6.3

Digital Video Transcoding

Shizhong Liu
Qualcomm Incorporated, San Diego

Alan C. Bovik
The University of Texas at Austin

1 Introduction

The rapid advent of digital multimedia technologies and the phenomenal growth of the Internet has created numerous networked multimedia applications such as Video on Demand (VoD), WebTV and video conferencing. The diversity of digital video content available on the Internet has grown explosively in the past few years. To conserve the storage space and communication bandwidth, most digital video content is represented in some kind of compressed format. There are many varieties of video formats which complicates the communication of video information between provider and viewer.

Generally, the following three conditions must be satisfied in order for a client to view a video over the Internet:

- The communication channel between the client device and the service provider must have sufficient bandwidth to support the bit rate of the compressed video.
- The client device must be able to decode the received compressed video bit stream.
- The client device must have the capability to display the decoded video.

It has become increasingly challenging to meet these conditions due to the diversity and heterogeneity of the Internet, both in terms of client device and network connection bandwidth. On the client side, new classes of pervasive computing devices such as personal digital assistants (PDAs), third generation (3G) mobile phones, and wearable computers are available that allow users to access multimedia information over the Internet. Different devices usually have different characteristics in terms of display capability, processing horsepower, storage capacity, and network access. For instance, a 3G mobile phone often has a smaller display screen, memory size, and limited processing power, compared with a desktop or laptop computer. Network connections are also highly diverse, ranging from several kilobits per second up to gigabits per second. Moreover, the bandwidth of a given connection may undergo dynamic changes over time. Consequently, to support universal multimedia access, the compressed video bit stream has to be manipulated dynamically to adapt to various client devices and network connections.

Various scalable video coding schemes have been proposed to support video communication over heterogeneous networks. The main idea of scalable video coding is to encode the video as one base layer and a few enhancement layers so that lower bit rates, spatial resolutions, and/or temporal resolutions could be obtained by simply truncating certain layers from the original bit stream. Several existing video coding standards such as MPEG-2/4, H.263 support scalable video coding. More details on scalable video coding can be found in Chapters 6.2, 6.4, 6.5, and the references therein. One problem of scalable video coding is that it does not support interoperability between different video coding formats. Consequently, a device that can only decode a

single video coding format may not be able to access those videos coded in other formats. In addition, scalable video coding usually suffers from lower coding efficiency, compared with single-layer video coding solutions. Since large numbers of video streams coded without scalability are available, such as digital video devices (DVDs), alternative technologies are needed to support universal access to those videos.

Video transcoding, where video is converted from one compressed format to another compressed format for adaptation of channel bandwidth or receiver or both, is another technique proposed to adaptively deliver video streams across heterogeneous networks. In general, video transcoding can be used for bit rate reduction, spatial resolution and/or temporal resolution reduction as well as for video coding format conversion. Quite a bit of research has been done on video transcoding due to its wide range of applications. Early research work on video transcoding mainly focused on bit rate reduction to adapt the compressed video to available channel bandwidth [1–4]. Owing to the coexistence of multiple video coding standards as well as the emergence of new classes of devices with limited display and processing power, heterogeneous video transcoding, where the video coding format, spatial resolution, and/or temporal resolution might all be changed, has also been investigated recently [5, 6]. Lately, error-resilience video transcoding has gained a significant amount of attention with the introduction of wireless video applications over various wireless networks [7, 8]. Figure 1 illustrates these common video transcoding operations.

In all video transcoding applications, it is possible to use a cascaded pixel-domain approach that decodes the incoming video stream, performs certain processing (if any), and reencodes the processed signal subject to any new constraints. However, this straightforward approach has a rather high computational complexity and the quest for more efficient techniques is the major driving force behind most video transcoding research activities. Certainly, any gains in efficiency should have negligible impact on the visual quality of the transcoded video. Since most current video coding standards are block-based video coding schemes that use the discrete cosine transform (DCT) and motion estimation and compensation (more details can be found in Chapter 6.1), we will concentrate on block-based video transcoding techniques in this chapter. Considering that the chrominance components can usually be handled similarly as the luminance components, we will only discuss the processing techniques done on the luminance components of the video.

The rest of this chapter is organized as follows. In Section 2, video transcoding techniques for bit rate reduction and corresponding architectures are reviewed. Section 3 reviews progress made in heterogeneous video transcoding, including spatial resolution reduction, motion vector reestimation after spatial and/or temporal resolution reduction, and macroblock coding-type decision. In Section 4, we discuss bit rate control techniques in video transcoding. Section 5 briefly discusses recent research activities on error resilient video transcoding for wireless video. Finally, we conclude the chapter in Section 6.

2 Video Transcoding for Bit Rate Reduction

Video transcoding for bit rate reduction has been studied intensively due to its wide range of applications including television broadcast and video streaming over the Internet. The most straightforward way to achieve this is to decode the incoming video bit stream and fully reencode the reconstructed video at the new bit rate as illustrated in Fig. 2. While this approach can achieve the best video transcoding results,

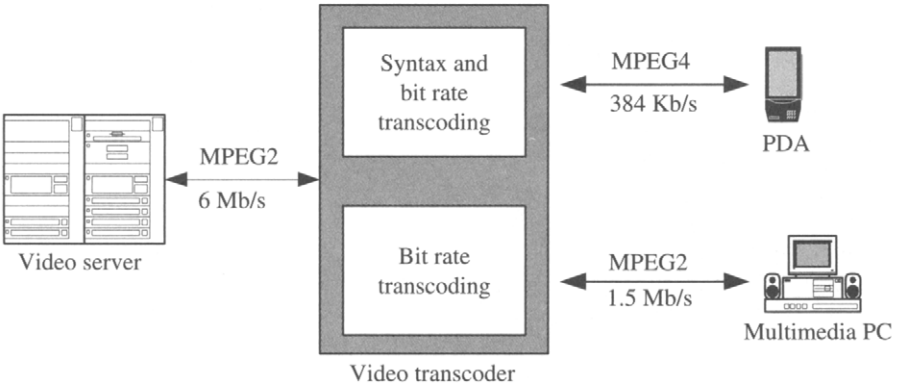

FIGURE 1 Illustration of common video transcoding applications. The source video is encoded at 6 Mb/s in MPEG-2 format. In one application, the source video is transcoded to another MPEG-2 bit stream with a lower bit rate of 1.5 Mb/s. In the other application, the source video is transcoded to a MPEG-4 bit stream with lower bit rate, spatial resolution, and/or temporal resolution due to the limited wireless channel bandwidth and the PDAs processing and display capabilities. Error resilient video transcoding may also be necessary in the second application to account for the high bit error rate of the wireless channel.

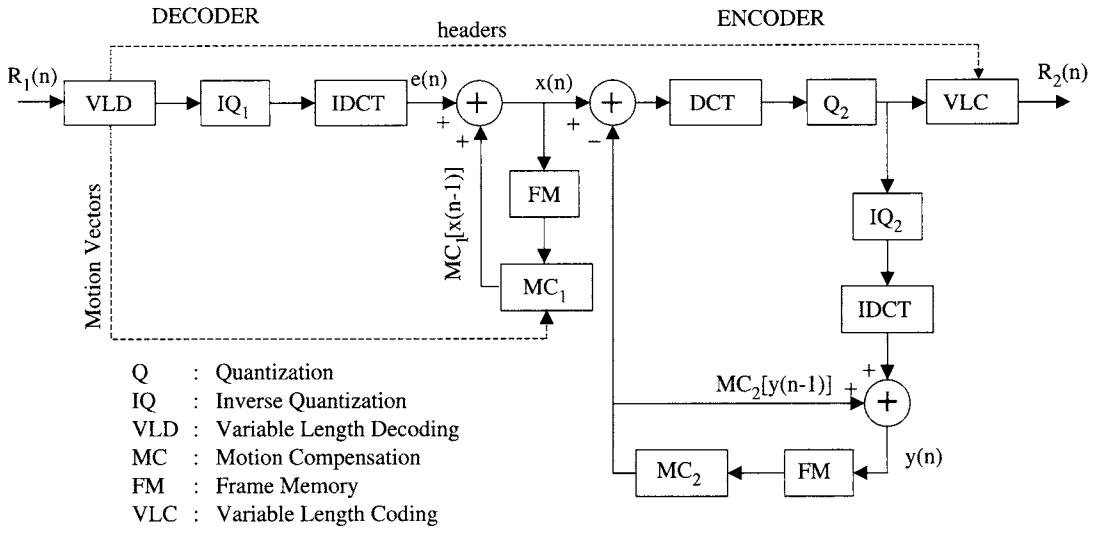

FIGURE 2 Diagram of the cascaded pixel domain video transcoder. $R_1(n)$ is the incoming video bit stream and $R_2(n)$ is the output stream with lower bit rate.

its high computational complexity prevents it from being a practical solution. It is the video transcoder's aim to significantly reduce the complexity, while still maintaining acceptable quality, by exploiting the information contained in the incoming video bit stream such as motion vectors.

