Heterogeneous Video Transcoding to Lower Spatio-Temporal Resolutions and Different Encoding Formats

Tamer Shanableh and Mohammed Ghanbari, *Senior Member, IEEE*

***Abstract*—In this work, transcoding of pre-encoded MPEG-1, 2 video into lower bit rates is realized through altering the coding algorithm into H.261/H.263 standards with lower spatio-temporal resolutions. For this heterogeneous transcoding, we extract and compose a set of candidate motion vectors, from the incoming bit stream, to comply with the encoding format of the output bit stream. For the spatial resolution reduction we generate one motion vector out of a set of input motion vectors operating on the higher spatial resolution image. Finally, for the temporal resolution reduction we compose new motion vectors from the dropped frames motion vectors. Throughout the paper, we discuss the impact of motion estimation refinement on the new motion vectors and show that for all cases a simple half-pixel refinement is sufficient for near-optimum results.**

***Index Terms*—Digital TV, image conversion, video compression, video signal processing.**

I. Introduction

THE forthcoming multimedia telecommunication services are expected to use pre-encoded video for storage and transmission. The heterogeneity of the present communication networks in addition to the user preference of quality, demands matching the bit rate of the video source to the channel constraints and characteristics.

In the past, various layered coding techniques have been devised for dynamic, yet limited, bit rate adaptation [1], [2]. In these coding techniques, an image sequence is encoded into various levels with varying degrees of importance. Although these techniques can provide a minimum quality of service, they can have several shortfalls, namely, the overall bit rate of a multilayer encoder can be much larger than a single layer one [3]. In the case of video on demand, the number of available layers can limit the user choice, or may not accommodate future networks, such as video over mobile networks at lower bit rates.

To resolve this problem, recently various video transcoding techniques have been developed to convert compressed bit streams into lower rates [4]–[10]. These are a cascade of video decoder and encoder, with the advantage that no new motion estimation is carried out. Since motion estimation, which comprises more than 60–70% of the encoding complexity, is no longer needed (see Table I), these transcoders can be made sufficiently fast, such that software based transcoding, with even today technology is quite feasible [4].

TABLE I
PERCENTAGE OF PROCESSING TIME REQUIRED TO CARRY OUT MOTION ESTIMATION AND MACROBLOCK DECISIONS IN AN MPEG-1 ENCODER

Category	Fast DCT		Brute-force DCT	
	Mobile	Claire	Mobile	Claire
P-frame ME	66.1%	68.4%	53.3%	56.1%
B-frame ME	58.2%	60.9%	46.2%	48.7%
P-frame MB-decisions	4.2%	3.3%	3.4%	2.9%
B-frame MB-decisions	9.6%	6.8%	5.7%	5.2%

So far, homogeneous transcoding of MPEG-2 to MPEG-2 [4], H.261 to H.261 [5], and H.263 to H.263 [6] have been investigated. However, there are requirements for heterogeneous transcoding, such that decoders of one type (e.g., H.263) might wish to receive video coded by another form (e.g., MPEG-2). This is becoming particularly important for transmitting video over low bandwidth channels or hostile environments such as the mobile networks and the Internet. The emergence of the forthcoming Universal Mobile Telecommunication System (UMTS) carrying video, voice, and data is a good example [11].

In this case, in contrast to homogeneous transcoding, picture type, picture resolution, directionality of motion vectors, and picture rate might all change. These impose a heavy processing burden on the operation of the transcoders. The main objective of this paper is to investigate these bottlenecks and propose solutions to alleviate the problems.

The organization of this paper is as follows. The heterogeneous video transcoder is introduced in Section II. In Section III we look at the extraction of motion vectors when picture-encoding format is changed. Section IV deals with the impact of spatial resolution reduction on the motion extraction. This impact on the temporal resolution reduction is presented in Section V. Section VI concludes the paper.

II. Heterogeneous Video Transcoder

A brute-force method for achieving video transcoding, is to fully decode the incoming bit stream and re-encoded it with a new type of encoder. For homogeneous transcoding, we had shown how the decoder and encoder loops could be combined

Manuscript received September 24, 1999; revised March 30, 2000. This work was supported by a research contract funded by the Engineering and Physical Sciences Research Council (EPSRC) of the U.K. The associate editor coordinating the review of this paper and approving it for publication was Dr. M. Reha Civanlar.

The authors are with the Department of Electronic Systems Engineering, University of Essex, Colchester CO4 3SQ, U.K. (e-mail: ghan@essex.ac.uk).

Publisher Item Identifier S 1520-9210(00)05315-3.

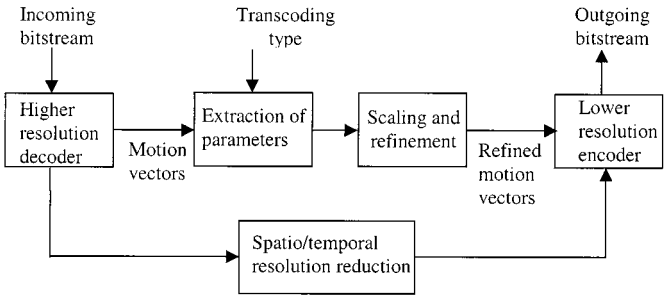


Fig. 1. A heterogeneous video transcoder.

to simplify the transcoder further [8]. However, in the heterogeneous transcoding, due to spatio-temporal sub-sampling, and different encoding structures of the input and output bit streams, these loops are no longer identical, and hence a generic heterogeneous transcoder might be realized, as the one shown in Fig. 1. In this figure, the incoming bitstream is fully decoded by a higher resolution decoder, compatible with the incoming bit stream. The extracted motion vectors are then post-processed according to the desired output encoding structure, and if required, they are properly scaled down to suit the lower spatio-temporal resolution encoder. In case post-processing is not sufficient, the extracted motion vectors are refined to improve the encoding efficiency. The decoded pictures are accordingly down-sampled spatially and or temporally, and the down-sampled images are encoded with the new motion vectors.

Note that, although it appears that two complete sets of decoder and encoder are used, but since the incoming motion vectors are re-employed, and other encoding decisions, such as macroblock types can be extracted from the incoming bit stream, the above decoder/encoder is much simpler than an off-the-shelf codec. Table I shows the portion of the processing time required by the full-search motion estimation and the macroblock decision in the MPEG Software Simulation Group (MSSG) MPEG-1 encoder [19], at 1.5 Mbit/s. The GOP size was $N = 12$ pictures and the picture sub-group size (distance between the anchor I/P pictures) was set to $M = 3$ pictures. A Pentium II processor was used to run the encoder and gather the statistics. The motion search size for P-pictures was set to 11×11 pixels, and for the B-pictures the search areas were 3×3 and 7×7 pixels for both forward and backward motion estimation, depending on their distances from the predicted anchor picture. Finally, since software based codecs might employ fast DCT, the measurements were also carried out on the brute-forced and fast DCT/IDCT parts.

As the table shows motion estimation, irrespective of the scene complexity, comprises about 66–68% of the processing power required to encode a P-picture with a software codec using fast DCT. This value for B-pictures, is slightly less, at 58–61%. B-pictures consume more processing in their macroblock decisions, which count about 7–10%. Overall, on the average macroblock decision plus the motion estimation comprise about 70% of all the processing power. Thus, if the motion scaling and refinement is not computationally intensive, then avoiding combined macroblock decision and motion estimation can speed up transcoding process by 100/(100-70), or roughly three times.

