the quantization levels are similar, redoing motion estimation although sometimes can result in different motion vectors which produce slightly better SAD, those motion vectors may not optimize the PSNR. Using the incoming motion vectors results in the same reference block as that used in the front encoder. The residual prediction errors tend to be more correlated with the residual errors in the first stage encoder, which results in smaller quantization errors when the quantization levels are similar. Fig. 11 implies that when the quantization step size difference is small, the distortion caused by the reuse of incoming motion vector is small. Thus it appears that we can skip the motion vector refinement when the quantization step size difference is small. However, the skipping of the motion vector refinement based solely on the quantization step size difference is not sufficient. In general, the need for motion vector refinement depends on the effect of the reconstruction errors relative to the strength of the motion-compensated prediction residual signal as shown previously in (5), and thus is signal dependent. For example, if a macroblock is quantized to zero in the front encoder, it will also be transcoded to zero when a much coarser quantization step size is used in the transcoder.

Intuitively, when the difference of the quantization step sizes is small, the effect of the term $\Delta_f^c(i, j) - \Delta_s^p(i+m, j+n)$ in (5) becomes small. Furthermore, when the quantization step size of the second encoder is much larger than that of the front-encoder, the effect of the last two terms in (5) can be roughly approximated to $\sum_i \sum_j |\Delta_s^p(i + Bx, j + By)|$ from the observation that the reconstruction error in the second encoder $\Delta_s^p(i+Bx, j+By)$ will dominate the first reconstruction error $\Delta_f^c(i, j)$. Note that $\Delta_s^p(i+Bx, j+By)$ can be calculated in the transcoder.

Based on these observations and simulations, we propose a criteria function sum of differential reconstruction error (SDRE) for the adaptive scheme:

$$\text{SDRE}(Bx, By) = \left| \left(\frac{q_f^c}{q_s^p} \right)^2 - 1 \right| \sum_i \sum_j \cdot |\Delta_s^p(i + Bx, j + By)| \qquad (12)$$

where q_f^c and $q_s^p$are the quantization step sizes used in the current frame of the front-encoder and the previous frame of the second-encoder, respectively. When q_s^p is close to q_f^c, SDRE will be small. When q_s^p is much larger than q_f^c, SDRE will be reduced to $\sum_i \sum_j |\Delta_s^p(i+Bx, j+By)|$. The square function was determined experimentally. When some frames are dropped as in Section IV-B, "p" and "c" can be replaced by the frame $(n-k-1)$ and the frame (n), respectively. Note that the computation of (12) is as simple as checking one search position in the motion estimation. Thus it does not require much new computation.

When an incoming motion vector has zero value, a higher threshold should be used to prefer the use of the zero incoming motion vector. This is because a nonzero motion vector will need more bits to code. The proposed adaptive algorithm is summarized as follows:

Compose the motion vector when frames are dropped
When a base motion vector is zero
    if (SDRE $>$ Threshold$_2$) apply the motion vector refinement
    otherwise, skip the motion vector refinement
Otherwise
    if (SDRE $>$ Threshold$_1$) apply the motion vector refinement
    otherwise, skip the motion vector refinement.

The simulation results which will be presented in Section VII show that the adaptive motion vector refinement can reduce the computational complexity significantly while achieving nearly the same quality achieved by applying the motion vector refinement at all times.

VI. Fast Search Algorithm

For the motion vector refinement scheme discussed in previous sections, the SAD's of all checking points in the new small search window are exhaustively computed to obtain the optimal delta motion vectors. The computational complexity required by the exhaustive search can be further reduced. In this section, we propose a fast search algorithm to further reduce computational complexity by minimizing the number of required checking points.