In the following, we first analyze the video transcoding of both intracoded frames and intercoded frames based on Fig. 2. In the analysis, we assume that the picture coding type does not change after transcoding. For instance, an intracoded frame will be transcoded into another intracoded frame. Then, we review and evaluate several fast video transcoding architectures proposed over the past few years for bit rate reduction. Finally, we will briefly discuss the recent advances in DCT-domain inverse-motion compensation used in DCT domain video transcoding architectures.

2.1 Transcoding of Intra-Coded Frame

Intracoded pictures are transcoded by coarsely encoding the decoded picture denoted by $x(n)$. Thus, from Fig. 2, we have

$$R_2(n) = Q_2[DCT(x(n))], \tag{1}$$

where $x(n)$ is given by

$$x(n) = IDCT[Q_1^{-1}(R_1(n))] \tag{2}$$

Substituting (2) to (1) gives the transcoding equation (3) for intracoded frames:

$$R_2(n) = Q_2[DCT[IDCT[Q_1^{-1}(R_1(n))]]] = Q_2[Q_1^{-1}(R_1(n))] \tag{3}$$

The transcoding distortion for intracoded frame consists of the following three components [3]:

1. Some nonzero AC coefficients of the input frame become zero after coarse requantization.
2. The quantization error for those nonzero coefficients.
3. The requantization error.

While the former two are well-known causes of distortions, the third one is not obvious. In certain cases, requantization can lead to an additional error, which would not be introduced had the original DCT coefficients been quantized with the same coarser quantization step size. This is illustrated in Fig. 3, which shows how requantization can lead to higher

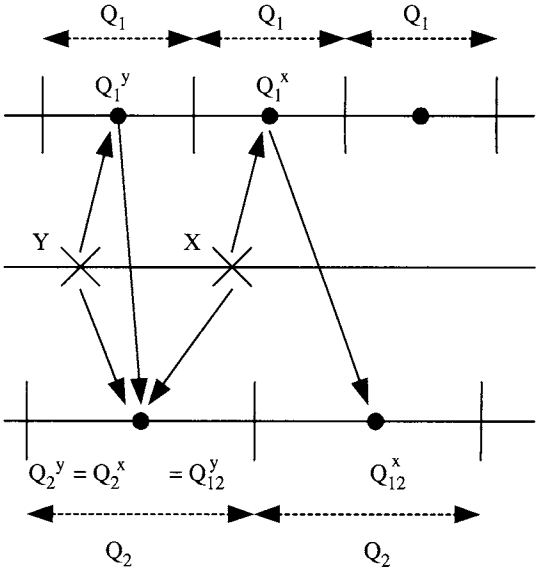

FIGURE 3 Requantization of DCT coefficients (adapted from [3]).

distortion than that produced by quantizing the original DCT coefficients using the same quantization step size.

In Fig. 3, the reconstructed levels of the DCT coefficients X, Y, quantized with a quantizer step size of Q_1, are ${Q_1}^X$ and ${Q_1}^Y$, respectively. If the coarser quantization step size Q_2 were used, both would be reconstructed to the same level ${Q_2}^X = {Q_2}^Y$. However, if X, Y are first quantized with Q_1 and then quantized with Q_2, the reconstructed level of Y will be the same as that of direct coarser quantization with Q_2 (*i.e.*, ${Q_2}^Y = {Q_{12}}^Y$), whereas in the case of X, the reconstruction value will be different from that of direct coarser quantization (*i.e.*, ${Q_2}^X \neq {Q_{12}}^X$). Therefore, the requantization error of Y is zero, whereas that of X is not.

Whenever the coarse quantization interval contains entirely the corresponding finer one, the direct coarse quantization and requantization distortions are equal. On the other hand, if the finer interval overlaps between two different coarser intervals, then the requantization distortion is larger in the case where the reconstruction value of the first quantization and the original coefficient fall into different coarser quantization intervals. Werner [9] analyzed the requantization problem theoretically and designed the quantizer Q_2 to reduce the overall requantization distortion by up to 1.3 db, compared with the quantizer used in the MPEG-2 reference coder TM5. In [10], a selective requantization method was proposed to reduce the requantization errors in video transcoding by avoiding critical ratios of the two cascaded quantizations that either lead to larger transcoding errors or require a higher bit budget.

2.2. Transcoding of Inter-Coded Frame

In Fig. 2, the incoming intercoded frame is first reconstructed at the decoder by motion compensation, for example,

$$x(n) = IDCT[Q_1^{-1}(R_1(n))] + MC_1[x(n-1)]. \tag{4}$$

At the encoder side, we have

$$R_2(n) = Q_2[DCT[x(n) - MC_2[y(n-1)]]] \tag{5}$$

Substituting (4) into (5) gives the transcoding equation (6) for inter-coded frames:

$$\begin{aligned} R_2(n) &= Q_2[DCT[IDCT[Q_1^{-1}(R_1(n))] + MC_1[x(n-1)] \\ &\quad - MC_2[y(n-1)]]] \\ &= Q_2[Q_1^{-1}(R_1(n) + DCT[MC_1(x(n-1)) \\ &\quad - MC_2(y(n-1))]]. \end{aligned} \tag{6}$$

Note that (6) implies that, for transcoding an intercoded frame, the transcoding error of the previous anchor picture has to be added to the incoming DCT coefficients and then coarsely quantized. Otherwise, the transcoding error in the intercoded frames will be accumulated until the next intracoded frame is met. This error accumulation is known as *drift*. In MPEG bit streams, intracoded frames can terminate the *drift* within a group of picture (GOP) structure. However, this refreshing by intracoded frames is not always available in other video streams. For instance, in H.263, the insertion of intracoded frames is not specified explicitly. Furthermore, in relatively complicated video sequences, the *drift* can be significant even with a small number of intercoded frames involved. In the following, various fast video transcoding architectures will be discussed and the *drift* in each transcoder will be investigated.

2.3 Fast Video Transcoding Architectures

Several different architectures for video transcoding have been proposed. A simple architecture uses open-loop transcoding as shown in Fig. 4, where the incoming bit rate is downscaled by truncating the high-frequency DCT coefficients or performing a requantization process [1].

Since the transcoding is done in the coded domain, a very simple and fast transcoder is possible. However, since the transcoding error associated with the anchor picture is not added to the subsequent intercoded frames, the open-loop transcoding produces an increasing distortion caused by the *drift*, which can degrade the video quality dramatically.

