Video Transcoding for Multipoint Videoconferencing

Chia-Wen Lin* (林嘉文) and Yung-Chang Chen** (陳永昌)
*Computer and Communications research Laboratories
Industrial Technology Research Institute
ljw@n100.ccl.itri.org.tw
**Department of Electrical Engineering
National Tsing Hua University
ycchen@ee.nthu.edu.tw

1. INTRODUCTION

Multimedia systems, which are the combination of text, graphics, speech, audio, and video, have a wide variety of applications. Networked multimedia services, such as teleconferencing, video on demand, and distance learning have been emerging in various network environments. In these applications, it is often needed to adapt the bit-rate of the video bit-streams to the available network bandwidth over various channels [1-2,13-16]. In heterogeneous networks as depicted in Fig. 1, bit-rate adaptation allows different end-users over different sub-networks to achieve different Quality of Service (QoS) based on their available network bandwidths. For some real-time video coding applications, the scaling of bit-rates can be achieved through the rate control in the video encoder [16]. However, for many other applications, such as video on demand, this can not be done since the video is already compressed at a certain bit-rate and stored in the server.

Video conferencing is a natural extension to voice-only communications. Due to the rapid progress in digital video processing, low cost video codecs, networking technologies, and international standards, video conferencing is becoming more and more widely used. Video conferencing can be over a circuit-switched network such as PSTN and ISDN, or over a packet-switched network such as IP-based network and Asynchronous Transfer Mode (ATM) network. International Telecommunication Union (ITU) has established various video conferencing terminal standards for different network environments. Some ITU standards are: H.320 [3] for N-ISDN, H.310 [4] and H.321 [5] for ATM, H.322 [6] for quaranteed QOS LANs, H.323 [7] for non-quaranteed QoS LANs, and H.324 [8] for PSTN.

Multipoint video conferencing is a natural extension of point-to-point video conferencing. With the rapid growth of video conferencing, the need for multipoint video conferencing is also growing. For a multipoint video conference over a wide-area network, the conference participants are connected to a multipoint control unit (MCU) in a central office. A video combiner in the MCU combines the multiple coded digital video streams from the conference participants into a coded video bit stream, and sends it back to the conference participants for decoding and presentation. The application scenario of four persons participating in a four-point video conference is shown in Fig. 2. An example of MCU with the gateway function between H.323 and H.324/I terminals is illustrated in Fig. 3. The conference participants can link to the MCU through standard H.323 or H.324/I terminals. The MCU combines the videos of the participants, converts the bandwidths of the combined videos to adapt to the participants' channel capacities and send back the combine video to each participant. The Multipoint Controller (MC) function is responsible for controlling the multipoint connection, and the Multipoint Processor (MP) deals with video combining and rate adaptation.

For rate adaptation, video transcoding is generally used to convert a previously compressed video bit-stream into another lower bit-rate video bit-stream (for lower bit-rate to higher bit-rate adaptation, zero stuffing is often used). Video transcoding has been studied recently in several literatures [13-18] because of its wide range of applications. In general, video transcoding deals with converting a previously compressed video signal into another compressed signal with a different format or bit rate. As the number of different video compression standards (e.g., H.261 [9], H.263 [10], MPEG-2 [11], MPEG-4 [12]) increases and the variety of bit rates at which they are operated for different applications, there is a growing need for video transcoding. In this article, we will focus on the specific problem of transcoding for bitrate reduction.

2. SINGLE POINT VIDEO TRANSCODING ARCHITECTURES

A simple architecture for the transcoder uses open-loop transcoding where the incoming bit-rate is down-scaled by truncating the DCT coefficients or performing a requantization process [16-17]. Since the transcoding is done in the coded domain, a very simple and fast transcoder is possible. However, the open-loop transcoding produces an increasing distortion caused by the drift due to the mismatched reconstructed pictures in the encoder and the decoder, which results in unacceptable video quality in many applications. Drift-free transcoding [18, 22-23] is possible by using a decoder to decode the incoming video and then using an encoder to re-encode the video at the lower rate.

The most straightforward approach to implement a transcoder is to cascade the decoder and encoder directly as shown in Fig. 4. And it's implementation based on the H.263 standard is shown in Fig. 5. This architecture, however, is too computationally costly to be adopted in practical real-time applications since it requires one video decoder and one video encoder for each transcoding operation.

2.1 Pixel-Domain Video Transcoders

With the cascaded architecture shown in Fig. 5, the motion vectors adopted in the video decoder and encoder are in general not the same, the architecture can be expressed as follows:

$$\begin{aligned} I_n^{(2)} &= P_n^{(2)} + R_n^{(2)} + E_n^{(2)} \\ &= I_n^{(1)} - R_n^{(2)} + R_n^{(2)} + E_n^{(2)} \\ &= I_n^{(1)} + E_n^{(2)} \end{aligned} \tag{1}$$

$$\begin{aligned} R_n^{(2)} &= I_n^{(1)} - P_n^{(2)} \\ &= I_n^{(1)} - S[I_{n-1}^{(2)}, \underline{V}_n^{(2)}] \\ &= I_n^{(1)} - S[I_{n-1}^{(1)}, \underline{V}_n^{(2)}] - S[E_{n-1}^{(2)}, \underline{V}_n^{(2)}] \end{aligned} \tag{2}$$

$$I_n^{(1)} = R_n^{(1)} + S[I_{n-1}^{(1)}, \underline{V}_n^{(1)}] \tag{3}$$

$$R_n^{(2)} = R_n^{(1)} + \{S[I_{n-1}^{(1)}, \underline{V}_n^{(1)}] - S[I_{n-1}^{(1)}, \underline{V}_n^{(2)}]\} - S[E_{n-1}^{(2)}, \underline{V}_n^{(2)}] \tag{4}$$

$$\begin{aligned} \mathrm{DCT}(R_n^{(2)}) = \mathrm{DCT}(R_n^{(1)}) &+ \mathrm{DCT}\{S[I_{n-1}^{(1)}, \underline{V}_n^{(1)}] - S[I_{n-1}^{(1)}, \underline{V}_n^{(2)}]\} \\ &- \mathrm{DCT}(S[E_{n-1}^{(2)}, \underline{V}_n^{(2)}]) \end{aligned} \tag{5}$$

where $S[I_n^{(i)}, \underline{V}_n]$ stands for the *shift* operation which locates the block translated by motion vector $\underline{V}_n$ for the n-th frame in the i-th stage frame memory.