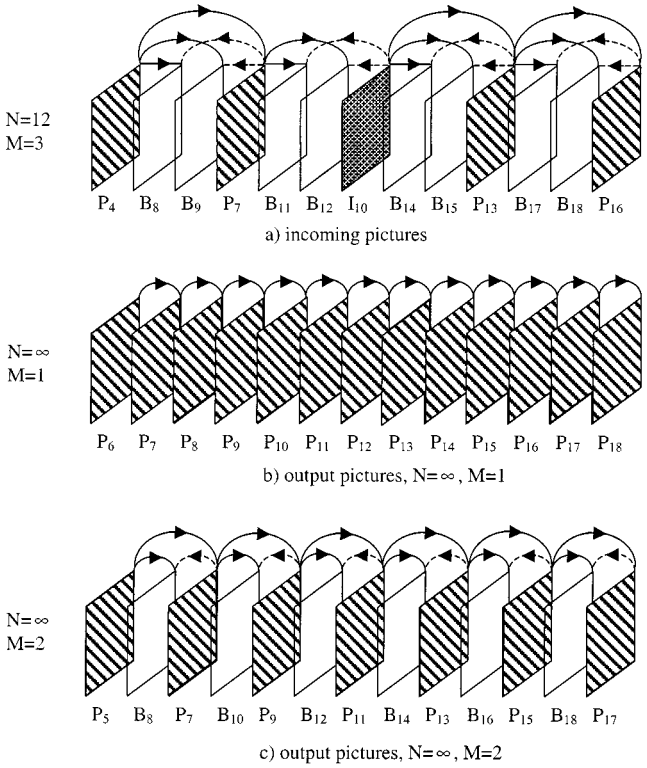


Fig. 2. Picture types of the input and output bit stream in the display order, but numbered in the encoding order.

III. Transcoding into a Different Coding Format

In a heterogeneous video transcoding, it is likely that the encoding format of a picture in the incoming bitstream is different from that of the outgoing bitstream, (e.g., transcoding of MPEG-2 into H.261 or H.263). For the standard codecs, a variety of format transcoding can exist. However to confine the discussion into a reasonable size, in the following we assume that the group of pictures (GOP) in the incoming bit stream has a length of 12 pictures $(N = 12)$ and the sub-group of three pictures $(M = 3)$ and the format of the output picture sequence is either that of H.261 or H.263, with the sequence structures of $(N = \infty, M = 1)$ and $(N = \infty, M = 2)$, respectively, (although H.263 is flexible to use limited N). We have considered these scenarios as the two most likely cases of the conversion, and the method can be easily generalized to other picture formats of the incoming and outgoing bit streams.

Since the main feature of the transcoding is to employ the motion vectors of the incoming bitstream in the outgoing one, then the extracted motion vectors have to be compatible with the encoding nature of the output bit stream. For example, the motion vectors of the incoming bit stream of our discussion $(N = 12, M = 3)$ vary from picture to picture, as shown in Fig. 2(a). I-pictures do not carry any motion vectors. The motion vectors in the P-pictures are referenced to the third preceding pictures, while those of B-pictures might use forward and backward predictions only from their anchor I/P-pictures. On the other hand, the motion vectors of the pictures in the output bit stream with the purely interframe coded pictures $(N = \infty, M = 1)$, require to be addressed to their immediate previous picture [Fig. 2(b)]. For $N = \infty$ and $M = 2$ of Fig. 2(c), every alternate picture

uses forward prediction (P-pictures) and the remaining ones use bi-directional predictions (B-pictures). Thus, the nature of extraction of the motion vectors, and their usage also depends on the picture type.

In Fig. 2, pictures are presented in the display order, but are numbered in the encoding order. Considering that the first input picture is I_0 and its second one is P_1, then due to reordering of pictures in the output stream, transcoding of every future anchor picture, such as P_1, has to be postponed after their corresponding bi-directional B-pictures (e.g., B_2, and B_3) are decoded. This introduces a transcoding delay, and its value is $M-1$ pictures. Also, due to this reordering, the encoding orders of the output P-pictures, become two pictures $(M-1=3-1)$ more than the encoding orders of their input anchor pictures, as shown in Fig. 2.

A. Picture Format Transcoding From $(N=12, M=3)$ Into $(N=\infty, M=1)$

For the start, let us look at the encoding format conversion of each picture type from $N=12$, $M=3$ into $N=\infty$, $M=1$, like MPEG-1 to H.261. As might be expected, in addition to the differences in picture type, not all the macroblocks of the input pictures carry appropriate forward motion vectors to be used directly at the output bit stream. In such cases, however, we can approximate them, by assuming the motion between the pictures is uniform, such that the forward and the reverse motion vectors are images of each other, or an interframe motion vector is a scaled version of a larger picture distance and so on. In case no motion vector is found, one might either use, a $(0,0)$ motion vector or at the worst case intraframe code the underlying macroblock. Finally, all the estimated motion vectors are compared, and the one that gives the least coding error in terms of sum of absolute differences (SAD) is chosen. Since the number of possible motion vector candidates is picture dependent, in the following we examine the most likely candidates for each input picture type.

For the first B-picture in the sub-group of an input bit stream, such as B_8, this picture is converted into an output P-picture P_7, with a prediction from P_6, as shown in Fig. 2(b). The new output motion vector $V^{\text{out}}_{6\to7}$ can be related to the input motion vectors with either of the following forms:

- to its forward motion vector, $V^{\text{fwd}}_{4\to8}$;
- its backward differential motion vector from its future anchor picture, $V^{\text{fwd}}_{4\to7}+V^{\text{bwd}}_{7\to8}$;
- one-third of its future anchor P-picture motion vector, $(1/3)V^{\text{fwd}}_{4\to7}$;
- image of the half of its backward motion vector, $-(1/2)V^{\text{bwd}}_{7\to8}$;
- half of its next B-picture forward motion vector, $(1/2)V^{\text{fwd}}_{4\to9}$.

The first two almost give the exact value of forward interframe motion vector. The last three assumes the motion is uniform, and the output motion vector is a properly scaled version of the available input motion vectors. To the above list, some poorer motion estimates, such as $-V^{\text{bwd}}_{7\to9}$, $-(V^{\text{bwd}}_{7\to8}-V^{\text{bwd}}_{7\to9})$ and others might be added. However, since some of these motion vectors may not be available, the more the candidates, the better is the estimate. To choose the best vector, the decoded picture is motion compensated with all the candidate motion vectors and the best one is chosen. In case no motion vector is available, $(0,0)$ motion vector or Intra coding is also tested. The number of comparisons can be limited to less than nine, which is equivalent to ±1 pixel search in motion estimation.