Drift-free transcoding is made possible by using a decoder to decode the incoming video and the using an encoder to reencode the decoded signal at a lower bit rate as shown

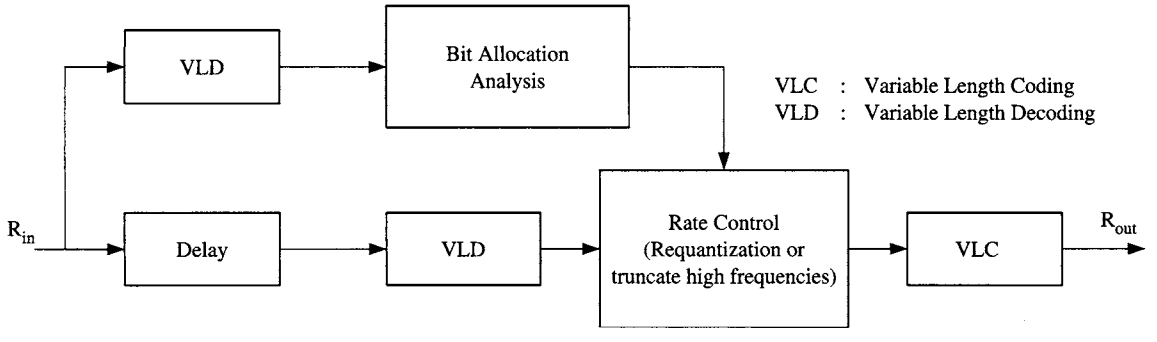

FIGURE 4 Fast pixel domain open-loop transcoder.

in Fig. 2. However, its high computational complexity makes it difficult to be used in real-time applications. Since an incoming video bit stream arriving at the transcoder already carries much useful information such as the picture type, motion vectors, quantization step size, bit-allocation statistics, and so forth, it is possible to reduce the complexity of the video transcoder by reusing some of the available information. By reusing the motion vectors and macroblock coding mode decision information contained in the incoming video bit stream, a simplified architecture with less computational cost can be obtained [2–4]. By reusing motion vectors (i.e., $MC_1 = MC_2 = MC$) equation (6) becomes

$$\begin{aligned} R_2(n) &= Q_2[Q_1^{-1}(R_1(n)) + DCT[MC_1(x(n-1)) \\ &\quad - MC_2(y(n-1))]] \\ &= Q_2[Q_1^{-1}(R_1(n)) + DCT[MC(x(n-1) - y(n-1))]], \end{aligned} \tag{7}$$

which leads to a simplified architecture depicted in Fig. 5. In this architecture, the transcoding error associated with the reference picture is compensated back to the predictive picture so that the error drift problem is eliminated. This operation is call *inverse motion compensation* as shown in Fig. 5.

This simplified architecture can save motion estimation, one frame memory and one IDCT operation compared to that shown in Fig. 2. The performance of different video transcoding architectures is depicted in Fig. 6 in which the *mobile* sequence was encoded using H.263 with the fixed quantization step size of $Q_1 = 5$. Figure 6 shows the PSNR (peak signal-to-noise ratio) value of each frame, obtained by the following three methods, respectively.

1. Encode the original mobile sequence using H.263 with fixed quantization step size of $Q_1 = 10$.
2. Transcode the bit stream with $Q_1 = 5$ into another one with $Q_2 = 10$, using the open-loop transcoder shown in Fig. 4.
3. Transcode the bit stream with $Q_1 = 5$ into another one with $Q_2 = 10$, using the drift free fast pixel domain transcoder shown in Fig. 5.

Clearly, the transcoding errors are accumulated in the open-loop video transcoder, causing a serious video-quality drift problem with more than 6-db PSNR degradation in the last frame. By contrast, the difference between that obtained by method 1 and that by method 3 is about 0.7 db for all frames.

In the fast pixel domain transcoder shown in Fig. 5, the coded quantization errors of the second-stage quantization are decoded into the pixel domain through inverse DCT transformation and then stored in the frame memory as pixel values. The motion compensation operation is performed in the pixel domain and then transformed into the DCT domain for the prediction operation. The whole process required one IDCT transform, one DCT transform, and one motion compensation. By keeping everything in the DCT domain, the computation for IDCT/DCT can be avoided, which leads to the DCT domain transcoding architecture shown in Fig. 7. In addition, for most video sequences, the DCT block is quite sparse hence the data volume to be processed can be significantly reduced while achieving the same functionality. The efficiency of this transcoder mainly depends on the computational complexity of DCT domain inverse motion compensation. In the next subsection, we will discuss the problem of DCT domain inverse motion compensation and briefly review several fast algorithms proposed in the last few years.

2.4 DCT Domain Inverse Motion Compensation

Inverse motion compensation is a necessary step in most video transcoding applications in order to convert the intercoded frames to intracoded frames or to compensate the transcoding errors in the anchor frames back to the prediction error frames for *drift-free* video transcoding. Since the data are organized

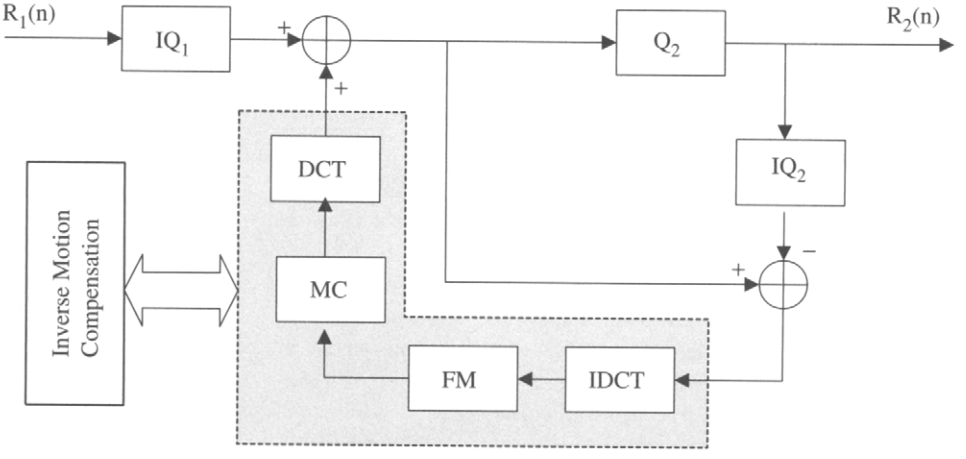

FIGURE 5 Fast pixel domain video transcoder.

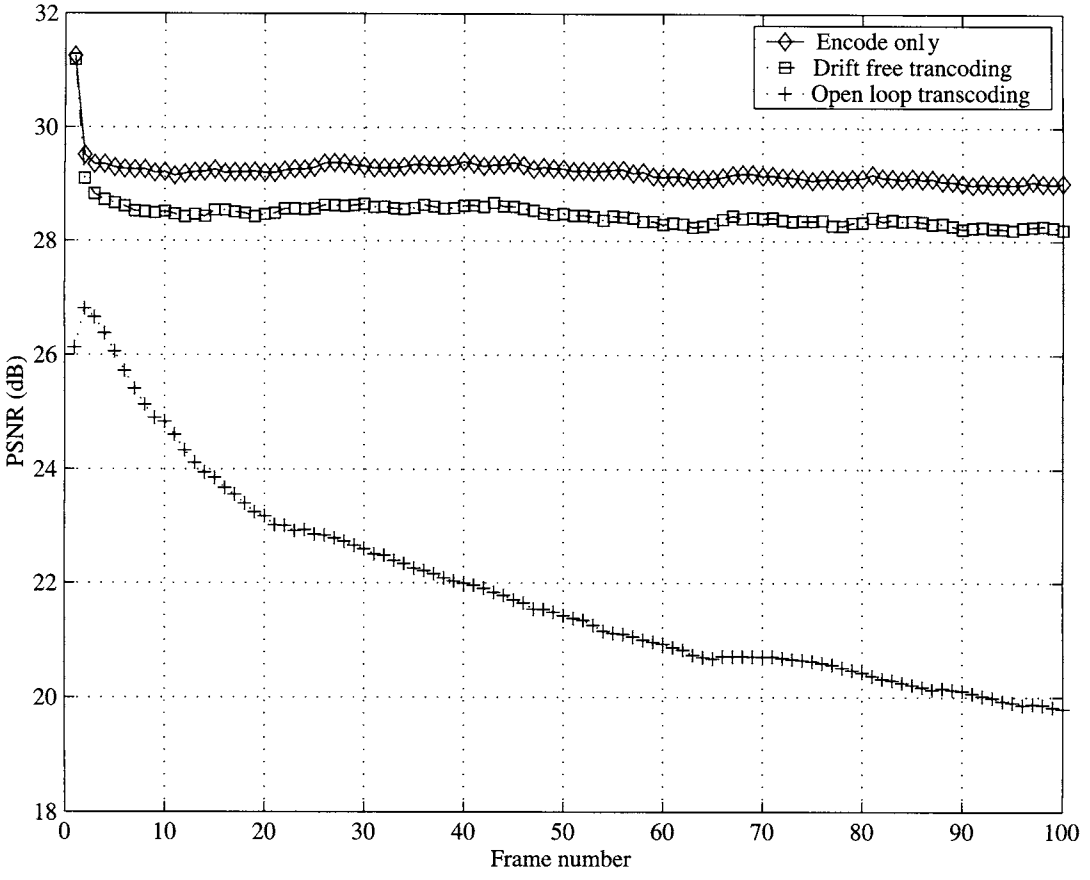

FIGURE 6 Performance of different video transcoding architectures (mobile sequence, $Q_1 = 5$, $Q_2 = 10$).