Since a pre-encoded video stream arriving at the transcoder already carries many useful information such as picture type, motion vectors, quantization step-size, bit-allocation statistics, and so forth, it is possible to construct transcoders with different complexity and performance in terms of coding efficiency and video quality. In Fig. 5, intuitively, most of the motion information and the mode decision information received in the video decoder can be reused in the video encoder without introducing significant degradation on visual quality. Based on this assumption, Keesman *et al*. [18] proposed a simplified architecture with less computational cost by reusing in the encoder stage the

same motion vectors from the decoder stage (i.e., $\underline{V}_n^{(1)} = \underline{V}_n^{(2)} = \underline{V}_n$). With motion vector reuse, Equation (5) becomes

$$\mathrm{DCT}(R_n^{(2)}) = \mathrm{DCT}(R_n^{(1)}) - \mathrm{DCT}(S[E_{n-1}^{(2)}, \underline{V}_n^{(2)}]\ ) \tag{6}$$

which leads to the simplified architecture shown in Fig. 6. This simplified architecture can save the motion estimation operation, one frame memory, and one IDCT operation. In general, the motion vector reuse approach is considered as an efficient scheme for complexity reduction for motion estimation for video transcoding. It was, however, shown in [13-14] that, in many applications the reuse of the incoming motion vector results in non-optimized outgoing motion vectors due to the significant quantization errors between the first stage quantizer and the second stage quantizer. Motion vector refinement schemes are proposed in [13-14] to keep the computation cost minimum while achieving the performance close to that obtained in the cascaded transcoder architecture using full-scale motion estimation.

2.2 DCT-Domain Video Transcoders

In the pixel-domain video transcoder shown in Fig. 6, the coded quantization errors of the second stage quantizer are decoded into pixel domain through DCT transform and then stored in the frame memory in 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 requires one DCT transform, one IDCT transform, and one block shift operation (motion compensation). This operation can also performed in the DCT domain without the need of DCT/IDCT computation for encoding/decoding process. Some special treatments, however, are required for the so-called "DCT-domain inverse motion compensation (MCD^{-1} in abbreviation)" to perform the inverse motion compensation operation fully in the DCT domain and were investigated in the literature [19-21, 23]. Fig. 7 shows the operations of the inverse motion compensation performed in pixel-domain and DCT-domain. This inverse motion compensation operation, regardless of being performed in the pixel domain or the DCT domain, is the most computationally critical part for video transcoders. To reduce the computational complexity of the inverse motion compensation process, it's possible to perform this process in DCT domain thus no DCT/IDCT operation is required. As shown in Fig. 8, the problem can be interpreted as computing the elements of a target DCT block B from the elements of its four neighboring DCT blocks, B_i, i = 1 to 4, where $B = \mathrm{DCT}(\mathbf{b})$ and $B_i = \mathrm{DCT}(\mathbf{b}_i)$

are the 8×8 blocks of the DCT coefficients of the associated segmented blocks **b** and $\mathbf{b}_i$ of the image data in the pixel domain.

The 8×8 2D-DCT transforms a block $\mathbf{x} = \{x(n,m)\}_{n,m=0}^{7}$ in the spatial domain into a matrix of frequency components $X = \{X(u,v)\}_{u,v=0}^{7}$ according to the following equation

$$X(u,v) = \frac{c(u)}{2}\frac{c(v)}{2}\sum_{n=0}^{7}\sum_{m=0}^{7} x(n,m)\cos\left(\frac{2n+1}{16}\cdot u\pi\right)\cos\left(\frac{2m+1}{16}\cdot v\pi\right) \tag{7}$$

In a matrix form, define the 8-point DCT matrix $S = \{s(u,n)\}_{u,n=0}^{7}$ where

$$s(u,n) = \frac{c(u)}{2}\cos\left(\frac{2n+1}{16}\cdot u\pi\right) \tag{8}$$

Then,

$$X = S\mathbf{x}S^{t} \tag{9}$$

A DCT-domain approach to exactly solve the problem was firstly proposed in [5], which represented the target pixel-domain block, **b**, as the sum of the four component blocks which are denoted as $\mathbf{b}_{14}$, $\mathbf{b}_{23}$, $\mathbf{b}_{32}$, and $\mathbf{b}_{41}$ corresponding to the four adjacent DCT blocks respectively as shown in Figs. 9 and 10.

Thus the target pixel-domain block **b** can be expressed as:

$$\mathbf{b} = \mathbf{b}_{14} + \mathbf{b}_{23} + \mathbf{b}_{32} + \mathbf{b}_{41} \tag{10}$$

Chang and Messerschmitt [19] firstly used some geometric transforms to derive the relationship among the elements of the DCT blocks B and $B_1 \sim B_4$ from the spatial geometric relationship among the associated pixel-domain blocks **b** and $\mathbf{b}_1 \sim \mathbf{b}_4$ so that the elements of the target block B can be computed from the elements of the neighboring blocks $B_1 \sim B_4$ directly in the DCT domain without the need of full decompression of the DCT blocks into the pixel domain. An example of the geometric relationship between the block $\mathbf{b}_4$ and $\mathbf{b}_{41}$ is discussed below. Other geometric relationships can be obtained in similar manners.