Fast search algorithms were extensively studied for the stand-alone encoder. Most existing algorithms assume that the SAD's decrease monotonically as the checking points move close to the global minimum point. Under this unimodal error surface assumption, several fast search algorithms to reduce the total number of checking points are possible [19]–[22]. However, the unimodal error surface assumption does not hold in most real-world video sequences with a large search window. Since many local minimum points may exist within the large search window, most proposed fast algorithms are likely to be trapped in a local minimum. Whether it will be

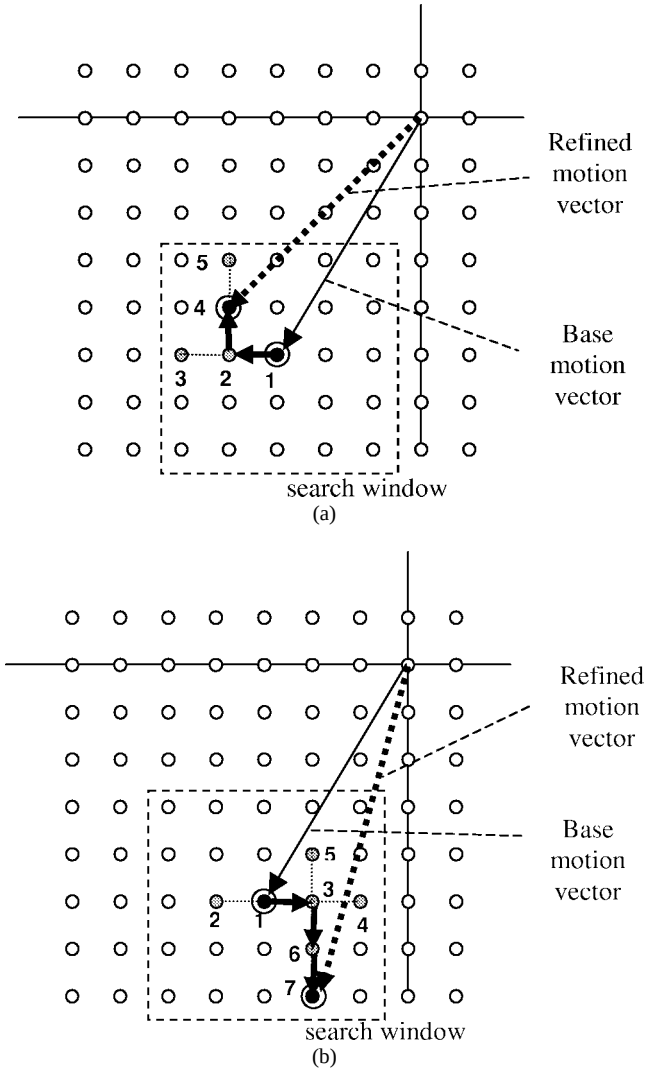


Fig. 12. Proposed HAVS method. (a) Best case: five points are checked and (b) worst case: seven points are checked.