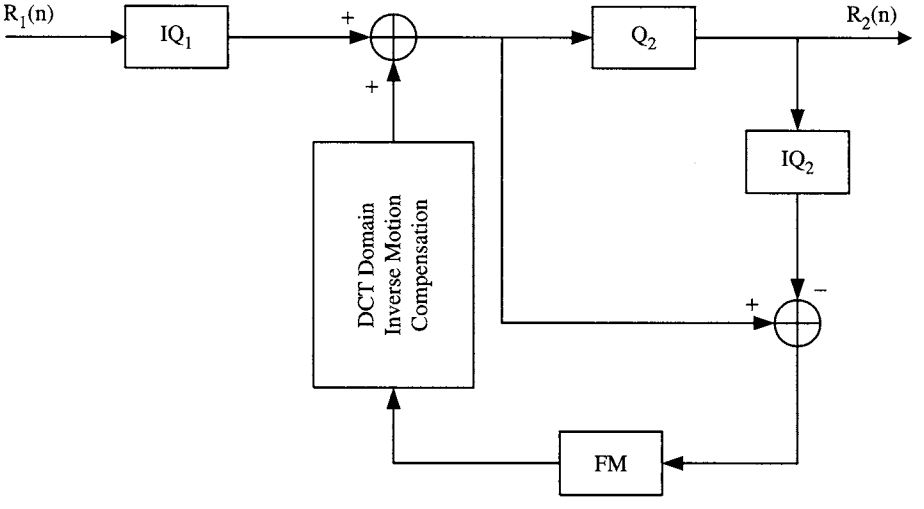

FIGURE 7 Fast DCT domain video transcoder.

block-by-block in the DCT domain, inverse motion compensation in the DCT domain is more complex than its counterpart in the spatial domain and hence becomes the bottleneck of DCT domain video transcoding algorithms.

The general setup of DCT domain inverse motion compensation is illustrated in Fig. 8, where $\hat{x}$ is the current block of interest, x_1, x_2, x_3 and x_4 are the reference blocks from which $\hat{x}$ is derived. In the spatial domain, $\hat{x}$ can be expressed as a superposition of the appropriate windowed and shifted versions of x_1, x_2, x_3 and x_4 as follows:

$$\hat{x} = \sum_{i=1}^{4} q_{i1} x_i q_{i2} \tag{8}$$

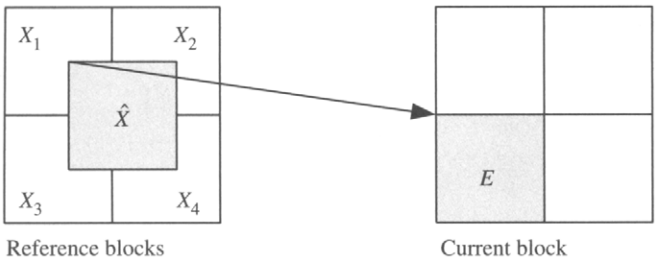

FIGURE 8 Illustration of DCT domain inverse motion compensation.

where q_{ij}, $i = 1, \cdots, 4$, $j = 1, 2$ are sparse 8×8 matrices of zeros and ones that perform windowing and shifting operations. For example, for $i = 1$,

$$q_{11} = \begin{pmatrix} 0 & I_h \\ 0 & 0 \end{pmatrix}, \; q_{12} = \begin{pmatrix} 0 & 0 \\ I_w & 0 \end{pmatrix}, \tag{9}$$

where I_h and I_w are identity matrices of dimension $h \times h$ and $w \times w$, respectively. The values h and w are determined by the motion vector corresponding to $\hat{x}$. By using the linear and distributive properties of the DCT, we can write the DCT domain inverse motion compensation as

$$\hat{X} = \sum_{i=1}^{4} Q_{i1} X_i Q_{i2} \tag{10}$$

where $\hat{X}$, X_i, Q_{i1} and Q_{i2} are the DCT's of $\hat{x}$, x_i, q_{i1} and q_{i2}, respectively. Note that Q_{i1} and Q_{i2} are constant and hence can be precomputed and stored in memory [11]. Obviously, the objective of DCT domain inverse motion compensation is to compute the DCT coefficients of $\hat{x}$ directly from the DCTs of x_i, $i = 1, \ldots 4$.

Brute-force computation of (10) in the case where the reference block $\hat{x}$ is not aligned in any direction with the block structure requires eight floating-point matrix multiplications and three matrix additions. Several algorithms have been proposed to reduce the computational complexity of the DCT domain inverse motion compensation. In [12], Merhav and Bhaskaran proposed to factorize the constant matrices Q_{ij} into a series of relatively sparse matrices instead of fully precomputing them. As a result, some of the matrix multiplications in (10) can be replaced by simple addition and permutation operations such that computational complexity can be reduced. In [3], the authors approximated the elements of Q_{ij} by binary numbers so that all multiplications can be implemented by basic integer operations such as *shift* and *add*. In general, motion compensation is done on a macroblock basis, meaning that all blocks in the same macroblock have the same motion vector. Based on this observation, Song and Yeo [13] presented a fast algorithm for DCT inverse-motion compensation by exploiting the shared information among the blocks within the same macroblock instead of constructing the DCT domain values of each target block independently like the previous two methods. However, this method does not apply to the case where one macroblock has multiple motion vectors. For instance, MPEG-4 and H.264 support four motion vectors per macroblock. While all three methods adopted the two-dimensional (2D) procedure shown in Fig. 8, Acharya and Smith [14] developed a separable implementation diagram in which the 2D problem was decomposed into two separate one-dimensional (1D) problems. They showed that the decomposition is more efficient than computing the combined operation.

In [15], the authors approached the problem of (10) from a different angle by analyzing the statistical properties of nature image/video data. The proposed algorithm first estimates the local bandwidth of the target block to be reconstructed from the reference blocks by modeling a natural image as a 2D separable Markov random field. The algorithm can reduce the processing time by avoiding the computations of those DCT coefficients outside the estimated local bandwidth. Note that the DCT coefficients inside the estimated local bandwidth can be computed by using other fast algorithms such as the ones discussed above. No significant distortion is introduced by the algorithm as shown in the experimental results in [15].

3 Heterogeneous Video Transcoding

Heterogeneous video transcoding has become increasingly important for supporting universal multimedia access due to the fact that multiple video coding standards coexist and a wide range of new devices with different characteristics have been developed for networked multimedia applications. In heterogeneous video transcoding, the incoming video bit stream may be subject to coding format conversion and spatial and/or temporal resolution reduction to satisfy the constraints imposed by the video communication channel and/or receiving devices. As such, the decoder and encoder loops in the heterogeneous video transcoder could not be combined to simplify the transcoder, as in the fast video transcoding architectures described in Section 2. Instead of simply cascading the video decoder and encoder, various efficient video transcoding techniques have been proposed in the literature by taking advantage of the information extracted from the incoming video bit stream (e.g., motion vectors, macroblock coding types as well as bit allocation statistics) to help reduce the computational complexity of the encoder. For instance, the motion vectors extracted from the input video can be reused or used to derive new motion vectors in the video encoder such that full-scale motion estimation, which comprises more than 60% of the encoding complexity, can be avoided [5].