The relation of the pixel-domain block $\mathbf{b}_{41}$ and the block $\mathbf{b}_4$ can be formulated by using a geometric transform as

$$\mathbf{b}_{41} = \mathbf{h}_{h_4}\mathbf{b}_4\mathbf{h}_{w_4}, \quad \text{where } \mathbf{h}_{h_4} = \begin{bmatrix} 0 & 0 \\ I_{h_4} & 0 \end{bmatrix} \mathbf{h}_{w_4} = \begin{bmatrix} 0 & I_{w_4} \\ 0 & 0 \end{bmatrix} \tag{11}$$

where I_{h_4} and I_{w_4} are identity matrices of size $h_4 \times h_4$ and $w_4 \times w_4$ respectively. Equation (11) is also illustrated in Fig. 11. Thus, through the distributive property of the DCT transform (viz. DCT(AB)=DCT(A)DCT(B))

$$B_{41} = H_{h_4} B_4 H_{w_4} \tag{12}$$

where $B_{41} = \mathrm{DCT}(\mathbf{b}_{41})$, $B_4 = \mathrm{DCT}(\mathbf{b}_4)$, $H_{h_4} = \mathrm{DCT}(\mathbf{h}_{h_4})$, and $H_{w_4} = \mathrm{DCT}(\mathbf{h}_{w_4})$

Through the geometric transform, the DCT coefficients of the target block B can be expressed as

$$B = \sum_{i=1}^{4} H_{h_i} B_i H_{w_i} \tag{13}$$

Direct computation of Equation (13) requires 8 matrix multiplications and 3 matrix additions. Note that, assume $h_4 = h$ and $w_4 = w$, the geometric transform matrices is tabulated as Table 1. From Table 1, the following equalities hold: $\mathbf{h}_{h_1} = \mathbf{h}_{h_2} = \begin{bmatrix} 0 & I_{8-h} \\ 0 & 0 \end{bmatrix}$, $\mathbf{h}_{h_3} = \mathbf{h}_{h_4} = \begin{bmatrix} 0 & 0 \\ I_h & 0 \end{bmatrix}$, $\mathbf{h}_{w_1} = \mathbf{h}_{w_3} = \begin{bmatrix} 0 & 0 \\ I_{8-w} & 0 \end{bmatrix}$, and $\mathbf{h}_{w_2} = \mathbf{h}_{w_4} = \begin{bmatrix} 0 & I_w \\ 0 & 0 \end{bmatrix}$.

Using these equalities, the number of operations in Equation (13) can be simplified as 6 matrix multiplications and 3 matrix additions. Because H_{h_i} and H_{w_i} are deterministic, they can be pre-computed and then pre-stored in memory. Therefore, there is no DCT transform operation required for the computation of Equation (13).

Table 1 Matrices $\mathbf{h}_{h_i}$ and $\mathbf{h}_{w_i}$

Neighboring blocks	Position	$\mathbf{h}_{h_i}$	$\mathbf{h}_{w_i}$
$\mathbf{b}_1$	Lower right	$\begin{bmatrix} 0 & I_{8-h} \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ I_{8-w} & 0 \end{bmatrix}$
$\mathbf{b}_2$	Lower left	$\begin{bmatrix} 0 & I_{8-h} \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & I_w \\ 0 & 0 \end{bmatrix}$
$\mathbf{b}_3$	Upper right	$\begin{bmatrix} 0 & 0 \\ I_h & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ I_{8-w} & 0 \end{bmatrix}$
$\mathbf{b}_4$	Upper left	$\begin{bmatrix} 0 & 0 \\ I_h & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & I_w \\ 0 & 0 \end{bmatrix}$

The computation requirement of Equation (13) is still not efficient enough for the computation of MCD^{-1}. In practical applications, the matrices, $B_1 \sim B_4$, are very likely to be sparse, since they represent the DCT coefficients of the quantization errors of the incoming DCT blocks from the second stage quantizer and the errors are often insignificant. Due to the sparseness of these matrices, Equation (13) can be computed efficiently using some existing sparse matrix manipulation algorithms. Moreover, [20-21] proposed a matrix factorization scheme which was originally proposed in JPEG coding standard [22] to factorize the DCT transform matrices S into the product of some matrices with simple and regular architectures as follows:

$$S = DPB_1B_2EA_1A_2A_3 \tag{14}$$

where D is a diagonal matrix consisting of real entries, P is a permutation matrix, A_i's and B_i's are sparse matrices with entries 1 and –1, and E is also a sparse matrix with real entries. Without taking into account the inherent sparseness of the DCT matrices, this matrix factorization scheme can achieve more than 40% computation saving and does not introduce any distortion.

On the other hand, [23] proposed a scheme to simplify the computation by approximating H_{h_i} and H_{w_i} using rounded matrices with elements which contain only a small number of powers of two. Thus the multiplication operation is simplified as addition + shifting instead of multiplication which leads to reduction of about 81% compared with the pixel-domain computation

In [26], an approach based on shared information in a macroblock was proposed to reduce the computation of DCT-domain inverse motion compensation. As depicted in Fig. 12, instead of sixteen contributing blocks for the four 8×8 target blocks when we treat the target blocks individually, there are only 9 different anchor blocks with five of them are shared among multiple target blocks and four of them are not shared, since the four target blocks belong to the same macroblock and adopt the same motion vector. For example, in Fig. 12, M_i, i = 0 to 3 are the contributing blocks to Q^M and N_i, i = 0 to 3 are the contributing blocks to Q^N, evidently, the target blocks Q^M and Q^N share two anchor blocks, i.e., $M_1 = N_0$ and $M_3 = N_2$. Similarly, it's easy to show that every two adjacent target blocks share two anchor blocks, and the four target blocks share the center anchor block. Using the shared information in a macro block as well as applying some simple pre-processing before and post-processing after the geometric transform mentioned above, about 47 % improvement was reported [26].