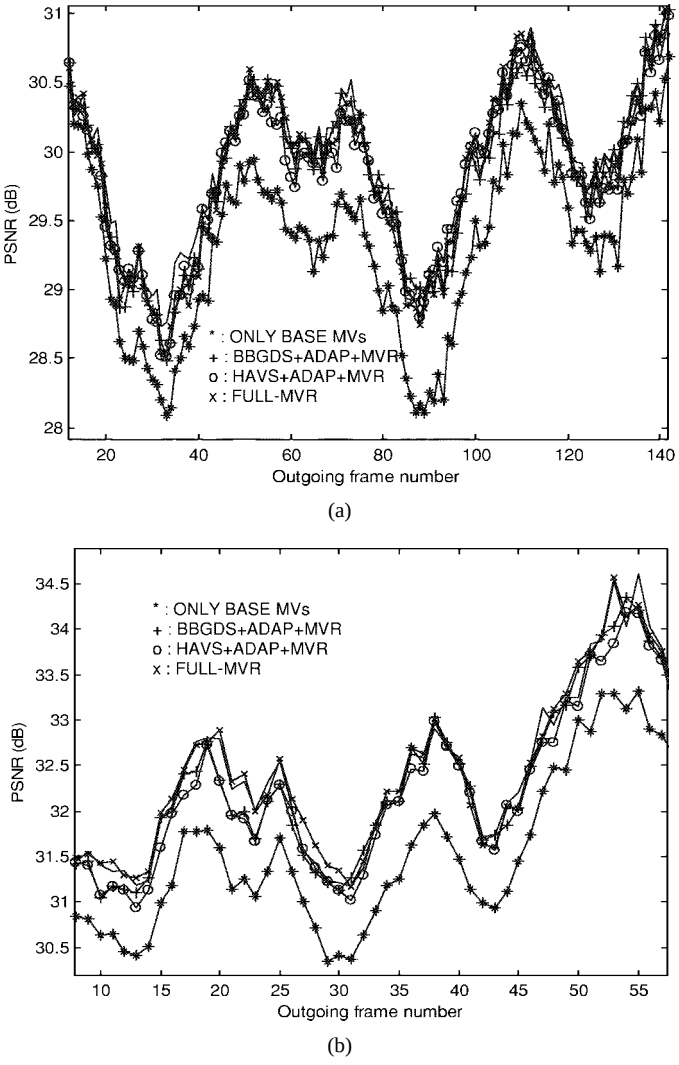


Fig. 13. Performance of the proposed fast-search, adaptive motion vector refinement when the frame-rate was changed: (a) to 30 frames/s and (b) to 10 frames/s. "Carphone" of 300 frames was encoded at 128 Kb/s with 30 frames/s, and then transcoded to 32 Kb/s. ($\text{Threshold}_1 = 300$, $\text{Threshold}_2 = 500$.) (a) Outgoing frame-rate: 30 frames/s and (b) outgoing frame-rate: 10 frames/s.

trapped in a local minimum or not depends on where we place the starting checking point. That is, if the checking points are closed to the global minimum, then the chance of hitting the optimal motion vector without being trapped in a local minimum will be higher [19], [20].

In previous sections, we discussed the perturbation due to the reconstruction errors. Since the perturbation is usually small, the base motion vector will be located near the globally optimal position. Furthermore, the unimodal error surface can be reasonably held around the point indicated by the base motion vector, especially in a small search window. Therefore, fast search algorithms are suitable for the motion vector refinement.

In this section, we propose a horizontal and vertical search (HAVS) scheme. Instead of searching all checking points within the search window, the HAVS searches first for a minimum point over the horizontal line and then over the vertical line. At the starting position in the horizontal search, only when the computed SAD on the left side is larger than that of the staring point, the points on the right side are searched. The vertical search is performed in a similar way. Fig. 12(a) shows the best case for a search range of ± 2 pixels. The HAVS compares the SAD's of the starting point at position 1 and the adjacent left point at position 2 to determine the direction for the next search. If the computed SAD at position 2 is smaller, then the point located on the left side at position 3 is checked. If the SAD at the position 2 is still smaller, a minimum point in the horizontal direction has been found. Next, we go up in the vertical direction. We compare the SAD's of position 4 and position 2. If the SAD of position 4 is smaller, we continue to go up and compare the SAD's of position 4 and position 5. If the SAD of position 4 is still smaller, we have found a minimum in both the horizontal and vertical dimensions. In this best case, only five checking points are required. Fig. 12(b) shows the worst case situation. The order of the search positions is also shown in the figure. It requires seven checking points. On average, six points are checked when a macroblock needs to perform the motion vector refinement. This number of checking points is smaller

TABLE II

PERFORMANCE OF PROPOSED FAST-SEARCH, ADAPTIVE MOTION VECTOR REFINEMENT. A TEST SEQUENCE "CARPHONE" WAS ENCODED AT 128 Kb/s WITH 30 frames/s, AND TRANSCODED INTO 32 Kb/s WITH DIFFERENT FRAME RATES. ["MEAN %MB" INDICATES AVERAGE NUMBER OF MACROBLOCKS TO WHICH THE MOTION VECTOR REFINEMENTS WERE APPLIED PER FRAME. "MEAN #CP" INDICATES AVERAGE NUMBER OF CHECKING POINTS PER MACROBLOCK NEEDED TO FIND THE MOTION VECTOR. ["SPEED-UP RATIO" = (MEAN #CP × MEAN %MB)/(100 × 961)] (a) OUTGOING FRAME-RATE: 30 frames/s, (b) OUTGOING FRAME-RATE: 15 frames/s, AND (c) OUTGOING FRAME-RATE: 10 frames/s

	FULL-ME	BASE-ONLY	FULL-MVR	BBGDS + ADAPTIVE + MVR	HAVS + ADAPTIVE + MVR
Mean PSNR (dB)	28.61	28.16	28.60	28.57	28.54
Mean % MB	100	0	100	73	73
Mean #CP	961	0	25	13	6
Speed-up ratio	1	0	0.026	0.0099	0.0046