Similarly, for the second B-picture in the sub-group, such as B_9, one can find several motion candidates. For example, the motion vector of the output P-picture, P_8, corresponding to the second input B-picture in the sub-group (B_9) can be

- forward differential motion vector, $V^{\text{fwd}}_{4\to9}-V^{\text{fwd}}_{4\to8}$;
- image of its backward differential motion vector, $-(V^{\text{bwd}}_{7\to8}-V^{\text{bwd}}_{7\to9})$;
- image of its backward motion vector, $-V^{\text{bwd}}_{7\to9}$;
- one half of its differential motion vector with the future anchor P-picture, $(1/2)(V^{\text{fwd}}_{4\to7}+V^{\text{bwd}}_{7\to9})$;
- one half of its forward motion vector, $(1/2)V^{\text{fwd}}_{4\to9}$;
- image of half of its previous B-picture backward motion vector, $-(1/2)V^{\text{bwd}}_{7\to8}$
- one third of its future anchor P-picture motion vector, $(1/3)V^{\text{fwd}}_{4\to7}$

and some more. It should be noted that in the B-pictures preceding an I-picture (e.g., B_{11} and B_{12}), due to the absence of motion vector in the future anchor picture (I-picture), the number of candidate motion vectors is less than these two cases.

The candidate motion vectors in transcoding of an input P-picture, such as P_7, to the output P-picture P_9 can be

- image of the backward motion vector of the second B-picture in its sub-group, $-V^{\text{bwd}}_{7\to9}$;
- its forward differential motion vector with the second B-picture in the sub-group, $V^{\text{fwd}}_{4\to7}-V^{\text{fwd}}_{4\to9}$;
- one third of its forward motion vector, $(1/3)V^{\text{fwd}}_{4\to7}$;
- image of half of the backward motion vector of the first B-picture in the sub-group, $-(1/2)V^{\text{bwd}}_{7\to8}$;
- image of the first B-picture in the sub-group, offset by its output motion vector, $-V^{\text{bwd}}_{7\to8}-V^{\text{out}}_{7\to8}$.

In transcoding of an I-picture such as I_{10} to an output P-picture, P_{12}, due to the absence of forward motion vector in the picture, possible candidate motion vectors are less, unless one uses more crude estimates. For this picture, good candidates are:

- image of the backward motion vector of the second B-picture in the sub-group, $-V^{\text{bwd}}_{10\to12}$;
- image of half of the backward motion vector of the first B-picture in the sub-group, $-(1/2)V^{\text{bwd}}_{10\to11}$;
- image of the first B-picture in the sub-group, offset by its output motion vector, $-V^{\text{bwd}}_{10\to11}-V^{\text{out}}_{10\to11}$;

Fig. 3 shows the quality of transcoded video of the Flower sequence from 4 Mbit/s coded with MPEG-1 with the GOP structure of $N=12$, $M=3$ into purely P-pictures $(N=\infty, M=1)$ at 2 Mbit/s. For comparison re-encoded video (decoded at 4 Mbit/s and encoded again at a new bit rate with full motion search) with $N=\infty$ and $M=1$ at 2 Mbit/s, which uses ±15.5 pixels full motion search is also shown.

As Fig. 3 shows, the candidate motion vectors are very close to those derived by the full search method in the re-encoding of

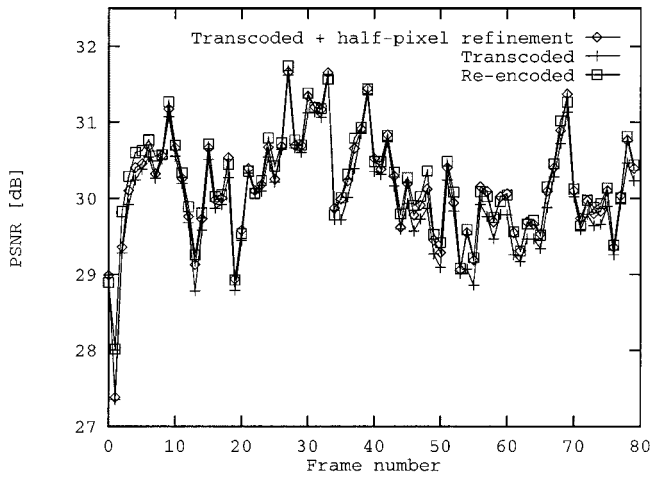


Fig. 3. Transcoding of 4 Mbit/s ($N = 12, M = 3$) video into 2 Mbit/s ($N = \infty$ and $M = 1$) of the Flower sequence.

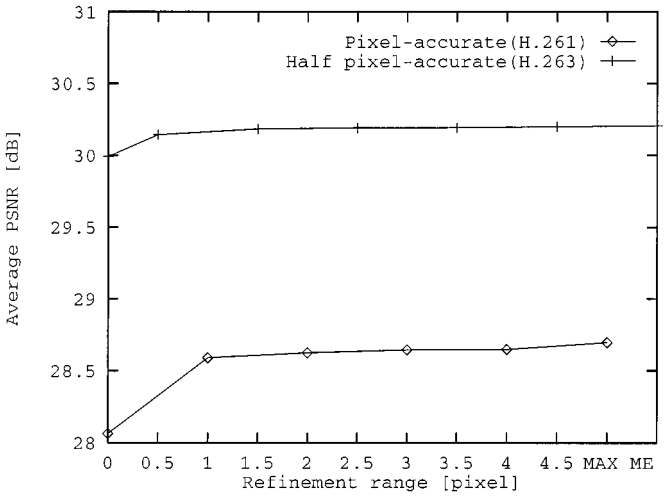


Fig. 4. Refinement of motion vectors with and without half pixel precision.

the decoded image. If the estimated motion vectors are refined to ± 0.5 pixel (± 0.5 pixel search around the best-derived motion vector), then its performance reaches to that of the re-encoding.

In order to evaluate the extent of required refinement, the search area around the best derived motion vector was increased from ± 0.5 pixel to a maximum of ± 15.5 pixel ($\mathrm{MAX\ ME} = 15.5$). Fig. 4 shows that refinement with ± 0.5 pixel is good enough to achieve an acceptable performance, and further refinement does not improve the motion compensation efficiency significantly. This indicates that the candidate motion vectors are very well chosen.

The figure also shows that when the estimated motion vectors are rounded to integer values (required in H.261), the motion compensation performance degrades. However, this is not much worse than re-encoding with the integer precision motion compensation (not shown in Fig. 3). Further refinement of the integer valued motion components, only improves the motion compensation by 0.6 dB. Again, significant improvement is noted with ± 1 pixel search, and larger refinement areas do not noticeably improve the motion compensation efficiency of the transcoder.

B. Picture Format Transcoding From $(N = 12, M = 3)$ Into $(N = \infty, M = 2)$

Transcoding the GOP structure into $N = \infty, M = 2$ can be done in a similar way to that of $N = \infty$, $M = 1$, described above. In this case, first the motion vectors between the output anchor P-pictures, which are two pictures apart, must be derived and then the bi-directional motion vectors of their neighboring B-pictures.