3. MULTI-POINT VIDEO TRANSCODING ARCHITECTURES

A multipoint videoconference is a videoconference that involves three or more conference participants. With the rapid growth of video conferencing, the need for multipoint video conferencing will also grow. In the multipoint video conference, the conference participants are connected to a Multipoint Control Unit (MCU) in a central office. A video bridge in the MCU combines the multiple compressed input video streams into a single bit-stream and sends it to the conference participants for presentation. Multi-point video transcoders are required for video bridges to transcode the input video streams into the output video bitstreams with the required bit rates and formats. Various realizations of multi-point video transcoder can be constructed by using the single point transcoding architecture mentioned above.

A straightforward realization of multi-point video transcoder, named Type I, is shown in Fig. 13. With Type I realization, the input data for the video transcoder consists of multiple video streams, which are received from corresponding terminals through heterogeneous network environments and contain encoded data based on H.263 standard. The video stream could be transmitted through either of GSTN, ISDN, and LAN transmission media with various bandwidth requirements. At the first stage, the input bit-streams are written into the input buffers from outer networks respectively. The decision of the input buffer size depends on the specifications of delay and network characteristics. For simplicity and without loss of generality, we assume each video stream is encoded in QCIF format and each conferee can see 4 participants in a CIF frame in a continuous presence fashion. In Type I realization, the received video streams in QCIF are firstly decoded through variable-length decoders (VLD) and then selected and combined into a CIF frame through a number of multiplexers and video combiners, as shown in Fig. 13. For video combining, the QCIF video bitstreams are partially decoded into the DCT coefficients of the video date and then assigned to the corresponding locations in the CIF frame. An example of the video combining of four QCIF frames into a single CIF frame is illustrated in Fig. 14. After these operations, a CIF bit stream is obtained which will require larger data rate. In the next stage, the simplified transcoding architectures proposed in the previous section are adopted to transcode the video data in CIF format to meet the required bit rates. Moreover, the rate control mechanism provides feedback messages to the transcoders for commanding the output data rates according to the contents of output buffers. The output buffers may have different sizes and different control schemes due to the heterogeneous

network environments.

Another realization scheme, called Type II, is shown in Fig. 15. In contrast to the architecture shown in Fig. 13, the input video streams are firstly transcoded in QCIF format and then the transcoded bitstreams are combined into a CIF frame through a number of multiplexers and video combiners. As depicted in Fig. 15, to meet the various bit-rate requirements, more than one transcoder might be required for the same video stream to generate multiple QCIF video data with different bit rates. This problem can be avoided by adopting a scalable video transcoder which can generate multiple video data with different target bitrates in a single bitstream. With this multiresolution feature which is supported in some forth-coming video coding standards such as MPEG-4 and H.263+, only a single video transcoder for each conferee is required to meet multiple channel bandwidth requirements. This type of realization is shown in Fig. 16. The main advantage of the Type III realization is that, since a video bitstream from a participant is often shared by many other participants, this architecture can make full use of the resource sharing property then Type I thus reducing the computation cost drastically.

Note that, the video combining in Type II and Type III realizations is performed in H.263 bitstream level rather than the DCT coefficients combining in Type I. The multiplexing unit for the video combiner is the GOB. Because of the difference of GOB allocations between CIF and QCIF, we need some modifications about the structure of GOB's. It may combine two GOB's of QCIF into a GOB of CIF in series connection.

4. SUMMARY

Video transcoding is an efficient way for rate adaptation and format conversion in networked video applications, especially for bitrate reduction. We discussed various architectures for implementing video transcoders in video conferencing. A straightforward approach to implement video transcoder is to cascade a video decoder followed by a video encoder. This cascaded architecture can avoid the drift problem, while its high complexity is unacceptable in real-time applications. Based on motion vector reuse scheme, a simplified pixel-domain video transcoder achieving significant computation saving can be constructed, though it may not perform as good as cascaded architecture in video quality. The simplified transcoder architecture can also be implemented in the DCT-domain without performing the DCT/IDCT operations. Instead, this approach uses an interpolation method to

estimate the DCT coefficients of a shifted macroblock. Some approaches were proposed to further reduce the complexity.

Three multipoint video transcoding architectures are discussed in this article. Different architectures can lead to different complexities in different applications.

Acknowledgement

The authors would like to thank Profs. Ming-Ting Sun and Jenq-Neng Hwang at University of Washington, and their Ph.D. students: Jeongnam Youn, I-Ming Pao, and Tzong-Der Wu for their kindly assistance in preparing this manuscript.

REFERENCES

[1] Tzong-Der Wu, Jenq-Neng Hwang and Ming-Ting Sun, "Video Transcoding," invited book chapter for Multimedia Image and Video Processing, L. Guan, S.Y. Kung, and J. larsen ed., CRC press.

[2] Ming-Ting Sun and I-Ming Pao, “Multipoint Video Conferencing,” Visual Communication and Image Processing, Marcel Dekker, C.W. Chen and Y.Q. Zhang ed., 1997.

[3] ITU-T Recommendation H.320, “Narrow-band Visual Telephone Systems and Terminal Equipment”.

[4] ITU-T Recommendation H.310, “Broadband Audiovisual Communication Systems and Terminals”.