Figure 9 shows a typical video transcoding architecture in which an input MPEG-2 video is transcoded to a MPEG-4

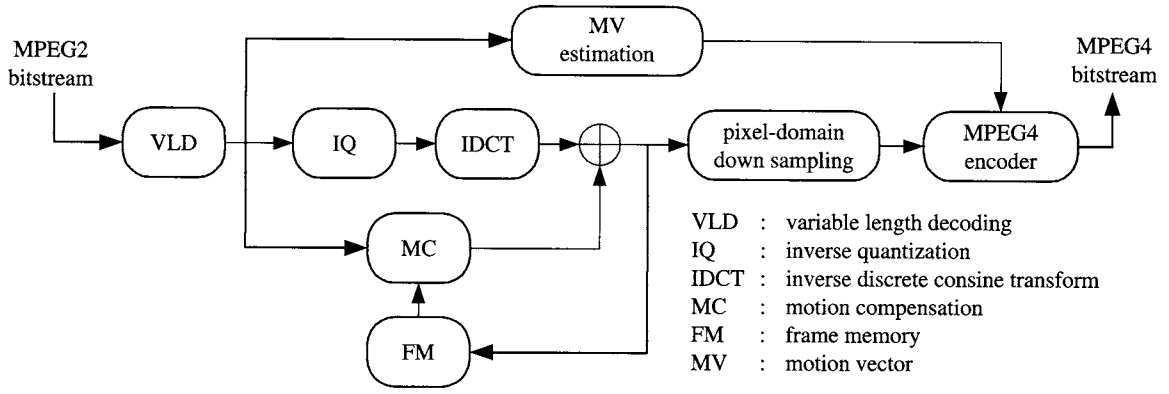

FIGURE 9 Pixel domain video transcoding architecture to transcode MPEG2 videos to MPEG4 bit streams with spatial and/or temporal resolution reduction.

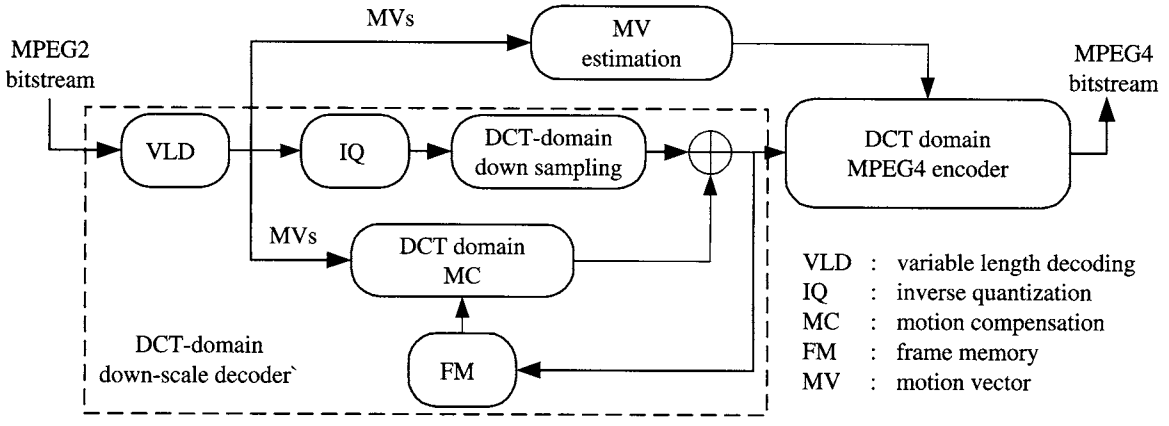

FIGURE 10 DCT domain video transcoder to transcode MPEG-2 videos to MPEG-4 bit streams with spatial and/or temporal resolution reduction.

video bit stream for wireless video applications. It first decodes the incoming MPEG-2 bit stream to the pixel domain by performing variable length decoding (VLD), inverse quantization (IQ), IDCT, and motion compensation (MC); then downscales the decoded video by a factor of two in both the horizontal and vertical directions in the pixel domain; and finally reencodes the downscaled video into an outgoing MPEG4 video bit stream. Note that the MC is performed using the original motion vectors. The motion vectors extracted from the incoming bit stream are used to speed up the motion estimation process in the MPEG-4 encoder.

If both the input and output video coding formats use the DCT, then DCT domain video transcoding architectures, where all operations are performed in the DCT domain to avoid DCT/IDCT process, can be used. Significant effort has been applied toward developing efficient DCT domain video transcoding architectures and algorithms, such as DCT domain spatial resolution reduction and DCT domain motion compensation. For example, in [16] a fast and memory-efficient DCT domain video transcoding architecture as illustrated in Fig. 10, where the input video is directly decoded to a lower resolution video by a so-called DCT domain downscale video decoder. In Fig. 10, the DCT domain MC is basically the same problem as the DCT domain inverse motion compensation (IMC) discussed in Section 2. Thus, fast algorithms for DCT domain IMC also apply to DCT domain MC. Compared with the pixel domain approach shown in Fig. 9, this transcoder can save more than 50% of required memory, since there is no buffer needed for the original high-resolution video frame. In addition, the computational cost is reduced by more than 70%. However, the video quality achieved by both approaches is hardly distinguishable at target bit rates of 384 kb/s and 256 kb/s according to the experimental results reported in [16].

In the following, we will review some recent progress made in heterogeneous video transcoding techniques, which include motion vector estimation, video down-sampling as well as macroblock coding type decision in the encoder. In the following discussion, we will assume the input video is down-sampled by two in both the horizontal and vertical resolutions. The techniques are extended to arbitrary scaling factors in [17–18].

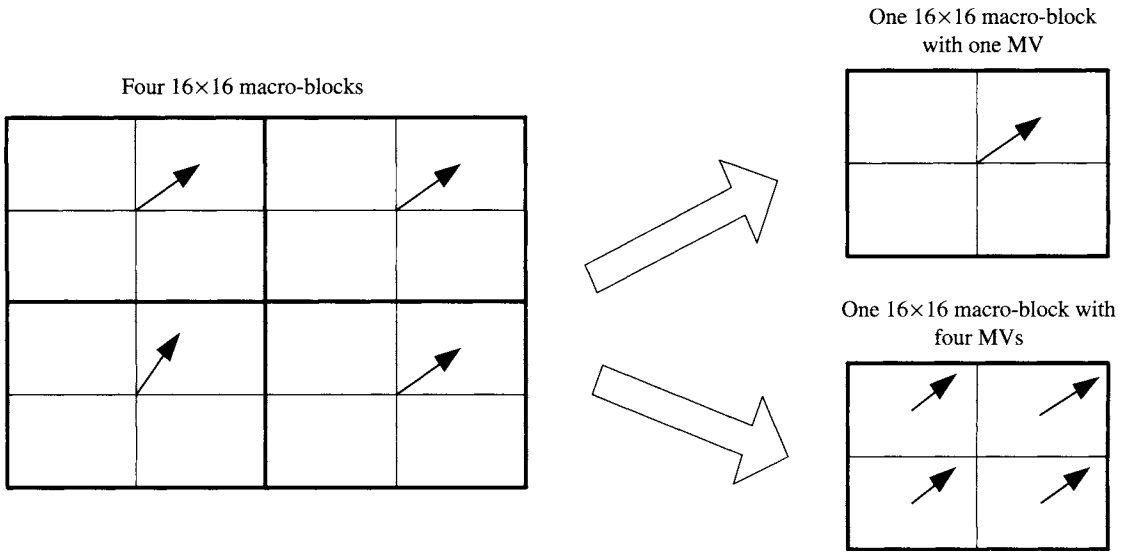

FIGURE 11 Illustration of motion vector (MV) mapping.

3.1 Motion Vector Estimation for Spatial Resolution Reduction

To avoid full-scale motion estimation in the encoder, the motion vectors extracted from the incoming video bit stream can be used to derive new motion vector(s) for the down-sampled outgoing video sequence. Due to the down-sampling in spatial resolution, one macroblock in the outgoing video corresponds to four macroblocks in the original video sequence. Each intercoded macroblock in the original video is assumed to have one motion vector. Thus, there will be four motion vectors associated with each 16×16 macroblock in the lower spatial resolution picture as shown in Fig. 11. If the outgoing video bit stream supports four 8×8 motion vectors in each 16×16 macroblock (such as MPEG-4), the input motion vectors of the original video can be used directly for each 8×8 block in the lower spatial resolution video with appropriate scaling by two. However, it is sometimes inefficient to use four motion vectors in each macroblock since more bits must be used to code four motion vectors. In the case where each 16×16 macroblock has one motion vector in the outgoing video, various techniques have been investigated to derive the output motion vector from the four input motion vectors. Three methods of deriving a new motion vector from the four input motion vectors are studied in [5]: the median value, the majority, and the mean value of the four input motion vectors. They showed that the median value gives the best results. The median vector is defined as one of the four input motion vectors that has the least Euclidean distance from all, for example,

$$\bar{v} = \arg\min_{v_k \in V} \{ \sum_{\substack{j=1 \\ j \neq k}}^{4} ||v_k - v_j|| \} \tag{11}$$

where $V = \{v_1, v_2, v_3, v_4\}$ and v_j, $j = 1, \ldots, 4$ are the incoming motion vectors and $\bar{v}$ is the candidate motion vector for the downscaled video. Note that the magnitude of the estimated vector v should be downscaled by a factor of two and those intracoded or skipped macroblocks in the incoming bit stream are usually viewed as predicted macroblocks with zero-valued motion vectors. Instead of simply averaging the four input motion vectors, it has been proposed [19] to estimate the new motion vector by using the weighted average of the four incoming motion vectors, where the weights correspond to the block activity. The proposed method produces higher video quality than the simple averaging method. However, the comparison between the median value method and the proposed method was not given in [19].