1) Derivation of the Motion Vectors for P-Pictures: Similar to the case of $M = 1$, depending on the encoding nature of the first-B, second-B, P and I input pictures within the sub-group of the input bit stream, derivation of the motion vectors for the output anchor-pictures can take different forms. For example, converting the input B_9-picture (second B-picture in the subgroup) of Fig. 2(a) into the output anchor P_7-picture of Fig. 2(c), the candidate motion vectors for $P^{\mathrm{out}}_{5\to 7}$ can be

- its forward motion vector, $V^{\mathrm{fwd}}_{4\to 9}$;
- differential motion vector of the future anchor P-picture and its backward motion vector, $V^{\mathrm{fwd}}_{4\to 7} + V^{\mathrm{bwd}}_{7\to 9}$;
- twice of its previous B-picture forward motion vector, $2V^{\mathrm{fwd}}_{4\to 8}$;
- two third of its future anchor p-picture motion vector, $(2/3)V^{\mathrm{fwd}}_{4\to 7}$;
- twice of the image of its backward motion vector, $-2V^{\mathrm{bwd}}_{7\to 9}$;
- image of the backward motion vector of its previous B-picture, $-V^{\mathrm{bwd}}_{7\to 8}$

and some others. Again, other input-picture types, might use different candidates. For example in converting input I_{10} of Fig. 2(a) into output P_{11} of Fig. 2(c), the possible candidate motion vectors for $P^{\mathrm{out}}_{9\to 11}$ can be

- image of the backward motion vector of B_{11}, $-V^{\mathrm{bwd}}_{10\to 11}$;
- twice the image of the motion vector of B_{12}, $-2V^{\mathrm{bwd}}_{10\to 12}$;
- forward motion vector of its second B-picture in the subgroup, $V^{\mathrm{fwd}}_{7\to 12}$;
- twice the forward motion vector of its first B-picture in the subgroup, $2V^{\mathrm{fwd}}_{7\to 11}$

and more. In a similar way, the candidate motion vectors of the first B- and P-input pictures can be derived.

2) Derivation of the Motion Vectors for B-Pictures: In general for $M = 2$, B-pictures can carry two types of motion information. They either have independent forward and or backward motion vectors, or their motion vectors are linked into their paired P-pictures, what is called PB-frames mode in H.263 [12].

In the case of independent motion vectors, the forward motion vectors are extracted similar to $M = 1$ of Section III-A, and similarly the backward motion vectors, but their directions are inverted. Then the best of these two or their combinations are chosen.

In the case of PB-frames mode, B-picture motion vectors are one half of their accompanying P-pictures plus a delta refinement. Hence, once the independent motion vectors of the B-pictures are known, their differences with the half of their accompanying P-picture motion vectors represent the delta refinement vector.

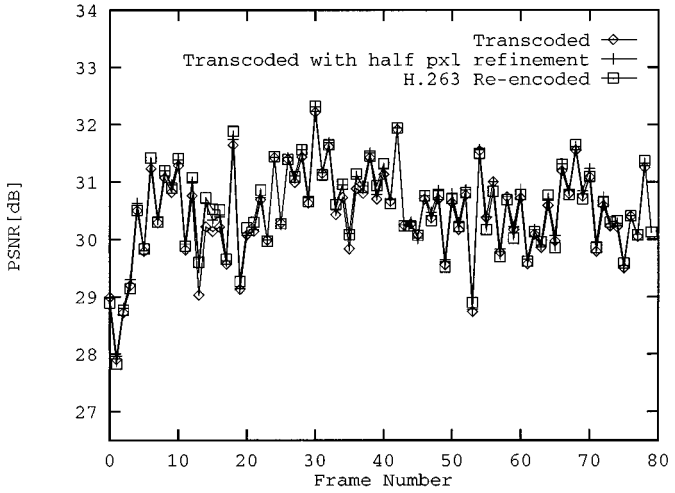


Fig. 5. Changing the picture format from $(N = 12, M = 3)$ video into $(N = \infty$ and $M = 2)$ PB-frames mode.

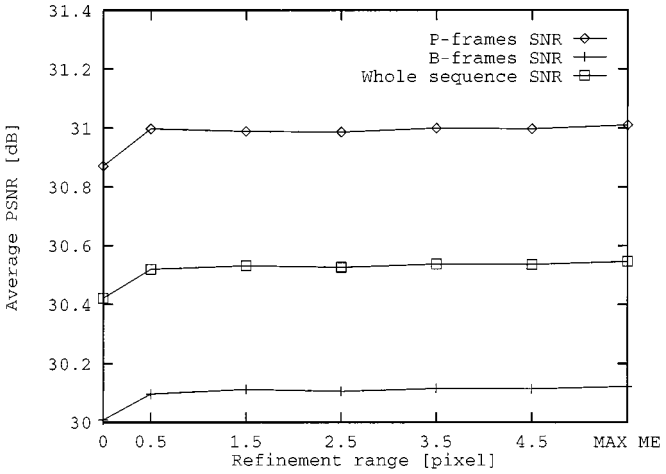


Fig. 6. Refinement of motion vectors for PB-frames mode.

To evaluate the accuracy of the candidate motion vectors for the P and B-pictures, the Flower sequence was transcoded from 4 Mbit/s with $N = 12$ and $M = 3$, into $N = \infty$ and $M = 2$ of PB-frames mode at 2 Mbit/s. Fig. 5 shows the performance of the estimated motion vectors for PB-frames mode. Again, it is seen that the candidate motion vectors are derived with reasonable accuracy, and the performance is very close to the re-encoding of the sequence with the full search within ± 15.5 pixel. Additional ± 0.5 refinement on the derived P and B pictures improves the performance to much closer to the re-encoding mode.

To find the extent of required refinement in the motion estimation, a search window from ± 0.5 pixel to the maximum of $(\mathrm{MAX\ ME} = \pm 15.5)$ pixel, was carried out around the best-derived motion vector. Fig. 6 shows the average quality of the transcoded image sequence with various refinement ranges, for individual P-, B-pictures and their combination. As the figure shows, for both picture types, ± 0.5 pixel refinement is sufficient, and larger search area is not needed. It should be noted that in this figure, some larger refinement appears to present lower quality, such as ± 2.5 pixel, compared to ± 0.5 pixel. However, this is not the case, and the inspection of the generated bits shows that at ± 2.5 pixel refinement 1 Kbit/s less data was generated than the refinement with ± 0.5 pixel. Had the encoder used the same bit rate, the performance would have been the least the same if not better. With a constant bit rate of a limited size test image sequence, it is hard to generate exactly the same bit rate for each mode. In addition, since generally B-pictures at the input bit stream are coded at poorer quality to P-pictures, this inferior quality is also preserved at the output stream.

IV. Transcoding into a Lower Spatial-Resolution

For transcoding into lower spatial resolution pictures, a new motion vector is to be calculated from a set of input motion vectors operating for a higher spatial resolution input sequence. For instance, transcoding a bit stream of SIF format into QSIF format requires calculating a new motion vector from four input motion vectors. This corresponds to transcoding four macroblocks into one macroblock.

In the H263 [12] standard, a macroblock can either have four 8×8 motion vectors or only one 16×16 motion vector. In the former, the same input motion vectors of the higher resolution can be employed directly for each 8×8 block, after the appropriate scaling, without any further processing. When one 16×16-motion vector per macroblock is used, a new motion vector from a set of four input motion vectors needs to be derived. The following section elaborates on the two scenarios.

A. One Motion Vector Per Macroblock

To generate one motion vector per macroblock of 16×16 pixels, we follow a two-step technique. First, a new motion vector is calculated using various heuristics depending on the four input motion vectors. Second, the calculated motion vector goes through a process of motion estimation refinement. However, similar to the transcoding of the encoding format of Section III, for the conversion to be efficient and worthwhile, the new motion vector should be calculated with sufficient accuracy, needing only minor refinement. In the following we discuss three methods of deriving a new motion vector from the four motion vectors available in the input bit stream information.