(a)

	FULL-ME	BASE-ONLY	FULL-MVR	BBGDS + ADAPTIVE + MVR	HAVS + ADAPTIVE + MVR
Mean PSNR (dB)	29.65	28.91	29.61	29.54	29.56
Mean % MB	100	0	100	71	70
Mean #CP	961	0	25	13	6
Speed-up ratio	1	0	0.026	0.0096	0.0044

(b)

	FULL-ME	BASE-ONLY	FULL-MVR	BBGDS + ADAPTIVE + MVR	HAVS + ADAPTIVE + MVR
Mean PSNR (dB)	30.53	29.62	30.51	30.26	30.28
Mean % MB	100	0	100	68	67
Mean #CP	961	0	25	13	6
Speed-up ratio	1	0	0.026	0.0092	0.0042

(c)

TABLE III

PERFORMANCE OF THE PROPOSED SCHEMES. THE BIT-RATE AND FRAME-RATE OF INCOMING BIT-STREAM IS 128 Kb/s AND 30 frames/s. ($\text{THRESHOLD}_1 = 300$, $\text{THRESHOLD}_2 = 500$)

(Unit : dB)

Test sequence	Outgoing Frame rate		Outgoing bit-rates		
			64 kbps	32 kbps	16 kbps
Claire	30	FULL-ME	39.60	37.07	34.49
		BASE-ONLY	39.50	36.75	33.63
		HAVS+ADAP+MVR	39.55	37.06	34.37
		Mean % MB	12 %	25 %	31 %
	15	FULL-ME	40.41	38.13	35.66
		BASE-ONLY	40.16	37.62	34.82
		HAVS+ADAP+MVR	40.40	38.03	35.63
		Mean % MB	10 %	22 %	30 %
	10	FULL-ME	40.58	39.17	37.41
		BASE-ONLY	40.20	38.60	36.63
		HAVS+ADAP+MVR	40.57	39.04	37.27
		Mean % MB	9 %	23 %	29 %
Suzie	30	FULL-ME	34.17	32.20	30.62
		BASE-ONLY	34.15	31.78	29.91
		HAVS+ADAP+MVR	34.12	32.17	30.52
		Mean % MB	45 %	64 %	68 %
	15	FULL-ME	34.88	33.25	31.32
		BASE-ONLY	34.52	32.71	30.43
		HAVS+ADAP+MVR	34.84	33.17	31.28
		Mean % MB	35 %	57 %	63 %
	10	FULL-ME	35.25	33.79	31.88
		BASE-ONLY	34.81	33.06	30.96
		HAVS+ADAP+MVR	35.23	33.68	31.83
		Mean % MB	37 %	53 %	62 %

than that of the popular block-based gradient descent search (BBGDS) [20] which may require on average of about 13 checking points.

VII. Simulation Results

In all the simulations presented in this paper, test sequences of QCIF (176×144) were encoded at high bitrate using a fixed quantization parameter, or using the rate-control implemented in [24]. At the front-encoder, the first frame was encoded as an intraframe (I-frame), and the remaining frames were encoded as interframes (P-frames). These picture-coding modes were preserved during the transcoding. In our simulations, bidirectional predicted frames (B-frames) were not considered. However, the idea of the proposed scheme can also be applied to B-frames. When no frame was dropped, the macroblock coding-modes were reused. In the frame-rate conversion, the macroblock coding-modes were recomputed. A fixed search-window size of $\pm$ 2 pixels for the motion vector refinement and fixed-threshold values ($\text{Threshold}_1 = 300$, $\text{Threshold}_2 = 500$) for the adaptive scheme were used for all the simulations.

Fig. 13 shows the simulation results of different motion vector refinement schemes at different frame-rates. The performances of the FULL-MVR (nonadaptive motion vector refinement), BBGDS + ADAP + MVR (adaptive motion vector refinement with BBGDS), and HAVS + ADAP + MVR (adaptive motion vector refinement with HAVS) were compared.