To further improve the output video quality, it has also been suggested to refine, within a small searching area, the motion vector derived from the four input vectors. The experimental results from [5] show that a refinement within 0.5 pixel range delivers satisfactory results. Nevertheless, the authors in [20] present an adaptive motion vector refinement algorithm called variable step-size search, where the search area is adapted to the motion vector magnitude itself. The algorithm achieves better video quality at the expense of higher computational cost.

3.2 Motion Vector Estimation for Temporal Resolution Reduction

To achieve higher bit rate reduction or to adapt to the processing power of the receiving device, the video transcoder may have to reduce the temporal resolution of the incoming video by dropping some of the encoded frames. For instance, a mobile terminal may only be capable of decoding and

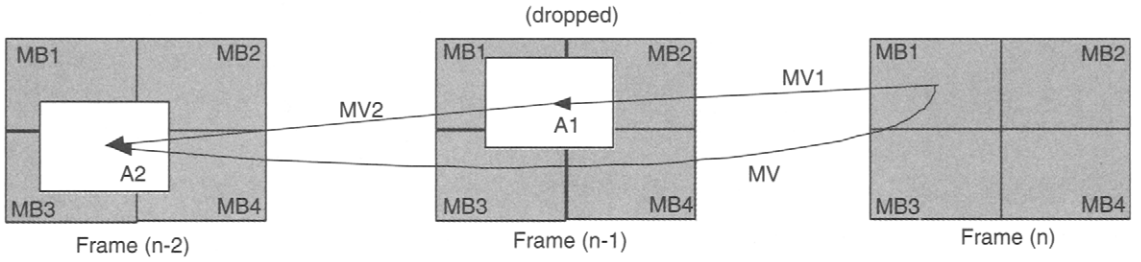

FIGURE 12 Motion vector reestimation. A new motion vector MV from frame n to frame n-2 will be derived from the incoming motion vectors MV1 and MV2 since the frame n-1 is dropped.

displaying 10 frames per second while most DVD videos are encoded at 30 frames per second.

As discussed earlier, motion vectors from the original video bit stream are usually reused or used to derive the new motion vectors for the outgoing bit stream in bit rate reduction and spatial resolution reduction video transcoders. For temporal resolution reduction, the problem is similar in that it is necessary to derive the motion vectors from the current frame to the previous nonskipped frame by using the motion vectors extracted from the skipped video frame(s). Figure 12 illustrates a situation where a frame is dropped. One way to obtain MV without performing full-scale motion estimation is to use the summation of MV1 and MV2. However, since the corresponding area of MV2 in frame n-1 is usually not aligned with macroblock boundary, MV2 is not available from the incoming bit stream. So the problem is boiled down to estimate MV2 from its four neighboring motion vectors obtained from the incoming bit stream. One method is to use bilinear interpolation from the four neighboring MVs. Another method is the forward dominant vector selection (FDVS) algorithm proposed in [21]. It selects the motion vector of the macroblock that has the largest overlapping portion with the area pointed at by the incoming motion vector MV1 in frame $n-1$. For instance, in Fig. 12, the FDVS would select the motion vector of MB1 as MV2 since the macroblock MB1 has the largest overlapping area with the area A1. The FDVS usually yields much better video performance than the bilinear interpolation scheme [21]. However, when the overlapping areas among the four macroblocks are very close, the motion vector selected by FDVS may not be optimal. On the basis of this observation, the authors in [20] proposed a so-called activity-dominant vector selection (ADVS) algorithm by utilizing the activity of each macroblock to determine which motion vector should be selected. The activity information of a macroblock can be represented by the number of nonzero DCT coefficients. From the simulation results reported in [20], the ADVS algorithm is superior to the FDVS method, especially in high-motion video sequences.

3.3 Spatial Resolution Reduction

In pixel domain video transcoding with spatial resolution reduction (Fig. 9), two methods can be used to perform spatial resolution reduction. The first one is pixel-averaging method in which every 2×2-pixel block is represented by a single pixel of their average value. The second is the subsampling method that applies low-pass filtering to the original image, then down-samples the low-pass filtered image by dropping every alternative pixel in both horizontal and vertical directions [22]. The first method is simpler to implement while the second usually gives better performance in video quality if a proper low-pass filter is used. With the popularity of DCT domain video transcoding techniques, DCT domain algorithms for spatial resolution reduction have also been studied. By utilizing the linear, distributive, and unity properties of the DCT transform, it is always possible to find a counterpart in the DCT domain for any given pixel domain linear operation [12, 23, 24]. Since most of the signal energy is concentrated at the lower frequency band in the DCT domain, the authors in [16] proposed to perform spatial resolution reduction by only decoding the top-left 4×4 DCT coefficients of each 8×8 block in the incoming bit stream. Then every four 4×4 DCT blocks are transformed into one 8×8 DCT block in the DCT domain as shown in Fig. 13.

To show this, let T and T_4 denote the 8×8 and 4×4 DCT operator matrices, respectively. Then we have

$$
\begin{aligned}
B = TbT^t &= [T_L \; T_R] \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix} \begin{bmatrix} T_L^t \\ T_R^t \end{bmatrix} \\
&= \frac{1}{2} \begin{bmatrix} T_L & T_R \end{bmatrix} \begin{bmatrix} T_4^t B_1 T_4 & T_4^t B_2 T_4 \\ T_4^t B_3 T_4 & T_4^t B_4 T_4 \end{bmatrix} \begin{bmatrix} T_L^t \\ T_R^t \end{bmatrix} \\
&= \frac{1}{2}(T_L T_4^t)[B_1(T_L T_4^t)^t + B_2(T_R T_4^t)^t] \\
&\quad + \frac{1}{2}(T_R T_4^t)[B_3(T_L T_4^t)^t + B_4(T_R T_4^t)^t]
\end{aligned} \tag{12}
$$

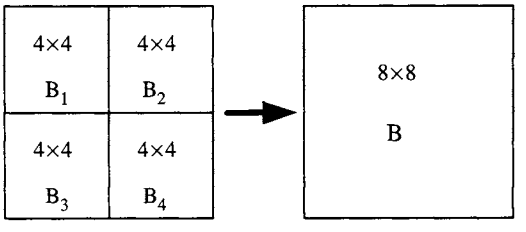

FIGURE 13 Convert four 4×4 subblocks to one 8×8 block.

where b, b_1, b_2, b_3, b_4 are the pixel-domain representations of the DCT blocks B, B_1, B_2, B_3, B_4 (shown in Fig. 13), respectively; T_L, T_R are 8×4 matrices denoting the first and last four columns of the eight-point DCT kernel T, respectively, and t denotes the matrix transpose. Let $C = T_L T_4^t + T_R T_4^t$ and $D = T_L T_4^t - T_R T_4^t$, then we have $T_L T_4^t = \frac{C+D}{2}$ and $T_R T_4^t = \frac{C-D}{2}$. Hence, (12) can be re-written as

$$\begin{aligned} B &= \frac{1}{8}\{[C(B_1 + B_3) + D(B_1 - B_3)](C + D)^t \\ &\quad + [C(B_2 + B_4) + D(B_2 - B_4)](C - D)^t\} \\ &= \frac{1}{8}[X(C + D)^t + Y(C - D)^t] \\ &= \frac{1}{8}[(X + Y)C^t + (X - Y)D^t] \end{aligned} \tag{13}$$

with

$$X = C(B_1 + B_3) + D(B_1 - B_3) \tag{14}$$

$$X = C(B_2 + B_4) + D(B_2 - B_4) \tag{15}$$

It can be shown that more than 50% of the elements in the matrices C and D are zeros, hence (14)–(15) can be computed very efficiently [28].