- Method One: the Median; let $V = \{v_1, v_2, v_3, v_4\}$ to represent the four adjacent motion vectors. The distance between each vector and the rest is calculated as the sum of their Euclidean distances as follows:

$$d_i = \sum_{\substack{j=1 \\ j \neq i}}^{4} \|v_i - v_j\|.$$

The median vector is defined as one of these vectors that has the least distance from all, i.e.

$$\mathrm{med}(V) = v_k \in V \quad \text{such that } \min d_i = d_k. \tag{1}$$

This method extracts the motion vector situated in the middle of the rest of the motion vectors. The magnitude of the selected motion vector is then scaled to reflect the reduction in the spatial resolution.

- Method Two: Moving with the majority; by calculating the average of those motion vectors that have the same direction, exploiting the high motion correlation between the neighboring macroblocks

$$V = \frac{1}{m}\sum_{i=1}^{m} V_i, \quad m \leq 4 \tag{2}$$

where m out of four motion vectors move at the same direction.
- Method Three: Calculating the average or mean of the input motion vectors, given by

$$\mathrm{V} = 1/4\sum_{i=1}^{4} V_i. \tag{3}$$

This method gives poor results especially, if the magnitude of one of the input motion vectors is significantly larger than the rest. A variety of other methods may also be used. For example, the weighted average of the incoming motion vectors, where each motion vector is weighted by the spatial activity of the perspective prediction error [13] or by selecting one of the incoming motion vectors in random [14]. Note that for all methods, the magnitude of the new motion vector is scaled down by half, to reflect the spatial resolution transcoding.

1) Motion Refinement: In contrast to the case of transcoding without reduction in the spatial resolution, where each input motion vector is mapped to an output one, in reducing picture sizes, the output motion vector has no resemblance to the input motion vectors, unless all the four input motion vectors are equal. Hence, the derived motion vectors inevitably need some refinement. We have tested the accuracy of the extracted motion vectors of the above three methods with converting the highly active moving sequence, Football, with SIF/25 Hz format encoded at 2 Mbit/s into QSIF/25 Hz at 0.7 Mbit/s, with $N = \infty$ and $M = 1$ (IPPPP...). For the refinement, in each case the derived motion vector is refined within a search area extending from ± 0.5 to a maximum of ± 15.5 pixels. Fig. 7 shows the quality of the motion compensated QSIF image, under the three schemes with various refinements. In this graph, the noncompressed SIF sequence was down-sampled to QSIF size to be used as the reference video for QSIF coded sequence.

Examining the figure, first, we see that the quality resulting from method-one "the median" proved the highest, followed by method-two "moving with the majority", while the lowest quality resulted from method-three "simple averaging."

Second, the candidate motion vectors of the median and majority methods are good enough to require only ± 0.5 pixel motion compensation refinement. On the other hand, the motion vector generated by averaging the incoming motion vectors requires larger refinements and hence is not a suitable method for motion extraction.

B. Four Motion Vectors Per Macroblock

To generate four motion vectors per macroblock we simply down-scale the incoming motion vectors by half and use them in the new macroblock. The down-scaling is performed whilst rounding the result to the nearest half-pixel resolution. Hence,

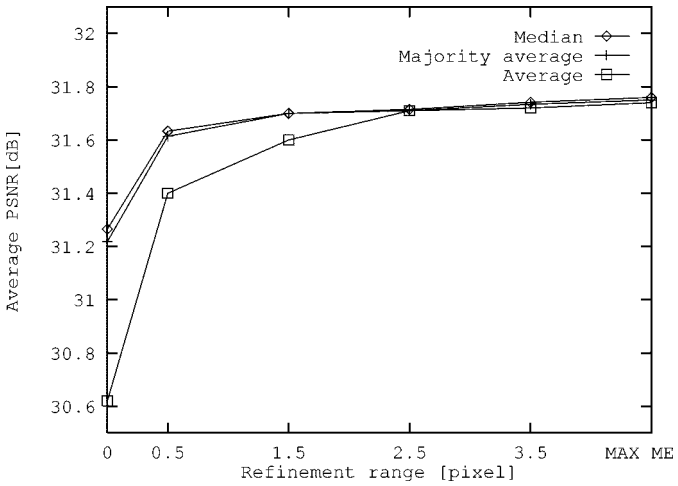


Fig. 7. Motion vector refinement from SIF to QSIF.

every 8×8 pixel block in the output bit stream uses the input motion vector of the input bit stream divided by two.

However, no matter whether one or four motion vectors are used at the input stream, the main difference between this mode of motion extraction with the one without spatial resolution reduction is that, here, a square of 2×2 macroblocks has to be transcoded into one 16×16 macroblock. Since the types of the input macroblocks might lack harmony, a new macroblock type has to be designated to the transcoded pictures in the output stream. This can be carried out in two ways. One is to derive a new macroblock type. The computational cost of carrying out new macroblock decision for each picture type is given in Table I. The other is to select a macroblock type from the four input macroblock types. For example, use the majority of the input macroblock type in the outgoing bit stream. However, in order to asses the performance of the estimated macroblock type, Fig. 8 shows the quality of the motion compensated Football sequence with various refinement ranges, when the macroblock types were derived fully by the macroblock decision rules at the encoder, or from the majority of the four input macroblock types. As can be seen, the difference is less than 0.2 dB, and the refinement of only 0.5 pixel reduces the difference to less than 0.1 dB.

C. Spatial Resolution Reduction

In transcoding with spatial resolution reduction, in addition to motion extraction, the spatial dimensions of the image should also be reduced (see Fig. 1). In the following, we examine three methods of spatial resolution reduction: pixel averaging and sub-sampling; filtering and sub-sampling and a DCT decimation method. We limit our discussion to 2:1 spatial resolution reduction, e.g., SIF to QSIF.

First, pixel averaging is the simplest method, where every 2×2 pixels are represented by a single pixel of their average value.

The second method used for down sampling is by employing a 7-tap filter with the following characteristics $(-1, 0, 9, 16, 9, 0, -1)/32$. This filter is used in both horizontal and vertical directions for luminance and chrominance, the

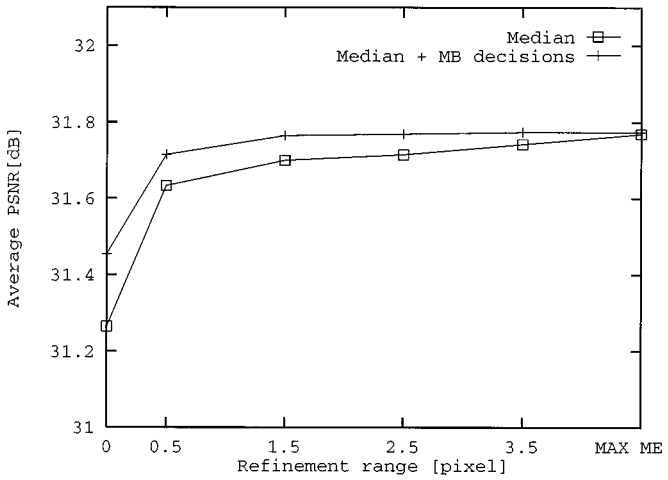


Fig. 8. Effect of new macroblock decision.