Based on our simulations, HAVS + ADAP + MVR has about the same performance as BBGDS + ADAP + MVR, but requires less computation. Furthermore, its performance is similar to FULL-MVR, which refines all incoming motion vectors. In the case of the adaptive refinement scheme, more than 30% of the incoming motion vectors were reused. These computational savings are significant and demonstrate the effectiveness of the proposed adaptive refinement scheme. The performances in terms of quality and speed are summarized in Table II where FULL-ME represents the full-scale full-search motion estimation. The speed-up ratio represents the ratio of the required number of checking points for the motion vector refinement compared to that of full-scale full-search motion estimation. Simulation results on different test sequences are summarized in Table III.

VIII. Conclusion

In this paper, we have discussed motion vector refinement for high performance transcoding. We have shown that in many transcoding applications, reusing the incoming motion vectors may introduce considerable quality degradation. We have presented a mathematical analysis to show that the degradation is due to the reconstruction errors. Quality can be significantly improved by refining the incoming motion vectors as opposed to applying a new full-scale motion estimation to find the optimal motion vectors. Starting with the base motion vector, the motion vector refinement scheme searches for a delta motion vector within a search area much smaller than that of the full-scale motion estimation. This small search window is computationally much less complex.

In addition, we have proposed an adaptive motion vector refinement scheme. This scheme significantly reduces the number of incoming motion vectors needing refinement. To further reduce the computational complexity, we have proposed a fast-search algorithm which requires less checking points compared to other fast-search algorithms. We have showed through simulations that combining the motion vector refinement with the adaptive and the fast-search scheme produces almost the same performance as the full-scale motion vector refinement. We have extended the motion vector refinement scheme to the case when the frame-rate conversion is needed. We have proposed a forward dominant vector selection (FDVS) composition method to compose an outgoing motion vector from the incoming motion vectors of the dropped frames. We have shown that the FDVS method performs better and requires less computation and memory than the bilinear interpolation method. Through extensive simulations we have showed that the proposed fast-search adaptive motion vector refinement scheme can improve the video quality to the level achieved by using the full-scale motion estimation, with minimal computational complexity.

Acknowledgment

The authors would like to thank the reviewers for their valuable comments.

References

[1] M. Ghanbari, "Two-layer coding of video signals for VBR networks," *IEEE J. Select. Areas Commun.*, vol. 7, pp. 771–781, June 1989.
[2] J. Moura, R. Jasinschi, H.-H. Shiojiri, and C. Lin, "Scalable video coding over heterogeneous networks," in *Proc. SPIE—Int. Soc. Opt. Eng.*, 1996, vol. 2602, pp. 294–306.
[3] N. Chaddha, "A software only scalable video delivery system for multimedia applications over heterogeneous networks," in *Proc. Int. Conf. Image Processing*, Washington, DC, Oct. 1995.
[4] A. T. Campbell and G. Coulson, "QoS adaptive transports: Delivering scalable media to the desktop," *IEEE Network*, pp. 18–27, Mar./Apr. 1997.
[5] N. Yeadon, F. Garcia, D. Hutchison, and D. Shepherd, "Filters: QoS support mechanisms for mulipeer communications," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 1245–1262, Sept. 1996.
[6] B. Shen and I. K. Sethi, "Adaptive motion vector resampling for compressed video down-scaling," in *Proc. Int. Conf. Image Processing*, Oct. 1997.
[7] ITU-T, ISO/IEC 13818-2, "Information technology—Generic coding for moving pictures and associated audio information: Video," Nov. 1994.
[8] Image Processing Lab., University of British Columbia, TMN (H.263+) encoder/decoder, Nov. 1997. [Online]. Available http://www.ece.ubc.ca/imagelab/.
[9] A. Eleftheriadis and D. Anastassiou, "Constrained and general dynamic rate shaping of compressed digital video," in *Proc. IEEE Int. Conf. Image Processing*, Washington, DC, Oct. 1995.
[10] G. Keesman *et al.*, "Transcoding of MPEG bitstreams," *Signal Process. Image Comm.*, vol. 8, pp. 481–500, 1996.
[11] P. N. Tudor and O. H. Werner, "Real-time transcoding of MPEG-2 video bit streams," in *Proc. Int. Broadcasting Conv.*, Amsterdam, The Netherlands, Sept. 1997, pp. 286–301.
[12] P. Assuncao and M. Ghanbari, "Post-processing of MPEG2 coded video for transmission at lower bit rates," in *ICASSP'96*, May 1996, vol. 4, pp. 1998–2001.
[13] D. G. Morrison, M. E. Nilsson, and M. Ghanbari, "Reduction of the bit-rate of compressed video while in its coded form," in *Proc. Sixth Int. Workshop Packet Video*, Portland, OR, Sept. 1994.
[14] R. J. Safranek, C. Kalmanek, and R. Garg, "Methods for matching compressed video to ATM networks," in *Proc. Int. Conf. Image*, Washington, DC, Oct. 1995.