3.4 Macroblock Coding Type Decision

In video transcoding with spatial resolution reduction, every four macroblocks in the incoming bit stream will collapse to one macroblock in the outgoing bit stream as shown in Figure 11. Since the four incoming macroblocks may have different coding types, a new macroblock coding type has to be designated to the output macroblock in the output bit stream. One way is to derive a new macroblock coding type by comparing the coding complexity of different coding types as in a stand-alone video encoder. However, the cost of carrying out a new macroblock coding type decision is rather high [5, 29]. Another method is to select a macroblock coding type from the four incoming macroblock coding types. For instance, we can use most of the input macroblock coding types. In other words, if three of the four input macroblocks are intercoded, then the output macroblock will be intercoded. The intracoded input macroblock will be considered as the intercoded one but with zero-valued motion vectors. Experimental results reported in [5] show that the video quality degradation in the second method is less than 0.2 db, compared with the one achieved by the first method.

4 Bit Rate Control in Video Transcoding

One important application of video transcoding is to adapt the bit rate of precoded video bit streams to various network connections. In video transcoding, the performance of bit rate adaptation heavily depends on the bit rate control technique used in the video transcoder. Therefore, bit rate control is one of the key components in a video transcoder. In the following, we discuss the difference between the problem of bit rate control in video transcoding and that in video encoding.

Generally speaking, bit rate control is a budget constrained bit allocation problem. A stand-alone video encoder usually relies on a preanalyzed rate-distortion (R-D) model to solve the bit allocation problem. The R-D model determines the relationship between the number of produced bits and the quantization step size. Various R-D models have been proposed in the literature based on information theory and the characteristics of the human vision system [30–34], such as the TMN8 and TMN10 models developed in the H.263 standard. Since the R-D model is usually derived from extensive experimental data, it may not match the R-D characteristics of any particular video sequence. The mismatch may produce large bit rate control error in video encoding. To reduce the mismatch, most bit rate control techniques update the R-D model by using the actual bit usage information in the previous frame or macroblock and then apply the updated R-D model to the current frame or macroblock. However, if the statistical properties of the current frame or macroblock are significantly different from the previous one, the mismatch problem still exists. For example, during a scene change, the current frame may be completely different from the previous one. Thus, the R-D model based on the statistics of the previous frame may not fit the current frame at all.

Although the bit rate control techniques used in video encoding can be directly used in video transcoding, the bit allocation and rate control in video transcoding are different from that in video encoding. The input of video transcoding is a preencoded video bit stream that contains much information such as bit usage of each macroblock, macroblock coding types, and motion vectors. The bit usage information extracted from the input bit stream can be reused by the video transcoder to build an R-D model that matches the actual statistics of the video frame or macroblock being encoded so that more accurate bit allocation and rate control can be achieved [35–37]. In [35], the authors proposed an accurate bit allocation and rate control algorithm for video transcoding by exploiting the approximate relationship between the number of VLC code words and the number of produced bits for encoding those VLC code words. Since both the number of VLC code words and the number of produced bits are already available in the incoming bit stream, an accurate bit allocation model can be derived from the information contained in the incoming bit stream. The proposed rate control scheme can adaptively allocate bits to each macroblock in a frame and make the total bits of a frame meet the target bits budget. Compared to TMN-8,

the proposed technique can allocate the bit budget more efficiently to the macroblocks in a frame to produce better video quality.

5 Error-Resilient Video Transcoding

With the emergence of various advanced wireless networks such as 3G cellular systems, wireless video has become increasingly popular and is attracting great interest [38–40]. Transmitting coded video stream over wireless channels is even more challenging than over wired channels since wireless channels usually have lower bandwidth and higher bit error rates than wired channels.

Error-resilient video transcoding increases the error resilience of a video stream to produce acceptable video quality at the receiver side. For example, the authors in [8] proposed an error-resilient transcoding technique that is built on three fundamental blocks:

1. Increase the spatial resilience by reducing the number of macroblocks per slice and enhance the temporal resilience by increasing the proportion of intracoded macroblocks in each frame. The amount of resilience is tailored to the content of the video and the prevailing error conditions, as characterized by bit error rate.
2. Derive an analytic model that characterizes how the bit errors propagate in a video that is compressed using motion-compensated encoding and subjected to bit errors.
3. Compute the optimal bit allocation between spatial resilience, temporal resilience, and video signal itself so that an optimal resilience will be injected into the video stream.

In [7], the authors presented a novel fully comprehensive mobile video communication system that exploits the useful rate management features of the video transcoders and combines them with error resilience for transmissions of coded video streams over general packet radio service (GPRS) mobile access networks. The error-resilient video transcoding operation takes place in a video proxy, which not only performs bit rate adaptation but also increases error resilience. In the proposed system, two resilience schemes are used, which are adaptive intrarefresh (AIR) and feedback control signaling (FCS) methods. These two schemes can work independently or combined. The system adjusts the output bit rate from the proxy by monitoring the current channel conditions such as bit error rate, delay, and so forth. Meanwhile, the amount of resilience added to the video data can also be controlled by monitoring the proxy output rate and the change in the error conditions of the network connection. AIR is mainly used to stop the error propagation across different frames. Experiments showed that the combined AIR-FCS method gave superior transcoding performances over error-prone GPRS channels relative to nonresilient video streams.

6 Concluding Remarks

Besides the transcoding techniques discussed in this chapter, other video transcoding techniques have also been proposed and developed for various applications. These include content-based transcoding [41, 42], joint transcoding of multiple video streams [43], transcoding for fast forward and reverse playback [44], and other more exotic schemes.

In summary, most video transcoding research activities have been focusing on the following two issues:

1. Reduce the computational cost of video transcoding for real-time operations. Due to the high data volume and computational complexity of video decoding and encoding, efficient algorithms are needed for real-time video transcoding.
2. Develop efficient bit rate control algorithms to achieve better video quality by exploiting the information extracted from the input bit stream and the properties of the HVS.

While a large number of video transcoding techniques have been developed, the two problems above are still not fully solved, especially the second issue. For any given target bit rate, various transcoding techniques can be used to achieve the target bit rate. For example, we can choose to increase the quantization step size, or reduce the spatial/temporal resolution, drop the chromatic components, etc. Yet, there is a lack of a unified video transcoding strategy that can automatically determine which transcoding technique to be used to achieve the best visual quality to the end users. This requires developing utility functions that can gauge a user's satisfaction of a coded video [45, 46]. Although great progress has been made in video quality assessment recently (see Chapters 8.2 through 8.4), further study is still needed toward developing algorithms that measure and compare transcoded video quality across spatiotemporal scales in a perceptually consistent way.

References

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

[2] Kou-Sou Kan and Kuo-Chin Fan, "Video transcoding architecture with minimum buffer requirement for compressed MPEG-2 bitstream," *Signal Processing*, 67, 223–235, 1998.

[3] Pedro A. A. Assuncao, and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams," *IEEE Trans. Circuits Syst. Video Technol.*, 8, 953–967, Dec. 1998.

[4] G. Keesman, R. Hellinghuizen, F. Hoeksema, and G. Heideman, "Transcoding of MPEG bitstreams," *Signal Processing: Image Communication*, 8, 481–500, 1996.

[5] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats," *IEEE Transactions on Multimedia*, 2(2), 101–110, June 2000.

[6] J. Nakajima, H. Tsuji, Y. Yashima, and N. Kobayashi, "Motionvector re-estimation for fast video transcoding from MPEG-2 to MPEG-4," MPEG-4. 2001 Proceedings of Workshop and Exhibition on MPEG-4, 87–90, 18–20 June 2001.

[7] S. Dogan, A. Cellatoglu, M. Uyguroglu, A. H. Sadka, and A. M. Kondoz, "Error-resilient video transcoding for robust internetwork communications using GPRS," *IEEE Transactions on Circuits and Systems for Video Technology*, 12(6), 453–464, June 2002.

[8] G. de los Reyes, A. R. Reibman, S.-F. Chang, and J. C.-I. Chuang, "Error-resilient transcoding for video over wireless channels," *IEEE Journal on Selected Areas in Commun.*, 18, 1063–1074, June 2000.

[9] O. Werner, "Requantization for transcoding of MPEG-2 intraframe," *IEEE Trans. on Image Processing*, 8, 179–191, Feb. 1999.

[10] H. Sorial, W. E. Lynch, and A. Vincent, "Selective requantization for transcoding of MPEG compressed video," in *Multimedia and Expo, ICME 2000*, 1, 217–220, 2000.

[11] S. F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE Journal of Selected Areas in Communications*, 13, 1–11, Jan. 1995.

[12] N. Merhav and V. Bhaskaran, "Fast algorithm for DCT-domain image down-sampling and for inverse motion compensation," *IEEE Trans. on Circuits and Systems for Video Tech.*, 7, 468–476, June 1997.

[13] J. Song and B.-L. Yeo, "A fast algorithm for DCT-domain inverse motion compensation based on shared information in a macroblock," *IEEE Trans. on Circuits and Systems for Video Tech.*, 10, 767–775, Aug. 2000.

[14] S. Acharya and B. Smith, "Compressed domain transcoding of MPEG," Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS), Austin, TX, June 1998.

[15] S. Liu and A. C. Bovik, "Local bandwidth constrained fast inverse motion compensation for DCT-domain video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, 12(5), 309–319, May 2002.

[16] S. Liu and A. C. Bovik, "A fast and memory efficient video transcoder for low bit rate wireless communications," Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2002, 2, 1969–1972, 2002.

[17] Y.-P. Tan, H. Sun, and Y. Q. Liang, "On the methods and applications of arbitrarily downsizing video transcoding," Multimedia and Expo, 2002. ICME '02. 1, 609–612, 26–29 Aug. 2002.

[18] Y. Q. Liang, L.-P. Chau, and Y.-P. Tan, "Arbitrary downsizing video transcoding using fast motion vector reestimation," *Signal Processing Letters, IEEE*, 9(11), 352–355, Nov. 2002.

[19] B. Shen, I. K. Sethi, and B. Vasudev, "Adaptive motion-vector resampling for compressed video downscaling," *Circuits and Systems for Video Technology, IEEE Transactions on*, 9(6), 929–936, Sept. 1999.

[20] M.-J. Chen, M.-C. Chu, and C.-W. Pan, "Efficient motion_estimation algorithm for reduced frame-rate video transcoder," *Circuits and Systems for Video Technology, IEEE Transactions on Circuits and Systems for Video Technology*, 12(4), 269–275, Apr. 2002.

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

[22] Video Codec Test Model, TM5, Jan. 31, 1995. ITU Telecommunication Standardization Sector LBC-95, Study Group 15, Working Party 15/1.

[23] W. Kou and T. Fjalbrant, "A direct computation of DCT coefficients for a signal block taken from two adjacent blocks," *IEEE Trans. Signal Processing*, 39, 1692–1695, July 1991.

[24] J. B. Lee and B. G. Lee, " Transform domain filtering based on pipelining structure," *IEEE Trans. Signal Processing*, 40, 2061–2064, Aug. 1992.

[25] Zhijun Lei and N. D. Georganas, "H.263 video transcoding for spatial resolution downscaling," Information Technology: Coding and Computing, 2002. Proceedings International Conference on Information Technology: Coding and Computing 2002, 425–430, 8–10 April 2002.

[26] Peng Yin, Min Wu, and Bede Liu, "Video transcoding by reducing spatial resolution," Image Processing, 2000, 1, 972–975, 10–13 Sept. 2000.

[27] Nam-Hyeong Kim, Yoon Kim, Goo-Rak Kwon, and Sung-Jea Ko, "A fast DCT domain downsampling technique for video transcoder," Consumer Electronics, 2003. ICCE. 2003 IEEE International Conference on Consumer Electronics, 2003, 36–37, June 2003.

[28] R. Dugad and N. Ahuja, "A fast scheme for image size change in compressed domain," *IEEE Trans. on Circuits and Systems for Video Tech.*, 11, 461–474, Apr. 2001.

[29] P. Yin, A. Vetro, B. Liu, and H. Sun, "Drift compensation for reduced spatial resolution transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, 12, 1009–1020, Nov. 2002.

[30] Y. Yang and Sheila S. Hemami, "Generalized rate-distortion optimization for motion-compensated video coders," *IEEE Trans. on CSVT*, 10, Sept. 2000.

[31] H.-M. Hang and J.-J. Chen, "Source model for transform video coder and its applications — Part I: fundamental theory," *IEEE Trans. on CSVT*, 7, Apr. 1997.

[32] Z. He, Y. K. Kim, and S. K Mitra, "Low-delay rate control for DCT video coding via ρ-domain source modeling," *IEEE Transactions on CSVT*, 11, 928–940, Aug. 2001.

[33] T. Chiang and Y.-Q. Zhang, "A new rate control scheme using quadratic rate distortion model," *IEEE Trans. on CSVT*, 1, Feb. 1997.

[34] A. Puri and R. Aravind, "Motion-compensated video coding with adaptive perceptual quantization," *IEEE Trans. on CSVT*, 1, Dec. 1991.

[35] Z. Lei and N. D. Georganas, "Accurate bit allocation and rate control for DCT domain video transcoding," Electrical and

Computer Engineering, 2002. IEEE CCECE 2002. Canadian Conference on Electrical and Computer Engineering, 2002, 2, 968–973, 12–15 May 2002.

[36] S. Liu and C.-C. J. Kuo, "Joint temporal-spatial rate control for adaptive video transcoding," Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on, 2, 225–228, 6–9 July 2003.

[37] Y. Sun, X.-H. Wei, and I. Ahmad, "Low delay rate-control in video transcoding," Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03., 2, 660–663, 25–28 May 2003.

[38] J. Cai and C. W. Chen, "A high-performance and low-complexity video transcoding scheme for video streaming over wireless links," Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE, 2, 913–917, 17–21 March 2002.

[39] O. Iwasaki, T. Uenoyama, A. Ando, T. Nishitoba, T. Yukitake, and M. EtoH, "Video transcoding technology for wireless communication systems," Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st, 2, 1577–1580, 15–18 May 2000.

[40] R. Asorey-Cacheda and F. G. Gonzalez-Castano, "Real-time transcoding and video distribution in IEEE 802.11b multicast networks," IEEE International Symposium Proceedings on Computers and Communication, 2003. (ISCC 2003), 693–698, 30 June–3 July 2003.

[41] A. Cavallaro, O. Steiger, and T. Ebrahimi, "Semantic segmentation and description for video transcoding," Multimedia and Expo, 2003. ICME '03, 3, 597–600, 6–9 July 2003.

[42] Y. Q. Liang, Y.-P. Tan, "A new content-based hybrid video transcoding method," Image Processing, 2001. Proceedings. 2001 International Conference on Image Processing, 1, 429–432, 7–10 Oct. 2001.

[43] H. Sorial, W. E. Lynch, and A. Vincent, "Joint transcoding of multiple MPEG video bitstreams," in *Proc. IEEE Int. Symp. Circuits Syst.*, Orlando, FL, May 1999.

[44] Y. P. Tan, Y. Q. Liang, and J. Yu, "Video transcoding for fast forward/reverse video playback," in *Proc. IEEE Int. Conf. Image Processing*, Rochester, NY, 1, 713–716, Sept. 2002.

[45] Z. Yuan and H. R Wu, "Human visual system based objective digital video quality metrics," Signal Processing Proceedings, 2000. WCCC-ICSP 2000. 5th International Conference on Signal Processing, 2, 1088–1095, 21–25 Aug. 2000.

[46] Zhou Wang, Ligang Lu, and A. C. Bovik, "Video quality assessment using structural distortion measurement," Image Processing. 2002. Proceedings. 2002 International Conference on Image Processing, 3, 65–68, 24–28 June 2002.