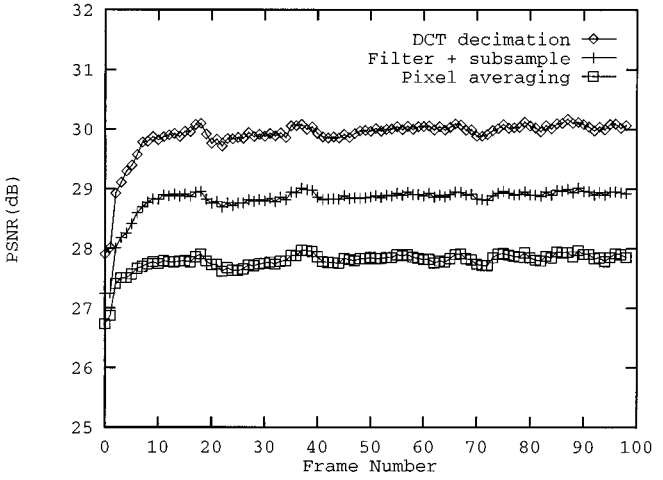


Fig. 9. SIF Salesman MPEG-encoded at 1.5 Mbit/s with $M = 12$, $N = 3$ and transcoded into H.263 P-frames with quarter of the spatial resolution/25 Hz at 0.6 Mbit/s.

image is then down-sampled by dropping every alternate pixel in the both horizontal and vertical directions [15].

In the third method, the DCT decimation, every four input blocks of 8×8 pixels, corresponding to an area of 16×16 pixels is first DCT transformed. The decimation is realized by retaining the top 4×4 coefficients of each block, and then inverse transforming by a $4 \times 4\ \mathrm{DCT}^{-1}$ to reconstruct 4×4 pixels [16]. Hence, the four blocks would become a new 8×8-pixel block. For this method not to produce visual artifacts, it should be noted that the normalizing term of the DCT transform pair for 8×8 and 4×4 pixels is 8 [16]. The attractive feature of this method is that, since in natural images, most of the energy is concentrated at the lower frequency band, then by retaining only lower 4×4 coefficients from 8×8, most of the energy of the original image is preserved.

In order to assess the quality of each method, considering that the resultant down-sampled images from the above methods have no corresponding original images with the same spatial resolution, for a fair comparison, one should up-sample and interpolate the resultant image to its original size and compare it with the original noncompressed one.

For the pixel averaging, up-sampling to the SIF size is realized through bilinear interpolation of the QSIF images i.e., substituting each missing pixel by the average of the neighboring pixels.

For the second method, up-sampling and interpolation of the filtered image is achieved by inserting zeros between every pixel of the QSIF image. The interpolation is first performed in the horizontal direction, and is then followed by the vertical direction to produce the SIF format. It should be noted that due to insertion of zeros, the filter must have a DC gain of two.

For the DCT interpolation, every 4×4 pixel block is first 4×4 DCT transformed. The resulting 4×4 coefficients are padded with zeroes at high frequencies to generate a block of 8×8 coefficients. Finally the 8×8 coefficients are inverse transformed with $8 \times 8\ \mathrm{DCT}^{-1}$ to generate interpolated 8×8 pixels [16].

The interpolated image can now be compared with the original noncompressed SIF image. Fig. 9 shows down-sampled and then up-sampled transcoded Salesman SIF/25 Hz test sequence. The original sequence was MPEG-1 encoded at 1.5 Mbit/s with $N = 12$, $M = 3$. It was then transcoded into H.263 P-pictures ($N = \infty$, $M = 1$, QSIF) with quarter of the spatial resolution/25 Hz at 0.6 Mbit/s, with either of the above spatial resolution reduction methods. Each resultant sequence was up-sampled and interpolated into the original size and compared with the original noncompressed images.

It can be seen that the results obtained through the DCT decimation are the highest. As expected, by retaining the most important information, one can deliver better quality images. The subjective quality of the images well correlate with the objective results of Fig. 9.

V. Transcoding into Lower Temporal Resolutions

For higher bit rate reductions, in case spatial resolution reduction is not sufficient to accommodate the bit rate in the available channel capacity, then the temporal resolution of the frames has to be reduced. This is done by dropping some of the encoded frames. Such a harsh bit rate reduction might arise, for instance, in the ATM available bit rate (ABR) service, where due to the bursty background traffic, the available bit rate to a source might be very limited. In ABR networks supporting video, the ATM switch may monitor the available bandwidth and instruct the sources not to generate more than their allocated channel rates [17]. The transcoder is then a useful tool to reduce the bit rate to the desired value [18]. Other situations demanding high compression ratios include video over low bandwidth networks with constant bit rate such as PSTN or mobile networks.

When the transcoder starts dropping frames, the incoming motion vectors of the remaining frames are no longer valid to be used at the output bit stream. One has to derive a new set of motion vectors, which takes into account the motion vectors of the dropped frames. Youn and Sun have devised a technique, called forward dominant vector selection (FDVS), which derives the motion vectors of the coded frames [7]. The best matched area pointed at by the motion vector of the current macroblock occurring after a dropped frame overlaps with at most four macroblocks in the previous dropped frame. The motion vector of

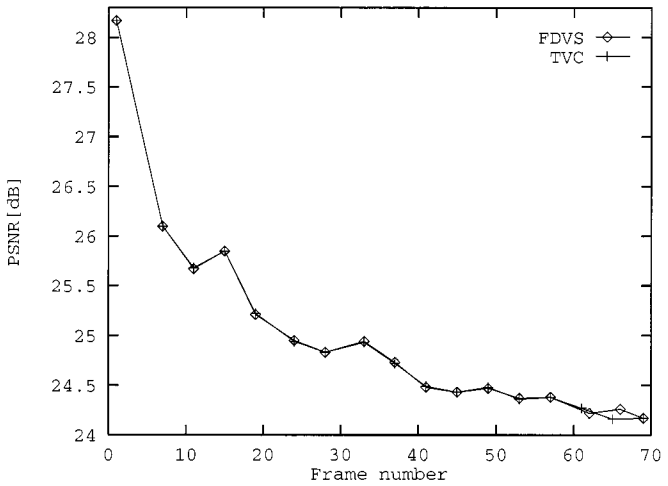


Fig. 10. Derived motion vectors for the dropped frames.

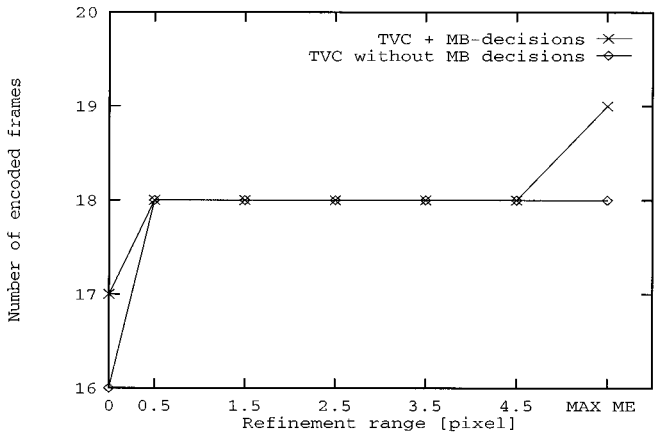


Fig. 11. Impact of motion refinement in transcoding 2 Mbit/s 25 Hz into 120 Kbit/s with the same spatial resolution, with and without new macroblock decision.

the macroblock with the largest overlapping portion is selected and added to the current motion vector. This process is repeated each time a frame is dropped until a new set of motion vectors is composed for the first encoded frame after the frame dropping. This technique also assumes a null motion vector for intra-coded macroblocks occurring in the dropped frames and emphasizes on the recalculation of the macroblock type after composing a new motion vector.