[15] H. Sun, W. Kwok, and J. W. Zdepski, "Architecture for MPEG compressed bitstream scaling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 191–199, Apr. 1996.
[16] J. Youn and M.-T. Sun, "Motion estimation for high performance transcoding," in *IEEE Int. Conf. Consumer Electronics*, Los Angeles, CA, June 1998.
[17] N. Bjork and C. Christopoulos, "Transcoder architecture for video coding," *IEEE Trans. Consumer Electron.*, vol. 44, pp. 88–98, Feb. 1998.
[18] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures.* Norwell, MA: Kluwer, 1995.
[19] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 438–442, Aug. 1994.
[20] L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 419–422, Aug. 1996.
[21] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COMM-29, pp. 1799–1806, Dec. 1981.
[22] B. Zeng, R. Li, and M. L. Liou, "Optimization of fast block motion estimation algorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 833–844, Dec. 1997.
[23] ITU-T Rec. H.263, "Video codec for low bit rate communication," May 1996.
[24] Telnor Codec, ITU-T/GS-15, "Video codec test model, TMN5," Telnor research, Jan. 95. [Online] Available http://www.nta.no/brukere/DVC/h263_software/.

Jeongnam Youn received the B.S. degree in electronic engineering from Hanyang University, Korea, in 1988 and the M.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Seoul, in 1990. Since 1996, he has been pursuing the Ph.D. degree in electrical engineering at the University of Washington, Seattle.

From 1990 to 1995, he was a member of the Technical Staff, Korea Telecom. His research interests are video coding, video transcoding, and networked multimedia applications.

Ming-Ting Sun (S'79–M'81–SM'89–F'96) received the B.S. degree from National Taiwan University, Taipei, Taiwan, R.O.C., in 1976, the M.S. degree from the University of Texas at Arlington in 1981, and the Ph.D. degree from the University of California, Los Angeles, in 1985, all in electrical engineering.

He joined the faculty at the University of Washington, Seattle, in September 1996. Before that, he was the Director of the Video Signal Processing Group at Bellcore. His research interests include video coding, multimedia networking, and VLSI architecture and implementation for real-time video signal processing. He has been awarded seven patents and has published more than 80 technical papers including several book chapters in the area of video technology. He developed new VLSI architectures for several critical functions in video compression. He was actively involved in the development of H.261, MPEG-1, and MPEG-2 video coding and systems standards.

Dr. Sun was the Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (T-CSVT) from 1995 to 1997, and the Express Letter Editor of T-CSVT from 1993 to 1994. He was a co-recipeint of the T-CSVT Best Paper Award in 1993. From 1988 to 1991, he served as the Chairman of the IEEE Circuits and Systems Society Standards Committee and established an IEEE Inverse Discrete Cosine Transform Standard. He received an Award of Excellence from Bellcore in 1987 for the work on the digital subscriber line.

Chia-Wen Lin received the M.S. degree in electrical engineering from National Tsing Hua University (NTHU), Hsinchu, Taiwan, R.O.C., in 1992. He is currently a Ph.D. degree candidate in electrical engineering at NTHU.

He joined the Computer and Communications Research Laboratories, Industrial Technology Research Institute (CCL/ITRI), Hsinchu, as a Design Engineer in 1992. He was promoted to Project Manager of SDH fiber optics loop access systems in 1997, and is currently a Section Manager leading the development of video customer premises equipment at CCL/ITRI. His research interests include video coding, multimedia technologies, and digital transmission systems design.