[5] ITU-T Recommendation H.321, “Adaptation of H.320 Visual Telephone Terminals to B-ISDN Environments”.

[6] ITU-T Recommendation H.322, “Visual Telephone Systems and Terminal Equipment for Local Area Networks Which Provide A Quaranteed Quality of Service”.

[7] ITU-T Recommendation H.323, “Visual Telephone Systems and Terminal Equipment for Local Area Networks Which Provide A Non-Quaranteed Quality of Service”.

[8] ITU-T Recommendation H.324, “Terminal for Low Bit Rate Multimedia Communication”.

[9] ITU-T Recommendation H.261, “Video Codecs for Audiovisual Services at p × 64 kbits”. March 1993.

[10] ITU-T Recommendation H.263, “Video Coding For Low Bit Rate Communication”. May 1997.

[11] ISO/IEC 13818-2 “Generic coding of moving pictures and associated audio”. (MPEG-2), Part 2: Video, Nov. 1993.

[12] ISO/IEC JTC1/SC29/WG11 “Coding of Moving Pictures and Associated Audio MPEG98/W2194”. (MPEG-4), March 1998.

[13] J. Youn, M.-T. Sun, and C.-W. Lin *“*Motion Estimation for High-Performance Transcoders,” *IEEE Trans. Consumer Electronics,* vol. 44, pp. 649-658, Aug. 1998.

[14] Jeongnam Youn, Ming-Ting Sun, and Chia-Wen Lin, "Motion Vector Refinement for High Performance Transcoding," *IEEE Trans. Multimedia*, vol. 1, pp. 30-40, March 1999.

[15] Chia-Wen Lin, Jeongnam Youn, Yung-Chang Chen, and Ming-Ting Sun, "A Study on Video Transcoder Architectures," *IEEE Int. Symp. Consumer Electronics,* Oct. 1998, Taipei, Taiwan.

[16] A. Eleftheriadis and D. Anastassiou, “Constrained and General Dynamic Rate Shaping of Compressed Digital Video,” ICIP ’95, 1995.

[17] H. Sun, W. Kwok and J.W. Zdepski, “Architecture for MPEG Compressed Bitstream Scaling”, *IEEE Trans. Circuits Syst. Video Technol*, vol. 6, No. 2, April 1996.

[18] G. Keesman et al., “Transcoding of MPEG Bitstream,” *Signal Proc.. Image Commun.*, pp. 481-500, 1996.

[19] S.-F. Chang and D. G. Messerschmitt, “Manipulation and Compositing of MC-DCT Compressed Video,” *IEEE Journal Selected Areas in Commun.,* vol. 13, pp. 1-11, Jan. 1995.

[20] M. Merhav and V. Bhaskaran, “Fast Algorithms for DCT-Domain Image Down-Sampling and for Inverse Motion Computation,” *IEEE Trans. Circuits Syst. Video Technol.,* vol. 7, pp. 468-476, Jun. 1997.

[21] M. Merhav and V. Bhaskaran, “Fast Inverse Motion Compensation Algorithms for MPEG-2 and for Partial DCT Information,” *HP Technical Report #HPL-96-53,* 1996.

[22] P. A. A. Assuncao and M. Ghanbari, “Optimal Transcoding of Compressed Video,” in *IEEE International Conference on Image Processing*, Chicago, Oct. 1997.

[23] P. 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.,* Vol. 8, No. 8, pp. 953-567, Dec. 1998.

[24] Y. Arai, T. Agui, and M. Nakajima, "A Fast DCT-SQ Scheme for Images," *Trans. of the IEICE,* E 71(11): 1095, Nov. 1988.

[25] B.-L. Yeo, "Efficient Processing of Compressed Images and Video," *Ph.D. Dissertation,* Princeton University, USA, Jan 1996.

[26] J. Song and B.-L. Yeo, "A Fast DCT Domain Inverse Motion Compensation Algorithm Based on Shard Information in a Macroblock," *IBM Research Report RC 10880,* June 1997.

[27] O. H. Werner, "Generic Quantizer for Transcoding of Hybrid Video," in *Picture Coding Symposium,* Berlin, Sep. 1997

[28] O. H. Werner, "Real-Time Transcoding of MPEG-2 Video Bit Stream," in *International Broadcast Convention,* Amsterdam, pp. 206-301, Sep. 1997.

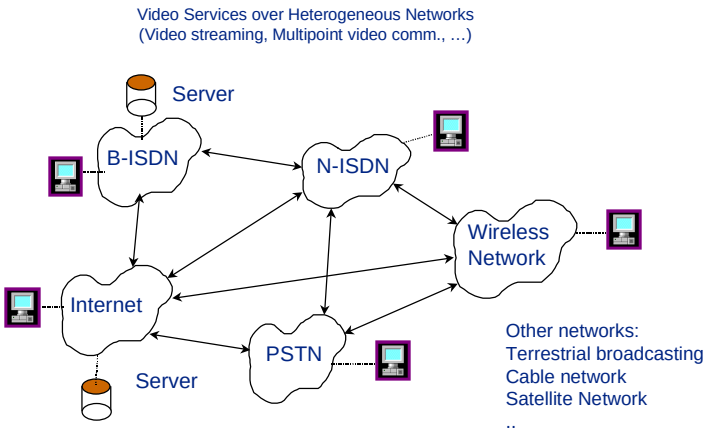


Fig. 1 Visual communication over heterogeneous networks

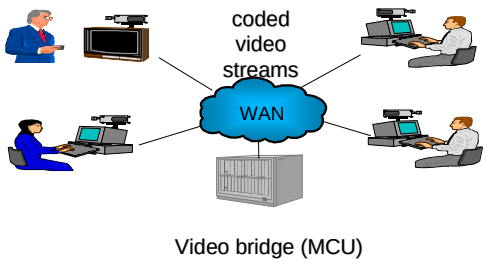


Fig. 2 Centralized multipoint videoconferencing

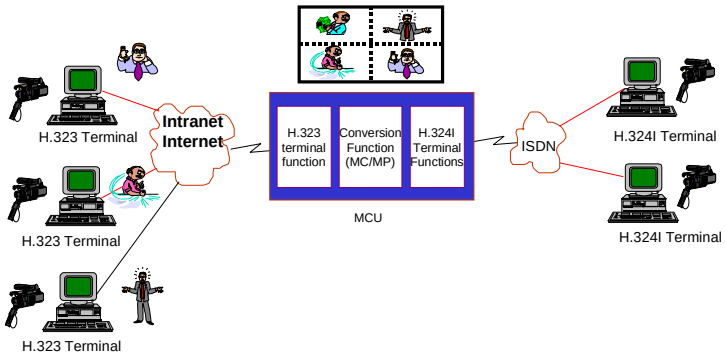


Fig. 3 An example of H.323/H.324 MCU/Gateway

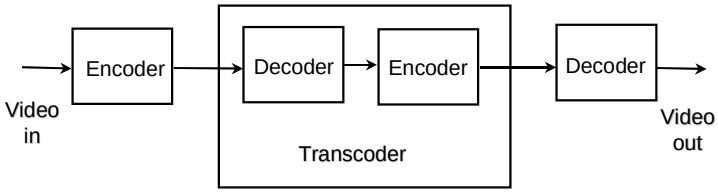


Fig. 4 Cascaded architecture of transcoder

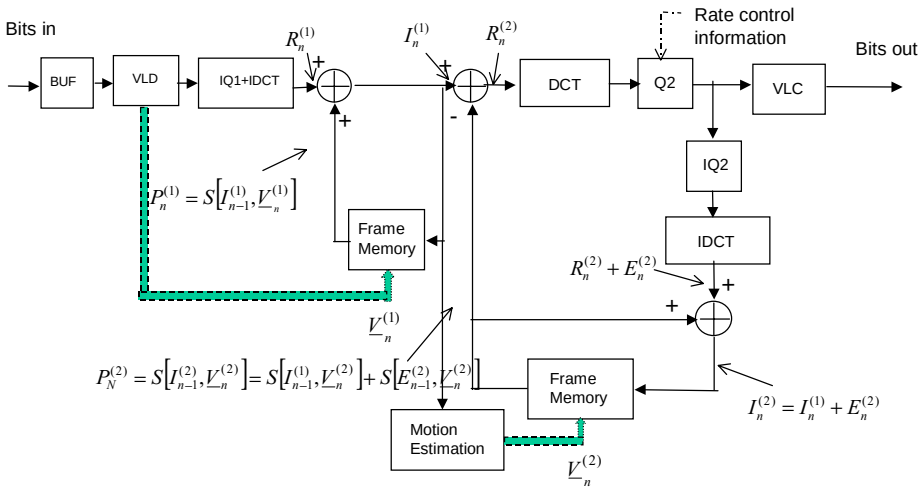


Fig. 5 Direct Implementation of transcoder

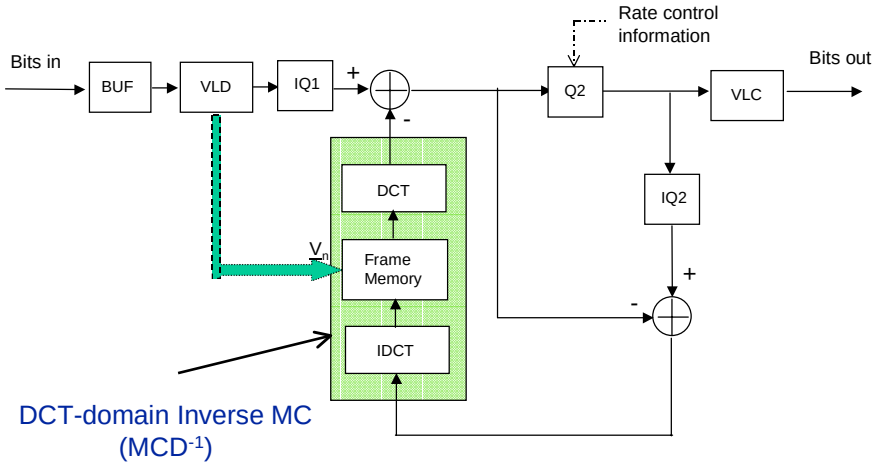


Fig. 6 Simplified architecture of transcoder with motion vector reuse

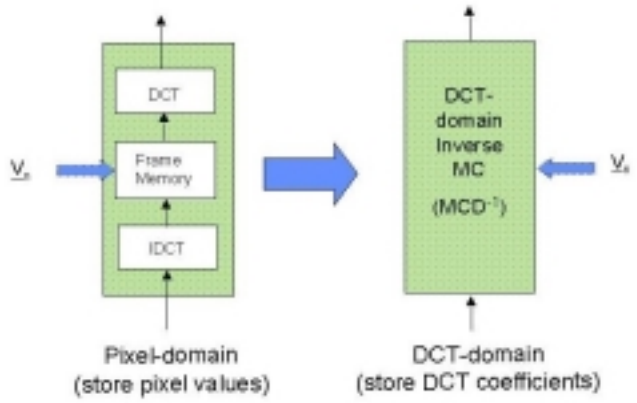


Fig. 7 Pixel-domain and DCT-domain inverse motion compensation

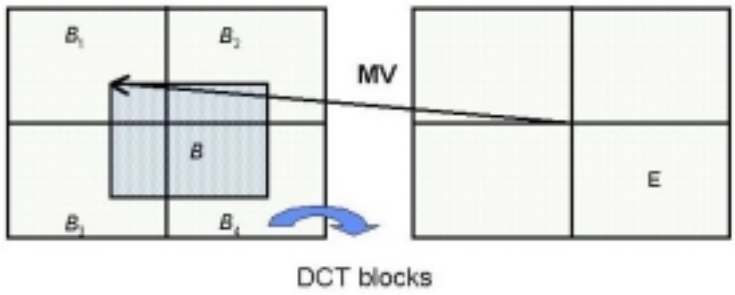


Fig. 8 The illustration of the DCT-domain inverse motion compensation

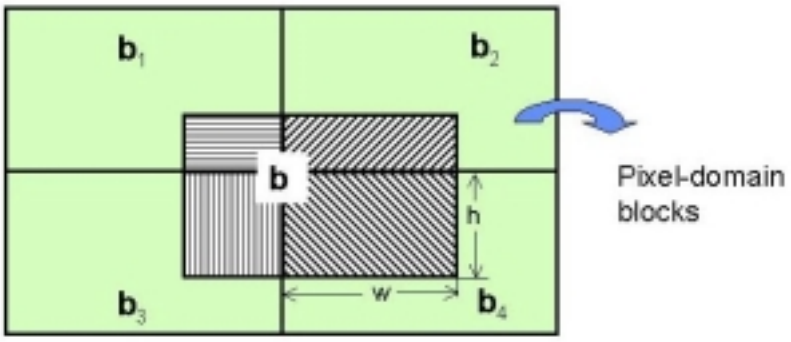


Fig. 9 The relation among the translated block **b** and its four neighboring blocks $\mathbf{b}_1 \sim \mathbf{b}_4$

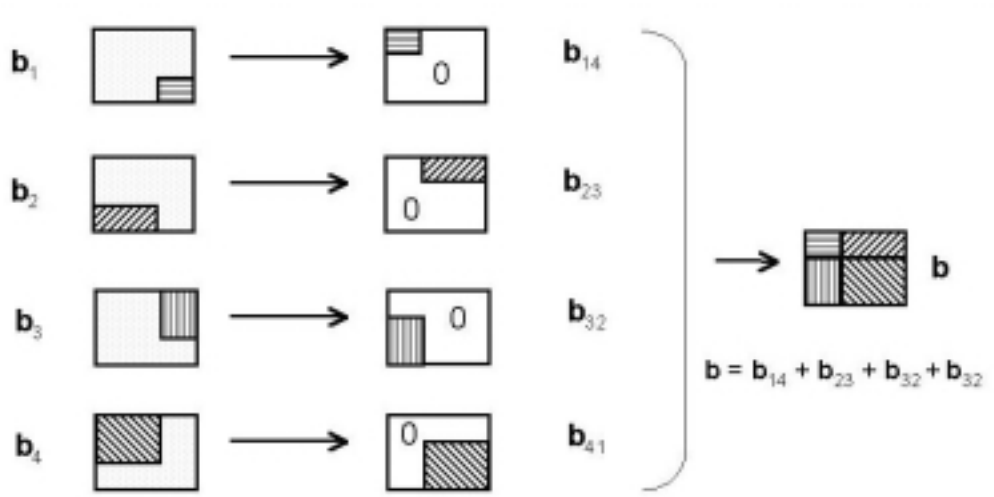


Fig. 10 Formation of the target block **b** from parts of $\mathbf{b}_1 \sim \mathbf{b}_4$ through geometric transform

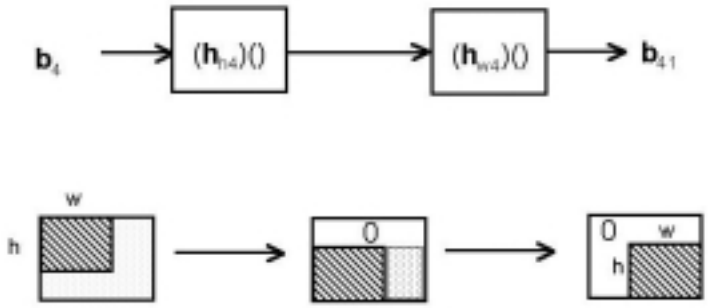


Fig. 11 Illustration of geometric transform

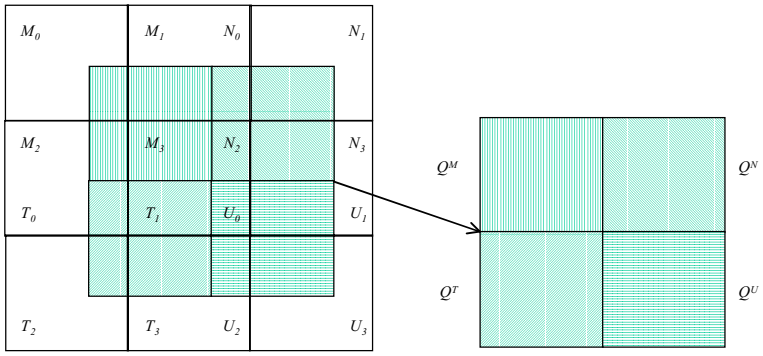


Fig. 12 Prediction of a 16×16 macroblock

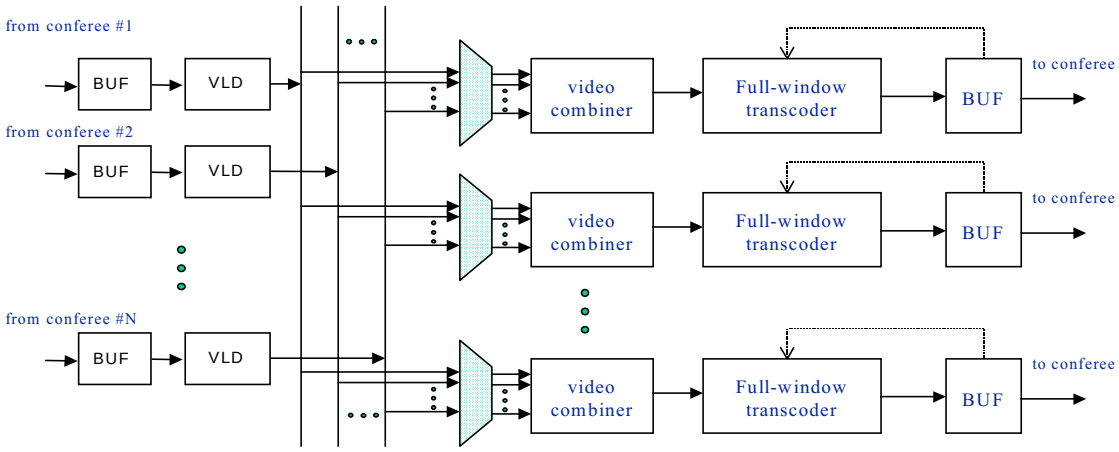
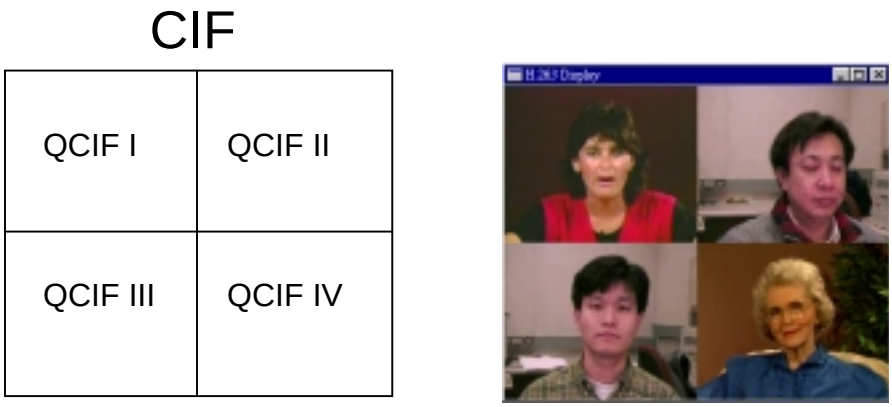


Fig. 13 Type I realization of multi-point video transcoder



(a) (b)

Fig. 14 Video combining of four QCIF frames into a CIF frame (a) sub-window arrangement; (b) a combine CIF image

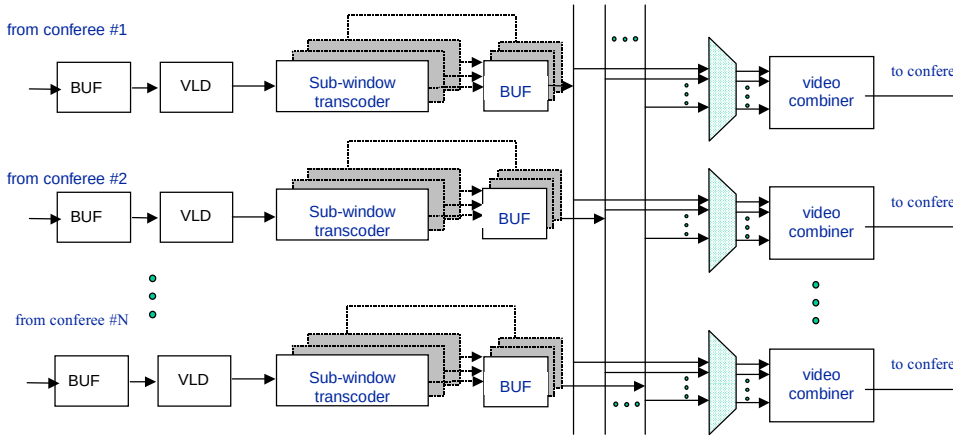


Fig. 15 Type II realization of multi-point video transcoder

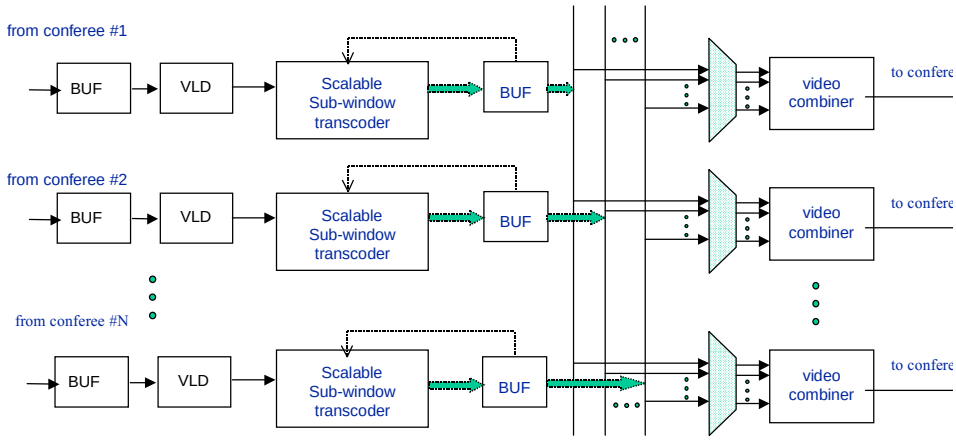


Fig. 16 Type III realization of multi-point video transcoder