A simpler technique is, however, to accumulate all the motion vectors of the corresponding macroblocks of the dropped frames and add each resultant composed motion vector to its correspondence in the current frame. Hereafter we will refer to this technique as telescopic vector composition (TVC).

Since frame dropping, in the content of this work, is a further means of bit rate compression which is normally required after the spatial resolution reduction, it follows that all the incoming motion vectors have to be down scaled by half beforehand. The best matched area pointed at by the down-scaled motion vector will always overlap most with the corresponding macroblock location at the previous dropped frame which dictates close results for both TVC and FDVS techniques. Although the input motion vector of this work is restricted to the range of $[-16, 16]$ and therefore, the resultant range of the down-scaled motion vector is $[-8, 8]$, nevertheless, even without down scaling, Fig. 10 shows that there is not much difference between the two methods. In the figure, seventy frames of the SIF Football sequence, originally coded at 2 Mbit/s with $N = 12$, $M = 3$, was transcoded into 120 Kbit/s with $N = \infty$ and $M = 1$, without spatial resolution reduction (SIF to SIF). Due the constraint in the channel rate, in both methods only 17 out of 70 frames are transcoded and the remaining ones are dropped. These coded frames are identified in the figure by their legends. As the figure shows, difference between the two methods is very marginal. Hence, hereinafter for simplicity, we use telescopic type of motion extraction for the dropped frames.

However, similar to the other modes of motion extractions described in Sections III and IV, motion extraction with the dropped frames loses some precision, which might require refinement. In addition, due to the dropped frames, the distance between the current frame and previous anchor frame becomes larger, and the extracted macroblock type may not be suitable. Fig. 11 shows the impact of motion refinement on the telescopic type motion extraction with and without new macroblock type decision. In this figure, 2 Mbit/s Football sequence with SIF format was transcoded into120 Kbit/s with the same spatial resolution. As the figure shows without motion refinement and new macroblock decision, only 16 frames out the 70 frames are retained for coding at 120 Kbit/s. When new macroblock decision is carried out, one more frame can be coded at the same bit rate. Increasing the motion refinement search area by ± 0.5 pixels, both refinements generate the same number of frames. Moreover, higher refinement combined with new macroblock decision added another frame.

At these low frame rates, transcoded frames become very jerky. To reduce frame jerkiness, temporal resolution can be traded with the spatial resolution. Fig. 12 shows the impact of motion refinement and new macroblock decision on the number of coded frames at 70 Kbit/s with spatial resolution reduction. The channel rate was chosen to experience some frame dropping and to investigate the impact of macroblock decision and motion refinement on this matter. In this graph, the SIF Football sequence at 2 Mbit/s was transcoded into QSIF at 70 Kbit/s. As can be seen, new macroblock decision does not improve the coding efficiency significantly.

Fig. 13 shows the quality of transcoded video from SIF $(N = 12, M = 3)$ at 2 Mbit/s into SIF and QSIF (with the 7-tap filtering method) of $N = \infty$ and $M = 1$ at 100 Kbit/s. With SIF output, only 13 frames of the 70 input frames are coded for transmission, the rest are dropped. The picture at this rate is very jerky. On the other hand, with the QSIF transcoded video, 69 out of 70 frames are coded, and only the second frame after the first I-frame was dropped. Both sequences have low quality, since for the SIF output, in order to skip frames, the quantizer step size is at its maximum of 62. The chosen bit rate is just enough for the QSIF to be at the border of frame dropping, hence its quantizer step size is also at its maximum level of 62. However, the QSIF sequence is almost free from jerkiness, and due to lower spatial resolution, where the correlation between the pixels is less than that of SIF size, the quality is poorer by less than 1 dB.

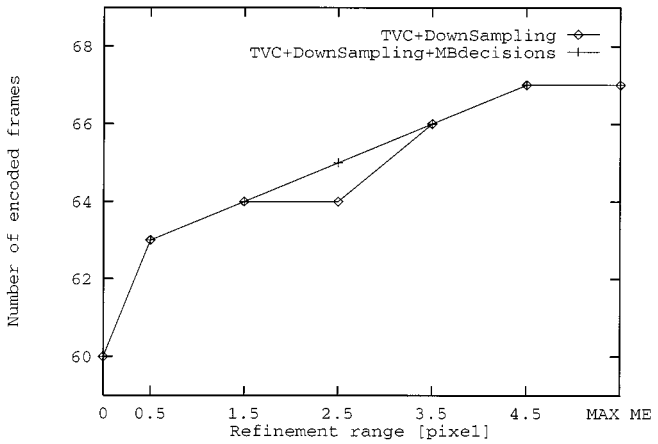


Fig. 12. Impact of motion refinement in transcoding 2 Mbit/s 25 Hz into 70 Kbit/s with quarter spatial resolution reduction.

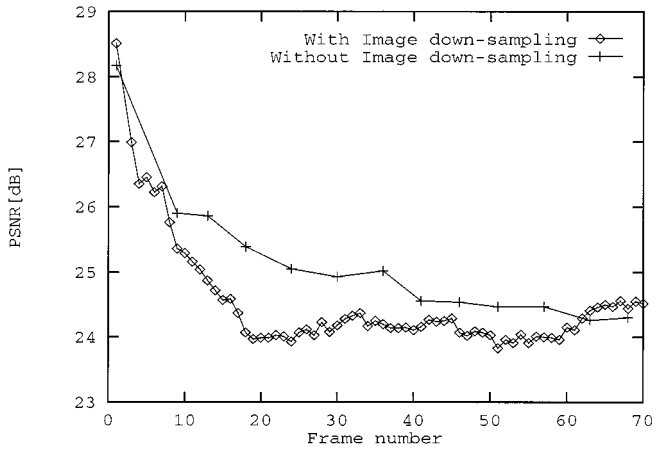


Fig. 13. FOOTBALL SIF sequence transcoded from 2 Mbit/s into SIF and QSIF size images at 100 kbit/s.

If the output channel rate is increased, say to 200 Kbit/s, at this rate SIF size output video still drops some frames. In this experiment 39 out of 70 frames were coded and the rest were dropped. Inevitably, those which remained, were encoded with the largest quantizer step size (otherwise there would not be frame dropping). With QSIF, all 70 frames were coded, but since at this time, the channel rate is sufficient, frames are coded at lower than the maximum quantizer step size and the quality is almost 2 dB better than the SIF size output video.

VI. Conclusion

In heterogeneous transcoding due to the spatio-temporal sub-sampling and different encoding format of the output sequence, encoder and decoder motion compensation loops differ and hence they cannot be combined into a single loop. Nevertheless, we have shown that the proposed transcoder architecture can speedup the processing time up to roughly three times by re-employing the incoming motion parameters. We have considered two scenarios for transcoding. First, transcoding from encoding format of $(N = 12, M = 3)$ into $(N = \infty, M = 1)$ e.g., MPEG-1, 2 into H.261/H.263. We have shown that the incoming motion parameters of a sub GOP of up to 3-frames can be manipulated to produce several candidate

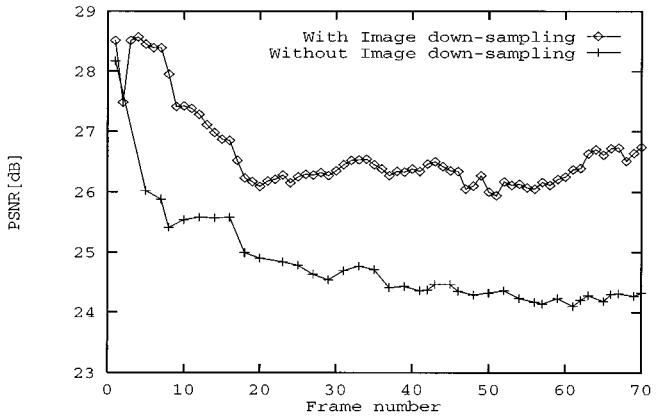


Fig. 14. FOOTBALL SIF sequence transcoded from 2 Mbit/s into SIF and QSIF size images at 200 Kbit/s.

motion vectors for the outgoing picture. The best motion vector was then refined by half-pixel (or one pixel in H.261) motion estimation to produce near-optimum results. Second, transcoding from encoding format of $(N = 12, M = 3)$ into $(N = \infty, M = 2)$ e.g., H.263 PB-frames. The new motion vectors of both P and B picture parts were calculated in a similar manner to that used in the first case. The new motion vectors of both picture parts were then shown to be half a pixel away from the optimum motion vector. For the spatial resolution reduction, combined with the transcoding of the encoded format, both median motion vector and average of the majority of the incoming motion vectors also proved to be nearly half a pixel away from the optimum motion vector. We have shown that the DCT decimation delivers better quality for image down-sampling over filtering/pixel-averaging and down-sampling. For the temporal resolution reduction, we have shown that disabling the spatial resolution reduction functionality of the transcoder produces poorer results in terms of picture quality and motion smoothness. Hence, lower bit rates require combined spatio-temporal resolution reductions. Finally, no macroblock decision is required, as the selected one with half a pixel refinement gives satisfactory results.

References

[1] M. Ghanbari, "Two-layer coding of video signals for VBR networks," *IEEE J. Select. Areas Commun.*, vol. SAC-7, pp. 771–781, 1987.

[2] G. Karlsson and M. Vetterli, "Packet video and its integration into the network architecture," *IEEE J. Select. Areas Commun.*, vol. SAC-7, pp. 739–751, 1987.

[3] M. Ghanbari and V. Seferridis, "Efficient H.261-based two-layer video codecs for ATM networks," *IEEE Trans. Ciruits Syst. Video Technol.*, vol. 5, pp. 171–175, Apr. 1995.

[4] P. A. A. Assuncao and M. Ghanbari, "A frequency-domain video transcoder for dynamic bitrate reduction of MPEG-2 bit streams," *IEEE Trans. Ciruits Syst. Video Technol.*, vol. 8, pp. 953–967, Dec. 1998.

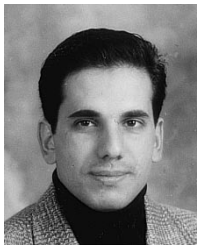
[5] D. G. Morrison, M. E. Nilsson, and M. Ghanbari, "Reduction of bit-rate of compressed video while in its coded form," in *Proc. 6th Int. Workshop on Packet Video*, Portland, OR, Sept. 1994, pp. 392–406.

[6] H. Sun, W. Kwok, and J. W. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Trans. Ciruits Syst. Video Technol.*, vol. 6, pp. 191–199, Apr. 1996.

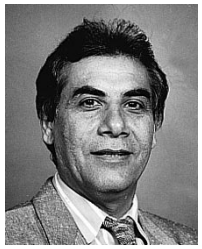
[7] J. Youn and M. Sun, "Motion vector refinement for high-performance transcoding," *IEEE Trans. Multimedia*, vol. 1, pp. 30–40, Mar. 1999.

[8] P. A. A. Assuncao and M. Ghanbari, "Transcoding of single-layer MPEG video into lower rates," *IEE Proc. Vision, Image, and Signal Processing*, vol. 144, no. 6, pp. 377–383, Dec. 1997.

[9] A. Eleftheriadis and D. Anastassiou, "Constrained and general dynamic rate shaping of compressed digital video," presented at the IEEE Int. Conf. Image Processing, Washington, DC, Oct. 1995.
[10] ——, "Meeting arbitrary QoS constraints using dynamic rate shaping of coded digital video," in *Proc. IEEE Int. Workshop on Network and Operating Systems for Digital Audio and Video*, Durham, NC, Apr. 1995, pp. 95–106.
[11] "A Regulatory Framework for UMTS,", UMTS Forum, June 1997.
[12] "Video Coding for Low Bit-Rate Communication,", Draft ITU-T Recommendation H.263, July 1995.
[13] B. Shen, I. K. Ishwar, and V. Bhaskaran, "Adaptive motion-vector resampling for compressed video downscaling," *IEEE Trans. Ciruits Syst. Video Technol.*, vol. 9, pp. 929–936, Sept. 1999.
[14] N. Bjork and C. Christopoulos, "Transcoder architecture for video coding," *IEEE Trans. Consumer Electron.*, vol. 44, pp. 88–98, Feb. 1998.
[15] *Video Codec Test Model, TM5*, Jan. 31, 1995. ITU Telecommunication Standardization Sector LBC-95, Study Group 15, Working Party 15/1.
[16] K. H. Tan and M. Ganbari, "Layered image coding using the DCT pyramid," *IEEE Trans. Image Processing*, vol. 4, pp. 512–516, Apr. 1995.
[17] R. Jain, S. Kalyanaraman, S. Fahmy, and R. Viswanathan, "The ERICA switch algorithm for ABR traffic management in ATM networks, part I: Description," *IEEE/ACM Trans. Networking*, Jan. 1997.
[18] E. Rosdinana and M. Ghanbari, "Subjective determination of MCR in ABR video," presented at the 9th Int. Workshop on Packet Video, New York, Apr. 1999.
[19] [Online]. Available: http://www.mpeg.org/MPEG/MSSG/



Tamer Shanableh was born in Scotland, U.K., in 1972. He received the B.Sc. degree in computer science in 1994 from Mu'tah University, Jordan, where he then worked as a computer programmer for a software house. In 1997, he received M.Sc. in software engineering (with distinction) from the University of Essex, U.K., where he is currently a Senior Research Officer and a part-time Ph.D. student in the Electronic Engineering Department. His research interests include video coding and transcoding, video networking, and error concealment.



Mohammed Ghanbari (M'78–SM'97) received the B.Sc. degree in electrical engineering from Aryamehr University of Technology, Tehran, Iran, in 1970, and the M.Sc. degree in telecommunications and the Ph.D. degree in electronics engineering, both from the University of Essex, U.K., in 1976 and 1979, respectively.

After working almost ten years in industry, he started his academic career as a Lecturer at the Department of Electronic Systems Engineering, University of Essex, in 1988, and was promoted to Senior Lecturer, Reader, and then Professor in 1993, 1995, and 1996, respectively. His research interests are video compression and video networking and he has published more than 220 papers in these fields. He is best known for his pioneering work on two-layer video coding for ATM networks. He is the author of the book *Video Coding: An Introduction to Standard Codecs*, (London, U.K.: IEE, 1999).

Dr. Ghanbari is the co-recipient of the 1995 A. H. Reeves premium prize for the year's best paper published in the *IEE Proceedings* on the theme of digital coding. He has been a member of the organizing committee of several international workshops and conferences. He was the chairman of the steering committee of the 1997 International Workshop on Packet Video, and currently is an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA.