during a transition period, both NTSC and DTV service will be simultaneously broadcast on different channels and DTV can only use the taboo channels. This approach allows a smooth transition to DTV, such that the services of the existing NTSC receivers will remain and gradually be phased out of existence in the year 2006. The simulcasting requirement causes some technical difficulties in DTV design. First, the high-quality HDTV program must be delivered in a 6-MHz channel to make efficient use of spectrum and to fit allocation plans for the spectrum assigned to television broadcasting. Second, a low-power and low-interference signal must be used so that simulcasting in the same frequency allocations as current NTSC service does not cause excessive interference with the existing NTSC receiving, since the taboo channels are generally unsuitable for broadcasting an NTSC signal due to high interference. In addition to satisfying the frequency spectrum requirement, the DTV standard has several important features, which allow DTV to achieve interoperability with computers and data communications. The first feature is the adoption of a layered digital system architecture. Each individual layer of the system is designed to be interoperable with other systems at the corresponding layers. For example, the square pixel and progressive scan picture format should be provided to allow computers access to the compression layer or picture layer depending on the capacity of the computers and the ATM-like packet format for the ATM network to access the transport layer. Second, the DTV standard uses a header/descriptor approach to provide maximum flexible operating characteristics. Therefore, the layered architecture is the most important feature of DTV standards. The additional advantage of layering is that the elements of the system can be combined with other technologies to create new applications. The system of DTV standard includes four layers: the picture layer, the compression layer, the transport layer, and the transmission layer.

17.2.2.1 Picture Layer

At the picture layer, the input video formats have been defined. The Executive Committee of the ATSC has approved release of statement regarding the identification of the HDTV and Standard Definition Television (SDTV) transmission formats within the ATSC DTV standards. There are six video formats in the ATSC DTV standard, which are "High Definition Television." These formats are listed in Table 17.1.

The remaining 12 video formats are not HDTV format. These formats represent some improvements over analog NTSC and are referred to as "SDTV." These are listed in Table 17.2.

These definitions are fully supported by the technical specifications for the various formats as measured against the internationally accepted definition of HDTV established in 1989 by the ITU and the definitions cited by the FCC during the DTV standard development process. These formats cover a wide variety of applications, which include motion picture film, currently available HDTV production equipment, the NTSC television standard, and computers such as personal computers and workstations. However, there is no simple technique which can convert images from one pixel

TABLE 17.1
HDTV Formats

Spatial Format (X × Y active pixels)	Aspect Ratio	Temporal Rate (Hz progressive scan)
1920 × 1080 (square pixel)	16:9	23.976/24
		29.97/30
		59.94/60
1280 × 720 (square pixel)	16:9	23.976/24
		29.97/30
		59.94/60

TABLE 17.2
SDTV Formats

Spatial Format (X × Y active pixels)	Aspect Ratio	Temporal Rate (Hz progressive scan)
704 × 480 (CCIR601)	16:9 or 4:3	23.976/24
		29.97/30
		59.94/60
640 × 480 (VGA, square pixel)	4:3	23.976/24
		29.97/30
		59.94/60

format and frame rate to another that achieve interoperability among film and the various worldwide television standards. For example, all low-cost computers use square pixels and progressive scanning, while current television uses rectangular pixels and interlaced scanning. The video industry has paid a lot of attention to developing format-converting techniques. Some techniques such as deinterlacing, down/up-conversion for format conversion have already been developed. It should be noted that the broadcasters, content providers, and service providers can use any one of these DTV format. This results in a difficult problem for DTV receiver manufacturers who have to provide all kinds of DTV receivers to decode all these formats and then to convert the decoded signal to its particular display format. On the other hand, this requirement also gives receiver manufacturers the flexibility to produce a wide variety of products that have different functionality and cost, and the consumers freedom to choose among them.

17.2.2.2 Compression Layer

The raw data rate of HDTV of 1920 × 1080 × 30 × 16 (16 bits per pixel corresponds to 4:2:2 color format) is about 1 Gbps. The function of the compression layer is to compress the raw data from about 1 Gbps to the data rate of approximately 19 Mbps to satisfy the 6-MHz spectrum requirement. This goal is achieved by using the main profile and high level of the MPEG-2 video standard. Actually, during the development of the Grand Alliance HDTV system, many research results were adopted by the MPEG-2 standard at the same time; for example, the support for interlaced video format and the syntax for data partitioning and scalability. The ATSC DTV standard is the first and most important application example of the MPEG-2 standard. The use of MPEG-2 video compression fundamentally enables ATSC DTV devices to interoperate with MPEG-1/2 computer multimedia applications directly at the compressed bitstream level.

17.2.2.3 Transport Layer

The transport layer is another important issue for interoperability. The ATSC DTV transport layer uses the MPEG-2 system transport stream syntax. It is a fully compatible subset of the MPEG-2 transport protocol. The basic function of the transport layer is to define the basic format of data packets. The purposes of packetization include:

- Packaging the data into the fixed-size cells or packets for forward error correction (FEC) encoding to protect the bit error due to the communication channel noise;
- Multiplexing the video, audio, and data of a program into a bitstream;
- Providing time synchronization for different media elements;
- Providing flexibility and extensibility with backward compatibility.

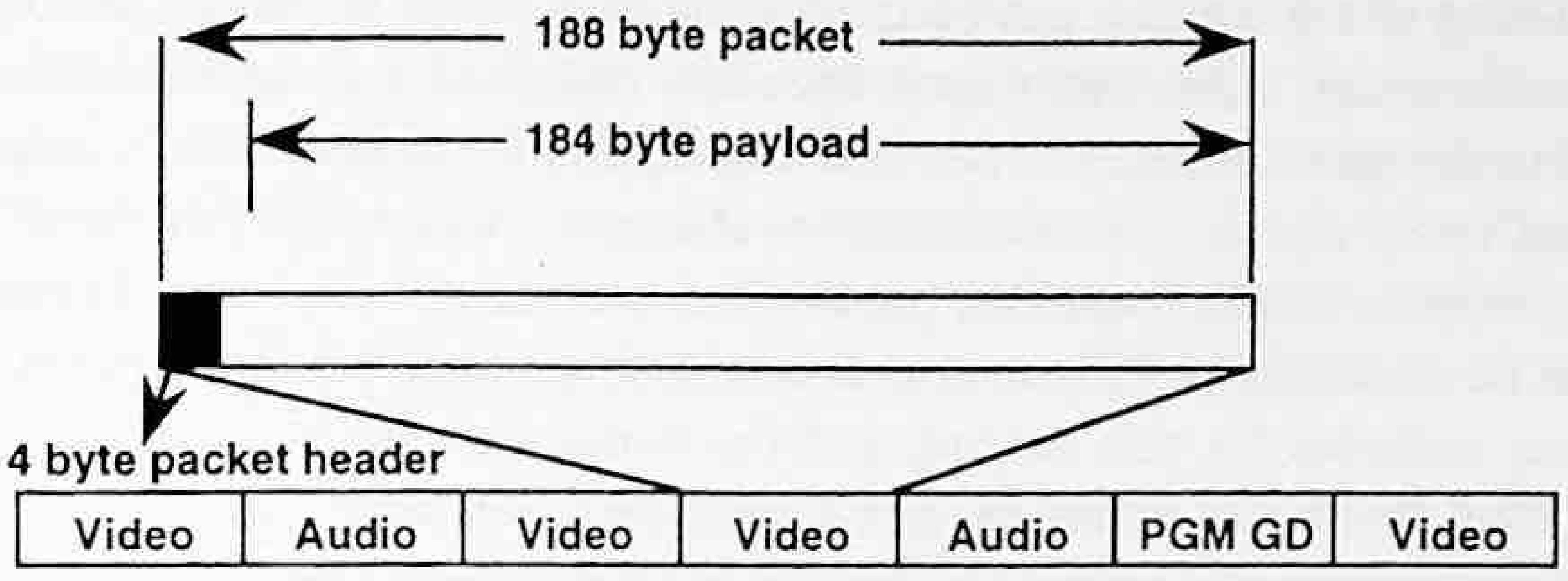


FIGURE 17.1 Packet structure of ATSC DTV transport layer.

The transport layer of ATSC DTV uses a fixed-length packet. The packet size is 188 bytes consisting of 184 bytes of payload and 4 bytes of header. Within the packet header, the13-bit packet identifier (PID) is used to provide the important capacity to combine the video, audio, and ancillary data stream into a single bitstream as shown in Figure 17.1. Each packet contains only a single type of data (video, audio, data, program guide, etc.) identified by the PID.

This type of packet structure packetizes the video, audio, and auxiliary data separately. It also provides the basic multiplexing function that produces a bitstream including video, five-channel surround-sound audio, and an auxiliary data capacity. This kind of transport layer approach also provides complete flexibility to allocate channel capacity to achieve any mix among video, audio, and other data services. It should be noted that the selection of 188-packet length is a trade-off between reducing the overhead due to the transport header and increasing the efficiency of error correction. Also, one ATSC DTV packet can be completely encapsulated with its header within four ATM packets by using 1 AAL byte per ATM header leaving 47 usable payload bytes times 4, for 188 bytes. The details of the transport layer is discussed in the chapter on MPEG systems.

Transmission Layer — The function of the transmission layer is to modulate the transport bitstream into a signal that can be transmitted over the 6-MHz analog channel. The ATSC DTV system uses a trellis-coded eight-level vestigial sideband (8-VSB) modulation technique to deliver approximately 19.3 Mbps in the 6-MHz terrestrial simulcast channel. VSB modulation inherently requires only processing the in-phase signal sampled at the symbol rate, thus reducing the complexity of the receiver, and ultimately the cost of implementation. The VSB signal is organized in a data frame that provides a training signal to facilitate channel equalization for removing multipath distortion. However, from several field-test results, the multipath distortion is still a serious problem of terrestrial simulcast receiving. The frame is organized into segments each with 832 symbols. Each transmitted segment consists of one synchronization byte (four symbols), 187 data bytes, and 20 R-S parity bytes. This corresponds to a 188-byte packet, which is protected by 20-byte R-S code. Interoperability at the transmission layer is required by different transmission media applications. The different media use different modulation techniques now, such as QAM for cable and QPSK for satellite. Even for terrestrial transmission, European DVB systems use OFDM transmission. The ATV receivers will not only be designed to receive terrestrial broadcasts, but also the programs from cable, satellite, and other media.

17.3 TRANSCODING WITH BITSTREAM SCALING

17.3.1 Background

As indicated in the previous chapters, digital video signals exist everywhere in the format of compressed bitstreams. The compressed bitstreams of video signals are used for transmission and storage through different media such as terrestrial TV, satellite, cable, the ATM network, and the

Internet. The decoding of a bitstream can be implemented in either hardware or software. However, for high-bit-rate compressed video bitstreams, specially designed hardware is still the major decoding approach due to the speed limitation of current computer processors. The compressed bitstream as a new format of video signal is a revolutionary change to video industry since it enables many applications. On the other hand, there is a problem of bitstream conversion. Bitstream conversion or transcoding can be classified as bit rate conversion, resolution conversion, and syntax conversion. Bit rate conversion includes bit rate scaling and the conversion between constant bit rate (CBR) and variable bit rate (VBR). Resolution conversion includes spatial resolution conversion and temporal resolution conversion. Syntax conversion is needed between different compression standards such as JPEG, MPEG-1, MPEG-2, H.261, and H.263. In this section, we will focus on the topic of bit rate conversion, especially on bit rate scaling since it finds wide application and readers can extend the idea to other kinds of transcoding. Also, we limit ourselves to focus on the problem of scaling an MPEG CBR-encoded bitstream down to a lower CBR. The other kind of transcoding, down-conversion decoder, will be presented in a separate section.

The basic function of bitstream scaling may be thought of as a black box, which passively accepts a precoded MPEG bitstream at the input and produces a scaled bitstream, which meets new constraints that are not known *a priori* during the creation of the original precoded bitstream. The bitstream scaler is a transcoder, or filter, that provides a match between an MPEG source bitstream and the receiving load. The receiving load consists of the transmission channel, the destination decoder, and perhaps a destination storage device. The constraint on the new bitstream may be bound by a variety of conditions. Among them are the peak or average bit rate imposed by the communications channel, the total number of bits imposed by the storage device, and/or the variation of bit usage across pictures due to the amount of buffering available at the receiving decoder.

While the idea of bitstream scaling has many concepts similar to those provided by the various MPEG-2 scalability profiles, the intended applications and goals differ. The MPEG-2 scalability methods (data partitioning, SNR scalability, spatial scalability, and temporal scalability) are aimed at providing encoding of source video into multiple service grades (that are predefined at the time of encoding) and multitiered transmission for increased signal robustness. The multiple bitstreams created by MPEG-2 scalability are hierarchically dependent in such a way that by decoding an increasing number of bitstreams, higher service grades are reconstructed. Bitstream scaling methods, in contrast, are primarily decoder/transcoder techniques for converting an existing precoded bitstream to another one that meets new rate constraints. Several applications that motivate bitstream scaling include the following:

1. Video-On-Demand — Consider a video-on-demand (VOD) scenario wherein a video file server includes a storage device containing a library of precoded MPEG bitstreams. These bitstreams in the library are originally coded at high quality (e.g., studio quality). A number of clients may request retrieval of these video programs at one particular time. The number of users and the quality of video delivered to the users are constrained by the outgoing channel capacity. This outgoing channel, which may be a cable bus or an ATM trunk, for example, must be shared among the users who are admitted to the service. Different users may require different levels of video quality, and the quality of a respective program will be based on the fraction of the total channel capacity allocated to each user. To accommodate a plurality of users simultaneously, the video file server must scale the stored precoded bitstreams to a reduced rate before it is delivered over the channel to respective users. The quality of the resulting scaled bitstream should not be significantly degraded compared with the quality of a hypothetical bitstream so obtained by coding the original source material at the reduced rate. Complexity cost is not such a critical factor because only the file server has to be equipped with the bitstream scaling hardware, not every user. Presumably, video service providers would be willing to pay a high cost for delivering the possible highest-quality video at a prescribed bit rate.

As an option, a sophisticated video file server may also perform scaling of multiple original precoded bitstreams jointly and statistically multiplex the resulting scaled VBR bitstreams into the channel. By scaling the group of bitstreams jointly, statistical gains can be achieved. These statistical gains can be realized in the form of higher and more uniform picture quality for the same channel capacity. Statistical multiplexing over a DirecTv transponder (Isnardi, 1993) is one example of an application of video statistical multiplexing.

2. Trick-play Track on Digital VTRs — In this application, the video bitstream is scaled to create a sidetrack on video tape recorders (VTRs). This sidetrack contains very coarse quality video sufficient to facilitate trick-modes on the VTR (e.g., FF and REW at different speeds). Complexity cost for the bitstream scaling hardware is of significant concern in this application since the VTR is a mass consumer item subject to mass production.
3. Extended-Play Recording on Digital VTRs — In this application, video is broadcast to users' homes at a certain broadcast quality (~6 Mbps for standard-definition video and ~24 Mbps for high-definition video). With a bitstream scaling feature in their VTRs, users may record the video at a reduced rate, akin to extended-play (EP) mode on today's VHS recorders, thereby recording a greater duration of video programs onto a tape at lower quality. Again, hardware complexity costs would be a major factor here.

17.3.2 Basic Principles of Bitstream Scaling

As described previously, the idea of scaling an MPEG-2-compressed bitstream down to a lower bit rate is initiated by several applications. One problem is the criteria that should be used to judge the performance of an architecture that can reduce the size or rate of an MPEG-compressed bitstream. Two basic principles of bitstream scaling are (1) the information in the original bitstream should be exploited as much as possible, and (2) the resulting image quality of the new bitstream with a lower bit rate should be as close as possible to a bitstream created by coding the original source video at the reduced rate. Here, we assume that for a given rate the original source is encoded in an optimal way. Of course, the implementation of hardware complexity also has to be considered. Figure 17.2 shows a simplified encoding structure of MPEG encoding in which the rate control mechanism is not shown.

In this structure, a block of image data is first transformed to a set of coefficients; the coefficients are then quantized with a quantizer step which is decided by the given bit rate budget, or number of bits assigned to this block. Finally, the quantized coefficients are coded in variable-length coding to the binary format, which is called the bitstream or bits.

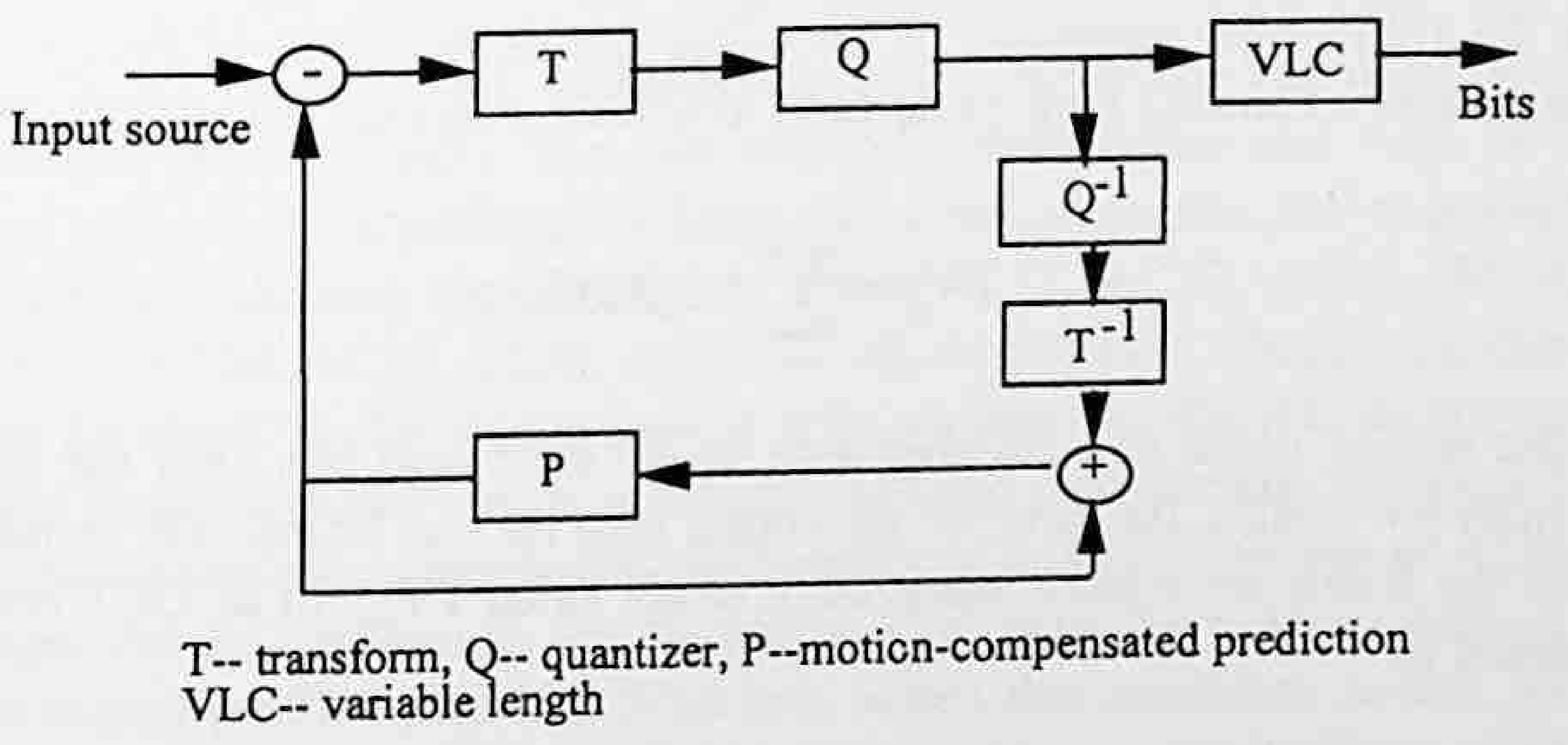

Input source
-
T
Q
VLC
Bits
Q^{-1}
T^{-1}
P
+
T-- transform, Q-- quantizer, P--motion-compensated prediction
VLC-- variable length

FIGURE 17.2 Simplified encoder structure. T = transform, Q = quantizer, P = motion-compensated prediction, VLC = variable length.

From this structure it is obvious that the performance of changing the quantizer step will be better than cutting higher frequencies when the same amount of rate needs to be reduced. In the original bitstream the coefficients are quantized with finer quantization steps which are optimized at the original high rate. After cutting the coefficients of higher frequencies, the rest of the coefficients are not quantized with an optimal quantizer. In the method of requantization all coefficients are requantized with an optimal quantizer which is determined by the reduced rate; the performance of the requantization method must be better than the method of cutting high frequencies to reach the reduced rate. The theoretical analysis is given in Section 17.3.4.

In the following, several different architectures that accomplish the bitstream scaling are discussed. The different methods have varying hardware implementation complexities; each has its own degree of trade-off between required hardware and resulting image quality.

17.3.3 Architectures of Bitstream Scaling

Four architectures for bitstream scaling are discussed. Each of the scaling architectures described has its own particular benefits that are suitable for a particular application.

Architecture 1:	The bitstream is scaled by cutting high frequencies.
Architecture 2:	The bitstream is scaled by requantization.
Architecture 3:	The bitstream is scaled by reencoding the reconstructed pictures with motion vectors and coding decision modes extracted from the original high-quality bitstream.
Architecture 4:	The bitstream is scaled by reencoding the reconstructed pictures with motion vectors extracted from the original high-quality bitstream, but new coding decisions are computed based on reconstructed pictures.

Architectures 1 and 2 are considered for VTR applications such as trick-play modes and EP recording. Architectures 3 and 4 are considered for and other applicable StatMux scenarios.

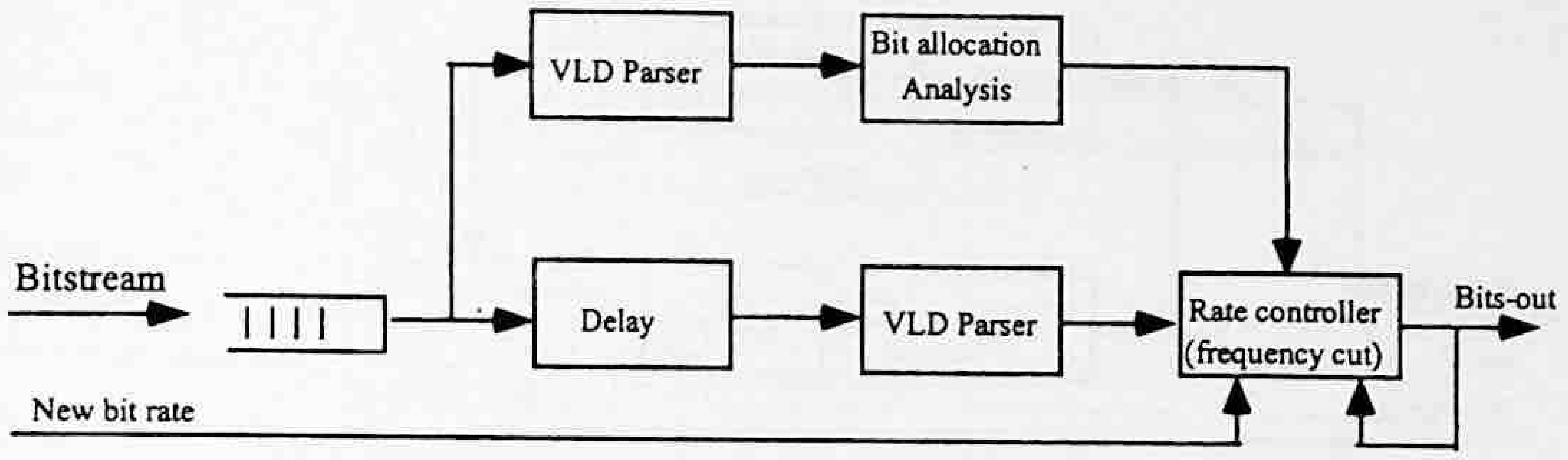
17.3.3.1 Architecture 1: Cutting AC Coefficients

A block diagram illustrating architecture 1 is shown in Figure 17.3a. The method of reducing the bit rate in architecture 1 is based on cutting the higher-frequency coefficients. The incoming precoded CBR stream enters a decoder rate buffer. Following the top branch leading from the rate buffer, a VLD is used to parse the bits for the next frame in the buffer to identify all the variable-length codewords that correspond to ac coefficients used in that frame. No bits are removed from the rate buffer. The codewords are not decoded, but just simply parsed by the VLD parser to determine codeword lengths. The bit allocation analyzer accumulates these ac bit counts for every macroblock in the frame and creates an ac bit usage profile as shown in Figure 17.3(b). That is, the analyzer generates a running sum of ac DCT coefficient bits on a macroblock basis:

$$PV_N = \sum AC\_BITS, \tag{17.1}$$

where PV_N is the profile value of a running sum of AC codeword bits until the macroblock N. In addition, the analyzer counts the sum of all coded bits for the frame, TB (total bits). After all macroblocks for the frame have been analyzed, a target value $\mathrm{TV_{AC}}$, of ac DCT coefficient bits per frame is calculated as

$$TV_{AC} = PV_{LS} - \alpha * TB - B_{EX}, \tag{17.2}$$



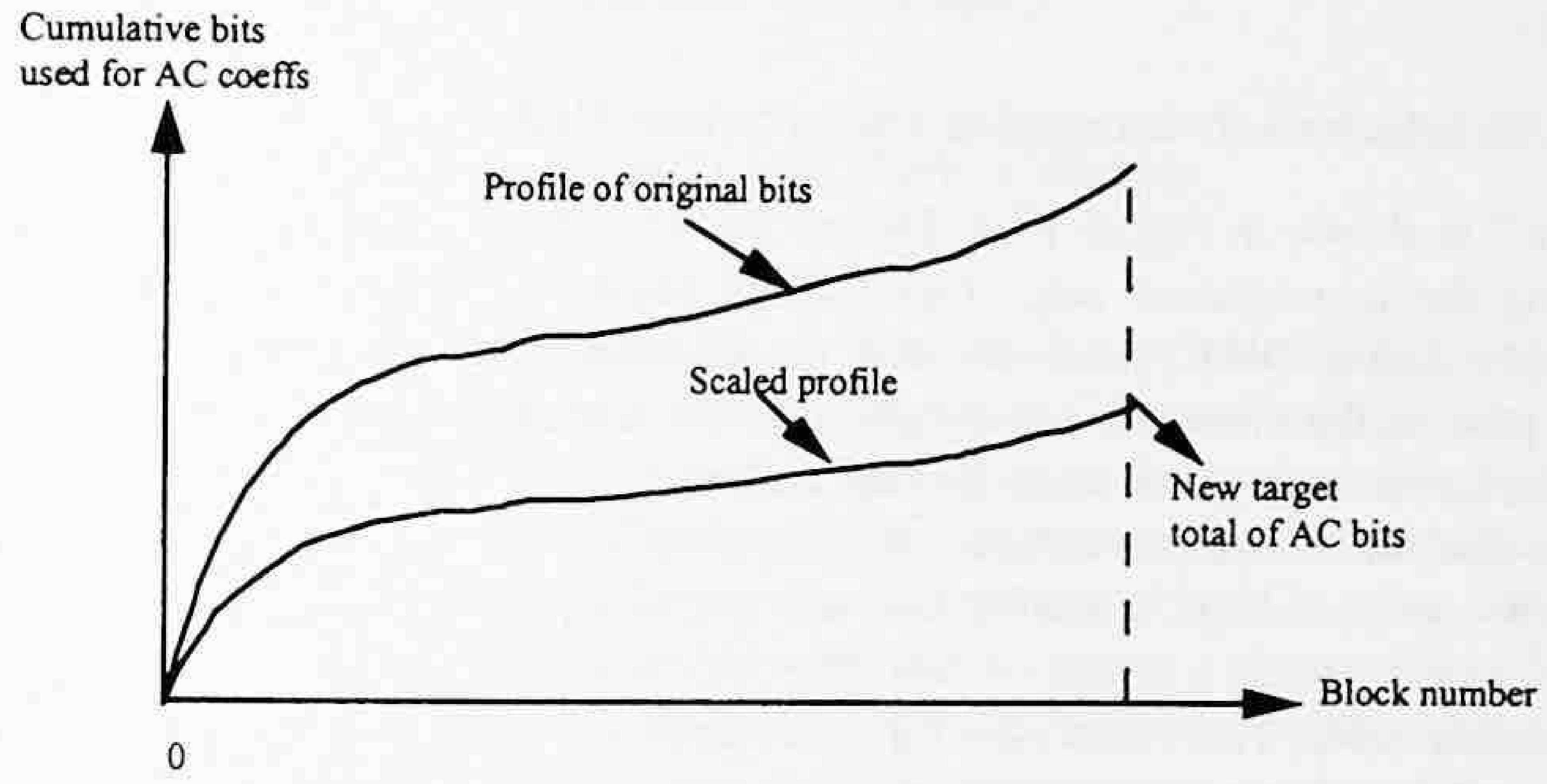


FIGURE 17.3 (a) Architecture 1: cutting high frequencies. (b) Profile map.

where TV_{AC} is the target value of AC codeword bits per frame, PV_{LS} is the profile value at the last macroblock, α is the percentage by which the preencoded bitstream is to be reduced, TB is the total bits, and B_{EX} is the amount of bits by which the previous frame missed its desired target. The profile value of AC coefficient bits is scaled by the factor TV_{AC}/PV_{LS}. Multiplying each PV_N performs scaling by that factor to generate the linearly scaled profile shown in Figure 17.3(b). Following the bottom branch from the rate buffer, a delay is inserted equal to the amount of time required for the top branch analysis processing to be completed for the current frame. A second VLD parser accesses and removes all codeword bits from the buffer and delivers them to a rate controller. The rate controller receives the scaled target bit usage profile for the amount of ac bits to be used within the frame. The rate controller has memory to store all coefficients associated with the current macroblock it is operating on. All original codeword bits at a higher level than ac coefficients (i.e., all fixed-length header codes, motion vector codes, macroblock type codes, etc.) are held in memory and will be remultiplexed with all AC codewords in that macroblock that have not been excised to form the outgoing scaled bitstream. The rate controller determines and flags in the macroblock codeword memory which AC codewords to keep and which to excise. AC codewords are accessed from the macroblock codeword memory in the order $AC11$, $AC12$, $AC13$, $AC14$, $AC15$, $AC16$, $AC21$, $AC22$, $AC23$, $AC24$, $AC25$, $AC26$, $AC31$, $AC32$, $AC33$, etc., where $ACij$ denotes the ith AC codewords from jth block in the macroblock if it is present. As the AC codewords are accessed from memory, the respective codeword bits are summed and continuously compared with the scaled profile value to the current macroblock, less the number of bits for insertion of EOB (end-of-block) codewords. Respective AC codewords are flagged as kept until the running sum of AC codewords bits exceeds the scaled profile value less EOB bits. When this condition is met, all remaining AC codewords are marked for being excised. This process continues until all macroblocks have their kept codewords reassembled to form the scaled bitstream.

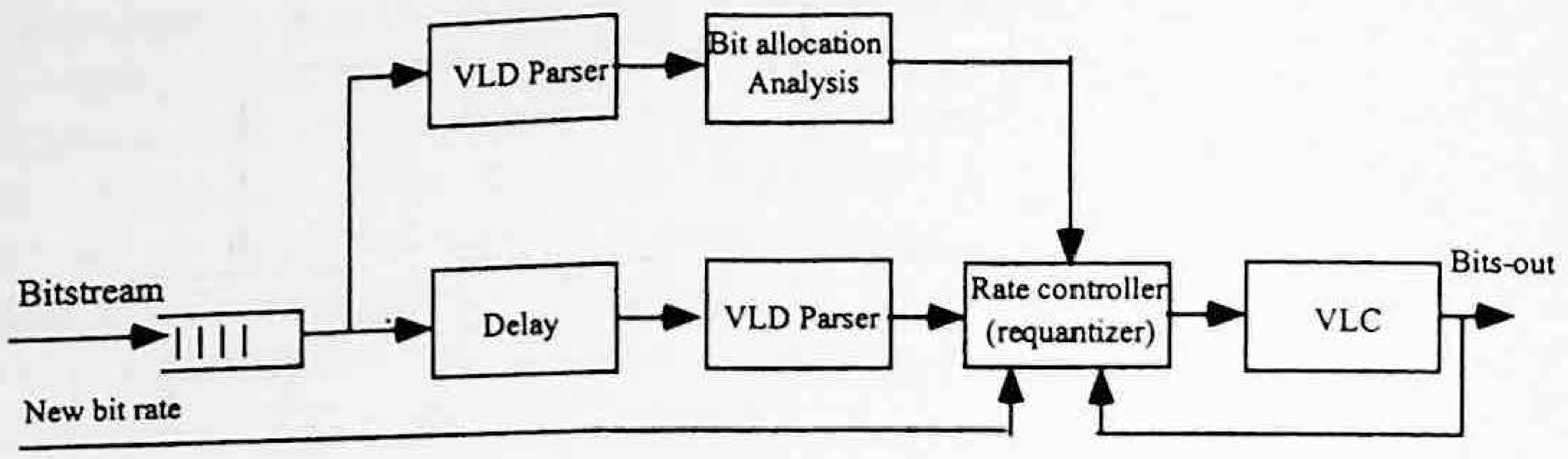


FIGURE 17.4 Architecture 2: increasing quantization step.

17.3.3.2 Architecture 2: Increasing Quantization Step

Architecture 2 is shown in Figure 17.4. The method of bitstream scaling in architecture 2 is based on increasing the quantization step. This method requires additional dequantizer/quantizer and variable-length coding (VLC) hardware over the first method. Like the first method, it also makes a first VLD pass on the bitstream and obtains a similar scaled profile of target cumulative codeword bits vs. macroblock count to be used for rate control.

The rate control mechanism differs from this point on. After the second-pass VLD is made on the bitstream, quantized DCT coefficients are dequantized. A block of finely quantized DCT coefficients is obtained as a result of this. This block of DCT coefficients is requantized with a coarser quantizer scale. The value used for the coarser quantizer scale is determined adaptively by making adjustments after every macroblock so that the scaled target profile is tracked as we progress through the macroblocks in the frame:

$$Q_N = Q_{NOM} + G * \left(\sum_{N-1} \left(BU - PV_{N-1} \right) \right), \tag{17.3}$$

where Q_N is the quantization factor for macroblock N, Q_{NOM} is an estimate of the new nominal quantization factor for the frame, $\sum_{N-1} BU$ is the cumulative amount of coded bits up to macroblock $N-1$, and G is a gain factor which controls how tightly the profile curve is tracked through the picture. Q_{NOM} is initialized to an average guess value before the very first frame, and updated for the next frame by setting it to Q_{LS} (the quantization factor for the last macroblock) from the frame just completed. The coarsely requantized block of DCT coefficients is variable-length-coded to generate the scaled bitstream. The rate controller also has provisions for changing some macroblock-layer codewords, such as the macroblock-type and coded-block pattern to ensure a legitimate scaled bitstream that conforms to MPEG-2 syntax.

17.3.3.3 Architecture 3: Reencoding with Old Motion Vectors and Old Decisions

The third architecture for bitstream scaling is shown in Figure 17.5. In this architecture, the motion vectors and macroblock coding decision modes are first extracted from the original bitstream, and at the same time the reconstructed pictures are obtained from the normal decoding procedure. Then the scaled bitstream is obtained by reencoding the reconstructed pictures using the old motion vectors and macroblock decisions from the original bitstream. The benefits obtained from this architecture compared with full decoding and reencoding is that no motion estimation and decision computation is needed.

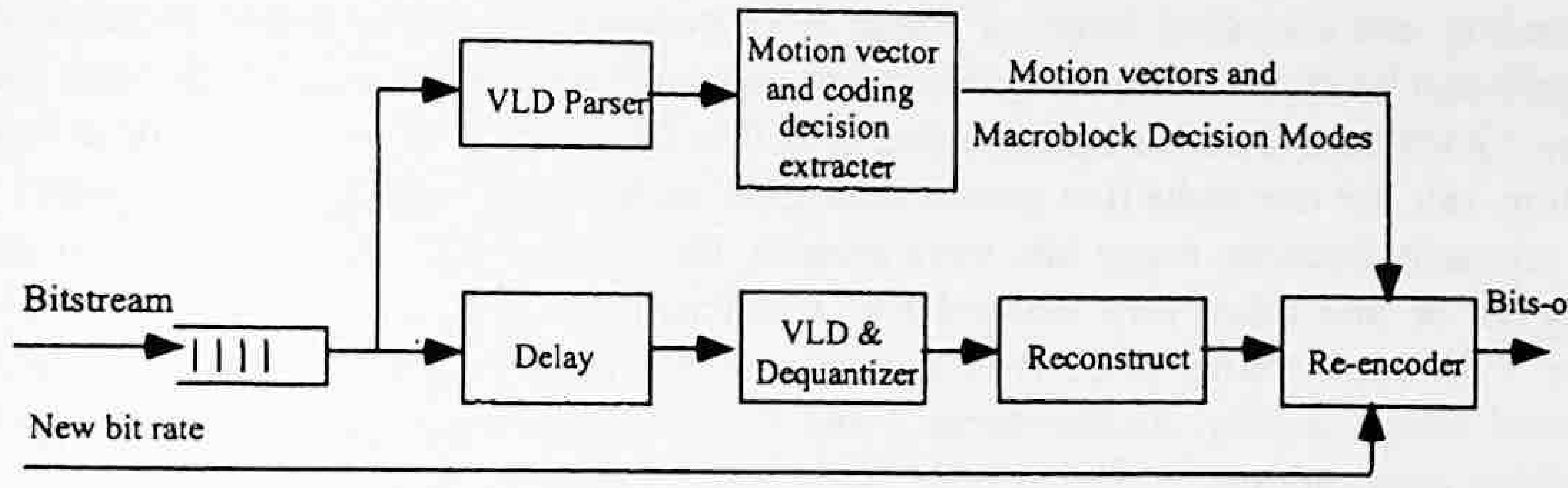

Bitstream
VLD Parser
Motion vector and coding decision extracter
Motion vectors and Macroblock Decision Modes
Delay
VLD & Dequantizer
Reconstruct
Re-encoder
Bits-out
New bit rate

FIGURE 17.5 Architecture 3.

17.3.3.4 Architecture 4: Reencoding with Old Motion Vectors and New Decisions

Architecture 4 is a modified version of architecture 3 in which new macroblock decision modes are computed during reencoding based on reconstructed pictures. The scaled bitstream created this way is expected to yield an improvement in picture quality because the decision modes obtained from the high-quality original bitstream are not optimal for reencoding at the reduced rate. For example, at higher rates the optimal mode decision for a macroblock is more likely to favor bidirectional field motion compensation over forward frame motion compensation. But at lower rates, only the opposite decision may be true. In order for the reencoder to have the possibility of deciding on new macroblock coding modes, the entire pool of motion vectors of every type must be available. This can be supplied by augmenting the original high-quality bitstream with ancillary data containing the entire pool of motion vectors during the time it was originally encoded. It could be inserted into the user data every frame. For the same original bit rate, the quality of an original bitstream obtained this way is degraded compared with an original bitstream obtained from architecture 3 because the additional overhead required for the extra motion vectors steals away bits for actual encoding. However, the resulting scaled bitstream is expected to show quality improvement over the scaled bitstream from architecture 3 if the gains from computing new and more accurate decision modes can overcome the loss in original picture quality. Table 17.3 outlines the hardware complexity savings of each of the three proposed architectures as compared with full decoding and reencoding.

17.3.3.5 Comparison of Bitstream Scaling Methods

We have described four architectures for bitstream scaling which are useful for various applications as described in the introduction. Among the four architectures, architectures 1 and 2 do not require

TABLE 17.3
Hardware Complexity Savings over Full Decoding/Reencoding

Coding Method	Hardware Complexity Savings
Architecture 1	No decoding loop, no DCT/IDCT, no frame store memory, no encoding loop, no quantizer/dequantizer, no motion compensation, no VLC, simplified rate control
Architecture 2	No decoding loop, no DCT/IDCT, no frame store memory, no encoding loop, no motion compensation, simplified rate control
Architecture 3	No motion estimation, no macroblock coding decisions
Architecture 4	No motion estimation

entire decoding and encoding loops or frame store memory for reconstructed pictures, thereby saving significant hardware complexity. However, video quality tends to degrade through the group of pictures (GOP) until the next I-picture due to drift in the absence of decoder/encoder loops. For large scaling, say, for rate reduction greater than 25%, architecture 1 produces poor-quality blocky pictures, primarily because many bits were spent in the original high-quality bitstream on finely quantizing the dc and other very low-order ac coefficients. Architecture 2 is a particularly good choice for VTR applications since it is a good compromise between hardware complexity and reconstructed image quality. Architectures 3 and 4 are suitable for VOD server applications and other StatMux applications.

17.3.4 Analysis

In this analysis, we assume that the optimal quantizer is obtained by assigning the number of bits according to the variance or energy of the coefficients. It is slightly different from MPEG standard which will be explained later, but the principal concept is the same and the results will hold for the MPEG standard. We first analyze the errors caused by cutting high coefficients and increasing the quantizer step. The optimal bit assignment is given by Jayant and Noll (1984):

$$R_{k0} = R_{av0} + \frac{1}{2}\log_2 \frac{\sigma_k^2}{\left(\prod_{i=0}^{N-1}\sigma_i^2\right)^{1/N}}, \quad k = 0, 1, \ldots, N-1, \tag{17.4}$$

where N is the number of coefficients in the block, R_{k0} is the number of bits assigned to the kth coefficient, R_{av0} is the average number of bits assigned to each coefficient in the block, i.e., $R_{T0} = N \cdot R_{av0}$, is the total bits for this block under a certain bit rate, and σ_k^2 is the variance of kth coefficient. Under the optimal bit assignment (17.4), the minimized average quantizer error, σ_{q0}^2, is

$$\sigma_{q0}^2 = \frac{1}{N}\sum_{k=1}^{N-1}\sigma_{qk}^2 = \frac{1}{N}\sum_{k=1}^{N-1}2^{-2R_{k0}}\cdot\sigma_k^2, \tag{17.5}$$

where σ_{qk}^2 is the quantizer error of kth coefficient. According to Equation 17.4, we have two major methods to reduce the bit rate, cutting high coefficients or decreasing the R_{av}, i.e., increasing the quantizer step. We are now analyzing the effects on the reconstructed errors caused by the method of cutting high-order coefficients. Assume that the number of the bits assigned to the block is reduced from R_{T0} to R_{T1}. Then the bits to be reduced, ΔR_1, are equal to $R_{T0} - R_{T1}$.

In the case of cutting high frequencies, say, the number of coefficients is reduced from N to M, then

$$R_{k0} = 0 \text{ for } K < M, \text{ and } \Delta R_1 = R_{T0} - R_{T1} = \sum_{k=M}^{N-1} R_{k0}. \tag{17.6}$$

the quantizer error increased due to the cutting is

$$\Delta\sigma_{q1}^2 = \sigma_{q1}^2 - \sigma_{q0}^2 = \frac{1}{N}\left(\sum_{k=0}^{M-1}2^{-2R_{k0}}\cdot\sigma_k^2 + \sum_{k=M}^{N-1}\sigma_k^2 - \sum_{k=0}^{N-1}2^{-2R_{k0}}\cdot\sigma_k^2\right) \tag{17.7}$$

$$= \frac{1}{N}\left(\sum_{k=M}^{N-1}\sigma_k^2 - \sum_{k=M}^{N-1}2^{-2R_{k0}}\cdot\sigma_k^2\right)$$

$$= \frac{1}{N}\sum_{k=M}^{N-1}\left(1-2^{-2R_{k0}}\right)\cdot\sigma_k^2,$$

where σ_{q1}^2 is the quantizer error after cutting the high frequencies.

In the method of increasing quantizer step, or decreasing the average bits, from R_{av0} to R_{av2}, assigned to each coefficient, the number of bits reduced for the block is

$$\Delta R_2 = R_{T0} - R_{T2} = N\cdot\left(R_{av0} - R_{av2}\right) \tag{17.8}$$

and the bits assigned to each coefficient become now

$$R_{k2} = R_{av2} + \frac{1}{2}\log_2\frac{\sigma_k^2}{\left(\prod_{i=0}^{N-1}\sigma_i^2\right)^{1/N}}, \quad k = 0, 1, \ldots, N-1, \tag{17.9}$$

The corresponding quantizer error increased by the cutting bits is

$$\begin{aligned}\Delta\sigma_{q2}^2 = \sigma_{q2}^2 - \sigma_{q0}^2 &= \frac{1}{N}\left(\sum_{k=0}^{N-1}2^{-2R_{k2}}\cdot\sigma_k^2 - \sum_{k=0}^{N-1}2^{-2R_{k0}}\cdot\sigma_k^2\right) \\ &= \frac{1}{N}\sum_{k=0}^{N-1}\left(2^{-2R_{k2}} - 2^{-2R_{k0}}\right)\cdot\sigma_k^2,\end{aligned} \tag{17.10}$$

where σ_{q2}^2 is the quantizer error at the reduced bit rate.

If the same number of bits is reduced, i.e., $\Delta R_1 = \Delta R_2$, it is obvious that $\Delta\sigma_{q2}^2$ is smaller than $\Delta\sigma_{q1}^2$ since σ_{q2}^2 is the minimized value at the reduced rate. This implies that the performance of changing the quantizer step will be better than cutting higher frequencies when the same amount of rate needs to be reduced. It should be noted that in the MPEG video coding, more sophisticated bit assignment algorithms are used. First, different quantizer matrices are used to improve the visual perceptual performance. Second, different VLC tables are used to code the DC values and the AC transform coefficients and the run-length coding is used to code the pairs of the zero-run length and the values of amplitudes. However, in general, the bits are still assigned according to the statistical model that indicates the energy distribution of the transform coefficients. Therefore, the above theoretical analysis will hold for the MPEG video coding.

17.4 DOWN-CONVERSION DECODER

17.4.1 BACKGROUND

Digital video broadcasting has had a major impact in both academic and industrial communities. A great deal of effort has been made to improve the coding efficiency at the transmission side and

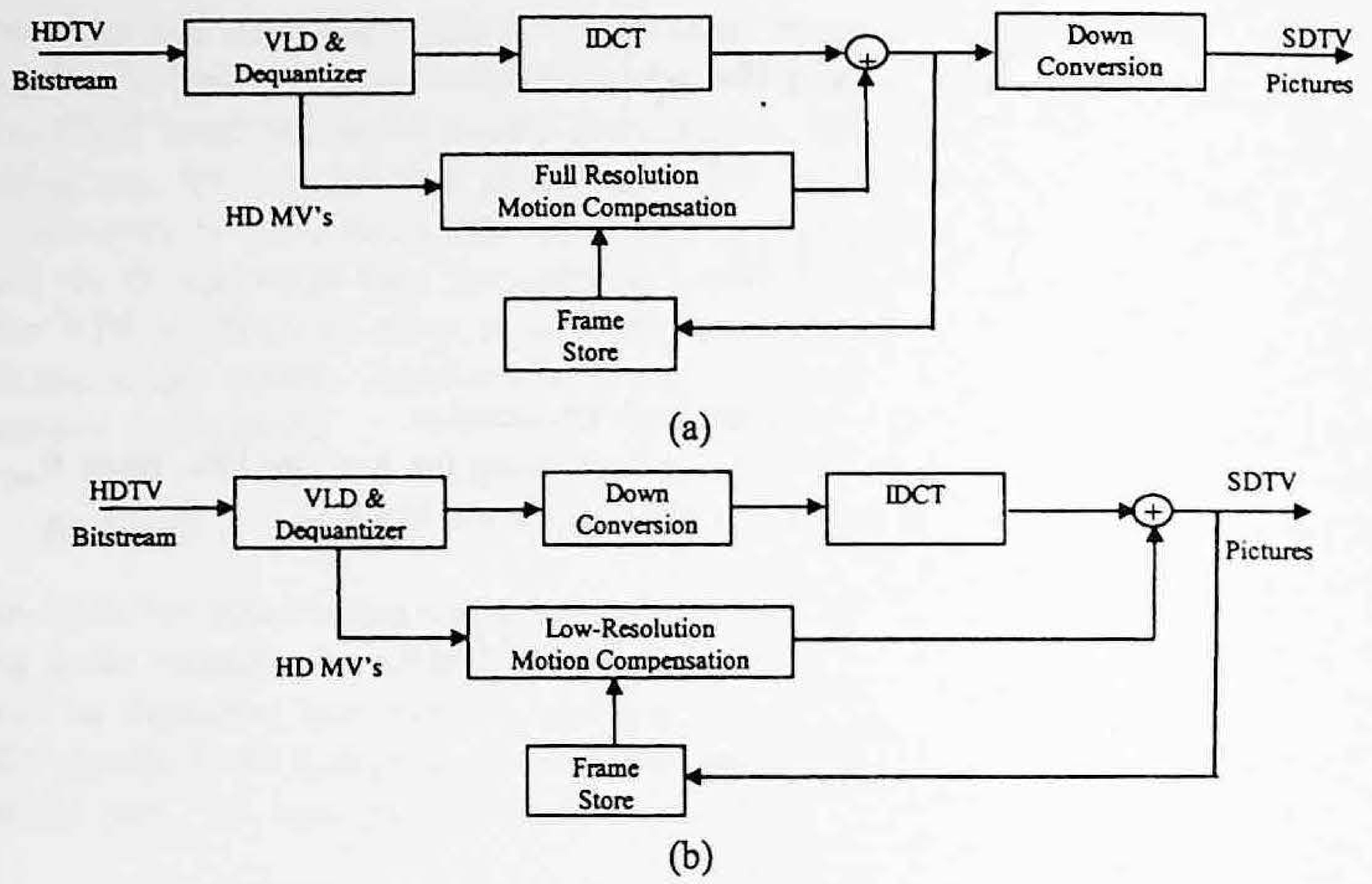


FIGURE 17.6 Decoder structures. (a) Block diagram of full-resolution decoder with down-conversion in the spatial domain. The quality of this output will serve as a drift-free reference. (b) Block diagram of low-resolution decoder. Down-conversion is performed within the decoding loop and is a frequency domain process. Motion compensation is performed from a low-resolution reference using motion vectors that are derived from the full-resolution encoder. Motion compensation is a spatial domain process.

offer cost-effective implementations in the overall end-to-end system. Along these lines, the notion of format conversion is becoming increasingly popular. On the transmission side, there are a number of different formats that are likely candidates for digital video broadcast. These formats vary in horizontal, vertical, and temporal resolution. Similarly, on the receiving side, there are a variety of display devices that the receiver should account for. In this section, we are interested in the specific problem of how to receive an HDTV bitstream and display it at a lower spatial resolution. In the conventional method of obtaining a low-resolution image sequence, the HD bitstream is fully decoded; then it is simply prefiltered and subsampled (ISO/IEC, 1993). The block diagram of this system is shown in Figure 17.6(a); it will be referred to as a full-resolution decoder (FRD) with spatial down-conversion. Although the quality is very good, the cost is quite high due to the large memory requirements. As a result, low-resolution decoders (LRDs) have been proposed to reduce some of the costs (Ng, 1993; Sun, 1993; Boyce et al., 1995; Bao et al., 1996). Although the quality of the picture will be compromised, significant reductions in the amount of memory can be realized; the block diagram for this system is shown in Figure 17.6(b). Here, incoming blocks are subject to down-conversion filters within the decoding loop. In this way, the down-converted blocks are stored into memory rather than the full-resolution blocks. To achieve a high-quality output with the low-resolution decoder, it is important to take special care in the algorithms for down-conversion and motion compensation (MC). These two processes are of major importance to the decoder as they have significant impact on the final quality. Although a moderate amount of complexity within the decoding loop is added, the reductions in external memory are expected to provide significant cost savings, provided that these algorithms can be incorporated into the typical decoder structure in a seamless way.

As stated above, the filters used to perform the down-conversion are an integral part of the low-resolution decoder. In Figure 17.6(b), the down-conversion is shown to take place before the IDCT. Although the filtering is not required to take place in the DCT domain, we initially assume that it takes place before the adder. In any case, it is usually more intuitive to derive a down-conversion filter in the frequency domain rather than in the spatial domain; this has been described by Pang et al. (1996), Merhav and Bhaskaran (1997), and Mokry and Anastassiou (1994). The major drawback of these approaches is that high frequency data is lost or not preserved very well. To overcome this, a method of down-conversion, which better preserves high-frequency data within the macroblock has been reported by Bao et al. (1996), Vetro and Sun (1998a); this method is referred to as frequency synthesis.

Although the above statement of the problem has only mentioned filtering-based approaches to memory reduction within the decoding loop, readers should be aware that other techniques have also been proposed. For the most part, these approaches rely on methods of embedded compression. For instance, de With et al. (1998) quantized the data being written to memory adaptively using a block predictive coding scheme; then a segment of macroblocks is fit into a fixed length packet. Similarly, Yu et al. (1999) proposed an adaptive min-max quantizer and edge detector. With this method, each macroblock is compressed to a fixed size to simplify memory access. Another, simpler approach may be to truncate the 8-bit data to 7 or 6 bits. However, in this case, it is expected the drift would accumulate very fast and result in poor reconstruction quality. Bruni et al. (1998) used a vectors quantization method, and Lei (1999) described a wavelet-based approach. Overall, these approaches offer exceptional techniques to reduce the memory requirements, but in most cases the reconstructed video would still be a high-resolution signal. The reason is that compressed high-resolution data are stored in memory rather than the raw, low-resolution data. For this reason, the remainder of this section emphasizes the filtering-based approach, in which the data stored in memory represent the actual low-resolution picture data.

The main novelty of the system that we describe is the filtering which is used to perform motion compensation from low-resolution anchor frames. It is well known that prediction drift has been difficult to avoid. It is partly due to the loss of high-frequency data from the down-conversion and partly due to the inability to recover the lost information. Although prediction drift cannot be totally avoided in a low-resolution decoder, it is possible to reduce the effects of drift significantly in contrast to simple interpolation methods. The solution that we described is optimal in the least-squares sense and is dependent on the method of down-conversion that is used (Vetro and Sun, 1998b). In its direct form, the solution cannot be readily applied to a practical decoding scheme. However, it is shown that a cascaded realization is easily implemented into the FRD-type structure (Vetro et al., 1998).

17.4.2 Frequency Synthesis Down-Conversion

The concept of frequency synthesis was first reported by Bao et al. (1996) and later expanded upon by Vetro and Sun (1998b). The basic premise is to better preserve the frequency characteristics of a macroblock in comparison to simpler methods which extract or cut specified frequency components of an 8×8 block. To accomplish this, the four blocks of a macroblock are subject to a global transformation — this transformation is referred to as frequency synthesis. Essentially, a single-frequency domain block can be realized using the information in the entire macroblock. From this, lower-resolution blocks can be achieved by cutting out the low-order frequency components of the synthesized block — this action represents the down-conversion process and is generally represented in the following way:

$$\tilde{\underline{A}} = X\underline{A}, \tag{17.11}$$

where $\underline{A}$ denotes the original DCT macroblock, $\underline{A}$ denotes the down-converted DCT block, and X is a matrix which contains the frequency synthesis coefficients. The original idea for frequency synthesis down-conversion was to extract an 8×8 block directly from the 16×16 synthesized block in the DCT domain as shown in Figure 17.7(a). The advantage of doing this is that the down-converted DCT block is directly applicable to an 8×8 IDCT (for which fast algorithms exist). The major drawback with regard to computation is that each frequency component in the synthesized

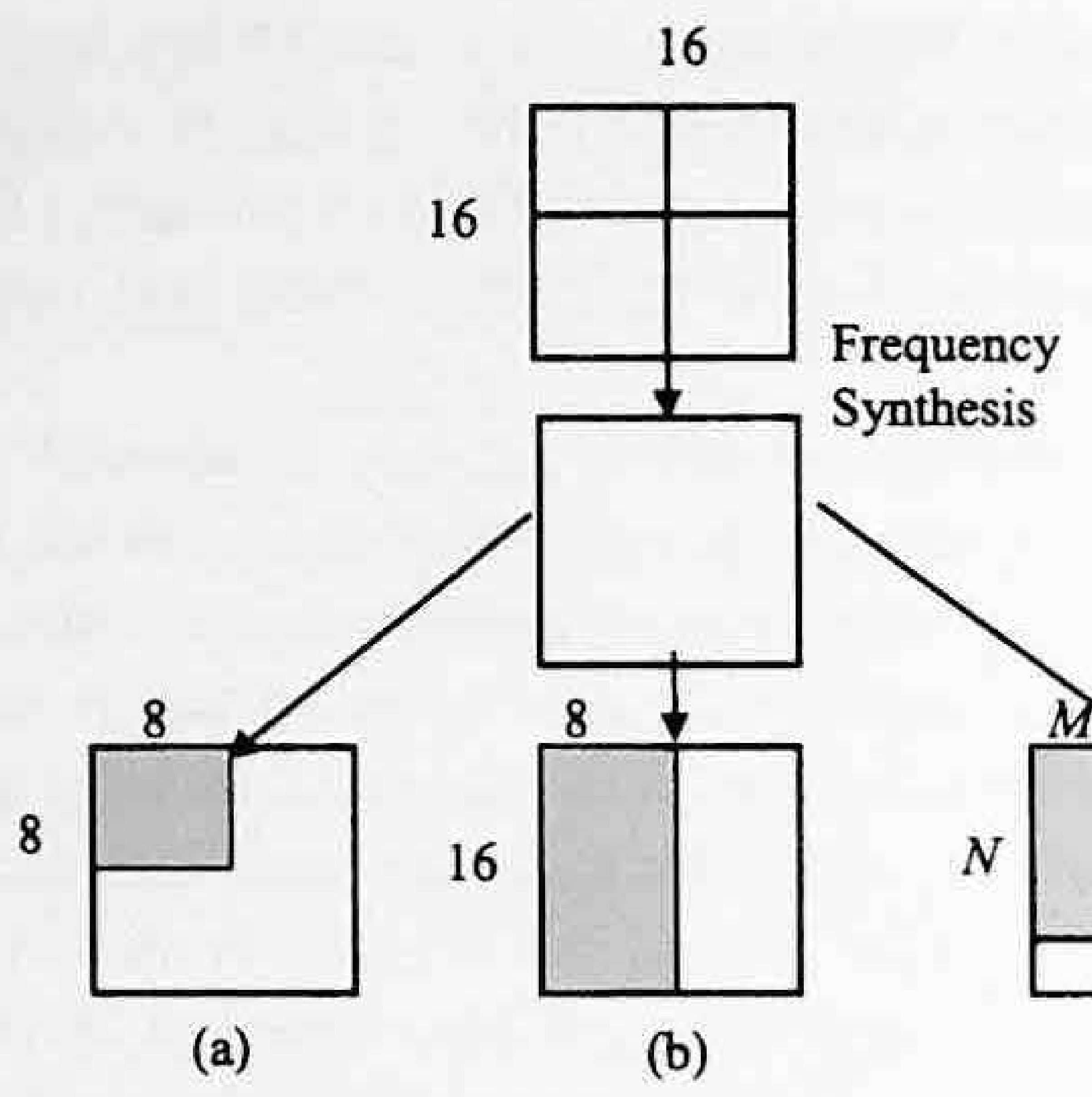


FIGURE 17.7 Concept of frequency synthesis down-conversion; (a) 256-tap filter applied to every frequency component to achieve vertical and horizontal down-conversion by a factor of two frame-based filtering; (b) 16-tap filter applied to frequency components in the same row to achieve horizontal down-conversion by two, picture structure is irrelevant; (c) illustration that the amount of synthesized frequency components which are retained is arbitrary.

block is dependent on all of the frequency components in each of the 8×8 blocks, i.e., each synthesized frequency component is the result of a 256-tap filter. The major drawback with regard to quality is that interlaced video with field-based predictions should not be subject to frame-based filtering (Vetro and Sun, 1998b). If frame-based filtering is used, it becomes impossible to recover the appropriate field-based data that is required to make field-based predictions. In areas of large motion, severe blocking artifacts will result.

Obviously, the original approach would incur too much computation and quality degradation, so, instead, the operations are performed separately and vertical down-conversion is performed on a field basis. In Figure 17.7(b), it is shown that a horizontal-only down-conversion can be performed. To perform this operation, a 16-tap filter is ultimately required. In this way, only the relevant row information is applied as the input to the horizontal filtering operation and the structure of the incoming video has no bearing on the down-conversion process. The reason is that the data in each row of a macroblock belong to the same field; hence the format of the output block will be unchanged. It is noteworthy that the set of filter coefficients is dependent on the particular output frequency index. For 1-D filtering, this means that the filters used to compute the second output index, for example, are different from those used to compute the fifth output index. Similar to the horizontal down-conversion, vertical down-conversion can also be applied as a separate process. As reasoned earlier, field-based filtering is necessary for interlaced video with field-based predictions.

However, since a macroblock consists of eight lines for the even field and eight lines for the odd field, and the vertical block unit is 8, frequency synthesis cannot be applied. Frequency synthesis is a global transformation and is only applicable when one wishes to observe the frequency characteristics over a larger range of data than the basic unit. Therefore, to perform the vertical down-conversion, we can simply cut the low-order frequency components in the vertical direction. This loss that we accept in the vertical direction is justified by the ability to perform accurate low-resolution MC that is free from severe blocking artifacts.

In the above, we have explained how the original idea to extract an 8×8 DCT block is broken down into separable operations. However, since frequency synthesis provides an expression for every frequency component in the new 16×16 block, it makes sense to generalize the down-conversion process so that decimation, which are multiples of $1/16$ can be performed. In Figure 17.7(c), an $M \times N$ block is extracted. Although this type of down-conversion filtering may not be appropriate before the IDCT operation and may not be appropriate for a bitstream containing

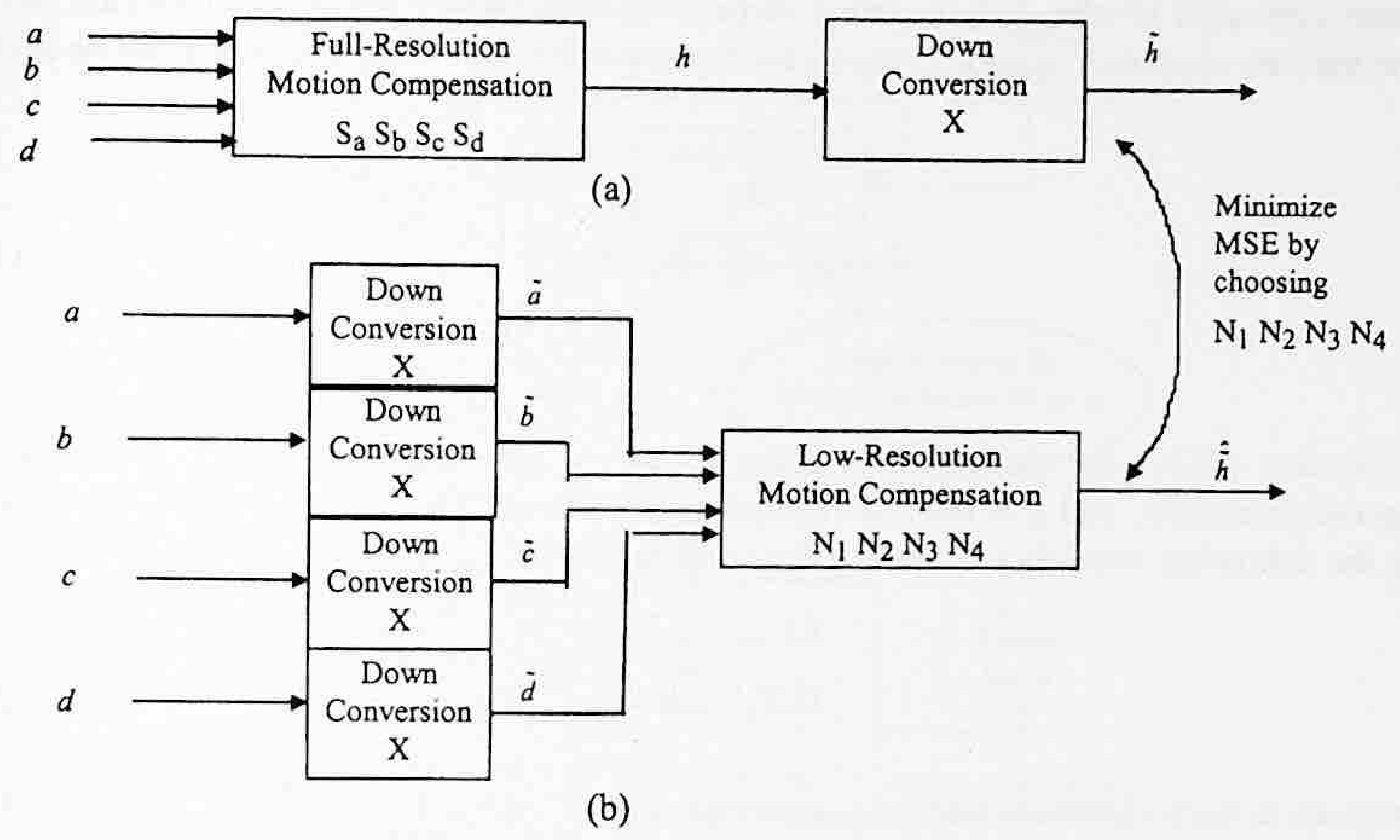


FIGURE 17.8 Comparison of decoding methods to achieve low-resolution image sequence. (a) FRD with spatial down-conversion; (b) LRD. The objective is to minimize the MSE between the two outputs by choosing N_1, N_2, N_3, and N_4 for a fixed down-conversion. (From Vetro, A. et al., *IEEE Trans. Consumer Elec.*, 44(3), 1998. With permission.)

field-based predictions, it may be applicable elsewhere, e.g., as a spatial domain filter somewhere else in the system and/or for progressive material. To obtain a set of spatial domain filters, an appropriate transformation can be applied. In this way, Equation 17.8 is expressed as

$$\underline{\tilde{a}} = x\underline{a}, \qquad (17.12)$$

where the lowercase counterparts denote spatial equivalents. The expression which transforms X to x is derived in Appendix A, Section 17.4.6.

17.4.3 Low-Resolution Motion Compensation

The focus of this section is to provide an expression for the optimal set of low-resolution MC filters given a set of down-conversion filters. The resulting filters are optimal in the least-squares sense as they minimize the mean squared error (MSE) between a reference block and a block obtained through low-resolution MC. The results that have been derived by Vetro and Sun (1998a) assume that a spatial domain filter, x, is applied to incoming macroblocks to achieve the down-conversion. The scheme shown in Figure 17.8(a) illustrates the process by which reference blocks are obtained. First, full-resolution motion compensation is performed on macroblocks $\underline{a}$, $\underline{b}$, $\underline{c}$, and $\underline{d}$ to yield $\underline{h}$. To execute this process, the filters $S_a^{(r)}, S_b^{(r)}, S_c^{(r)}$, and $S_d^{(r)}$ are used. Basically, these filters represent the masking/averaging operations of the motion compensation in a matrix form. More on the composition of these filters can be found in Appendix B, Section 17.4.7. Once $\underline{h}$ is obtained, it is down-converted to $\underline{\tilde{h}}$ via the spatial filter, x:

$$\underline{\tilde{h}} = x\underline{h}. \qquad (17.13)$$

The above block is considered to be the drift-free reference. On the other hand, in the scheme of Figure 17.8(b), the blocks $\underline{a}$, $\underline{b}$, $\underline{c}$, and $\underline{d}$ are first subject to the down-conversion filter, x, to yield the down-converted blocks, $\underline{\tilde{a}}$, $\underline{\tilde{b}}$, $\underline{\tilde{c}}$, and $\underline{\tilde{d}}$, respectively. By using these down-converted blocks as input to the low-resolution motion compensation process, the following expression can be assumed:

$$\underline{\hat{\tilde{h}}} = \begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix} \begin{bmatrix} \underline{\tilde{a}} \\ \underline{\tilde{b}} \\ \underline{\tilde{c}} \\ \underline{\tilde{d}} \end{bmatrix}, \tag{17.14}$$

where N_k, $k = 1,2,3,4$ are the unknown filters which are assumed to perform the low-resolution motion compensation, and $\underline{\tilde{h}}$ is the low-resolution prediction. These filters are solved by differentiating the following objective function (Vetra and Sun, 1998a):

$$J\{N_k\} = \left\| \underline{\tilde{h}} - \underline{\hat{\tilde{h}}} \right\|^2, \tag{17.15}$$

with respect to each unknown filter and setting each result equal to zero. It can be verified that the optimal least-squares solution for these filters is given by

$$\begin{aligned} N_1^{(r)} &= xS_a^{(r)}x^+; \quad N_2^{(r)} = xS_b^{(r)}x^+ \\ N_3^{(r)} &= xS_c^{(r)}x^+; \quad N_4^{(r)} = xS_d^{(r)}x^+, \end{aligned} \tag{17.16}$$

where

$$x^+ = x^T\left(xx^T\right)^{-1} \tag{17.17}$$

is the Moore–Penrose inverse (Lancaster and Tismenetsky, 1985) for an $m \times n$ matrix with $m \le n$. In the solution of Equation 17.16, the superscript r is added to the filters, N_k, due to their dependency on the full-resolution motion compensation filters. In using these filters to perform the low-resolution motion compensation, the MSE between $\underline{\tilde{h}}$ and $\underline{\hat{\tilde{h}}}$ is minimized. It should be emphasized that Equation 17.16 represents a generalized set of MC filters which are applicable to any x, which operates on a single macroblock. For the special case of the 4×4 cut, these filters are equivalent to the ones that were determined by Morky and Anastassiou (1994) to minimize the drift.

In Figure 17.9, two equivalent MC schemes are shown. However, for implementation purposes, the optimal MC scheme is realized in a cascade form rather than a direct form. The reason is that the direct-form filters are dependent on the matrices, which perform full-resolution MC. Although, these matrices were very useful in analytically expressing the full-resolution MC process, they require a huge amount of storage due to their dependency on the prediction mode, motion vector, and half-pixel-accuracy. Instead, the three linear processes in Equation 17.13 are separated, so that an up-conversion, full-resolution MC, and down-conversion can be performed. Although one may be able to guess such a scheme, we have proved here that it is an optimal scheme provided the up-conversion filter is a Moore–Penrose inverse of the down-conversion filter. Vetro and Sun (1998b), the optimal MC scheme, which employs frequency synthesis, to a nonoptimal MC scheme, which employs bilinear interpolation, and an optimal MC scheme, which employs the 4×4 cut down-conversion. Significant reductions in the amount of drift were realized by both optimal MC schemes over the method, which used bilinear interpolation as the method of up-conversion. But more importantly, a 35% reduction in the amount of drift was realized by the optimal MC scheme using frequency synthesis over the optimal MC scheme using the 4×4 cut.

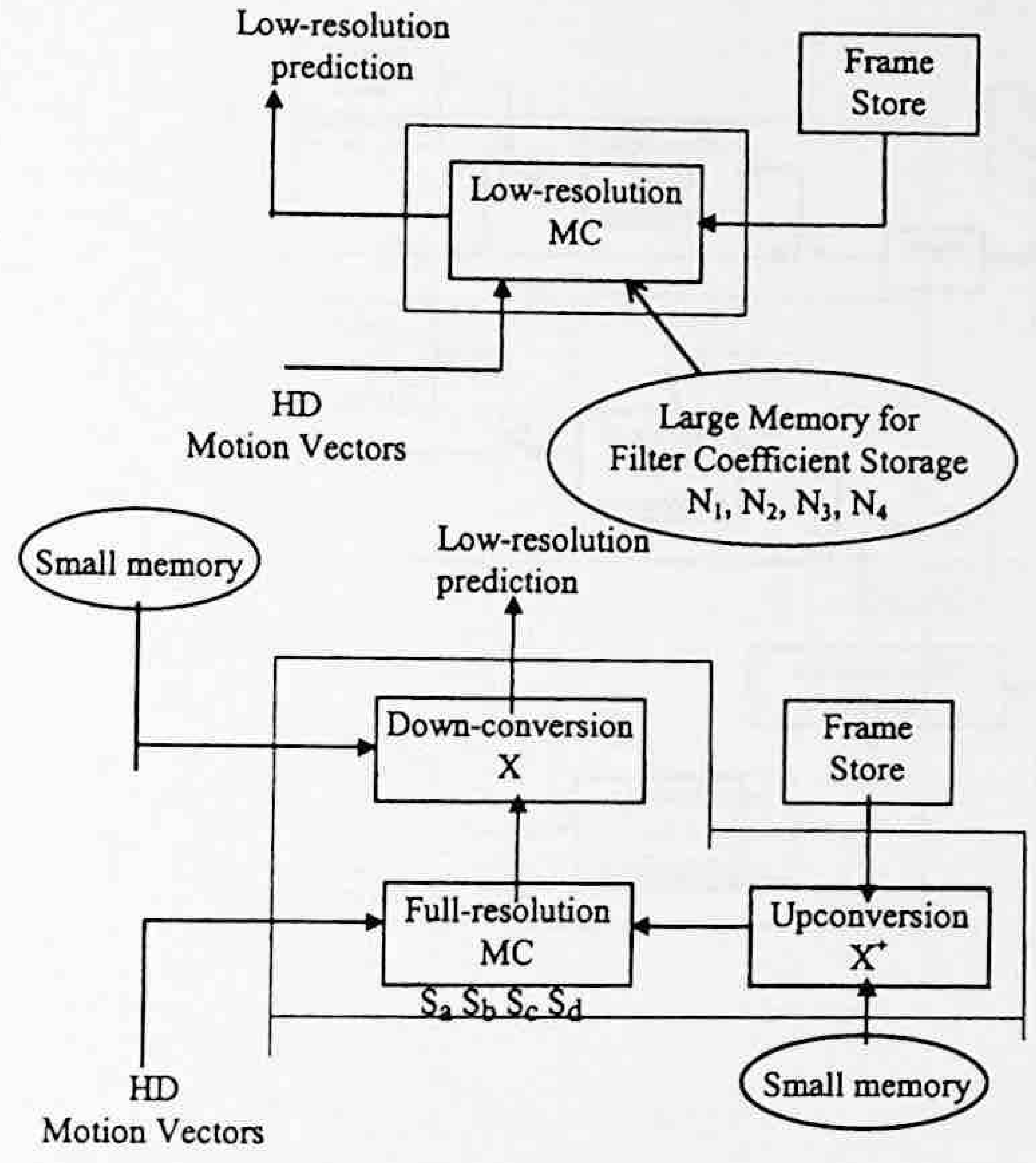
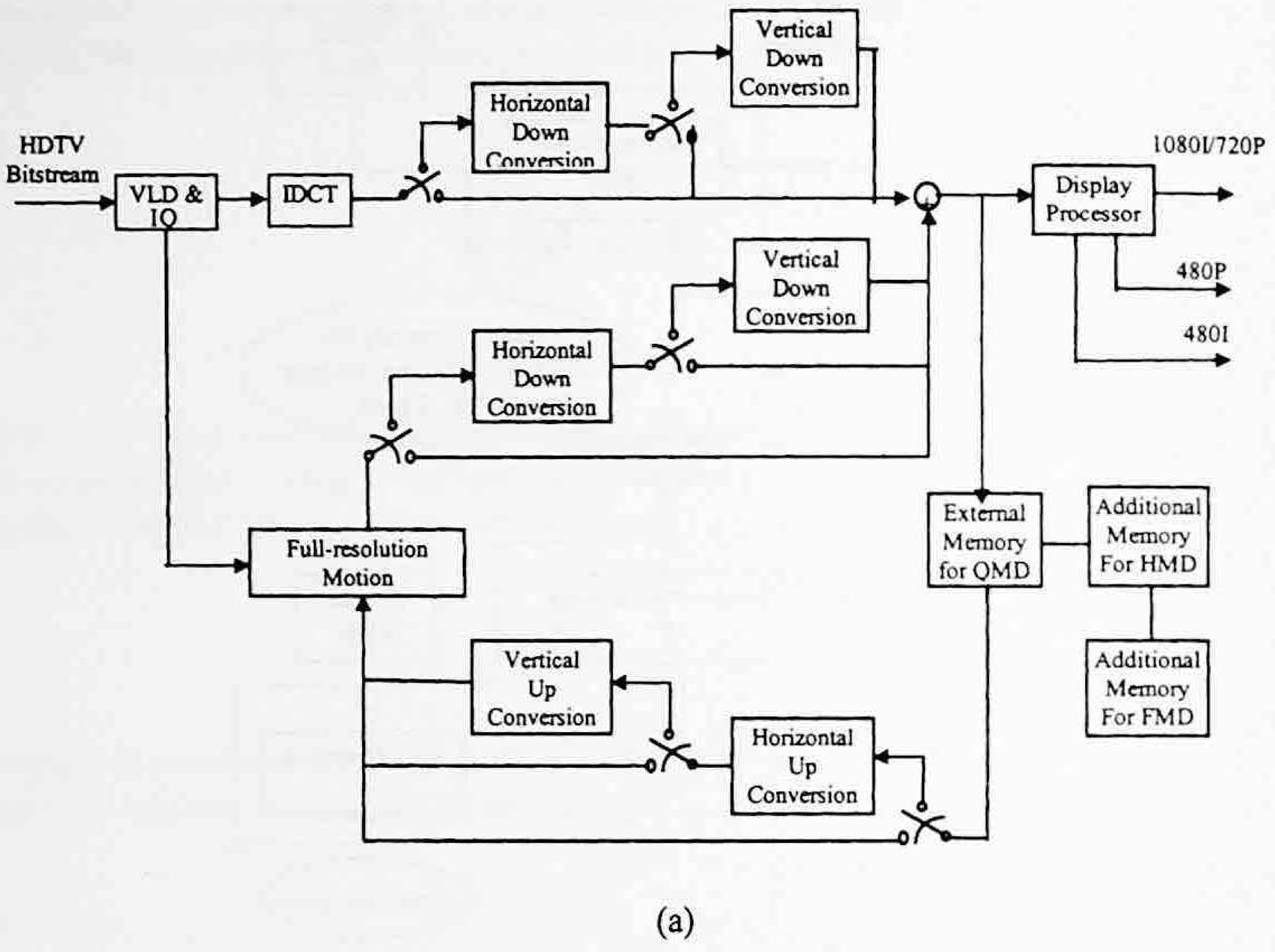


FIGURE 17.9 Optimal low-resolution MC scheme: direct form (top) vs. cascade form (bottom). Both forms yield equivalent quality, but vary significantly in the amount of internal memory. (From Vetra, A., et al., *IEEE Trans. Consumer Elec.*, 44(3), 1998. With permission.)

17.4.4 Three-Layer Scalable Decoder

In this section, we show how the key algorithms for down-conversion and motion compensation are integrated into a three-layer scalable decoder. The central concept of this decoder is that three layers of resolution can be decoded using a decreased amount of memory for the lower resolution layers. Also, regardless of which layer is being decoded, much of the logic can be shared. Three possible decoder configurations are considered: full-memory decoder (FMD), half-memory decoder (HMD), and quarter-memory decoder (QMD). The low-resolution decoder configurations are based on the key algorithms, which were described for down-conversion and motion compensation. In the following, three possible architectures are discussed that provide equal quality, but vary in system-level complexity. The first (ARCH1) is based on the low-resolution decoder modeled in Figure 17.6(b), the second (ARCH2) is very similar, but attempts to reduce the IDCT computation, while the third (ARCH3) is concerned with the amount of interface with an existing high-level decoder.

With regard to functionality, all of the architectures share similar characteristics. For one, an efficient implementation is achieved by arranging the logic in a hierarchical manner, i.e., employing separable processing. In this way, the FMD configuration is the simplest and serves as the logic core on which other decoder configurations are built. In the HMD configuration, an additional horizontal down-conversion and up-conversion are performed. In the QMD configuration, all of the logic components from the HMD are utilized, such that an additional vertical down-conversion is performed after a horizontal down-conversion, and an additional vertical up-conversion is performed after a horizontal up-conversion. In summary, the logic for the HMD is built on the logic for the FMD, and the logic for the QMD is built on the logic of the HMD. The total system contains a moderate increase in logic, but HD bitstreams may be decoded to a lower resolution with a smaller amount of external memory. By simply removing external memory, lower layers can be achieved at a reduced cost.



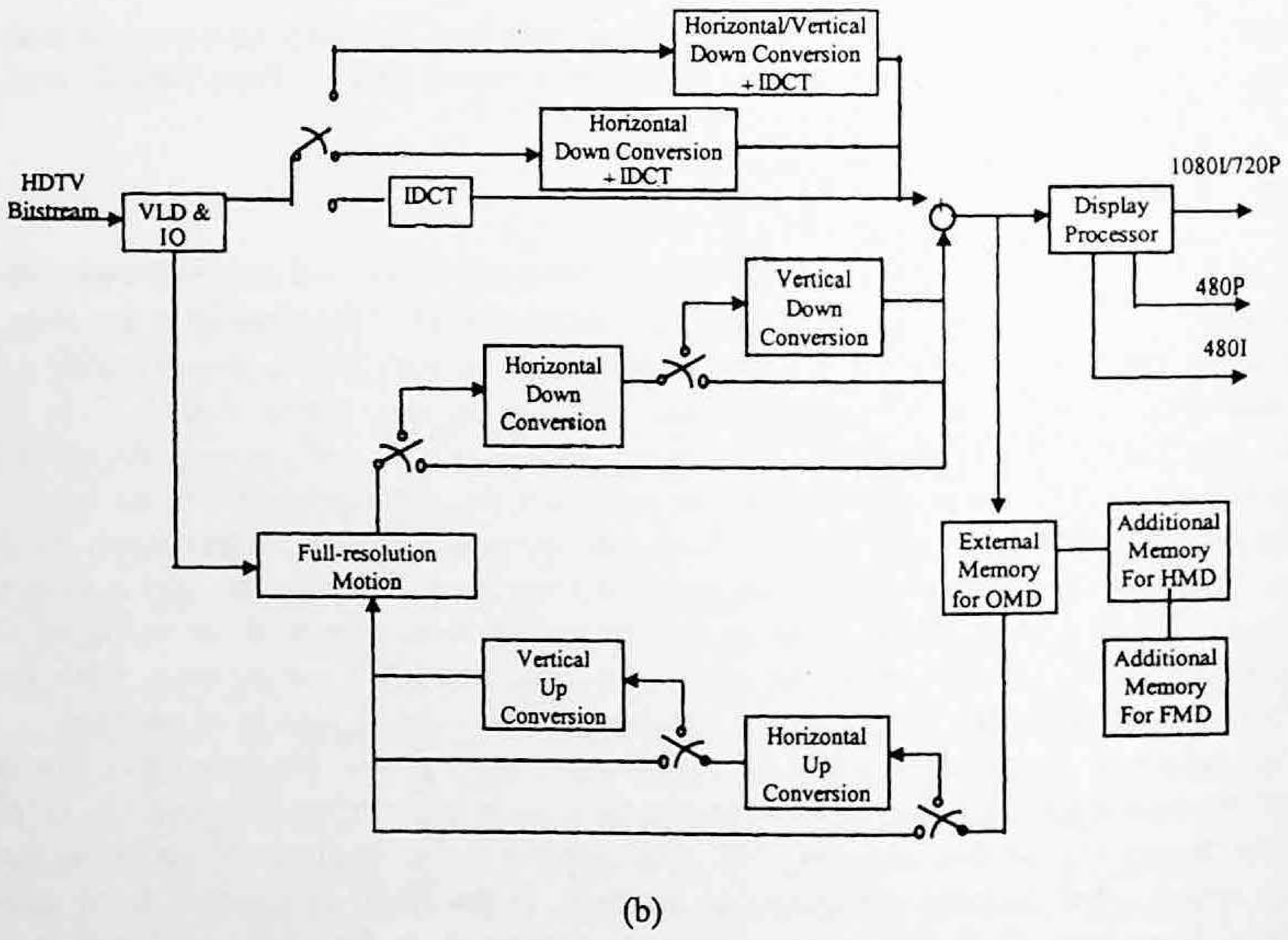
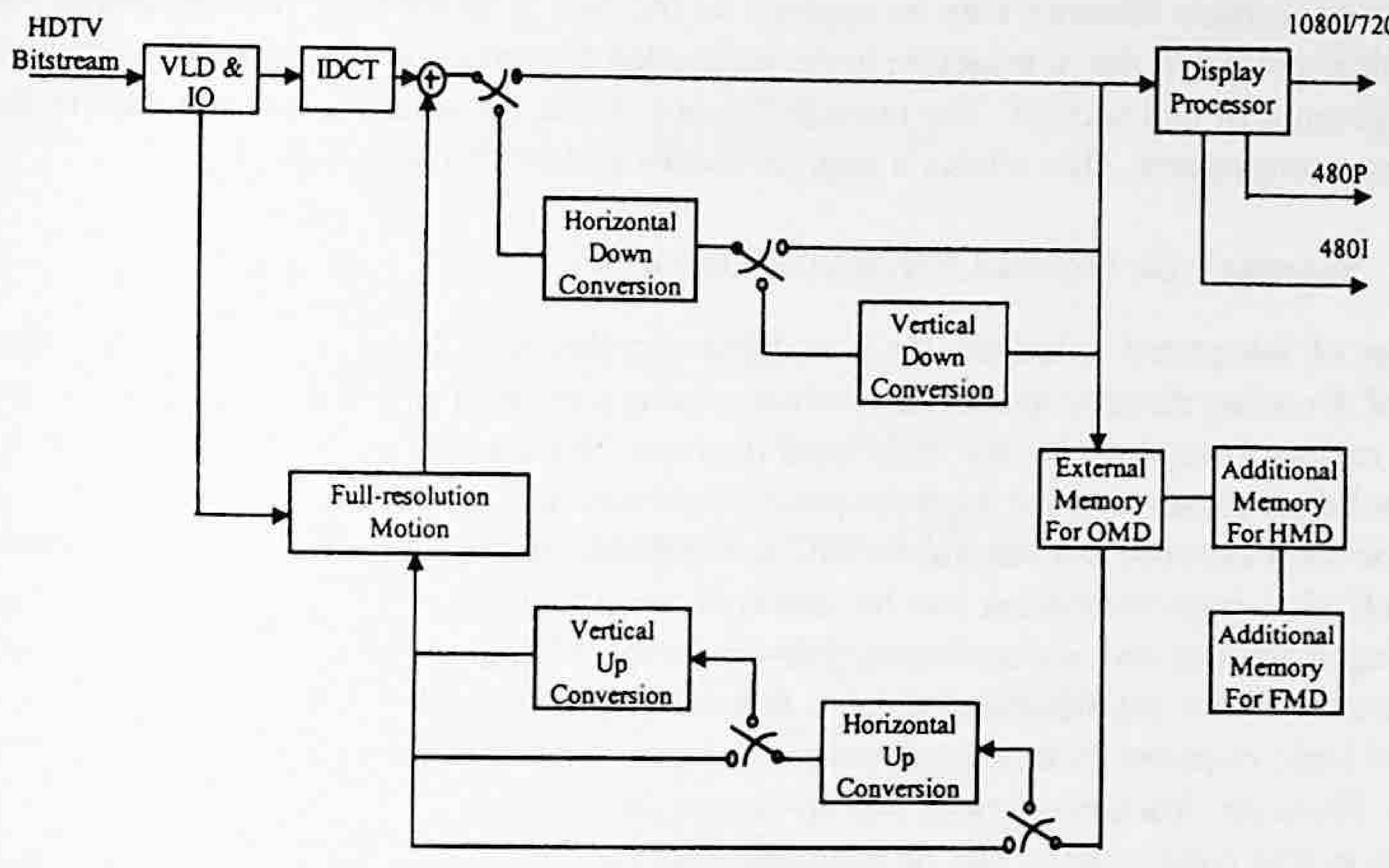


FIGURE 17.10 Block diagram of various three-layer scalable decoder architectures; all architectures provide equal quality with varying system complexity: (a) ARCH1, derived directly from block diagram of assumed low-resolution decoder; (b) ARCH2, reduce computation of IDCT by combining down-conversion and IDCT filters; (c) ARCH3, minimize interface with existing HL decoder by moving linear filtering for down-conversion outside of the adder. (From Vetro, A. et al., *IEEE Trans. Consumer Elec.*, 44(3), 1998. With permission.)



(c)

FIGURE 17.10 (continued)

The complete block diagram of ARCH1 is shown in Figure 17.10(a). The diagram shown here assumes two things: (1) the initial system model of a low-resolution decoder from Figure 17.6(b) is assumed, and (2) the down-conversions in the incoming branch are performed after the IDCT to avoid any confusion regarding macroblock format conversions in the DCT domain (Vetro and Sun, 1998b). In looking at the resulting system, it is evident that full computation of the IDCT is required, and that two independent down-conversion operations must be performed. The latter is necessary so that low-resolution predictions are added to low-resolution residuals. Overall, the increase in logic for the added feature of memory savings is quite small. However, it is evident that ARCH1 is not the most cost-effective implementation, but it represents the foundation of previous assumptions, and allows us to analyze better the impact of the two modified architectures to follow.

In Figure 17.10(b), the block diagram of ARCH2 is shown. In this system, realizing that the IDCT operation is simply a linear filter reduces the combined computation for the IDCT and down-conversion. In the FMD, we know that a fast IDCT is applied separately to the rows and columns of an 8×8 block. For the HMD, our goal is to combine the horizontal down-conversion with the horizontal IDCT. In the 1-D case, an 8×16 matrix can represent the horizontal down-conversion, and an 8×8 matrix can represent the horizontal IDCT. Combining these processes, such that the down-conversion operates on the incoming DCT rows first, results in a combined 8×16 matrix. To complete the transformation, the remaining columns can then be applied to the fast IDCT. In the above description, computational savings are achieved in two places: first, the horizontal IDCT is fully absorbed into the down-conversion computation which must take place anyway, and, second, the fast IDCT is utilized for a smaller amount of columns. In the case of the QMD, these same principles can be used to combine the vertical down-conversion with the vertical IDCT. In this case, one must be aware of the macroblock type (field-DCT or frame-DCT) so that an appropriate filter can be applied. In contrast to the previous two architectures, ARCH3 assumes that the entire front-end processing of the decoder is used; it is shown in Figure 17.5. In this way, the adder is always a full-resolution adder, whereas in ARCH1 and ARCH2, the adder needed to handle all three layers of resolution. The major benefit of ARCH3 is that it does not require many interfaces with the existing decoder structure. The memory is really the only place where a new interface needs to be defined. Essentially, a down-conversion filtering may be applied before storing the data, and an up-conversion filtering may be applied, as the data is needed for full-resolution MC. This final architecture is similar in principle to the embedded compression schemes that were mentioned in the beginning of this section. The main difference is that the resolution of the data is decreased rather than compressed. This allows a simpler means of low-resolution display.

17.4.5 Summary of Down-Conversion Decoder

A number of integrated solutions for a scalable decoder have been presented. Each decoder is capable of decoding directly to a lower resolution using a reduced amount of memory in comparison with the memory required by the high-level decoder. The method of frequency synthesis is successful in better preserving the high-frequency data within a macroblock, and the filtering that is used to perform optimal low-resolution MC is capable of minimizing the drift. It has been shown that a realizable implementation can be achieved, such that the filters for optimal low-resolution MC are equivalent to an up-conversion, full-resolution MC, and for down-conversion, where the up-conversion filters are determined by a Moore–Penrose inverse of the down-conversion. The amount of logic required by these processes is kept minimal since they are realized in a hierarchical structure. Since the down-conversion and up-conversion processes are linear, the architecture design is flexible in that equal quality can be achieved with varying levels of system complexity. The first architecture that we examined came from the initial assumptions that were made on the low-resolution decoder, i.e., a down-conversion is performed before the adder. It was noted that a full IDCT computation was required and that a down-conversion must be performed in two places. As a result, a second architecture was presented to reduce the IDCT computation, and a third was presented to minimize the amount of interface with the existing high-level decoder. The major point here is that the advantages of ARCH2 and ARCH3 cannot be realized by a single architecture. The reason is that performing a down-conversion in the incoming branch reduces the IDCT computation; therefore, a down-conversion must be performed after the full-resolution MC as well. In any case, equal quality is offered by each architecture and the quality is of commercial grade.

17.4.6 DCT-to-Spatial Transformation

Our objective in this section is to express the following DCT domain relationship:

$$\tilde{A}(k,l) = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \left[X_{k,l}(p,q) A(p,q) \right] \tag{17.18}$$

as

$$\tilde{a}(i,j) = \sum_{s=0}^{M-1} \sum_{t=0}^{N-1} \left[x_{i,j}(s,t) a(s,t) \right], \tag{17.19}$$

where $\tilde{A}$ and $\tilde{a}$ are the DCT and spatial output, A and a are the DCT and spatial input, and X and x are the DCT and spatial filters, respectively. By definition, the $M \times N$ DCT transform is defined by

$$A(k,l) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} a(i,j) \psi_k^M(i) \psi_l^N(j) \tag{17.20}$$

and its inverse, the $M \times N$ IDCT by

$$a(i,j) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} A(k,l) \psi_k^M(i) \psi_l^N(j), \tag{17.21}$$

where the basis function is given by

$$\psi_k^N = \sqrt{\frac{2}{N}}\,\alpha(k)\cos\left(\frac{2i+1}{2N}k\pi\right) \tag{17.22}$$

and

$$\alpha(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for} \quad k = 0; \\ 1 & \text{for} \quad k \neq 0. \end{cases} \tag{17.23}$$

By substituting Equation 17.22 into the expression for the IDCT yields

$$\begin{aligned} \tilde{a}(i,j) &= \sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\psi_k^M(i)\psi_l^N(j)\cdot\left[\sum_{p=0}^{M-1}\sum_{q=0}^{N-1}X_{k,l}(p.q)A(p.q)\right] \\ &= \sum_{p=0}^{M-1}\sum_{q=0}^{N-1}A(p,q)\cdot\left[\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}X_{k,l}(p,q)\psi_k^M(i)\psi_l^N(j)\right]. \end{aligned} \tag{17.24}$$

Substituting the DCT definition into the above gives the following,

$$\tilde{a}(i,j) = \sum_{p=0}^{M-1}\sum_{q=0}^{N-1}\left[\sum_{s=0}^{M-1}\sum_{t=0}^{N-1}a(s,t)\psi_p^M(s)\psi_q^N(t)\right]\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\left[X_{k,l}(p,q)\cdot\psi_p^M(i)\psi_q^N(j)\right]. \tag{17.25}$$

Finally, Equation 17.17 can be formed with

$$x_{i,j}(s,t) = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\psi_k^M(i)\cdot\psi_l^N(j)\left[\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\left(X_{k,l}(p,q)\cdot\psi_p^M(s)\psi_q^N(t)\right]\right. \tag{17.26}$$

and the transformation is fully defined.

17.4.7 Full-Resolution Motion Compensation in Matrix Form

In 2-D, a motion compensated macroblock may have contributions from at most four macroblocks per motion vector. As noted in Figure 17.11, macroblocks *a*, *b*, *c*, and *d* include four 8×8 blocks each. These subblocks are raster-scanned so that each macroblock can be represented as a vector. According to the motion vector, (dx, dy), a local reference, (y_1,y_2), is computed to indicate where the origin of the motion compensated block is located; the local reference is determined by

$$\begin{aligned} y_1 &= dy - 16\cdot\left[Integer(dy/16) - \gamma(dy)\right] \\ y_2 &= dx - 16\cdot\left[Integer(dx/16) - \gamma(dx)\right], \end{aligned} \tag{17.27}$$

where

$$\gamma(d) = \begin{cases} 1, & \text{if} \quad d < 0 \text{ and } d \bmod 16 = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{17.28}$$

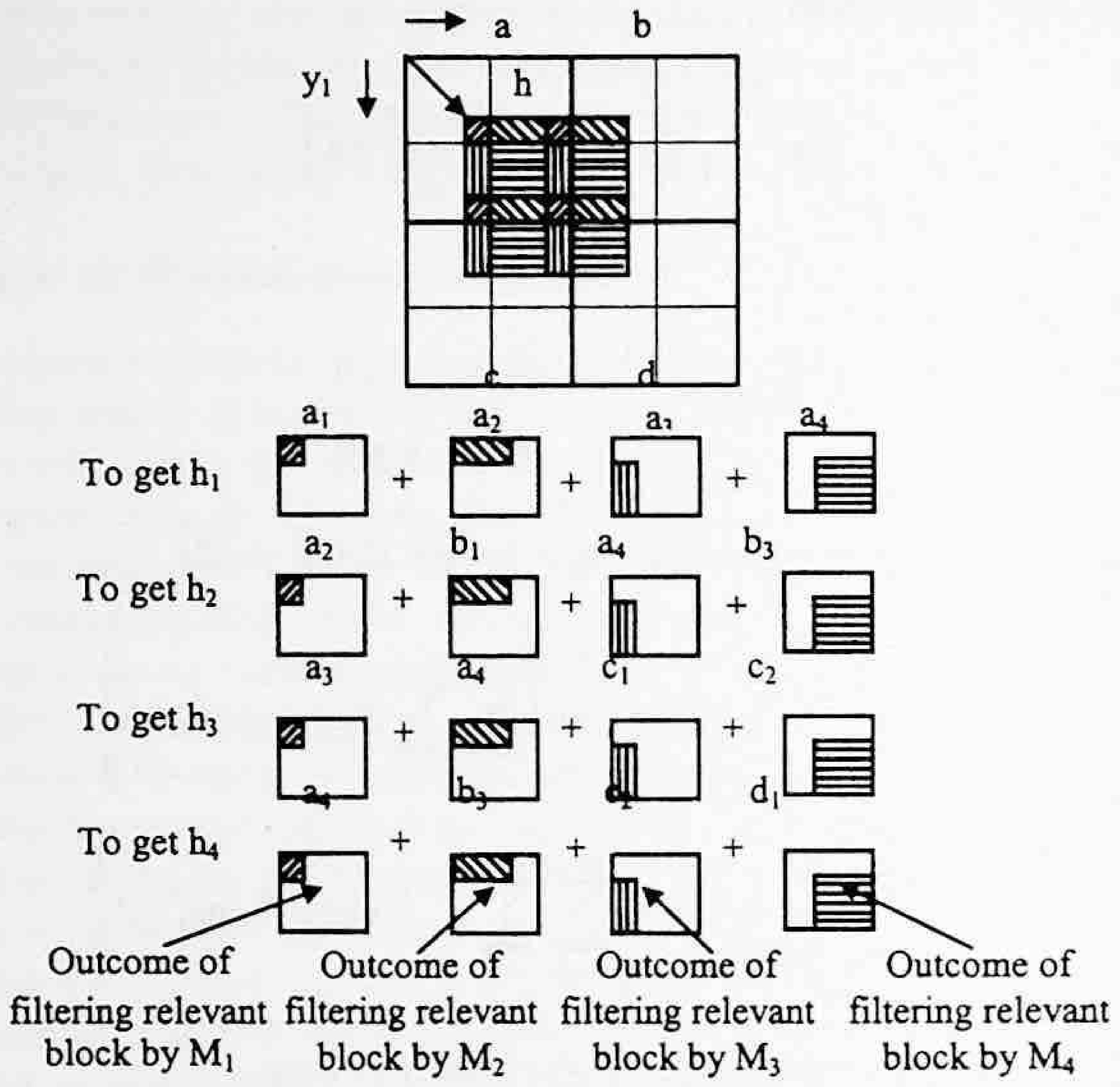

a
b
y_1
h
c
d
To get h_1
a_1 + a_2 + a_3 + a_4
To get h_2
a_2 + b_1 + a_4 + b_3
To get h_3
a_3 + a_4 + c_1 + c_2
To get h_4
a_4 + b_3 + c_2 + d_1
Outcome of filtering relevant block by M_1
Outcome of filtering relevant block by M_2
Outcome of filtering relevant block by M_3
Outcome of filtering relevant block by M_4

FIGURE 17.11 Relationship between the input and output blocks of the motion compensation process in the FRD. (From Vetro, A. et al., *IEEE Trans. Consumer Elec.*, 44(3), 1998. With permission.)

The reference point for this value is the origin of the upper-left-most input macroblock. With this, the motion-compensated prediction may be expressed as

$$\underline{h} = \begin{bmatrix} \underline{h_1} \\ \underline{h_2} \\ \underline{h_3} \\ \underline{h_4} \end{bmatrix} = \begin{bmatrix} S_a^{(r)} & S_b^{(r)} & S_c^{(r)} & S_d^{(r)} \end{bmatrix} \cdot \begin{bmatrix} \underline{a} \\ \underline{b} \\ \underline{c} \\ \underline{d} \end{bmatrix}; \quad r = 1, 2, 3, 4. \tag{17.29}$$

As an example, Figure 17.11 considers $(y_1, y_2) \in [0,7]$, which implies that $r = 1$. In this case the motion compensation filters are given by

$$S_a^{(1)} = \begin{bmatrix} M_1 & M_2 & M_3 & M_4 \\ 0 & M_1 & 0 & M_3 \\ 0 & 0 & M_1 & M_2 \\ 0 & 0 & 0 & M_1 \end{bmatrix}, \quad S_b^{(1)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ M_2 & 0 & M_4 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & M_2 & 0 \end{bmatrix},$$

$$S_c^{(1)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ M_3 & M_4 & 0 & 0 \\ 0 & M_3 & 0 & 0 \end{bmatrix}, \quad S_c^{(1)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ M_4 & 0 & 0 & 0 \end{bmatrix}. \tag{17.30}$$

In the above equations, the M_1, M_2, M_3, and M_4 matrices operate on the relevant 8×8 blocks of a, b, c, and d. Their elements will vary according to the amount of overlap as indicated by (y_1, y_2) and the type of prediction. The type of prediction may be frame based or field based and is predicted with half-pixel accuracy. As a result, the matrices $S_a^{(r)}$, $S_b^{(r)}$, $S_c^{(r)}$, and $S_d^{(r)}$, are extremely sparse and may only contain nonzero values of 1, ½, and ¼. For different values of (y_1,y_2) the configuration of the above matrices will change: $y_1 \in [0,7]$ and $y_2 \in [8,15]$ implies $r = 2$; $y_1 \in [8,15]$ and $y_2 \in [0,7]$ implies $r = 3$; $y_1, y_2 \in [8,15]$ implies $r = 4$. The resulting matrices can easily be formed using the concepts illustrated in Figure 17.11.

17.5 ERROR CONCEALMENT

17.5.1 BACKGROUND

Practical communications channels available for delivery of compressed digital video are characterized by occasional bit error and/or packet loss, although the actual impairment mechanism varies widely with the specific medium under consideration. The class of decoder error concealment schemes described here is based on identification and predictive replacement of picture regions affected by bit error or data loss. It is noted that this approach is based on conversion (via appropriate error/loss detection mechanisms) of the transmission medium into an erasure channel in which all error or loss events can be identified in the received bit-stream. In a block-structured compression algorithm such as MPEG, all channel impairments are manifested as erasures of video units (such as MPEG macroblocks or slices). Concealment at the decoder is then based on exploiting temporal and spatial picture redundancy to obtain an estimate of erased picture areas. The efficiency of error concealment depends on redundancies in pictures and on redundancies in the compressed bitstream that are not removed by source coding. Block compression algorithms do not remove a considerable amount of inter-block redundancies, such as structure, texture, and motion information about objects in the scene.

To be more specific, error resilience for compressed video can be achieved through the addition of suitable transport and error concealment methods, as outlined in the system block diagram shown in Figure 17.12.

The key elements of such a robust video delivery system are outlined below:

- The video signal is encoded using an appropriate video compression syntax such as MPEG. Note that we have restricted consideration primarily to the practical case in which the video compression process itself is not modified, and robustness is achieved through additive transport and decoder concealment mechanisms (except for I-frame motion described in Section 17.4.3). This approach simplifies encoder design, since it separates

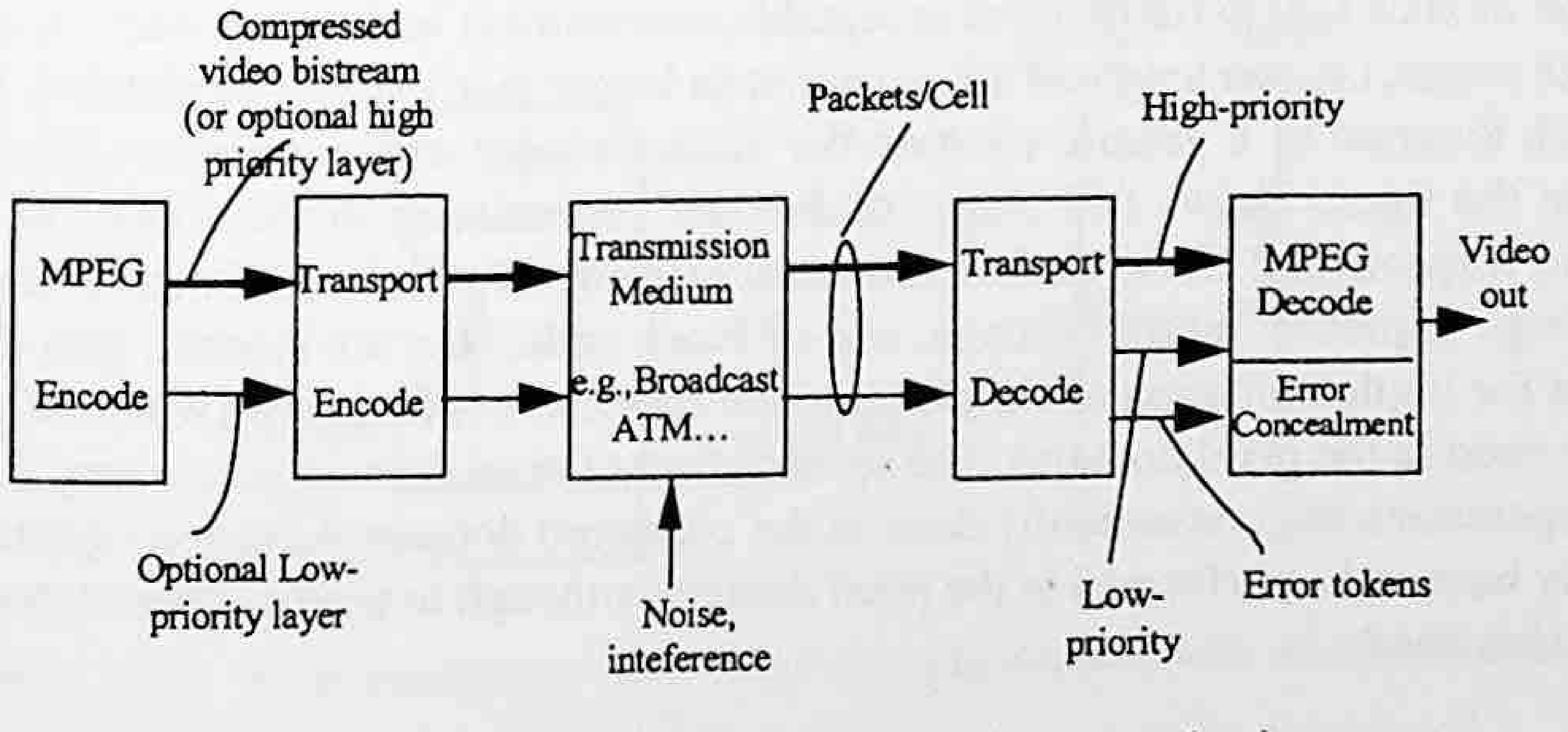


FIGURE 17.12 System block diagram of visual communication system.

media-independent video compression functions from media-dependent transport operations. On the receiver side, although a similar separation is substantially maintained, the video decoder must be modified to support an "error token" interface and error concealment functionality.

- Compressed video data is organized into a systematic data structure with appropriate headers for identification of the temporal and spatial pixel-domain location of encoded data (Joseph et al., 1992b). When an erroneous/lost packet is detected, these video units serve as resynchronization points for resumption of normal decoding, while the headers provide a means for precisely locating regions of the picture that were not correctly received. Note that two-tier systems may require additional transport-level support for high- and low-priority (HP/LP) resynchronization (Siracusa et al., 1993).
- The video bitstream may optionally be segregated into two layers for prioritized transport (Ghanbari, 1989; Kishno et al., 1989; Karlsson and Vetterli, 1989; Zdepski et al., 1989; Joseph et al., 1992a,b; Siracusa, 1993) when a high degree of error resilience is required. Note that separation into high and low priorities may be achieved either by using a hierarchical (layered) compression algorithm (Ghanbari, 1989; Siracusa, 1993) or by direct codeword parsing (Zdepski et al., 1989; 1990). Note that both these layering mechanisms have been accepted for standardization by MPEG-2 (ISO/IEC, 1995).
- Once the temporal and spatial location(s) corresponding to lost or incorrectly received packets is determined by the decoder, it will execute an error-concealment procedure for replacement of lost picture areas with subjectively acceptable material estimated from available picture regions (Harthanck et al., 1986; Jeng and Lee, 1991; Wang and Zhu, 1991). Generally, this error concealment procedure will be applied to all erased blocks in one-tier (single-priority) transmission systems, while for two-tier (HP/LP) channels the concealment process may optionally ignore loss of LP data.
- In the following subsections, the technical detail of some commonly used error concealment algorithms is provided. Specifically, we focus on the recovery of codeword errors and errors that affect the pixels within a macroblock.

17.5.2 Error Concealment Algorithms

In general, design of specific error-concealment strategies depends on the system design. For example, if two-layered transmission is used, the receiver should be designed to conceal high-priority error and low-priority error with different strategies. Moreover, if some redundancy ("steering information") could be added to the encoder the concealment could be more efficient. However, we first assume that the encoder is defined for maximum compression efficiency, and that concealment is only performed in the receiver. It should be noted that some exemptions exist for this assumption. These exemptions include the use of I-frame motion vectors, scalability concealment, and limitation of slice length (to perform acceptable concealment in the pixel domain the limitation of slice length exists, i.e., the length of slices cannot be longer than one row of picture). Figure 17.13 shows a block diagram of a generic one/two-tier video decoder with error concealment.

Note that the figure shows two stages of decoder concealment in the codeword domain and pixel domain, respectively. Codeword domain concealment, in which locally generated decodable codewords (e.g., B-picture motion vectors, end-of-block code, etc.) are inserted into the bitstream, is convenient for implementation of simple temporal replacement functions (which in principle can also be performed in the pixel domain). The second stage of pixel domain processing is for temporal and spatial operations not conveniently done in the codeword domain. Advanced spatial processing will generally have to be performed in the pixel domain, although limited codeword domain options can also be identified.

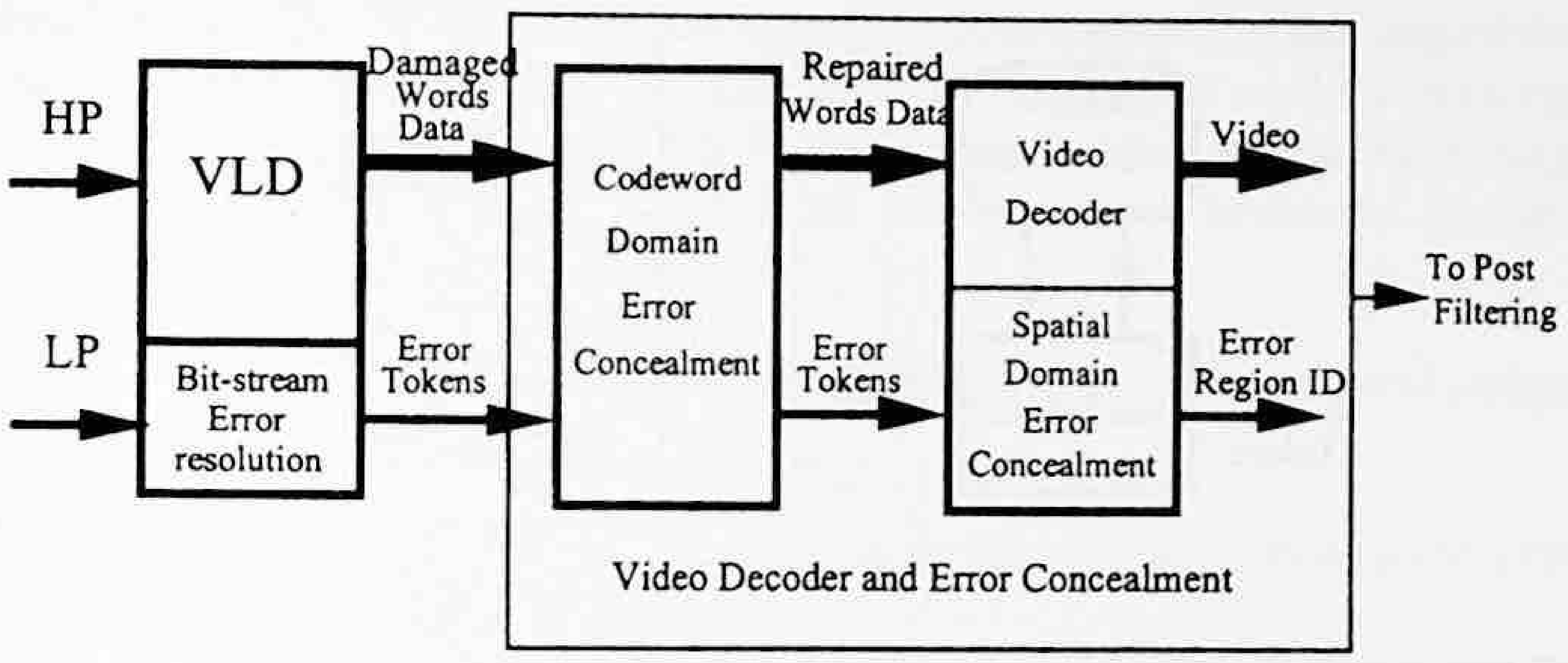

HP
LP
VLD
Bit-stream Error resolution
Damaged Words Data
Error Tokens
Codeword Domain Error Concealment
Repaired Words Data
Error Tokens
Video Decoder
Spatial Domain Error Concealment
Video
Error Region ID
To Post Filtering
Video Decoder and Error Concealment

FIGURE 17.13 MPEG video decoder with error concealment.

17.5.2.1 Codeword Domain Error Concealment

The codeword domain concealment receives video data and error tokens from the transport processor/VLD. Under normal conditions, no action is taken and the data are passed along to the video decoder. When an error token is received, damaged data are repaired to the extent possible by insertion of locally generated codewords and resynchronization codes. An error region ID is also created to indicate the image region to be concealed by subsequent pixel domain processing. Two mechanisms have been used in codeword domain error concealment: neglect the effect of lost data by declaring an end of block (EOB), or replace the lost data with a pseudo-code to handle the macroblock-types or other VLC codes. If high-level data such as dc or macroblock header is lost, the codeword domain concealment with pseudo-codes can only provide signal resynchronization (decodability) and replaces the image scene with a fixed gray level in the error region. Obviously, further improvement is needed in the video decoder. This task is implemented with the error concealment in the video decoder. It is desirable to replace erased I- or P-picture regions with a reasonably accurate estimate to minimize the impact of frame-to-frame propagation.

17.5.2.2 Spatiotemporal Error Concealment

In general, two basic approaches are used for spatial domain error concealment: temporal replacement and spatial interpolation. In temporal replacement, as shown in Figure 17.14, the spatially corresponding ones in the previously decoded data with motion compensation replace the damaged blocks in the current frame if motion information is available. This method exploits temporal redundancy in the reconstructed video signals and provides satisfactory results in areas with small

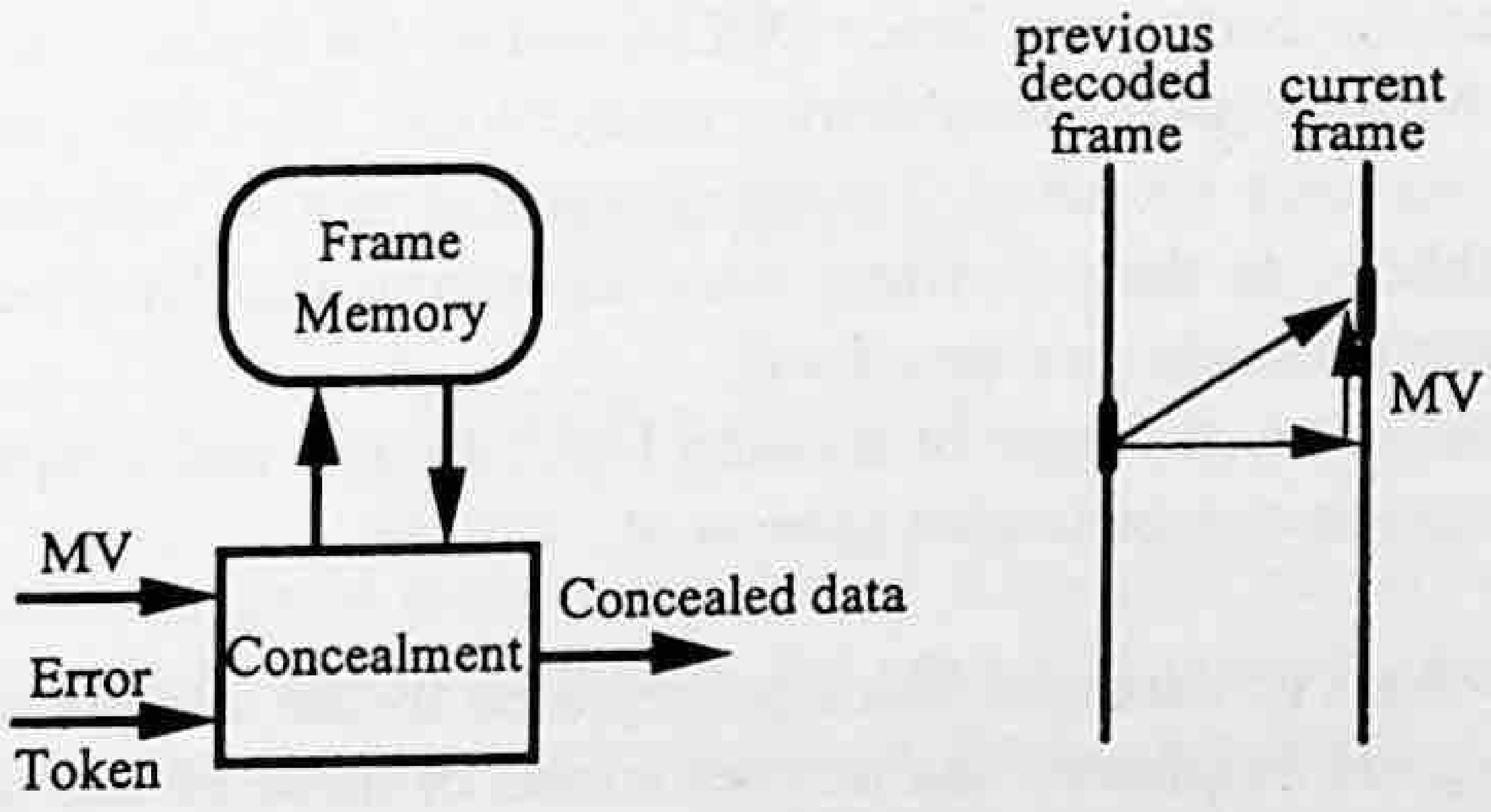


FIGURE 17.14 Error concealment uses temporal replenishment with motion compensation.

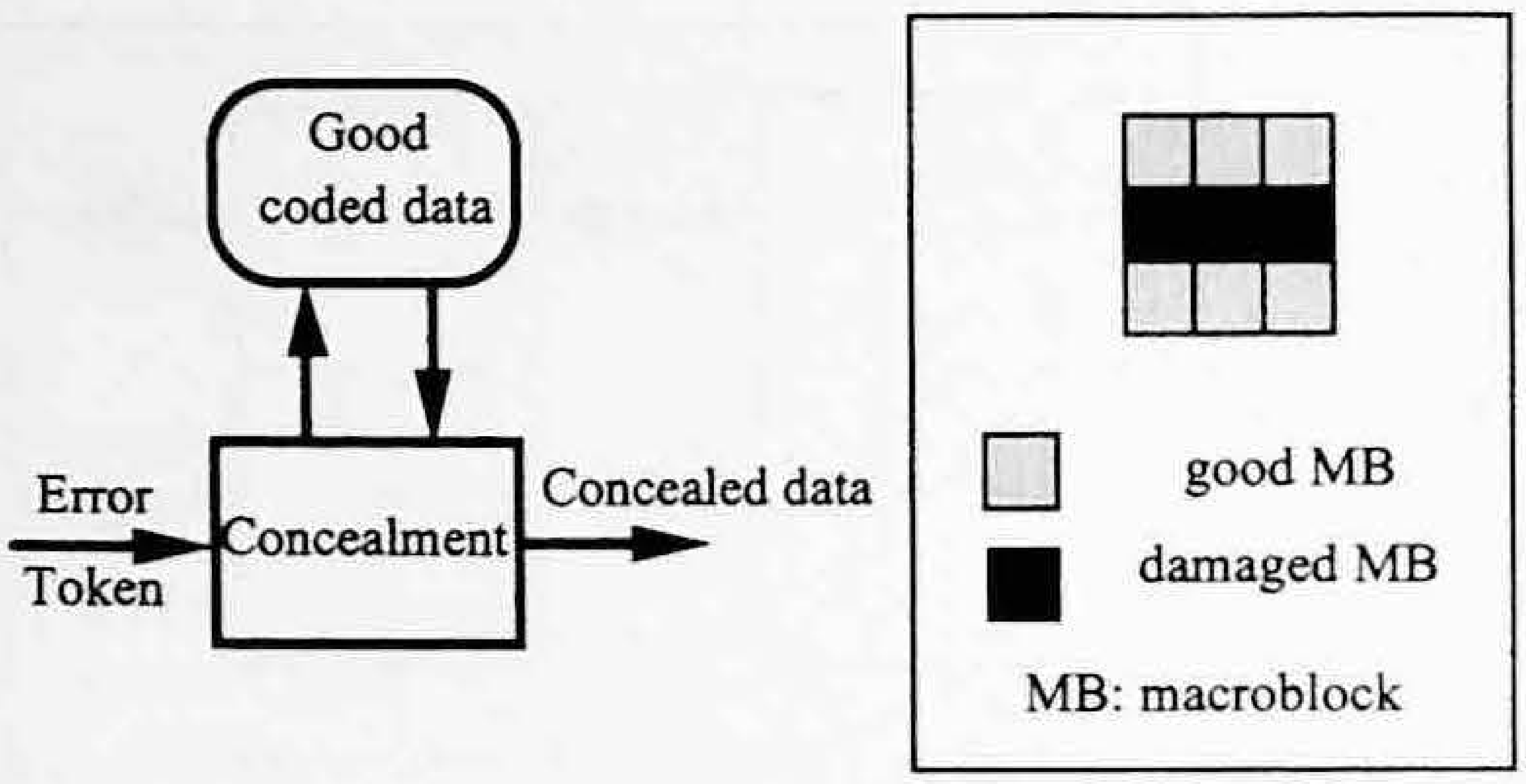


FIGURE 17.15 Error concealment uses spatial interpolation with the data from good neighbors. (From Sun, H. and Kwok, W. *IEEE Trans. Image Proc.*, 4(4), 470–477, 1995. With permission.)

motion and for which motion vectors are provided. If motion information is lost, this method will fail in the moving areas. In the method of spatial interpolation as shown in Figure 17.15, the lost blocks are interpolated by the data from the adjacent nonerror blocks with maximally smooth reconstruction criteria or other techniques.

In this method, the correlation between adjacent blocks in the received and reconstructed video signals is exploited. However, severe blurring will result from this method if data in adjacent blocks are also lost. In an MPEG decoder, temporal replacement outlined above is based on previously decoded anchor (I, P) pictures that are available in the frame memory. If motion vectors corresponding to pixels in the erasure region can also be estimated, this temporal replacement operation can be improved via motion compensation. Also, in the MPEG decoder, groups of video pixels (blocks, macroblocks, or slices) are separately decoded, so that pixel values and motion information corresponding to adjacent picture regions are generally available for spatial concealment. However, estimation from horizontally adjacent blocks may not always be useful since cell loss tends to affect a number of adjacent blocks (due to the MPEG and ATM data structures); also differential encoding between horizontally adjacent blocks tends to limit the utility of data obtained from such neighbors. Therefore, most of the usable spatial information will be located in blocks above or below the damaged region. That is, vertical processing/concealment is found to be most useful due to the transmission order of the data.

For I-pictures, the damaged data can be reconstructed by either temporal replacement from the previously decoded anchor frame or by spatial interpolation from good neighbors. These two methods will be discussed later. For P- and B-pictures, the main strategy to conceal the lost data is to replace the region with pixels from the corresponding (and possibly motion-compensated) location in the previously decoded anchor. In this replacement the motion vectors play a very important role. In other words, if "good" estimates of motion information can be obtained, its use may be the least noticeable correction. Since DPCM coding for motion vectors only exploited the correlations between the horizontally neighboring macroblocks, the redundancy between the vertical neighborhood still exists after encoding. Therefore, the lost motion information can be estimated from the vertical neighbors. In the following, three algorithms that have been developed for error concealment in the video decoder are described.

Algorithm 1: Spatial interpolation of missing I-picture data and temporal replacement for P- and B-pictures with motion compensation (Sun et al., 1992a):

For I-pictures, dc values of damaged block are replaced by the interpolation from the closest top and bottom good neighbors; the ac coefficients of those blocks are synthesized from the dc values of the surrounding neighboring blocks.

For P-pictures, the previously decoded anchor frames with motion compensation replace the lost blocks. The lost motion vectors are estimated by interpolation of the ones from the top and bottom macroblocks. If motion vectors in both top and bottom macroblocks are not available, zero motion vectors are used. The same strategy is used for B-pictures; the only difference is that the closest anchor frame is used. In other words, the damaged part of the B-picture could be replaced by either the forward or backward anchor frame, depending on its temporal position.

Algorithm 2: Temporal replacement of missing I-picture data and temporal replacement for P- and B-pictures with top motion compensation:

For I-pictures, the damaged blocks are replaced with the colocated ones in the previously decoded anchor frame.

For P- and B-pictures, the closest previously decoded anchor frame replaces the damaged part with motion compensation as in the Algorithm 1. The only difference is that the motion vectors are estimated only from the closest top macroblock instead of interpolation of top and bottom motion vectors. This makes the implementation of this scheme much easier. If these motion vectors are not available, then zero motion vectors are used.

In the above two algorithms, the damaged blocks in an I-picture (anchor frame) are concealed by two methods: temporal replacement and spatial interpolation. Temporal replacement is able to provide high-resolution image data to substitute for lost data; however, in motion areas, a big difference might exist between the current intracoded frame and the previously decoded frame. In this case, temporal replacement will produce large shearing distortion unless some motion-based processing can be applied at the decoder. However, this type of processing is not generally available since it is a computationally demanding task to compute motion trajectories locally at the decoder. In contrast, the spatial interpolation approach synthesizes lost data from the adjacent blocks in the same frame. Therefore, the intraframe redundancy between blocks is exploited, while the potential problem of severe blurring due to insufficient high-order ac coefficients for active areas. To alleviate this problem, an adaptive concealment strategy can be used as a compromise; this is described in Algorithm 3.

Algorithm 3: Adaptive spatiotemporal replacement of missing I-picture data and temporal replacement with motion compensation for P- and B-pictures:

For I-pictures, the damaged blocks are concealed with temporal replacement or spatial interpolation according to the decision made by the top and bottom macroblocks, which is shown in Figure 17.16. The decision of which concealment method to use will be based on the more cheaply obtained measures of image activity from the neighboring top and bottom macroblocks. One candidate for the decision processor is to make the decision based on prediction error statistics measured in the neighborhood. The decision region is shown in Figure 17.16, where

$$\begin{aligned} \mathrm{VAR} &= E\left[(x-\hat{x})^2\right], \\ \mathrm{VAROR} &= E\left[x^2\right]-\mu^2, \end{aligned} \tag{17.31}$$

and x is the neighboring good macroblock data, $\hat{x}$ is the data of the corresponding macroblock in the previously decoded frame at the colocated position, and μ is the average value of the neighboring good macroblock data in the current frame. One can appreciate that VAR is indicative of the local motion and VAROR of the local spatial detail. If VAR > VAROR and VAR > T, where T is a preset threshold value which is set to 5 in the experiments, the concealment method is spatial interpolation; if VAR < VAROR or VAR < T, the concealment method is temporal replacement.

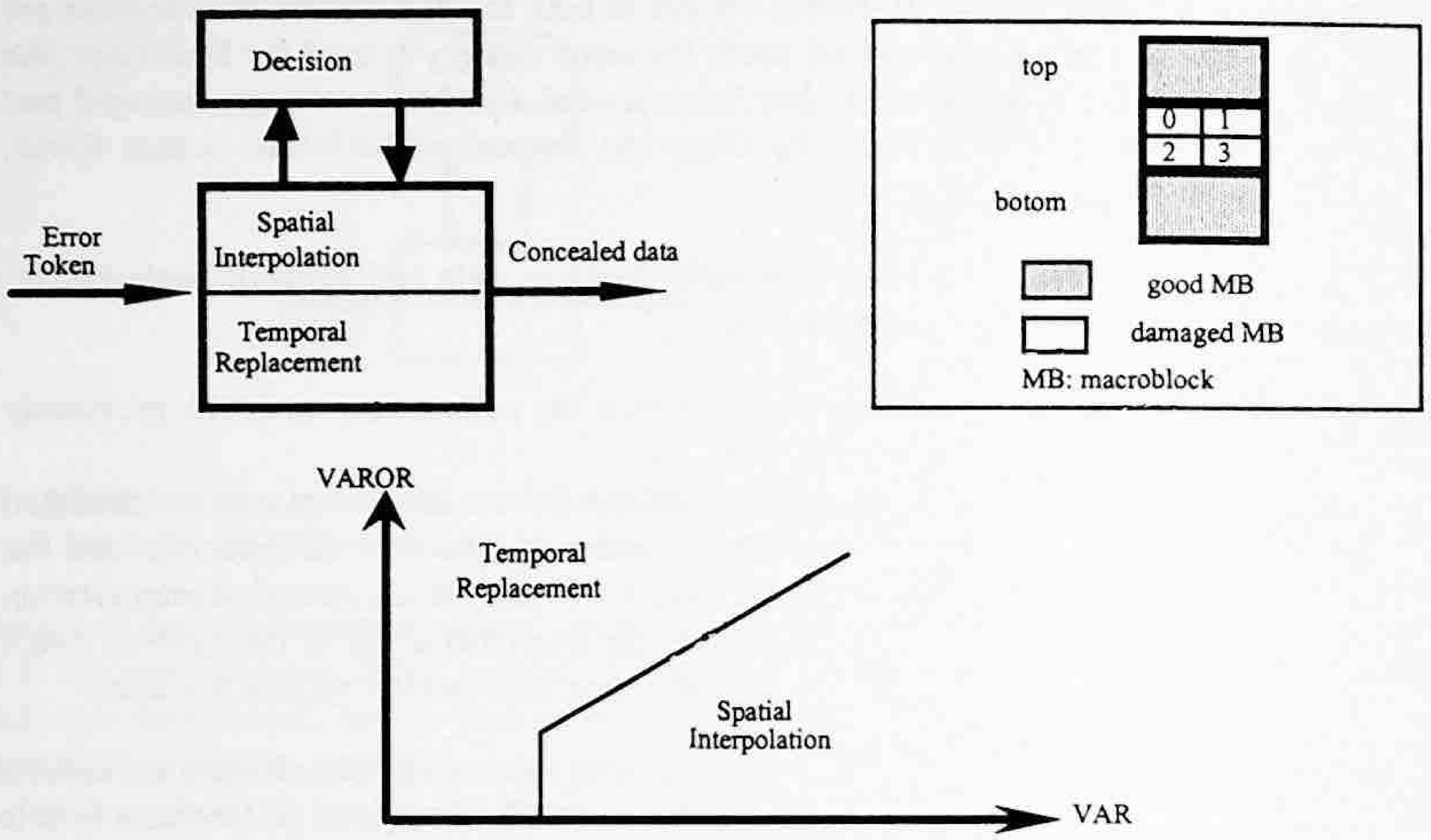


FIGURE 17.16 Adaptive error concealment strategy. (From Sun, H. and Kwok, W., *IEEE Trans. Image Proc.*, 4(4), 470–477, 1995. With permission.)

It should be noted that the concealment for luminance is performed on a block basis instead of macroblock basis, while the chrominance is still on the macroblock basis. The detailed decisions for the luminance blocks are described as follows:

- If both top and bottom are temporally replaced, then four blocks (0, 1, 2, and 3) are replaced by the colocated ones (colocated means no motion compensation) in the previously decoded frame.
- If top is temporally replaced and bottom is spatially interpolated, then blocks 0 and 1 are replaced by the colocated ones in the previously decoded anchor frame and blocks 2 and 3 are interpolated from the block boundaries.
- If top is spatially interpolated and bottom is temporally replaced, then blocks 0 and 1 are interpolated from the boundaries, and blocks 2 and 3 are replaced by the colocated ones in the previously decoded anchor frame.
- If both top and bottom are not temporally replaced, all four blocks are spatially interpolated.

In spatial interpolation, a maximal smoothing technique with boundary conditions under certain smoothness measures is used. The spatial interpolation process is carried out with two steps: the mean value of the damaged block is first bilinearly interpolated with ones from the neighboring blocks; then spatial interpolation for each pixel is performed with a Laplacian operator. Minimizing the Laplacian on the boundary pixels using the iterative process (Wang and Zhu, 1991) enforces the process of maximum smoothness.

For P- and B-pictures a similar concealment method is used as in Algorithm 2 except motion vectors from top and bottom neighboring macroblocks are used for top two blocks and bottom two blocks, respectively.

A schematic block diagram for implementation of adaptive error concealment for intracoded frames is given in Figure 17.17. Corrupted macroblocks are first indicated by error tokens obtained via the transport interface. Then, a decision regarding which concealment method (temporal replacement or spatial interpolation) should be used is based on easily obtained measures of image activity

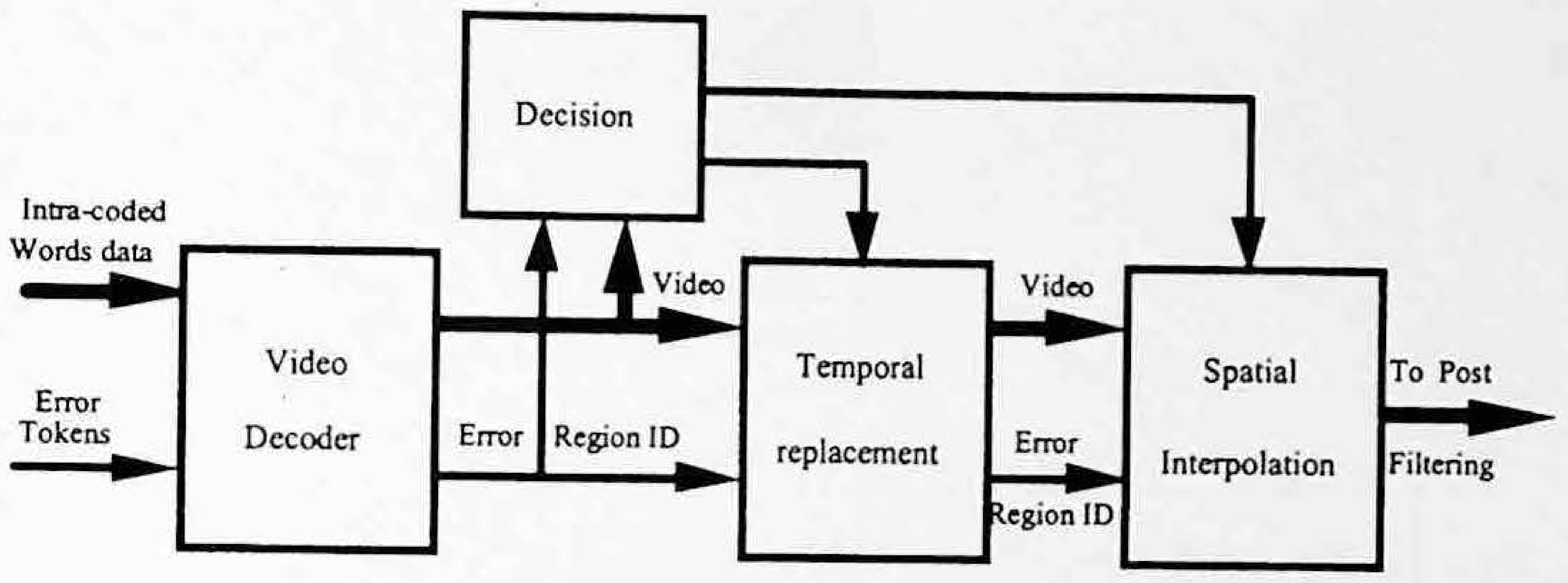


FIGURE 17.17 Two-stage error concealment strategy. (From Sun, H. and Kwok, W., *IEEE Trans. Image Proc.*, 4(4), 470–477, 1995. With permission.)

from the neighboring top and bottom macroblocks. The corrupted macroblocks are first classified into two classes according to the local activities. If local motion is smaller than spatial detail, the corrupted macroblocks are defined as the first class and will be concealed by temporal replacement; when local motion is greater than local spatial detail, the corrupted macroblocks are defined as the second class and will be concealed by spatial interpolation. The overall concealment procedure consists of two stages. First, temporal replacement is applied to all corrupted macroblocks of the first class throughout the whole frame. After the temporal replacement stage, the remaining unconcealed damaged macroblocks of the second class are more likely to be surrounded by valid image macroblocks. A stage of spatial interpolation is then performed on them. This will now result in less blurring, or the blurring will be limited to smaller areas. Therefore, a good compromise between shearing (discontinuity or shift of edge or line) and blurring can be obtained.

17.5.3 Algorithm Enhancements

As discussed above, I-picture errors, which are imperfectly concealed, will tend to propagate through all frames in the group of pictures (GOP). Therefore, it is desirable to develop enhancements for the basic spatiotemporal error concealment technique to improve further the accuracy with which missing I-picture pixels are replaced. Three new algorithms have been developed for this purpose. The first is an extension of the spatial restoration technique outlined earlier, and is based on processing of edge information in a large local neighborhood to obtain better restoration of the missing data. The second and third are variations which involve encoder modifications aimed at improved error concealment performance. Specifically, information such as I-picture pseudo-motion vectors, or low-resolution data in a hierarchical compression system are added in the encoder. These redundancies can significantly benefit error concealment in the decoders that must operate under higher cell loss/error conditions, while having a relatively modest impact on nominal image quality.

17.5.3.1 Directional Interpolation

Improvements in spatial interpolation algorithms (for use with MPEG I-pictures) have been proposed (Kwok and Sun, 1993; Sun and Kwok, 1995). In these studies, additional smoothness criteria and/or directional filtering are used for estimating the picture area to be replaced. The new algorithms utilize spatially correlated edge information from a large local neighborhood of surrounding pixels and perform directional or multidirectional interpolation to restore the missing block. The block diagram illustrating the general principle of the restoration process is shown in Figure 17.18.

Three parts are included in the restoration processing: edge classification, spatial interpolation, and pattern mixing. The function of the classifier is to select the top one, two, or three directions that strongly characterize edge orientations in the surrounding neighborhood. Spatial interpolation

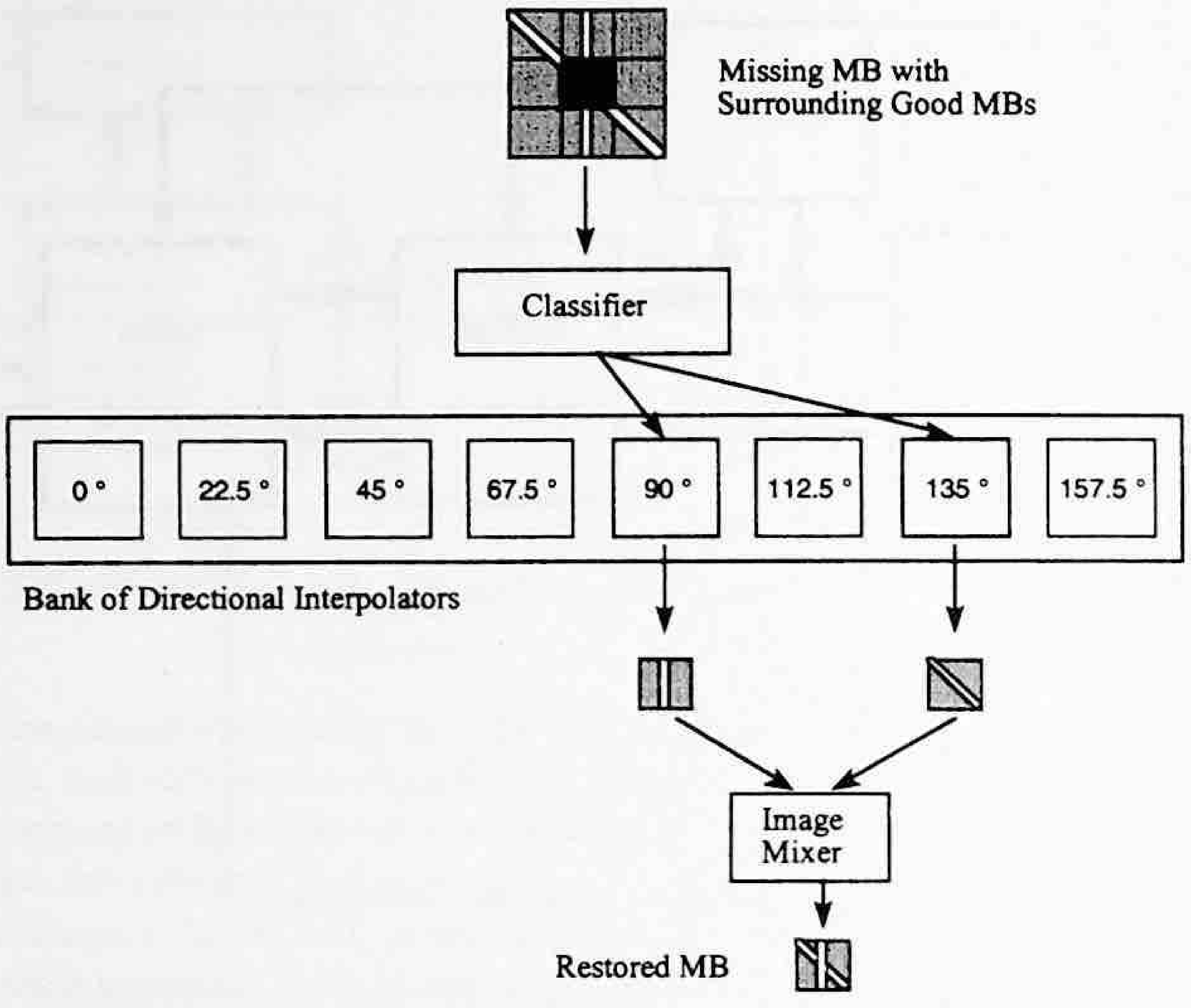


FIGURE 17.18 The multidirectional edge restoration process. (From Sun, H. and Kwok, W., *IEEE Trans. Image Proc.,* 4(4), 470–477, 1995. With permission.)

is performed for each of the directions determined by the classifier. For a given direction, a series of 1-D interpolations are carried out along that direction. All of the missing pixels are interpolated from a weighted average of good neighborhood pixels. The weights depend inversely on the distance from the missing pixel to the good neighborhood pixels. The purpose of pattern mixing is to extract strong characteristic features of two or more images and merge them into one image, which is then used to replace the corrupted one. Results show that these algorithms are capable of providing subjectively better edge restoration in missing areas, and may thus be useful for I-picture processing in high-error-rate scenarios. However, the computational practicality of these edge-filtering techniques needs further investigation for given application scenarios.

17.5.3.2 I-Picture Motion Vectors

Motion information is very useful in concealing losses in P- and B-frames, but is not available for I-pictures. This limits the concealment algorithm to spatial or direct temporal replacement options described above, which may not always be successful in moving areas of the picture. If motion vectors are made available for all MPEG frames (including intracoded ones) as an aid for error concealment (Sun et al., 1992a), good error concealment performance can be obtained without the complexity of adaptive spatial processing. Therefore, a syntax extension has been adopted by the MPEG-2 where motion vectors can be transmitted in an I-picture as the redundancy for error-concealment purposes (Sun et al., 1992b). The macroblock syntax is unchanged, however, motion vectors are interpreted in the following way: the decoded forward motion vectors belong to the macroblock spatially below the current macroblock, and describe how that macroblock can be replaced from the previous anchor frame in the event that the macroblock cannot be recovered. Simulation results have shown that subjective picture quality with I-picture motion vectors is noticeably superior to conventional temporal replacement, and that the overhead for transmitting the additional motion vectors is less than 0.7% of the total bit rate at a bit rate of about 6 to 7 Mbps.

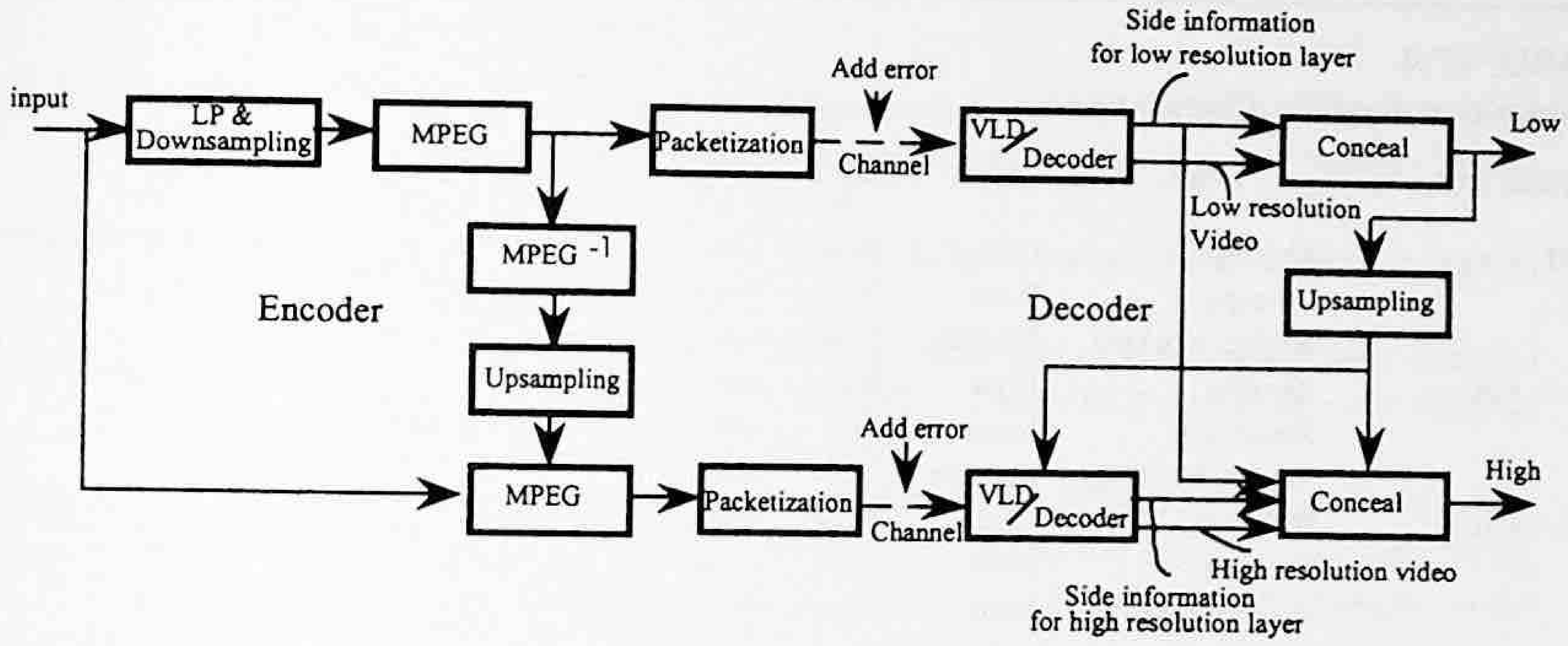


FIGURE 17.19 Block diagram of spatial scalability with error concealment.

17.5.3.3 Spatial Scalable Error Concealment

This approach for error concealment of MPEG video is based on the scalability (or hierarchy) feature of MPEG-2 (ISO/IEC, 1995). Hierarchical transmission provides more possibilities for error concealment, when a corresponding two-tier transmission media is available. A block diagram illustrating the general principle of coding system with spatial scalability and error concealment is shown in Figure 17.19.

It should be noted that the concept of scalable error concealment is different from the two-tier concept with data partitioning. Scalable concealment uses the spatial scalability feature in MPEG-2, while the two-tier case uses the data partitioning feature of MPEG-2, in which the data corresponds to the same spatial resolution layer but is partitioned to two parts with a breakpoint. In spatial scalability, the encoder produces two separate bitstreams: one for the low-resolution base layer and another for the high-resolution enhancement. The high-resolution layer is encoded with an adaptive choice of temporal prediction from previous anchor frames and compatible spatial prediction (obtained from the up-sampled low-resolution layer) corresponding to the current temporal reference. In the decoder, redundancies that exist in the scaling data greatly benefit the error concealment processing. In a simple experiment with spatially scalable MPEG-2, we consider a scenario in which losses in the high-resolution MPEG-2 video are concealed with information from the low-resolution layer. Actually, there are two kinds of information in the lower layer that can be used to conceal the data loss in the high-resolution layer: up-sampled picture data and scaled motion information. Therefore, three error concealment approaches are possible:

1. *Up-sampled substitution*: Lost data are replaced by colocated up-sampled data in the low-resolution decoded frame. The up-sampled picture is obtained from the low-resolution picture with proper up-sampling filter.
2. *Mixed substitution*: Lost macroblocks in I-picture are replaced by colocated up-sampled macroblocks in the low-resolution decoded frame, while lost macroblocks in P- and B-picture are temporally replaced by the previously decoded anchor frame with the motion vectors for the low-resolution layer.
3. *Motion vector substitution*: The previously decoded anchor frame with the motion vectors replaces lost macroblocks for the low-resolution layer appropriately scaled.

Since motion vectors are not available for I-pictures, obviously, method 3 does not work for I-pictures (unless I-picture motion vectors, concealment motion vectors, of MPEG-2 are generated

TABLE 17.4
Subjective Quality Comparison

Picture Material	Items	Alg 1	Alg 2	Alg 3	Comments
Still	Blurring	High	None	Low	Temporal replacement works very well in no-motion areas
	Shearing	None	None	None	
	Artifact blocking	Medium	None	Low	
Slow motion	Blurring	High	None	Low	Temporal replacement works well in slow-motion areas
	Shearing	None	Low	Low	
	Artifact blocking	Medium	None	Low	
Fast motion	Blurring	High	None	Medium	Temporal replacement causes more shearing; spatial interpolation results in blurring; adaptive strategy limits blurring in smaller areas
	Shearing	None	High	Low	
	Artifact blocking	High	Low	Medium	
Overall	The adaptive strategy of steering the temporal replacement and spatial interpolation according to the measures of local activity and local motion gives a good compromise between shearing and blurring				

in encoder). Simulation results have shown that, on average, the up-sampled substitution outperforms the other two, and mixed substitution also provides acceptable results in the case of video with smooth motion.

17.5.4 Summary of Error Concealment

In this section, a general class of error-concealment algorithms for MPEG video has been discussed. The error-concealment approaches that have been described are practical for current MPEG decoder implementations, and have been demonstrated to provide significant robustness. Specifically, it has been shown that the adaptive spatiotemporal algorithm can provide reasonable picture quality at cell loss ratios (CLR) as high as 10^{-3} when used in conjunction. These results confirm that compressed video is far less fragile than originally believed when appropriate transport and concealment techniques are employed. The results can be summarized as in Table 17.4.

Several concealment algorithm extensions based on directional filtering, I-picture pseudo-motion vectors, and MPEG-2 scalability were also considered and shown to provide performance gains that may be useful in certain application scenarios. In view of the practical benefits of robust video delivery, it is recommended that such error resilience functions (along with associated transport structures) be important for implementation in emerging TV, HDTV, teleconferencing, and multimedia systems if the cell loss rates on these transmission systems are significant. Particularly for terrestrial broadcasting and ATM network scenarios, we believe that robust video delivery based on decoder error concealment is an essential element of a viable system design.

17.6 SUMMARY

In this chapter, several application issues of MPEG-2 are discussed. The most successful application of MPEG-2 is the U.S. HDTV standard. The other application issues include transcoding with bitstream scaling, down-conversion decoding, and error concealment. Transcoding is a very interesting topic that converts the bitstreams between different standards. The error concealment is very useful in the noisy communication channels such as terrestrial television broadcasting. The down-conversion decoder responds to the market requirement during the DTV transition-period and long-term need for displaying DTV signals on computer monitors.

17.7 EXERCISES

17-1. In DTV applications, describe the advantages and disadvantages of interlaced format and progressive format. Explain why the computer industry favors progressive format and TV manufacturers like interlaced format.

17-2. Do all DTV formats have square pixel format? Why is square pixel format important for digital television?

17-3. The bitstream scaling is one kind of transcoding; according to your knowledge, describe several other kinds of transcoding (such as MPEG-1 to JPEG) and propose a feasible solution to achieve the transcoding requirements.

17-4. What type of MPEG-2 frames will cause a higher degree of error propagation if errors occur? What technique of error concealment is allowed by the MPEG-2 syntax? Using this technique, perform simulations with several images to determine the penalty in the case of no errors.

17-5. To reduce the drift in a down-conversion decoder, what coding parameters can be chosen at the encoder? Will these actions affect the coding performance?

17-6. What are the advantages and disadvantages of a down-conversion decoder in the frequency domain and spatial domain?

REFERENCES

Bao, J., H. Sun, and T. Poon, HDTV down-conversion decoder, *IEEE Trans. Consumer Elec.*, 42(3), 402-410, 1996.

Boyce, J., J. Henderson, and L. Pearlestien, An SDTV decoder with HDTV capability: an all-format ATV decoder, presented at SMPTE Fall Conference, New Orleans, 1995.

Bruni, R., A. Chimienti, M. Lucenteforte, D. Pau, and R. Sannino, A novel adaptive vector quantization method for memory reduction in MPEG-2 HDTV receivers, *IEEE Trans. Consumer Elec.*, 44(3), 537-544, 1998.

de With, P. H. N., P. H. Frencken, and M. v.d. Schaar-Mitrea, An MPEG decoder with embedded compression for memory reduction, *IEEE Trans. Consumer Elec.*, 44(3), 545-555, 1998.

Ghanbari, M. Two-layer coding of video signals for VBR networks, *IEEE J. Selected Areas Comm.*, 7(5), 771-781, 1989.

Gonzalez, R. C. and P. Wintz, *Digital Image Processing*, 2nd ed., Addison-Wesley, Reading, MA, 1987, 232-233.

Grand Alliance, HDTV System Specification Version 2.0, December 7, 1994.

Harthanck, W., W. Keesen, and D. Westerkamp, Concealment techniques for block encoded TV-signals, presented at Picture Coding Symposium, 1986.

Isnardi, M. A. Consumers seek easy-to-use products, *IEEE Spect.*, 64, Jan. 1993.

ISO/IEC, MPEG Test Model 5, ISO/IEC JTC/SC29/WG11 Document. April, 1993.

ISO/IEC, MPEG-2 International Standard. Video Recommendation ITU-T H.262, ISO/IEC 13818-2, Jan. 10, 1995.

Jayant, N. N. and P. Noll, *Digital Coding of Waveforms to Speech and Video*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

Jeng, F.-C. and S. H. Lee, Concealment of bit error and cell loss in inter-frame coded video transmission, *ICC Proceeding*, ICC'91, 496-500, 1991.

Joseph, K., S. Ng, D. Raychaudhuri, R. Saint Girons, T. Savatier, R. Siracusa, and J. Zdepski, MPEG++: A robust compression and transport system for digital HDTV, *Signal Proc. Image Comm.*, 4, 307-323, 1992a.

Joseph, K., S. Ng, D. Raychaudhuri, R. Saint Girons, R. Siracusa, and J. Zdepski, Prioritization and transport in the ADTV digital simulcast system, *Proceedings ICCE '92*, 1992b.

Karlsson, G. and M. Vetterli, Packet video and its integration into the network architecture, *IEEE J. Selected Areas Communication*, 739-751, 1989.

Kishino, F., K. Manabe, Y. Hayashi, and H. Yasuda, Variable bit-rate coding of video signals for ATM networks, *IEEE J. Selected Areas Comm.*, 7(5), 801-806, 1989.

Kwok, W. and H. Sun, Multi-directional interpolation for spatial error concealment, *IEEE Trans. Consumer Elec.*, 455-460, 1993.

Lancaster, P. and M. Tismenetsky, *The Theory of Matrices with Application*, Academic Press, Boston, 1985.

Lei, S., A quadtree embedded compression algorithm for memory saving DTV decoders, *Proceedings International Conference on Consumer Electronics*, Los Angeles, CA, June 1999.

Merhav, N. and V. Bhaskaran, Fast algorithms for DCT-domain image down-sampling and for inverse motion compensation, *IEEE Trans. Circ. Syst. Video Technol.*, 7(3), 468-476, 1997.

Mokry, R. and D. Anastassiou, Minimal error drift in frequency scalability for motion-compensated DCT coding, *IEEE Trans. Circ. Syst. Video Technol.*, 4(4), 392-406, 1994.

Ng, S. Thompson Consumer Electronics, Low Resolution HDTV Receivers, U.S. patent 5,262,854, Nov. 16, 1993.

Pang, K. K., H. G. Lim, S. Dunstan, and J. M. Badcock, Frequency domain decimation and interpolation techniques, presented at Picture Coding Symposium, Melbourne, Australia, March 1996.

Perkins, M. and D. Arnstein. Statistical multiplexing of multiple MPEG-2 video programs in a single channel, *SMPTE J.*, September 1995.

Reitmeier, G. A. The U.S. advanced television standard and its impact on VLSI, *VLSI and Signal Processing, Systems for Signal, Image, and Video Technology*, Kluwer Academic Press, 1997.

Siracusa, R., K. Joseph, J. Zdepski, and D. Raychaudhuri, Flexible and robust packet transport for digital HDTV, *IEEE J. Selected Areas Comm.*, 11(1), ISACEM, 88-98, 1993.

Sun, H. Hierarchical decoder for MPEG compressed video data *IEEE Trans. Consumer Elec.*, 39(3), 559-562, 1993.

Sun, H., K. Challapali, and J. Zdepski, Error concealment in simulcast AD-HDTV decoder, *IEEE Trans. Consumer Elec.*, 38(3), 108-118, 1992.

Sun, H., M. Uz, J. Zdepski, and R. Saint Girons, A Proposal for Increased Error Resilience, ISO-IEC/JTC1?SC29/WG11, MPEG92, Sept. 30, 1992b.

Sun, H. and W. Kwok, Restoration of damaged block transform coded image using projection onto convex sets, *IEEE Trans. Image Process.*, 4(4), IIPRE4, 470-477, 1995.

Vetro, A. and H. Sun, On the motion compensation within a down-conversion decoder, *J. Elec. Imag.*, 7(3), 1998a.

Vetro, A. and H. Sun, Frequency domain down-conversion using an optimal motion compensation scheme, *J. Imag. Sci. Technol.*, 9(4), 1998b.

Vetro, A., H. Sun, P. DaGraca, and T. Poon, Minimum drift architectures for three-layer scalable DTV decoding, *IEEE Trans. Consumer Elec.*, 44(3), 1998.

Wang, Y. and Q.-F. Zhu, Signal loss recovery in DCT-based image and video codecs, *Proceedings of SPIE on Visual Communication and Image Processing*, Boston, 667-678, Nov. 1991.

Yu, H., W.-M. Lam, B. Canfield, and B. Beyers, Block-based image processor for memory efficient MPEG video decoding, *Proceedings International Conference on Consumer Electronics*, Los Angeles, CA, June 1999.

Zdepski, J. et al., Packet transport of rate-free interframe DCT compressed digital video on a CSMA/CD LAN, *Proceedings IEEE Global Conference on Communications*, Dallas TX, Nov. 1989.

Zdepski, J. et al., Prioritized Packet Transport of VBR CCITT H.261 Format Compressed Video on a CSMA/CD LAN, Third International Workshop on Packet Video, Morristown, NJ, March 22-23, 1990.

18 MPEG-4 Video Standard: Content-Based Video Coding

This chapter provides an overview of the ISO MPEG-4 standard. The MPEG-4 work includes natural video, synthetic video, audio and systems. Both natural and synthetic video have been combined into a single part of the standard, which is referred to as MPEG-4 visual (ISO/IEC, 1998a). It should be emphasized that neither MPEG-1 nor MPEG-2 considers synthetic video (or computer graphics) and the MPEG-4 is also the first standard to consider the problem of content-based coding. Here, we focus on the video parts of the MPEG-4 standard.

18.1 INTRODUCTION

As we discussed in the previous chapters, MPEG has completed two standards: MPEG-1 that was mainly targeted for CD-ROM applications up to 1.5 Mbps and MPEG-2 for digital TV and HDTV applications at bit rates between 2 and 30 Mbps. In July 1993, MPEG started its new project, MPEG-4, which was targeted at providing technology for multimedia applications. The first working draft (WD) was completed in November 1996, and the committee draft (CD) of version 1 was completed in November 1997. The draft international standard (DIS) of MPEG-4 was completed in November of 1998, and the international standard (IS) of MPEG-4 version 1 was completed in February of 1999. The goal of the MPEG-4 standard is to provide the core technology that allows efficient content-based storage, transmission, and manipulation of video, graphics, audio, and other data within a multimedia environment. As we mentioned before, there exist several video-coding standards such as MPEG-1/2, H.261, and H.263. Why do we need a new standard for multimedia applications? In other words, are there any new attractive features of MPEG-4 that the current standards do not have or cannot provide? The answer is yes. The MPEG-4 has many interesting features that will be described later in this chapter. Some of these features are focused on improving coding efficiency; some are used to provide robustness of transmission and interactivity with the end user. However, among these features the most important one is the content-based coding. MPEG-4 is the first standard that supports content-based coding of audio visual objects. For content providers or authors, the MPEG-4 standard can provide greater reusability, flexibility, and manageability of the content that is produced. For network providers, MPEG-4 will offer transparent information, which can be interpreted and translated into the appropriate native signaling messages of each network. This can be accomplished with the help of relevant standards bodies that have the jurisdiction. For end users, MPEG-4 can provide much functionality to make the user terminal have more capabilities of interaction with the content. To reach these goals, MPEG-4 has the following important features:

The contents such as audio, video, or data are represented in the form of primitive audio visual objects (AVOs). These AVOs can be natural scenes or sounds, which are recorded by video camera or synthetically generated by computers.

The AVOs can be composed together to create compound AVOs or scenes.

The data associated with AVOs can be multiplexed and synchronized so that they can be transported through network channels with certain quality requirements.

18.2 MPEG-4 REQUIREMENTS AND FUNCTIONALITIES

Since the MPEG-4 standard is mainly targeted at multimedia applications, there are many requirements to ensure that several important features and functionalities are offered. These features include the allowance of interactivity, high compression, universal accessibility, and portability of audio and video content. From the MPEG-4 video requirement document, the main functionalities can be summarized by the following three aspects: content-based interactivity, content-based efficient compression, and universal access.

18.2.1 Content-Based Interactivity

In addition to provisions for efficient coding of conventional video sequences, MPEG-4 video has the following features of content-based interactivity.

18.2.1.1 Content-Based Manipulation and Bitstream Editing

The MPEG-4 supports the content-based manipulation and bitstream coding without the need for transcoding. In MPEG-1 and MPEG-2, there is no syntax and no semantics for supporting true manipulation and editing in the compressed domain. MPEG-4 provides the syntax and techniques to support content-based manipulation and bitstream editing. The level of access, editing, and manipulation can be done at the object level in connection with the features of content-based scalability.

18.2.1.2 Synthetic and Natural Hybrid Coding (SNHC)

The MPEG-4 supports combining synthetic scenes or objects with natural scenes or objects. This is for "compositing" synthetic data with ordinary video, allowing for interactivity. The related techniques in MPEG-4 for supporting this feature include sprite coding, efficient coding of 2-D and 3-D surfaces, and wavelet coding for still textures.

18.2.1.3 Improved Temporal Random Access

The MPEG-4 provides and efficient method to access randomly, within a limited time, and with the fine resolution parts, e.g., video frames or arbitrarily shaped image objects from an audiovisual sequence. This includes conventional random access at very low bit rate. This feature is also important for content-based bitstream manipulation and editing.

18.2.2 Content-Based Efficient Compression

One initial goal of MPEG-4 is to provide a highly efficient coding tool with high compression at very low bit rates. But this goal has now extended to a large range of bit rates from 10 Kbps to 5 Mbps, which covers QSIF to CCIR601 video formats. Two important items are included in this requirement.

18.2.2.1 Improved Coding Efficiency

The MPEG-4 video standard provides subjectively better visual quality at comparable bit rates compared with the existing or emerging standards, including MPEG-1/2 and H.263. MPEG-4 video contains many new tools, which optimize the code in different bit rate ranges. Some experimental results have shown that it outperforms MPEG-2 and H.263 at the low bit rates. Also, the content-based coding reaches the similar performance of the frame-based coding.

18.2.2.2 Coding of Multiple Concurrent Data Streams

The MPEG-4 provides the capability of coding multiple views of a scene efficiently. For stereoscopic video applications, MPEG-4 allows the ability to exploit redundancy in multiple viewing points of the same scene, permitting joint coding solutions that allow compatibility with normal video as well as the ones without compatibility constraints.

18.2.3 Universal Access

The another important feature of the MPEG-4 video is the feature of universal access.

18.2.3.1 Robustness in Error-Prone Environments

The MPEG-4 video provides strong error robustness capabilities to allow access to applications over a variety of wireless and wired networks and storage media. Sufficient error robustness is provided for low-bit-rate applications under severe error conditions (e.g., long error bursts).

18.2.3.2 Content-Based Scalability

The MPEG-4 video provides the ability to achieve scalability with fine granularity in content, quality (e.g., spatial and temporal resolution), and complexity. These scalabilities are especially intended to result in content-based scaling of visual information.

18.2.4 Summary of MPEG-4 Features

From above description of MPEG-4 features, it is obvious that the most important application of MPEG-4 will be in a multimedia environment. The media that can use the coding tools of MPEG-4 include computer networks, wireless communication networks, and the Internet. Although it can also be used for satellite, terrestrial broadcasting, and cable TV, these are still the territories of MPEG-2 video since MPEG-2 already has made such a large impact in the market. A large number of silicon solutions exist and its technology is more mature compared with the current MPEG-4 standard. From the viewpoint of coding theory, we can say there is no significant breakthrough in MPEG-4 video compared with MPEG-2 video. Therefore, we cannot expect to have a significant improvement of coding efficiency when using MPEG-4 video over MPEG-2. Even though MPEG-4 optimizes its performance in a certain range of bit rates, its major strength is that it provides more functionality than MPEG-2. Recently, MPEG-4 added the necessary tools to support interlaced material. With this addition, MPEG-4 video does support all functionalities already provided by MPEG-1 and MPEG-2, including the provision to compress efficiently standard rectangular-sized video at different levels of input formats, frame rates, and bit rates.

Overall, the incorporation of an object- or content-based coding structure is the feature that allows MPEG-4 to provide more functionality. It enables MPEG-4 to provide the most elementary mechanism for interactivity and manipulation with objects of images or video in the compressed domain without the need for further segmentation or transcoding at the receiver, since the receiver can receive separate bitstreams for different objects contained in the video. To achieve content-based coding, the MPEG-4 uses the concept of a video object plane (VOP). It is assumed that each frame of an input video is first segmented into a set of arbitrarily shaped regions or VOPs. Each such region could cover a particular image or video object in the scene. Therefore, the input to the MPEG-4 encoder can be a VOP, and the shape and the location of the VOP can vary from frame to frame. A sequence of VOPs is referred to as a video object (VO). The different VOs may be encoded into separate bitstreams. MPEG-4 specifies demultiplexing and composition syntax which provide the tools for the receiver to decode the separate VO bitstreams and composite them into a

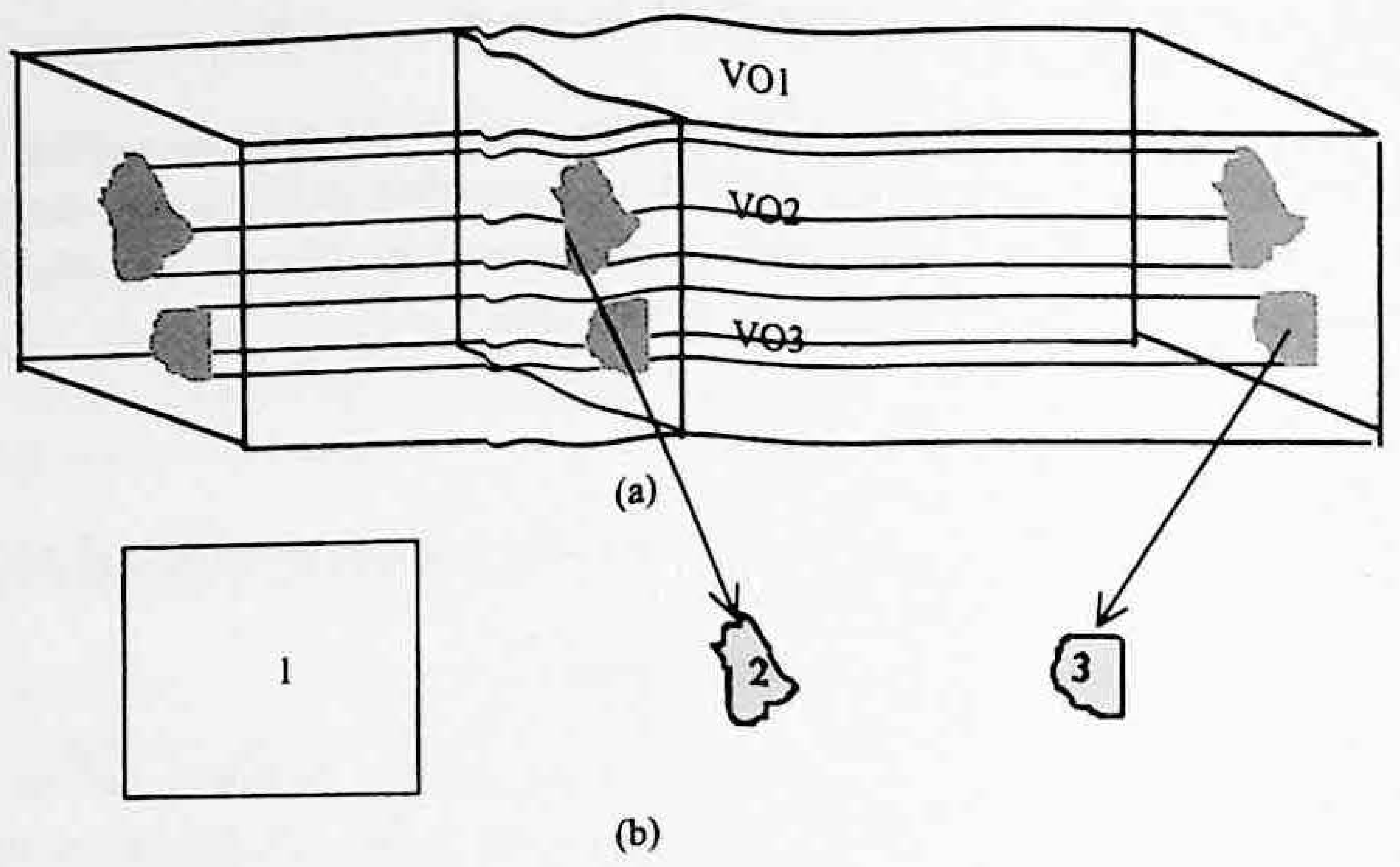


FIGURE 18.1 Video object definition and format: (a) video object, (b) VOPs.

frame. In this way, the decoders have more flexibility to edit or rearrange the decoded video objects. The detailed technical issues will be addressed in the following sections.

18.3 TECHNICAL DESCRIPTION OF MPEG-4 VIDEO

18.3.1 Overview of MPEG-4 Video

The major feature of MPEG-4 is to provide the technology for object-based compression, which is capable of separately encoding and decoding video objects. To explain the idea of object-based coding clearly, we should review the set of video object-related definitions. An image scene may contain several objects. In the example of Figure 18.1, the scene contains the background and two objects. The time instant of each video object is referred to as the VOP. The concept of a VO provides a number of functionalities of MPEG-4, which are either impossible or very difficult in MPEG-1 or MPEG-2 video coding. Each video object is described by the information of texture, shape, and motion vectors. The video sequence can be encoded in a way that will allow the separate decoding and reconstruction of the objects and allow the editing and manipulation of the original scene by simple operation on the compressed bitstream domain. The feature of object-based coding is also able to support functionality such as warping of synthetic or natural text, textures, image, and video overlays on reconstructed video objects.

Since MPEG-4 aims at providing coding tools for multimedia environments, these tools not only allow one to compress natural video objects efficiently, but also to compress synthetic objects, which are a subset of the larger class of computer graphics. The tools of MPEG-4 video includes the following:

- Motion estimation and compensation
- Texture coding
- Shape coding
- Sprite coding
- Interlaced video coding
- Wavelet-based texture coding
- Generalized temporal and spatial as well as hybrid scalability
- Error resilience.

The technical details of these tools will be explained in the following sections.

18.3.2 Motion Estimation and Compensation

For object-based coding, the coding task includes two parts: texture coding and shape coding. The current MPEG-4 video texture coding is still based on the combination of motion-compensated prediction and transform coding. Motion-compensated predictive coding is a well-known approach for video coding. Motion compensation is used to remove interframe redundancy, and transform coding is used to remove intraframe redundancy, as in the MPEG-2 video-coding scheme. However, there are lots of modifications and technical details in MPEG-4 for coding a very wide range of bit rates. Moreover, MPEG-4 coding has been optimized for low-bit-rate applications with a number of new tools. In other words, MPEG-4 video coding uses the most common coding technologies, such as motion compensation and transform coding, but at the same time, it modifies some traditional methods such as advanced motion compensation and also creates some new features, such as sprite coding.

The basic technique to perform motion-compensated predictive coding for coding a video sequence is motion estimation (ME). The basic ME method used in the MPEG-4 video coding is still the block-matching technique. The basic principle of block matching for motion estimation is to find the best-matched block in the previous frame for every block in the current frame. The displacement of the best-matched block relative to the current block is referred to as the motion vector (MV). Positive values for both motion vector components indicate that the best-matched block is on the bottom right of the current block. The motion-compensated prediction difference block is formed by subtracting the pixel values of the best-matched block from the current block, pixel by pixel. The difference block is then coded by a texture-coding method. In MPEG-4 video coding, the basic technique of texture coding is a discrete cosine transformation (DCT). The coded motion vector information and difference block information is contained in the compressed bitstream, which is transmitted to the decoder. The major issues in the motion estimation and compensation are the same as in the MPEG-1 and MPEG-2 which include the matching criterion, the size of search window (searching range), the size of matching block, the accuracy of motion vectors (one pixel or half-pixel), and inter/intramode decision. We are not going to repeat these topics and will focus on the new features in the MPEG-4 video coding. The feature of the advanced motion prediction is a new tool of MPEG-4 video. This feature includes two aspects: adaptive selection of 16×16 block or four 8×8 blocks to match the current 16×16 block and overlapped motion compensation for luminance block.

18.3.2.1 Adaptive Selection of 16 × 16 Block or Four 8 × 8 Blocks

The purpose of the adaptive selection of the matching block size is to enhance coding efficiency further. The coding performance may be improved at low bit rate since the bits for coding prediction difference could be greatly reduced at the limited extra cost for increasing motion vectors. Of course, if the cost of coding motion vectors is too high, this method will not work. The decision in the encoder should be very careful. For explaining the procedure of how to make decisions, we define $\{C(i,j),\ i,j = 0, 1,\ldots, N-1\}$ to be the pixels of the current block and $\{P(i,j),\ i,j = 0, 1, \ldots, N-1\}$ to be the pixels in the search window in the previous frame. The sum of absolute difference (SAD) is calculated as

$$SAD_N(x,y) = \begin{cases} \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \left|C(i,j) - P(i,j)\right| - T & \text{if}(x,y) = (0,0) \\ \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \left|C(i,j) - P(i+x,j+y)\right| & \text{otherwise,} \end{cases} \tag{18.1}$$

where (x, y) is the pixel within the range of searching window, and T is a positive constant. The following steps then make the decision:

Step 1: To find $SAD_{16}(MV_x, MV_y)$;
Step 2: To find $SAD_8(MV1_x, MV1_y)$, $SAD_8(MV2_x, MV2_y)$, $SAD_8(MV3_x, MV3_y)$, and $SAD_8(MV4_x, MV4_y)$;
Step 3: If

$$\sum_{i=1}^{4} SAD_8\left(MV_{ix}, MV_{iy}\right) < SAD_{16}\left(MV_x, MV_y\right) - 128,$$

then choose 8×8 prediction; otherwise, choose 16×16 prediction.

If the 8×8 prediction is chosen, there are four motion vectors for the four 8×8 luminance blocks that will be transmitted. The motion vector for the two chrominance blocks is then obtained by taking an average of these four motion vectors and dividing the average value by a factor of two. Since each motion vector for the 8×8 luminance block has half-pixel accuracy, the motion vector for the chrominance block may have a sixteenth pixel accuracy.

18.3.2.2 Overlapped Motion Compensation

This kind of motion compensation is always used for the case of four 8×8 blocks. The case of one motion vector for a 16×16 block can be considered as having four identical 8×8 motion vectors, each for an 8×8 block. Each pixel in an 8×8 of the best-matched luminance block is a weighted sum of three prediction values specified in the following equation:

$$p'(i,j) = \left(H_0(i,j) \cdot q(i,j) + H_1(i,j) \cdot r(i,j) + H_2(i,j) \cdot s(i,j)\right)/8, \tag{18.2}$$

where division is with round-off. The weighting matrices are specified as:

$$H_0 = \begin{bmatrix} 4 & 5 & 5 & 5 & 5 & 5 & 5 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 6 & 6 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 4 & 5 & 5 & 5 & 5 & 5 & 5 & 4 \end{bmatrix}, \quad H_1 = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix}, \text{ and}$$

$$H_2 = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \end{bmatrix}$$

It is noted that $H_0(i,j) + H_1(i,j) + H_2(i,j) = 8$ for all possible (i,j). The value of $q(i,j)$, $r(i,j)$, and $s(i,j)$ are the values of the pixels in the previous frame at the locations,

$$
\begin{aligned}
q(i,j) &= p\left(i + MV_x^0, j + MV_y^0\right),\\
r(i,j) &= p\left(i + MV_x^1, j + MV_y^1\right),\\
s(i,j) &= p\left(i + MV_x^2, j + MV_y^2\right),
\end{aligned}
\tag{18.3}
$$

where (MV_x^0, MV_y^0) is the motion vector of the current 8×8 luminance block $p(i,j)$, (MV_x^1, MV_y^1) is the motion vector of the block either above (for $j = 0,1,2,3$) or below (for $j = 4,5,6,7$) the current block and (MV_x^2, MV_y^2) is the motion vector of the block either to the left (for $i = 0,1,2,3$) or right (for $i = 4,5,6,7$) of the current block. The overlapped motion compensation can reduce the prediction noise at a certain level.

18.3.3 Texture Coding

Texture coding is used to code the intra-VOPs and the prediction residual data after motion compensation. The algorithm for video texture coding is based on the conventional 8×8 DCT with motion compensation. DCT is performed for each luminance and chrominance block, where the motion compensation is performed only on the luminance blocks. This algorithm is similar to those in H.263 and MPEG-1 as well as MPEG-2. However, MPEG-4 video texture coding has to deal with the requirement of object-based coding, which is not included in the other video-coding standards. In the following we will focus on the new features of the MPEG-4 video coding. These new features include the intra-DC and AC prediction for I-VOP and P-VOP, the algorithm of motion estimation and compensation for arbitrary shape VOP, and the strategy of arbitrary shape texture coding. The definitions of I-VOP, P-VOP, and B-VOP are similar to the I-picture, P-picture, and B-picture in Chapter 16 for MPEG-1 and MPEG-2.

18.3.3.1 Intra-DC and AC Prediction

In the intramode coding, the predictive coding is not only applied on the DC coefficients but also the AC coefficients to increase the coding efficiency. The adaptive DC prediction involves the selection of the quantized DC (QDC) value of the immediately left block or the immediately above block. The selection criterion is based on comparison of the horizontal and vertical DC gradients around the block to be coded. Figure 18.2 shows the three surrounding blocks "A," "B," and "C" to the current block "X" whose QDC is to be coded where block "A", "B," and "C" are the immediately left, immediately left and above, and immediately above block to the "X," respectively. The QDC value of block "X," QDC_X, is predicted by either the QDC value of block "A," QDC_A,

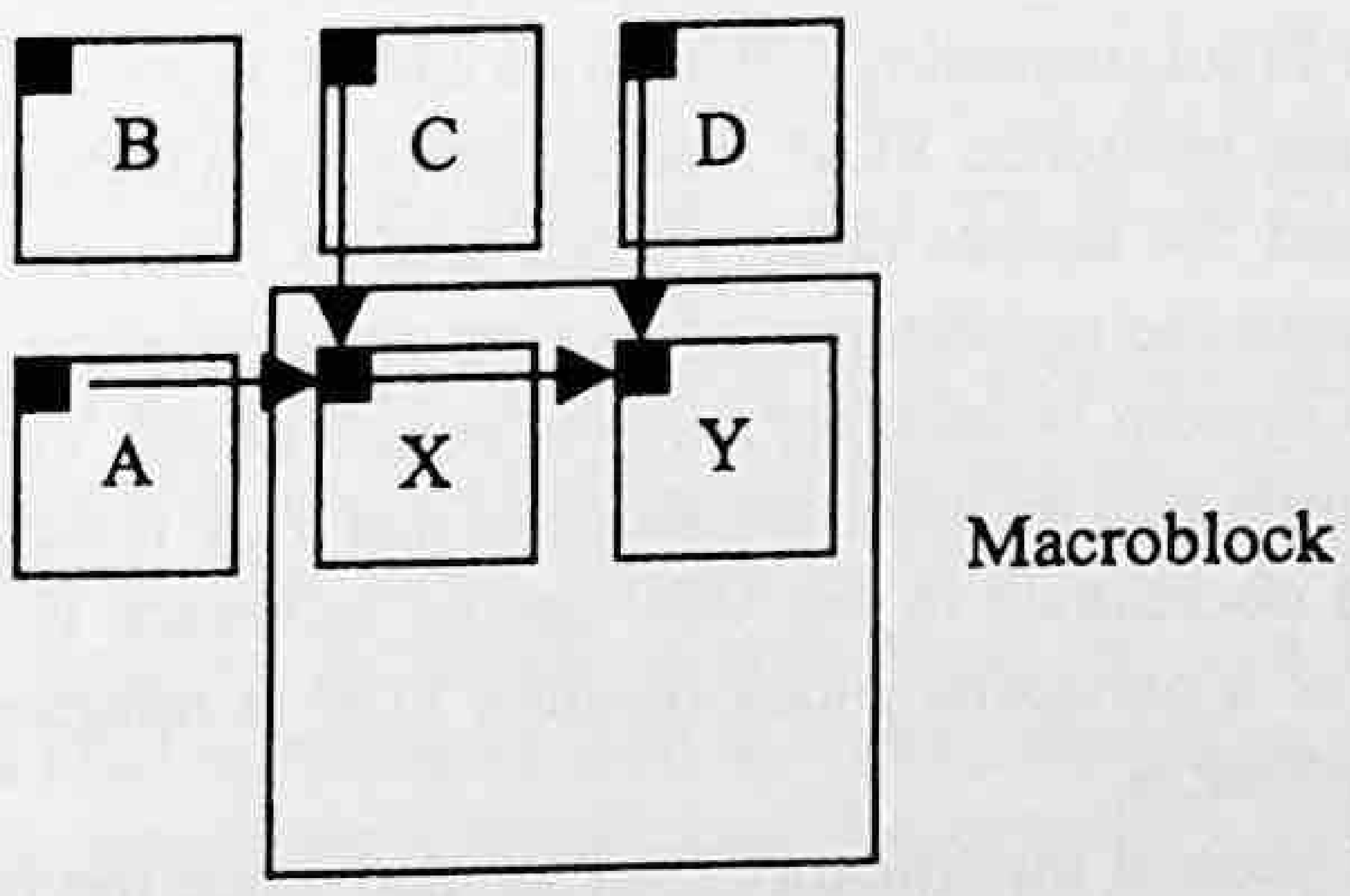


FIGURE 18.2 Previous neighboring blocks used in DC prediction. (From ISO/IEC 14496-2 Video Verification Model V.12, N2552, Dec. 1998. With permission.)

or the QDC value of block "C," QDC_C, based on the comparison of horizontal and vertical gradients as follows:

$$\begin{array}{ll} \text{If} & \left|QDC_A - QDC_B\right| < \left|QDC_B - QDC_C\right|, \quad QDC_P = QDC_C; \\ \text{Otherwise} & QDC_P = QDC_A. \end{array} \tag{18.4}$$

The differential DC is then obtained by subtracting the DC prediction, QDC_P, from QDC_X. If any of block "A", "B," or "C" are outside of the VOP boundary, or they do not belong to an intracoded block, their QDC value are assumed to take a value of 128 (if the pixel is quantized to 8 bits) for computing the prediction. The DC prediction is performed similarly for the luminance and each or the two chrominance blocks.

For AC coefficient prediction, either coefficients from the first row or the first column of a previous coded block are used to predict the cosited (same position in the block) coefficients in the current block. On a block basis, the same rule for selecting the best predictive direction (vertical or horizontal direction) for DC coefficients is also used for the AC coefficient prediction. A difference between DC prediction and AC prediction is the issue of quantization scale. All DC values are quantized to the 8 bits for all blocks. However, the AC coefficients may be quantized by the different quantization scales for the different blocks. To compensate for differences in the quantization of the blocks used for prediction, scaling of prediction coefficients becomes necessary. The prediction is scaled by the ratio of the current quantization step size and the quantization step size of the block used for prediction. In the cases when AC coefficient prediction results in a larger range of prediction errors as compared with the original signal, it is desirable to disable the AC prediction. The decision of AC prediction switched on or off is performed on a macroblock basis instead of a block basis to avoid excessive overhead. The decision for switching on or off AC prediction is based on a comparison of the sum of the absolute values of all AC coefficients to be predicted in a macroblock and that of their predicted differences. It should be noted that the same DC and AC prediction algorithm is used for the intrablocks in the intercoded VOP. If any blocks used for prediction are not intrablocks, the QDC and QAC values used for prediction are set to 128 and 0 for DC and AC prediction, respectively.

18.3.3.2 Motion Estimation/Compensation of Arbitrarily Shaped VOP

In previous sections we discussed the general issues of motion estimation (ME) and motion compensation (MC). Here we are going to discuss the ME and MC for coding the texture in the arbitrarily shaped VOP. In an arbitrarily shaped VOP, the shape information is given by either binary shape information or alpha components of a gray-level shape information. If the shape information is available to both encoder and decoder, three important modifications have to be considered for the arbitrarily shaped VOP. The first is for the blocks, which are located in the border of VOP. For these boundary blocks, the block-matching criterion should be modified. Second, a special padding technique is required for the reference VOP. Finally, since the VOPs have arbitrary shapes rather than rectangular shapes, and the shapes change from time to time, an agreement on a coordinate system is necessary to ensure the consistency of motion compensation. At the MPEG-4 video, the absolute frame coordinate system is used for referencing all of the VOPs. At each particular time instance, a bounding rectangle that includes the shape of that VOP is defined. The position of upper-left corner in the absolute coordinate in the VOP spatial reference is transmitted to the decoder. Thus, the motion vector for a particular block inside a VOP is referred to as the displacement of the block in absolute coordinates.

Actually, the first and second modifications are related since the padding of boundary blocks will affect the matching of motion estimation. The purpose of padding aims at more accurate block matching. In the current algorithm, the repetitive padding is applied to the reference VOP for performing motion estimation and compensation. The repetitive padding process is performed as the following steps:

Define any pixel outside the object boundary as a zero pixel.
Scan each horizontal line of a block (one 16×16 for luminance and two 8×8 for chrominance). Each scan line is possibly composed of two kinds of line segments: zero segments and nonzero segment. It is obvious that our task is to pad zero segments. There are two kinds of zero segments: (1) between an end point of the scan line and the end point of a nonzero segment, and (2) between the end points of two different nonzero segments. In the first case, all zero pixels are replaced by the pixel value of the end pixel of nonzero segment; for the second kind of zero segment, all zero pixels take the averaged value of the two end pixels of the nonzero segments.
Scan each vertical line of the block and perform the identical procedure as described for the horizontal line.
If a zero pixel is located at the intersection of horizontal and vertical scan lines, this zero pixel takes the average of two possible values.
For the rest of zero pixels, find the closest nonzero pixel on the same horizontal scan line and the same vertical scan line (if there is a tie, the nonzero pixel on the left or the top of the current pixel is selected). Replace the zero pixel by the average of these two nonzero pixels.

For a fast-moving VOP, padding is further extended to the blocks outside the VOP but immediately next to the boundary blocks. These blocks are padded by replacing the pixel values of adjacent boundary blocks. This extended padding is performed in both horizontal and vertical directions. Since block matching is replaced by polygon matching for the boundary blocks of the current VOP, the SAD values are calculated by the modified formula:

$$SAD_N(x,y) = \begin{cases} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left|c(i,j) - p(i,j)\right| \cdot \alpha(i,j) - C & \text{if} \quad (x,y) = (0,0); \\ \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left|c(i,j) - p(i+x,j+y)\right| \cdot \alpha(i,j) - C & \text{otherwise,} \end{cases} \tag{18.5}$$

where $C = N_B/2 + 1$ and N_B is the number of pixels inside the VOP and in this block and $\alpha(i,j)$ is the alpha component specifying the shape information, and it is not equal to zero here.

18.3.3.3 Texture Coding of Arbitrarily Shaped VOP

During encoding the VOP is represented by a bounding rectangle that is formed to contain the video object completely but with minimum number of macroblocks in it, as shown in Figure 18.3. The detailed procedure of VOP rectangle formation is given in MPEG-4 video VM (ISO/IEC, 1998b).

There are three types of macroblocks in the VOP with arbitrary shape: the macroblocks that are completely located inside of the VOP, the macroblocks that are located along the boundary of the VOP, and the macroblocks outside of the boundary. For the first kind of macroblock, there is no need for any particular modified technique to code them and just use of normal DCT with entropy coding of quantized DCT coefficients such as coding algorithm in H.263 is sufficient. The second kind of macroblocks, which are located along the boundary, contains two kinds of 8×8 blocks: the blocks lie along the boundary of VOP and the blocks do not belong to the arbitrary shape but lie inside the rectangular bounding box of the VOP. The second kind of blocks are referred

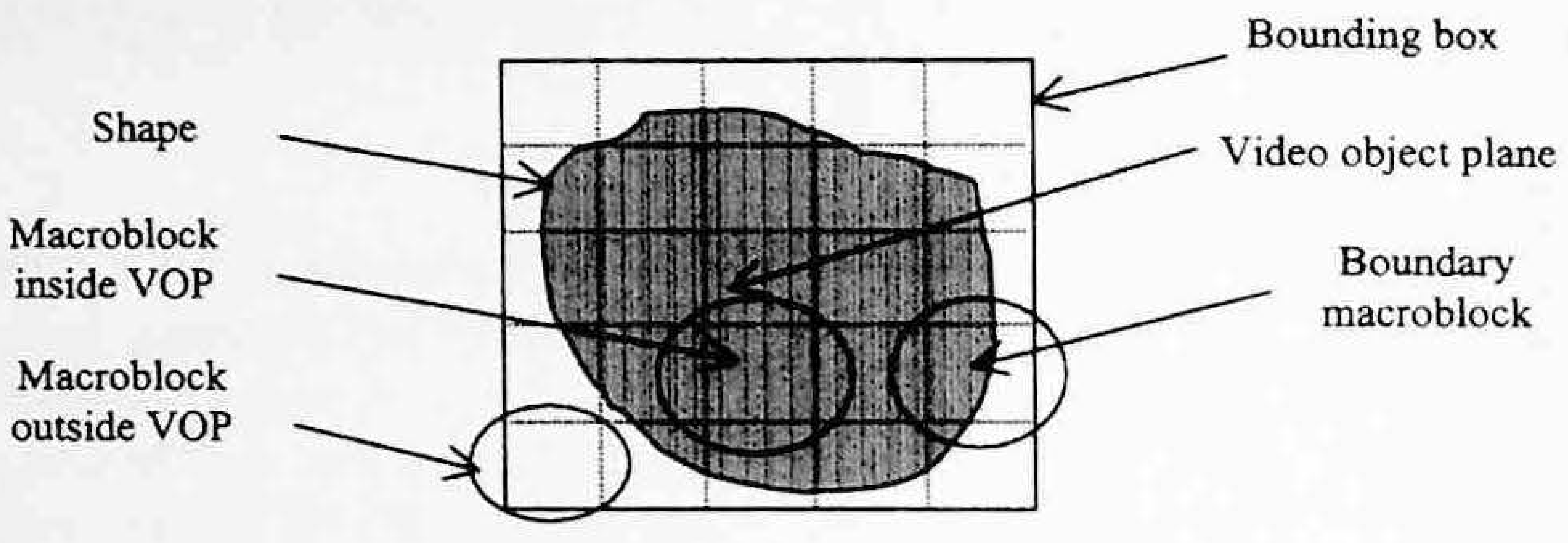


FIGURE 18.3 A VOP is represented by a bounding rectangular box.

to as transparent blocks. For those 8×8 blocks that do lie along the boundary of VOP, there are two different methods that have been proposed: low-pass extrapolation (LPE) padding and shape-adaptive DCT (SA-DCT). All blocks in the macroblock outside of boundary are also referred to as transparent blocks. The transparent blocks are skipped and not coded at all.

1. Low-pass extrapolation padding technique: This block-padding technique is applied to intracoded blocks, which are not located completely within the object boundary. To perform this padding technique we first assign the mean value of those pixels that are located in the object boundary (both inside and outside) to each pixel outside the object boundary. Then an average operation is applied to each pixel $p(i,j)$ outside the object boundary starting from the upper-left corner of the block and proceeding row by row to the lower-right corner pixel:

$$p(i,j) = \left[p(i,j-1) + p(i-1,j) + p(i,j+1) + p(i+1,j)\right]/4. \tag{18.6}$$

   If one or more of the four pixels used for filtering are outside of the block, the corresponding pixels are not considered for the average operation and the factor ¼ is modified accordingly.
2. SA-DCT: The shape-adaptive DCT is only applied to those 8×8 blocks that are located on the object boundary of an arbitrarily shaped VOP. The idea of the SA-DCT is to apply 1-D DCT transformation vertically and horizontally according to the number of active pixels in the row and column of the block, respectively. The size of each vertical DCT is the same as the number of active pixels in each column. After vertical DCT is performed for all columns with at least one active pixel, the coefficients of the vertical DCTs with the same frequency index are lined up in a row. The DC coefficients of all vertical DCTs are lined up in the first row, the first-order vertical DCT coefficients are lined up in the second row, and so on. After that, horizontal DCT is applied to each row. As the same as for the vertical DCT, the size of each horizontal DCT is the same as the number of vertical DCT coefficients lined up in the particular row. The final coefficients of SA-DCT are concentrated into the upper-left corner of the block. This procedure is shown in the Figure 18.4.

The final number of the SA-DCT coefficients is identical to the number of active pixels of the image. Since the shape information is transmitted to the decoder, the decoder can perform the inverse shape-adapted DCT to reconstruct the pixels. The regular zigzag scan is modified so that the nonactive coefficient locations are neglected when counting the runs for the run-length coding of the SA-DCT coefficients. It is obvious that for a block with all 8×8 active pixels, the SA-DCT becomes a regular 8×8 DCT and the scanning of the coefficients is identical to the zigzag scan. All SA-DCT coefficients are quantized and coded in the same way as the regular DCT coefficients

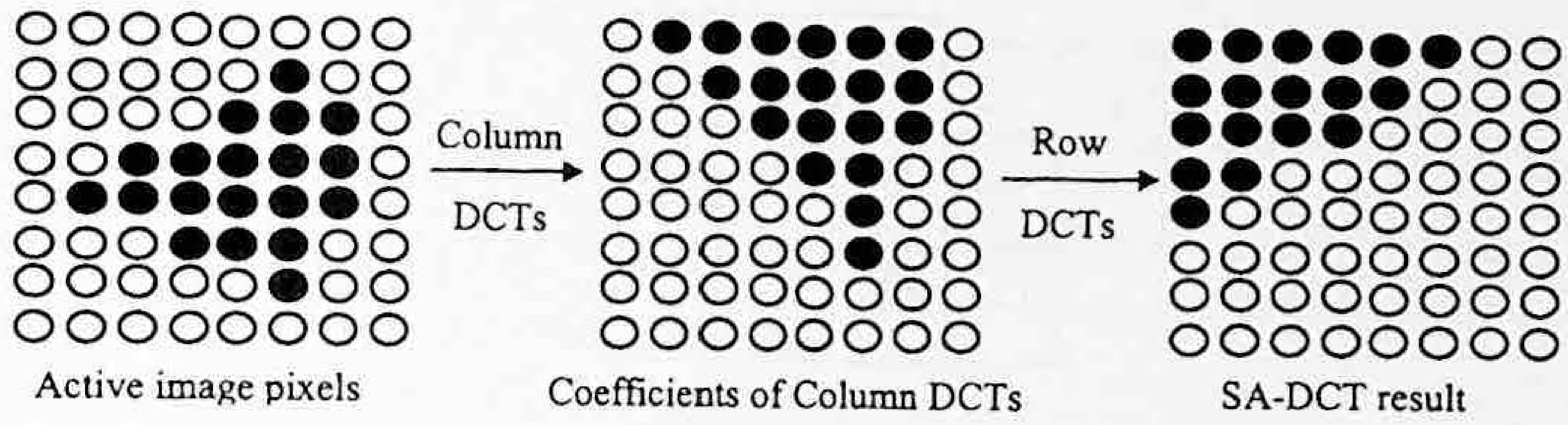


FIGURE 18.4 Illustration of SA-DCT. (From ISO/IEC 14496-2 Video Verification Model V.12, N2552, Dec. 1998. With permission.)

employing the same quantizers and VLC code tables. The SA-DCT is not included in MPEG-4 video version 1, but it is being considered for inclusion into version 2.

18.3.4 Shape Coding

Shape information of the arbitrarily shaped objects is very useful not only in the field of image analysis, computer vision, and graphics, but also in object-based video coding. MPEG-4 video coding is the first to make an effort to provide a standardized approach to compress the shape information of objects and contain the compressed results within a video bitstream. In the current MPEG-4 video coding standard, the video data can be coded on an object basis. The information in the video signal is decomposed to shape, texture, and motion. This information is then coded and transmitted within the bitstream. The shape information is provided in binary format or gray scale format. The binary format of shape information consists of a pixel map, which is generally the same size as the bounding box of the corresponding VOP. Each pixel takes on one of two possible values indicating whether it is located within the video object or not. The gray scale format is similar to the binary format with the additional feature that each pixel can take on a range of values, i.e., times an alpha value. Alpha typically has a normalized value of 0 to 1. The alpha value can be used to blend two images on a pixel-by-pixel basis in this way: new pixel = (alpha)(pixel A color) + (1 − alpha)(pixel B color).

Now let us discuss how to code the shape information. As we mentioned, the shape information is classified as binary shape or gray scale shape. Both binary and gray scale shapes are referred to as an alpha plane. The alpha plane defines the transparency of an object. Multilevel alpha maps are frequently used to blend different images. A binary alpha map defines whether or not a pixel belongs to an object. The binary alpha planes are encoded by modified content-based arithmetic encoding (CAE), while the gray scale alpha planes are encoded by motion-compensated DCT coding, which is similar to texture coding. For binary shape coding, a rectangular box enclosing the arbitrarily shaped VOP is formed as shown in Figure 18.3. The bounded rectangle box is then extended in both vertical and horizontal directions on the right-bottom side to the multiple of 16×16 blocks. Each 16×16 block within the rectangular box is referred to as binary alpha block (BAB). Each BAB is associated with colocated macroblock. The BAB can be classified as three types: transparent block, opaque block, and alpha or shape block. The transparent block does not contain any information about an object. The opaque block is entirely located inside the object. The alpha or shape block is located in the area of the object boundary; i.e., a part of block is inside of object and the rest of block is in background. The value of pixels in the transparent region is zero. For shape coding, the type information will be included in the bitstream and signaled to the decoder as a macroblock type. But only the alpha blocks need to be processed by the encoder and decoder. The methods used for each shape format contain several encoding modes. For example, the binary shape information can be encoded using either an intra- or intermode. Each of these modes can be further divided into lossy and lossless options. Gray scale shape information also contains intra- and intermodes; however, only a lossy option is used.

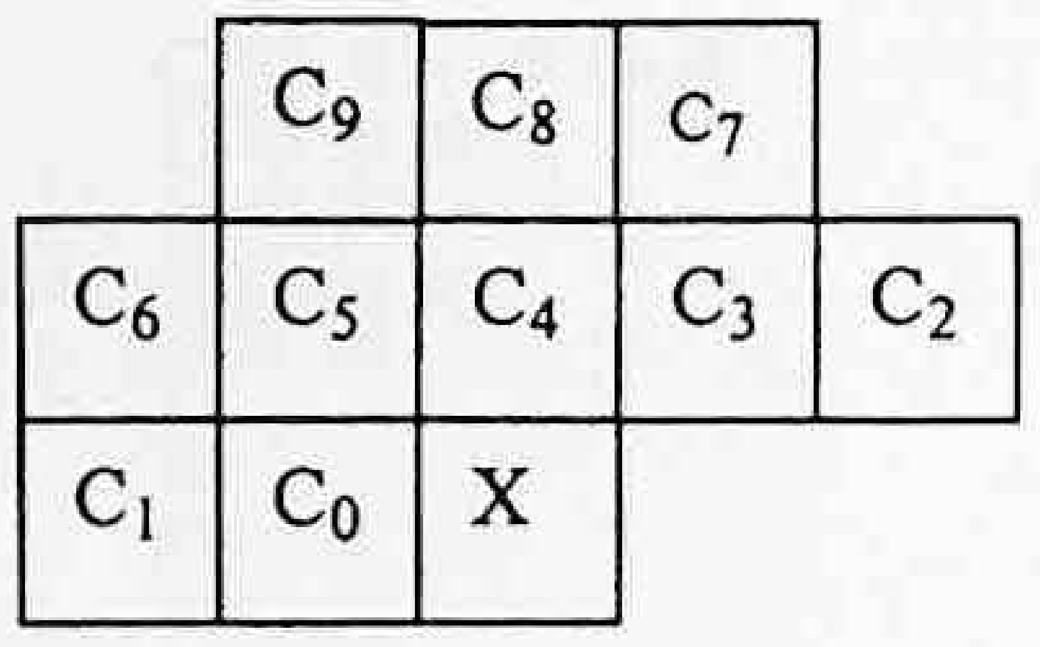


FIGURE 18.5 Template for defining the context of the pixel, X, to be coded in intramode. (From ISO/IEC 14496-2 Video Verification Model V.12, N2552, Dec. 1998. With permission.)

18.3.4.1 Binary Shape Coding with CAE Algorithm

As mentioned previously, the CAE is used to code each binary pixel of the BAB. For a P-VOP, the BAB may be encoded in intra- or intermode. Pixels are coded in scan-line order, i.e., row by row for both modes. The process for coding a given pixel includes three steps: (1) compute a context number, (2) index a probability table using the context number, and (3) use the indexed probability to drive an arithmetic encoder. In intramode, a template of 10 pixels is used to define the causal context for predicting the shape value of the current pixel as shown in Figure 18.5. For the pixels in the top and left boundary of the current macroblock, the template of causal context will contain the pixels of the already transmitted macroblocks on the top and on the left side of the current macroblock. For the two rightmost columns of the VOP, each undefined pixel such as C_7, C_3, and C_2, of the context is set to the value of its closest neighbor inside the macroblock, i.e., C_7 will take the value of C_8 and C_3 and C_2 will take the value of C_4.

A 10-bit context is calculated for each pixel, X as

$$C = \sum_{k=0}^{9} C_k \cdot 2^k . \tag{18.7}$$

This causal context is used to predict the shape value of the current pixel. For encoding the state transition, a context-based arithmetic encoder is used. The probability table of the arithmetic encoder for the 1024 contexts was derived from sequences that are outside of the test set. Two bytes are allocated to describe the symbol probability for each context; the table size is 2048 bytes. To increase coding efficiency and rate control, the algorithm allows lossy shape coding. In lossy shape coding a macroblock can be down-sampled by a factor of two or four resulting in a subblock of size 8×8 pixels or 4×4 pixels, respectively. The subblock is then encoded using the same method as for full-size block. The down-sampling factor is included in the encoded bitstream and then transmitted to the decoder. The decoder decodes the shape data and then up-samples the decoded subblock to full macroblock size according to the down-sampling factor. Obviously, it is more efficient to code shape using a high down-sampling factor, but the coding errors may occur in the decoded shape after up-sampling. However, in the case of low-bit-rate coding, lossy shape coding may be necessary since the bit budget may not be enough for lossless shape coding. Depending on the up-sampling filter, the decoded shape can look somewhat blocky. Several up-sampling filters were investigated. The best-performing filter in terms of subjective picture quality is an adaptive nonlinear up-sampling filter. It should be noted that the coding efficiency of shape coding also depends on the orientation of the shape data. Therefore, the encoder can choose to code the block as described above or transpose the macroblock prior to arithmetic coding. Of course, the transpose information has to be signaled to the decoder.

For shape coding in a P-VOP or B-VOP, the intermode may be used to exploit the temporal redundancy in the shape information with motion compensation. For motion compensation, a 2-D integer pixel motion vector is estimated using full search for each macroblock in order to minimize

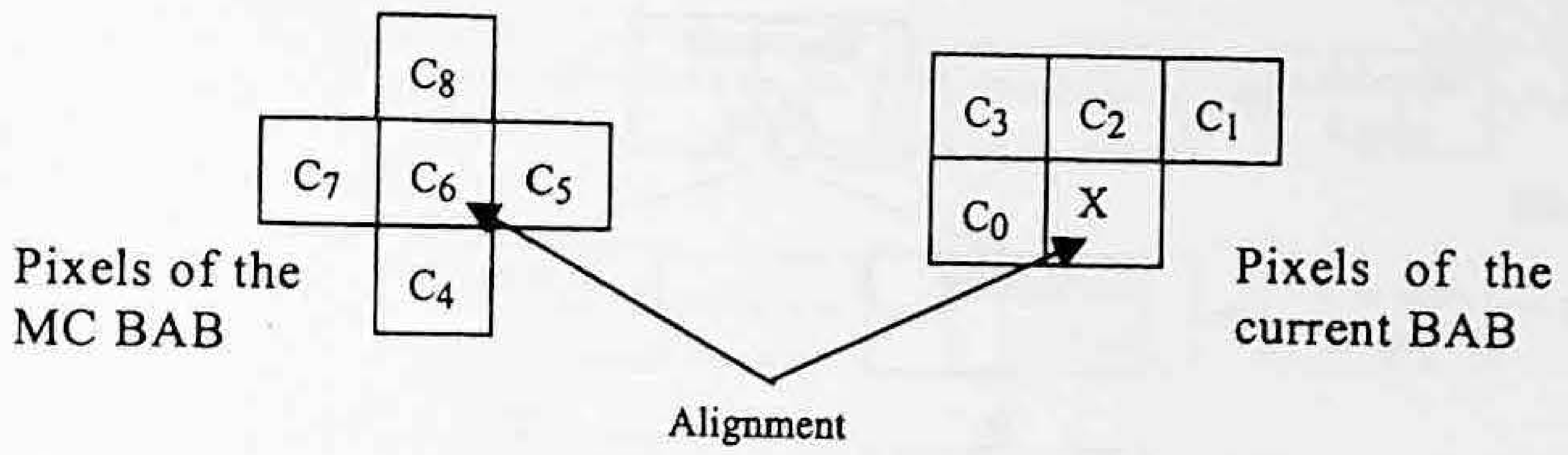


FIGURE 18.6 Template for defining the context of the pixel, X, to be coded in intermode. (From ISO/IEC 14496-2 Video Verification Model, N2552, Dec. 1998. With permission.)

the prediction error between the previously coded VOP shape and the current VOP shape. The shape motion vectors are predictively encoded with respect to the shape motion vectors of neighboring macroblocks. If no shape motion vector is available, texture motion vectors are used as predictors. The template for intermode differs from the one used for intramode. The intermode template contains 9 pixels among which 5 pixels are located in the previous frame and 4 are the current neighbors as shown in Figure 18.6.

The intermode template defines a context of 9 pixels. Accordingly, a 9-bit context or 512 contexts, can be computed in a similar way to Equation 18.7:

$$C = \sum_{k=0}^{8} C_k \cdot 2^k . \tag{18.8}$$

The probability for one symbol is also described by 2 bytes giving a probability table size of 1024 bytes. The idea of lossy coding can also be applied to the intermode shape coding by down-sampling the original BABs. For intermode shape coding, the total bits for coding the shape consist of two parts, one part for coding motion vectors and another for prediction residue. The encoder may decide that the shape representation achieved by just using motion vectors is sufficient; thus bits for coding the prediction error can be saved. Actually, there are seven modes to code the shape information of each macroblock: (1) transparent, (2) opaque, (3) intra, inter (4) with and (5) without shape motion vectors, and inter (6) with and (7) without shape motion vectors and prediction error coding. These different options with optional down-sampling and transposition allow for encoder implementations of different coding efficiency and implementation complexity. Again, this is a problem of encoder optimization, which does not belong to the standard.

18.3.4.2 Gray Scale Shape Coding

The gray scale shape information is encoded by separately encoding the shape and transparency information as shown in Figure 18.7. For a transparent object, the shape information is referred to as the support function and is encoded using the binary shape-coding method. The transparency or alpha values are treated as the texture of luminance and encoded using padding, motion compensation, and the same 8×8 block DCT approach for the texture coding. For an object with varying alpha maps, shape information is encoded in two steps. The boundary of the object is first losslessly encoded as a binary shape, and then the actual alpha map is encoded as texture coding.

The binary shape coding allows one to describe objects with constant transparency, while gray scale shape coding can be used to describe objects with arbitrary transparency, providing for more flexibility for image composition. One application example is a gray scale alpha shape that consists of a binary alpha shape with the value around the edges tapered from 255 to 0 to provide for a smooth composition with the background. The description of each video object layer includes the information to give instruction for selecting one of six modes for feathering. These six modes

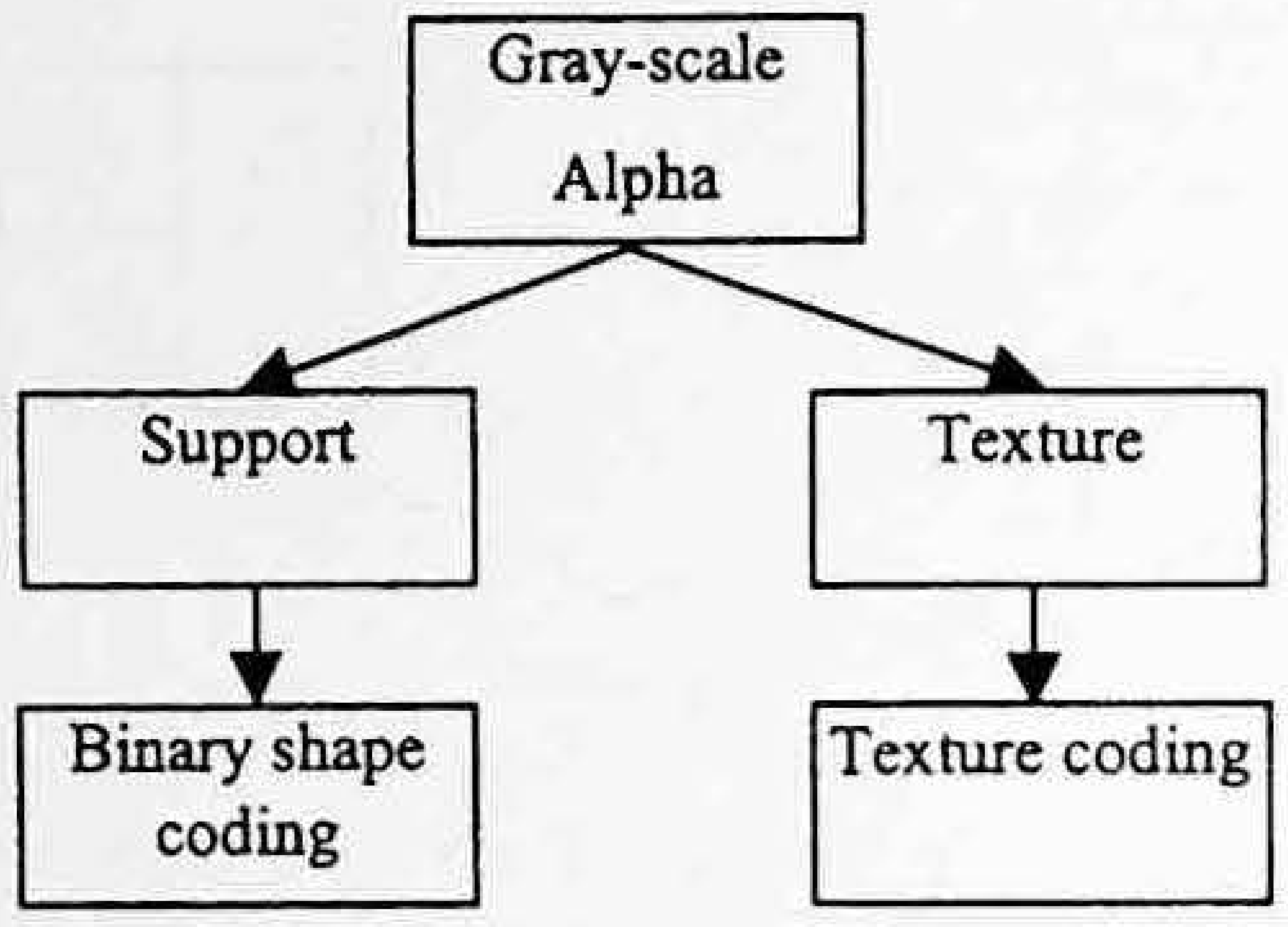


FIGURE 18.7 Gray scale shape coding.

include (1) no effects, (2) linear feathering, (3) constant alpha, (4) linear feathering and constant alpha, (5) feathering filter, and (6) feathering filter and constant alpha. The detailed description of the function of these modes are given in the reference of version 12 (ISO/IEC, 1998b).

18.3.5 Sprite Coding

As mentioned previously, MPEG-4 video has investigated a number of new tools, which attempt to improve the coding efficiency at low bit rates compared with MPEG-1/2 video coding. Among these tools, sprite coding is an efficient technology to reach this goal. A sprite is a specially composed video object that is visible throughout an entire piece of video sequence. For example, the sprite generated from a panning sequence contains all the visible pixels of the background throughout the video sequence. Portions of the background may not be seen in certain frames due to the occlusion of the foreground objects or the camera motion. This particular example is one of the static sprites. In other words, a static sprite is a possible still image. Since the sprite contains all visible background scenes of a segment video sequence where the changes within the background content are mainly caused by camera parameters, the sprite can be used for direct reconstruction of the background VOPs or as the prediction of the background VOPs within the video segment. The sprite-coding technology first efficiently transmits this background to the receiver and then stores it in a frame at both encoder and decoder. The camera parameters are then transmitted to the decoder for each frame so that the appropriate part of the background scene can be either used as the direct reconstruction or as the prediction of the background VOP. Both cases can significantly save the coding bits and increase the coding efficiency. There are two types of sprites, static sprite and dynamic sprite, which are being considered as coding tools for MPEG-4 video. A static sprite is used for a video sequence in which the objects in a scene can be separated into foreground objects and a static background. A static sprite is a special VOP, which is generated by copying the background from a video sequence. This copying includes the appropriate warping and cropping. Therefore, a static sprite is always built off-line. In contrast, a dynamic sprite is dynamically built during the predictive coding. It can be built either online or off-line. The static sprite has shown significant coding gain over existing compression technology for certain video sequences. The dynamic sprite is more complicated in the real-time application due to the difficulty of updating the sprite during the coding. Therefore, the dynamic sprite has not been adopted by version 1 of the standard. Additionally, both sprites are not easily applied to the generic scene content. Also, there is another kind of classification of sprite coding according to the method of sprite generation, namely, off-line and online sprites. Off-line is always used for static sprite generation. Off-line sprites are well suited for synthetic objects and objects that mostly undergo rigid motion. Online sprites are only used for dynamic sprites. Online sprites provide a no-latency solution in the case of natural sprite objects. Off-line dynamic sprites provide an enhanced predictive coding environment. The sprite is built with a similar way in both off-line and online methods. In particular, the same global motion estimation algorithm is exploited. The difference is that the off-line sprite is

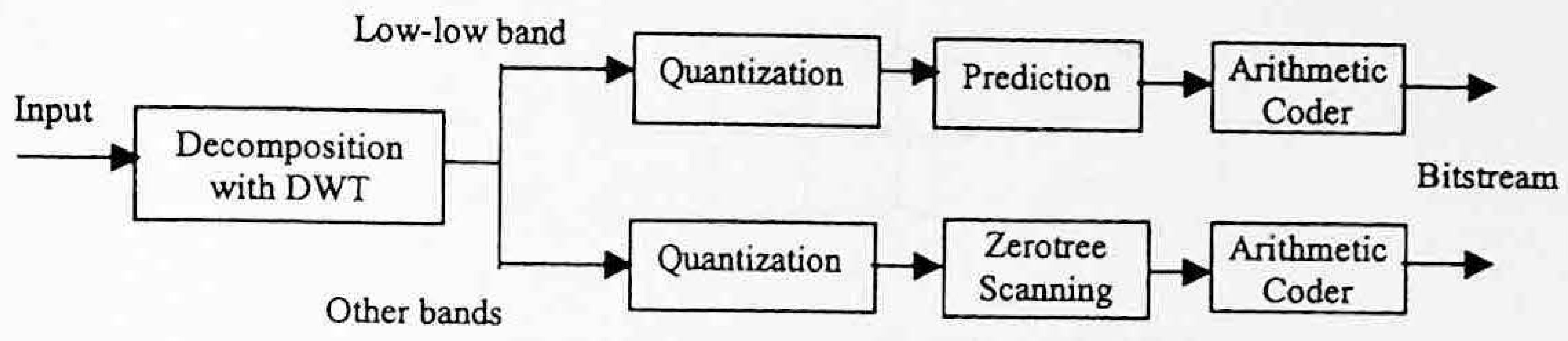


FIGURE 18.8 Block diagram of encoder of wavelet-based texture coding, DWT stands for discrete wavelet transform.

built before starting the encoding process while, in the online sprite case, both the encoder and the decoder build the same sprite from reconstructed VOPs. This is why the online dynamic sprites are more complicated in the implementation. The online sprite is not included in version 1, and will most likely not be considered for version 2 either. In sprite coding, the chrominance components are processed in the same way as the luminance components, with the properly scaled parameters according to the video format.

18.3.6 Interlaced Video Coding

Since June of 1997, MPEG-4 has extended its application to support interlaced video. Interlaced video consists of two fields per frame, which are referred to as the even field and the odd field. MPEG-2 has a number of tools, which are used to deal with field structure of video signals. These tools include frame/field-adaptive DCT coding and frame/field-adaptive motion compensation. However, the field issue in MPEG-4 has to be considered on a VOP basis instead of the conventional frame basis. When field-based motion compensation is specified, two field motion vectors and the corresponding reference fields are used to generate the prediction from each reference VOP. Shape information has to be considered in the interlaced video for MPEG-4.

18.3.7 Wavelet-Based Texture Coding

In MPEG-4 there is a texture-coding mode which is used to code the texture or still image such as in JPEG. The basic technique used in this mode is wavelet-based transform coding. The reason for adopting wavelet transform instead of DCT for still texture coding is not only its high coding efficiency, but also because the wavelet can provide excellent scalability, both spatial scalability and SNR scalability. Since the principle of wavelet-based transform coding for image compression has been explained in Chapter 8, we just briefly describe the coding procedure of this mode. The block diagram of the encoder is shown in Figure 18.8.

18.3.7.1 Decomposition of the Texture Information

The texture or still image is first decomposed into bands using a bank of analysis filters. This decomposition can be applied recursively on the obtained bands to yield a decomposition tree of subbands. An example of decomposition to depth 2 is shown in Figure 18.9.

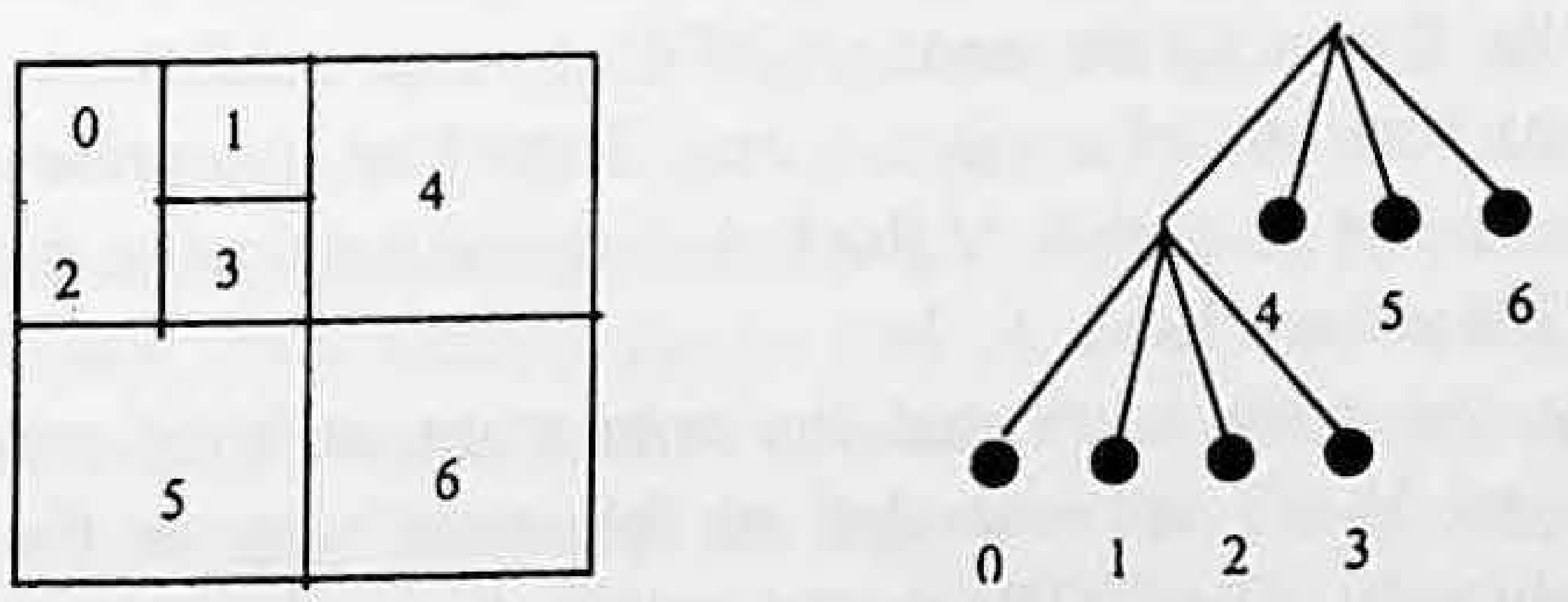
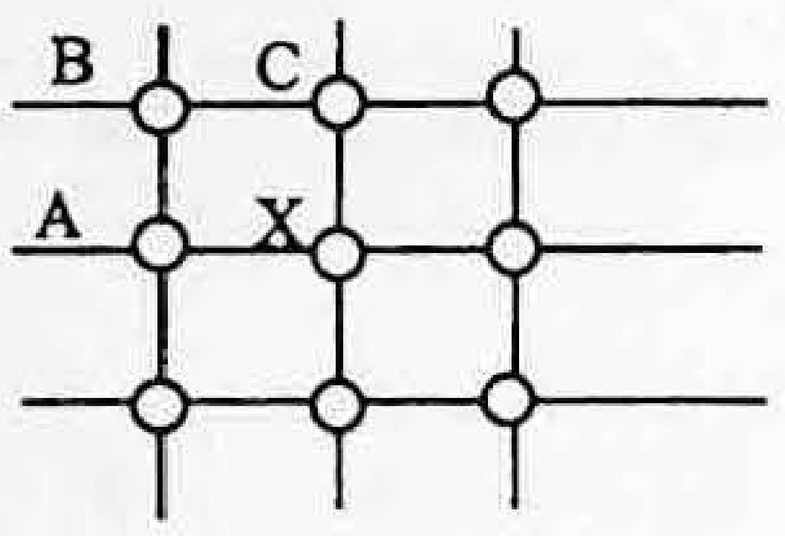


FIGURE 18.9 An example of wavelet decomposition of depth 2.



If $|W_A - |W_B| < |W_B - W_C|$, $W_{Xp} = W_C$, else $W_{Xp} = W_A$.

FIGURE 18.10 Adaptive DPCM coding of the coefficients in the lowest band.

18.3.7.2 Quantization of Wavelet Coefficients

After decomposition, the coefficients of the lowest band are coded independently of the other bands. These coefficients are quantized using a uniform midriser quantizer. The coefficients of high bands are quantized with a multilevel quantization. The multilevel quantization provides a very flexible approach to support the correct trade-off between levels and type of scalability, complexity, and coding efficiency for any application. All quantizers for the higher bands are uniform midrise quantizers with a dead zone that is twice the quantizer step size. The levels and quantization steps are determined by the encoder and specified in the bitstream. To achieve scalability, a bi-level quantization scheme is used for all multiple quantizers. This quantizer is also uniform and midrise with a dead zone that is twice the quantization step. The coefficients outside of the dead zone are quantized to 1 bit. The number of quantizers is equal to the maximum number of bit planes in the wavelet transform representation. In this bi-level quantizer, the maximum number of bit planes instead of a quantization step size is specified in the bitstream.

18.3.7.3 Coding of Wavelet Coefficients of Low–Low Band and Other Bands

The quantized coefficients at the lowest band are DPCM coded. Each of the current coefficients is predicted from three other quantized coefficients in its neighborhood in a way shown in Figure 18.10.

The coefficients in high bands are coded with the zerotree algorithm (Shapiro, 1993), which has been discussed in Chapter 8.

18.3.7.4 Adaptive Arithmetic Coder

The quantized coefficients and the symbols generated by the zerotree are coded using an adaptive arithmetic coder. In the arithmetic coder three different tables which correspond to the different statistical models have been utilized. The method used here is very similar to one in Chapter 8. Further detail can be found in MPEG-4 (ISO/IEC, 1998a).

18.3.8 Generalized Spatial and Temporal Scalability

The scalability framework is referred to as generalized scalability that includes the spatial and the temporal scalability similar to MPEG-2. The major difference is that MPEG-4 extends the concept of scalability to be content based. This unique functionality allows MPEG-4 to be able to resolve objects into different VOPs. By using the multiple VOP structure, different resolution enhancement can be applied to different portions of a video scene. Therefore, the enhancement layer may only be applied to a particular object or region of the base layer instead of to the entire base layer. This is a feature that MPEG-2 does not have.

In spatial scalability, the base layer and the enhancement layer can have different spatial resolutions. The base-layer VOPs are encoded in the same way as the nonscalable encoding technique described previously. The VOPs in the enhancement layer are encoded as P-VOPs or B-VOPs, as shown in Figure 18.11. The current VOP in the enhancement layer can be predicted

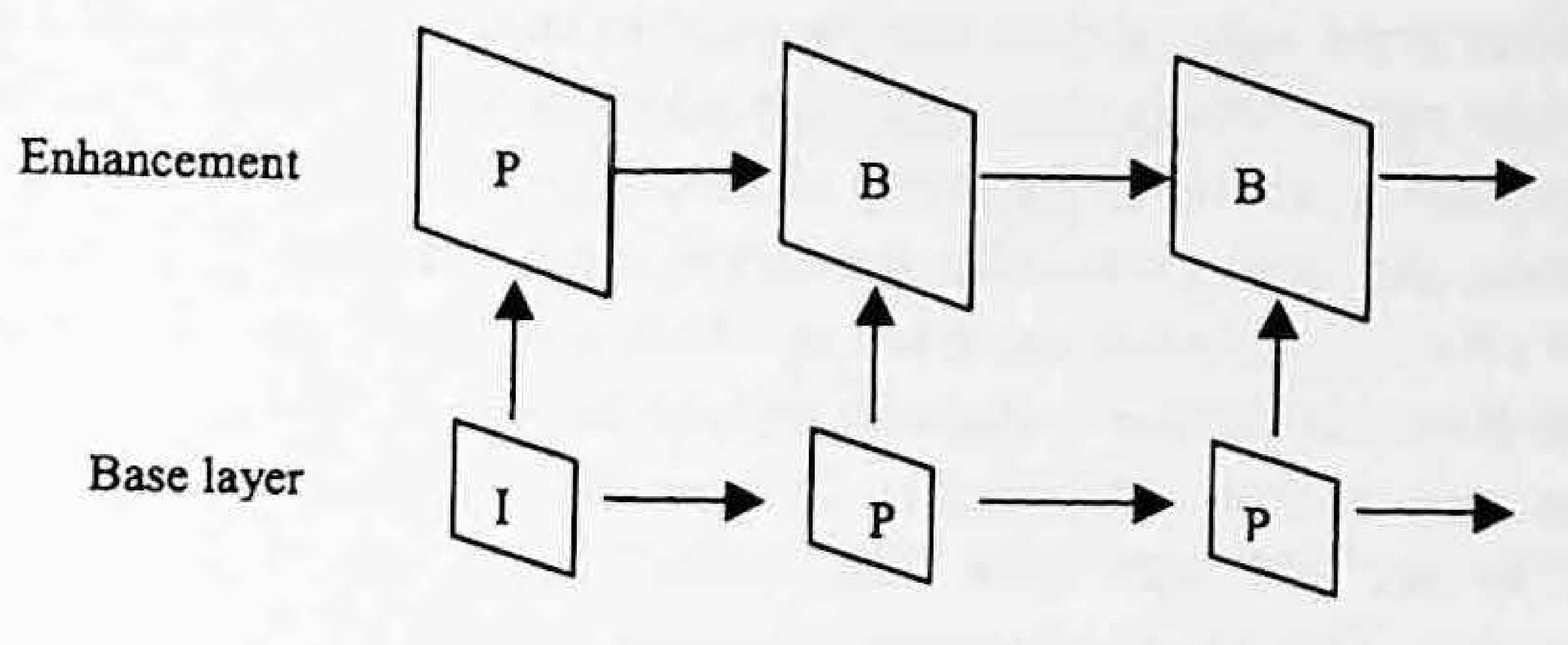


FIGURE 18.11 Illustration of spatial scalability.

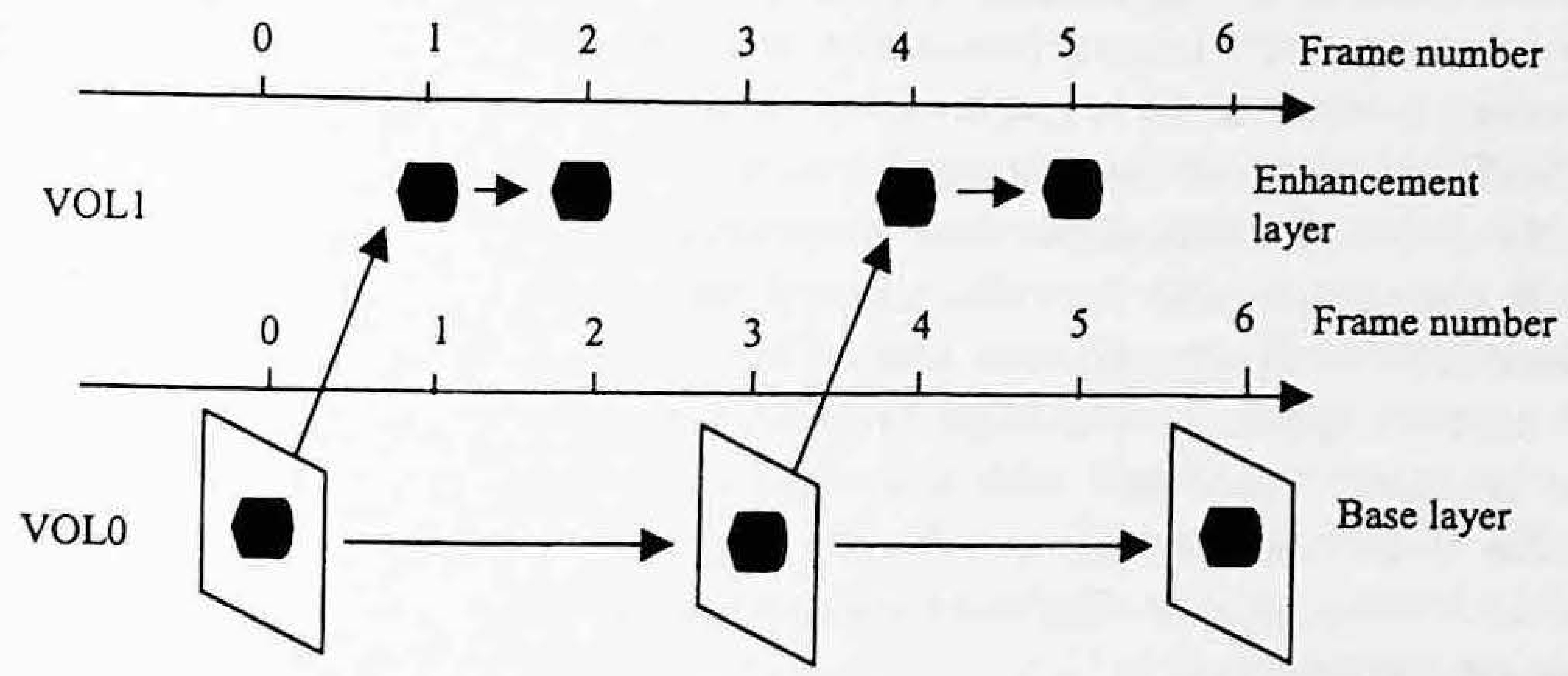


FIGURE 18.12 An example of temporal scalability. (From ISO/IEC 14496-2 Video Verification Model V.12, N2552, Dec. 1998. With permission.)

from either the up-sampled base layer VOP or the previously decoded VOP at the same layer as well as both of them. The down-sampling and up-sampling processing in spatial scalability is not a part of the standard and can be defined by the user.

In temporal scalability, a subsequence of subsampled VOP in the time domain is coded as a base layer. The remaining VOPs can be coded as enhancement layers. In this way, the frame rate of a selected object can be enhanced so that it has a smoother motion than other objects. An example of temporal scalability is illustrated in Figure 18.12. In Figure 18.12, the VOL_0 is the entire frame with both an object and a background, while VOL_1 is a particular object in VOL_0. VOL_0 is encoded with a low frame rate and VOL_1 is the enhancement layer. The high frame rate can be reached for the particular object by combining the decoded data from both the base layer and the enhancement layer. Of course, the B-VOP is also used in temporal scalability for coding the enhancement layer, which is another type of temporal scalability. As in spatial scalability, the enhancement layer can be used to improve either the entire base layer frame resolution or only a portion of the base layer resolution.

18.3.9 Error Resilience

The MPEG-4 visual coding standard provides error robustness and resilience to allow access of image and video data over a wide range of storage and transmission media. The error resilience tool development effort is divided into three major areas, which include resynchronization, data recovery, and error concealment. As with other coding standards, MPEG-4 makes heavy use of variable-length coding to reach high coding performance. However, if even 1 bit is lost or damaged, the entire bitstream becomes undecodable due to loss of synchronization. The resynchronization tools attempt to enable resynchronization between the decoder and the bitstream after a transmission error or errors have been detected. Generally, the data between the synchronization point prior to the error and the first point, where synchronization is reestablished, are discarded. The purpose of resynchronization is to localize effectively the amount of data discarded by the decoder; then the other methods such as error concealment can be used to conceal the damaged areas of a decoded picture. Currently, the resynchronization approach adopted by MPEG-4 is referred to as a packet approach. This approach is similar to the group of block (GOB) structure used in H.261 and H.263. In the GOB structure, the GOB contains a start code, which provides the location information of the GOB. MPEG-4 adopted a similar approach in which a resynchronization marker is periodically inserted into the bitstream at the particular macroblock locations. The resynchronization marker is used to indicate the start of new video packet. This marker is distinguished from all possible VLC codewords as well as the VOP start code. The packet header information is then provided at the start of a video packet. The header contains the information necessary to restart the decoding process. This includes the macroblock number of the first macroblock contained in this packet and the quantization parameter necessary to decode the first macroblock. The macroblock number provides the necessary spatial resynchronization while the quantization parameter allows the differential decoding process to be resynchronized. It should be noted that when error resilience is used within MPEG-4, some of the compression efficiency tools need to be modified. For example, all predictively encoded information must be contained within a video packet to avoid error propagation. In conjunction with the video packet approach to resynchronization, MPEG-4 has also adopted a fixed-interval synchronization method which requires that VOP start-codes and resynchronization markers appear only at legal fixed-interval locations in the bitstream. This will help to avoid the problems associated with start-code emulation. In this case, when fixed-interval synchronization is utilized, the decoder is only required to search for a VOP start-code at the beginning of each fixed interval. The fixed-interval synchronization method extends this approach to any predetermined interval.

After resynchronization is reestablished, the major problem is recovering lost data. A new tool called reversible variable-length codes (RVLC) is developed for the purpose of data recovery. In this approach, the variable-length codes are designed such that the codes can be read both in the forward and the reverse direction. An example of such a code includes codewords like 111, 101, 010. All these codewords can be read reversibly. However, it is obvious that this approach will reduce the coding efficiency that is achieved by the entropy coder. Therefore, this approach is used only in the cases where error resilience is important.

Error concealment is an important component of any error-robust video coding. The error-concealment strategy is highly dependent on the performance of the resynchronization technique. Basically, if the resynchronization method can efficiently localize the damaged data area, the error concealment strategy becomes much more tractable. Error concealment is actually a decoder issue if there is no additional information provided by the encoder. There are many approaches to error concealment, which are referred to in Chapter 17.

18.4 MPEG-4 VISUAL BITSTREAM SYNTAX AND SEMANTICS

The common feature of MPEG-4 and MPEG-1/MPEG-2 is the layered structure of the bitstream. MPEG-4 defines a syntactic description language to describe the exact binary syntax of an audio-visual object bitstream, as well as that of the scene description information. This provides a consistent and uniform way to describe the syntax in a very precise form, while at the same time simplifying bitstream compliance testing. The visual syntax hierarchy includes the following layers:

- Video session (VS)
- Video object (VO)
- Video object layer (VOL) or texture object layer (TOL)
- Group of video object plane (GOV)
- Video object plane (VOP)

A typical video syntax hierarchy is shown in Figure 18.13.

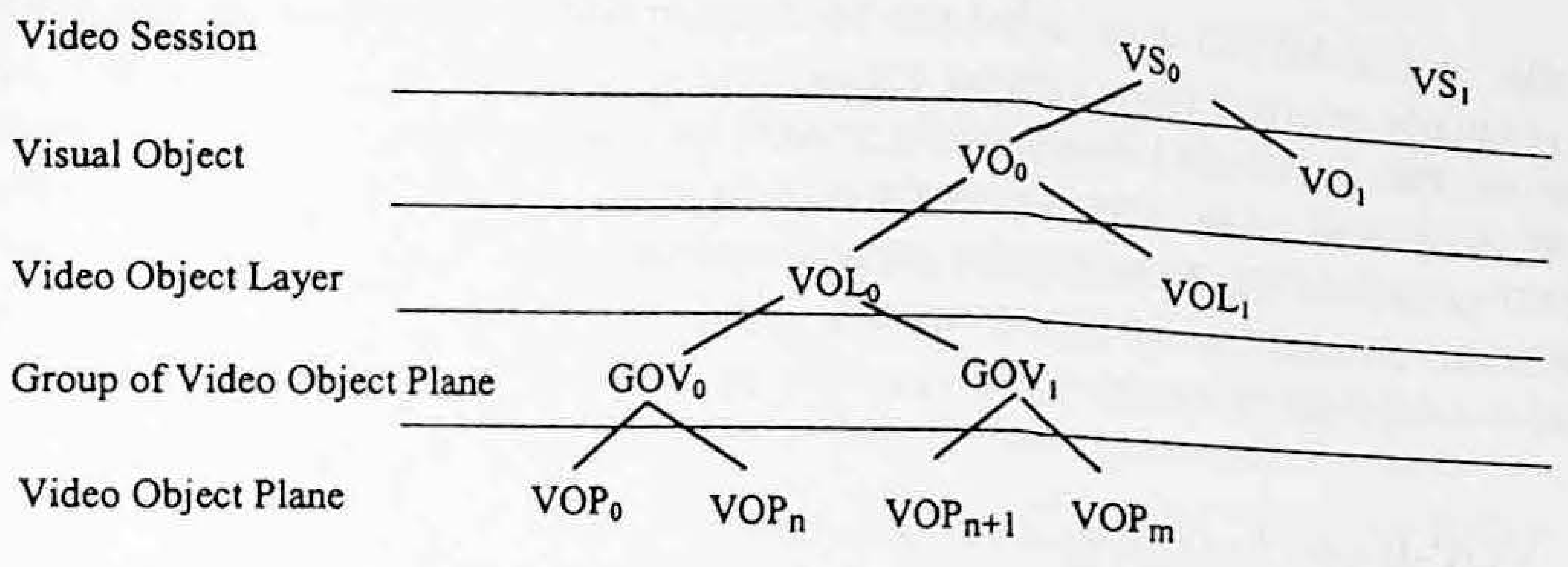


FIGURE 18.13 MPEG-4 video syntax hierarchy.

The video session (VS) is the highest syntactic structure of the coded video bitstream. A VS is a collection of one or more VOs. A VO can consist of one or more layers. Since MPEG-4 is extended from video coding to visual coding, the type of visual objects not only includes video objects, but also still texture objects, mesh objects, and face objects. These layers can be either video or texture. Still texture coding is designed for high-visual-quality applications in transmission and rendering of texture. The still coding algorithm supports a scalable representation of image or synthetic scene data such as luminance, color, and shape. This is very useful for progressive transmission of images or 2-D/3-D synthetic scenes. The images can be gradually built up in the terminal monitor as they are received. The bitstreams for coded mesh objects are nonscalable; they define the structure and motion of a 2-D mesh. The texture of the mesh has to be coded as a separate video object. The bitstreams for face objects are also nonscalable; these bitstreams contain the face animation parameters. VOs are coded with different types of scalability. The base layer can be decoded independently and the enhancement layers can only be decoded with the base layer. In the special case of a single rectangular VO, all of the MPEG-4 layers can be related to MPEG-2 layers. That is, VS is the same as VO since in this case a single VO is a video sequence, VOL or TOL is the same as the sequence scalable extension, GOV is like the GOP, and VOP is a video frame. VO sequence may contain one or more VOs coded concurrently. The VO header information contains the start code followed by profile and level identification and a VO identification to indicate the type of object, which may be a VO, a still texture object, a mesh object, or a face object. The VO may contain one or more VOLs. In the VOL, the VO can be coded with spatial or temporal scalability. Also, the VO may be encoded in several layers from coarse to fine resolution. Depending on the application need, the decoder can choose the number of layers to decode. A VO at a specified time is called a video object plane (VOP). Thus, a VO contains many VOPs. A scene may contain many VOs. Each VO can be encoded to an independent bitstream. A collection of VOPs in a VOL is called a group of VOPs (GOV). This concept corresponds to the group of pictures (GOP) in MPEG-1 and MPEG-2. A VOP is then coded by shape coding and texture coding, which is specified at lower layers of syntax, such as the macroblock and block layer. The VOP or higher-than-VOP layer always commences with a start code and is followed by the data of lower layers, which is similar to the MPEG-1 and MPEG-2 syntax.

18.5 MPEG-4 VIDEO VERIFICATION MODEL

Since all video-coding standards define only the bitstream syntax and decoding process, the use of test models to verify and optimize the algorithms is needed during the development process. For this purpose a common platform with a precise definition of encoding and decoding algorithms has to be provided. The test model (TM) of MPEG-2 took the above-mentioned role. The TM of MPEG-2 was updated continually from version 1.0 to version 5.0, until the MPEG-2 Video IS (International Standard) was completed. MPEG-4 video uses a similar tool during the development process; this tool in MPEG-4 is called the Verification Model (VM). So far, the MPEG-4 video VM has gradually evolved from version 1.0 to version 12.0 and in the process has addressed an increasing number of desired functionalities such as content-based scalability, error resilience, coding efficiency, and so on. The material presented in this section is different from Section 18.3. Section 18.3 presented the technologies adopted or that will be adopted by MPEG-4, while this section provides an example of how to use the standard, for example, how to encode or generate the MPEG-4-compliant bitstream. Of course, the decoder is also included in the VM.

18.5.1 VOP-Based Encoding and Decoding Process

Since the most important feature of MPEG-4 is an object-based coding method, the input video sequence is first decomposed into separate VOs, these VOs are then encoded into separate bitstreams so that the user can access and manipulate (cut, paste, etc.) the video sequence in the bitstream domain. Instances of VOs in a given time are called a video object plane (VOP). The bitstream also contains the composition information to indicate where and when each VOP is to be displayed. At the decoder, the user may be allowed to change the composition of the scene displayed by interactively changing the composition information.

18.5.2 Video Encoder

For object-based coding, the encoder consists mainly of two parts: the shape coding and the texture coding of the input VOP. The texture coding is based on the DCT coding with traditional motion-compensated predictive coding. The VOP is represented by means of a bounding rectangular as described previously. The phase between luminance and chrominance pixels of the bounding rectangular has to be correctly set to the 4:2:0 format as in MPEG-1/2. The block diagram of encoding structure is shown in Figure 18.14.

The core technologies used in VOP coding of MPEG-4 have been described previously. Here we are going to discuss several encoding issues. Although these issues are essential to the performance and application, they are not dependent on the syntax. As a result, they are not included as normative parts of the standard, but are included as informative annexes.

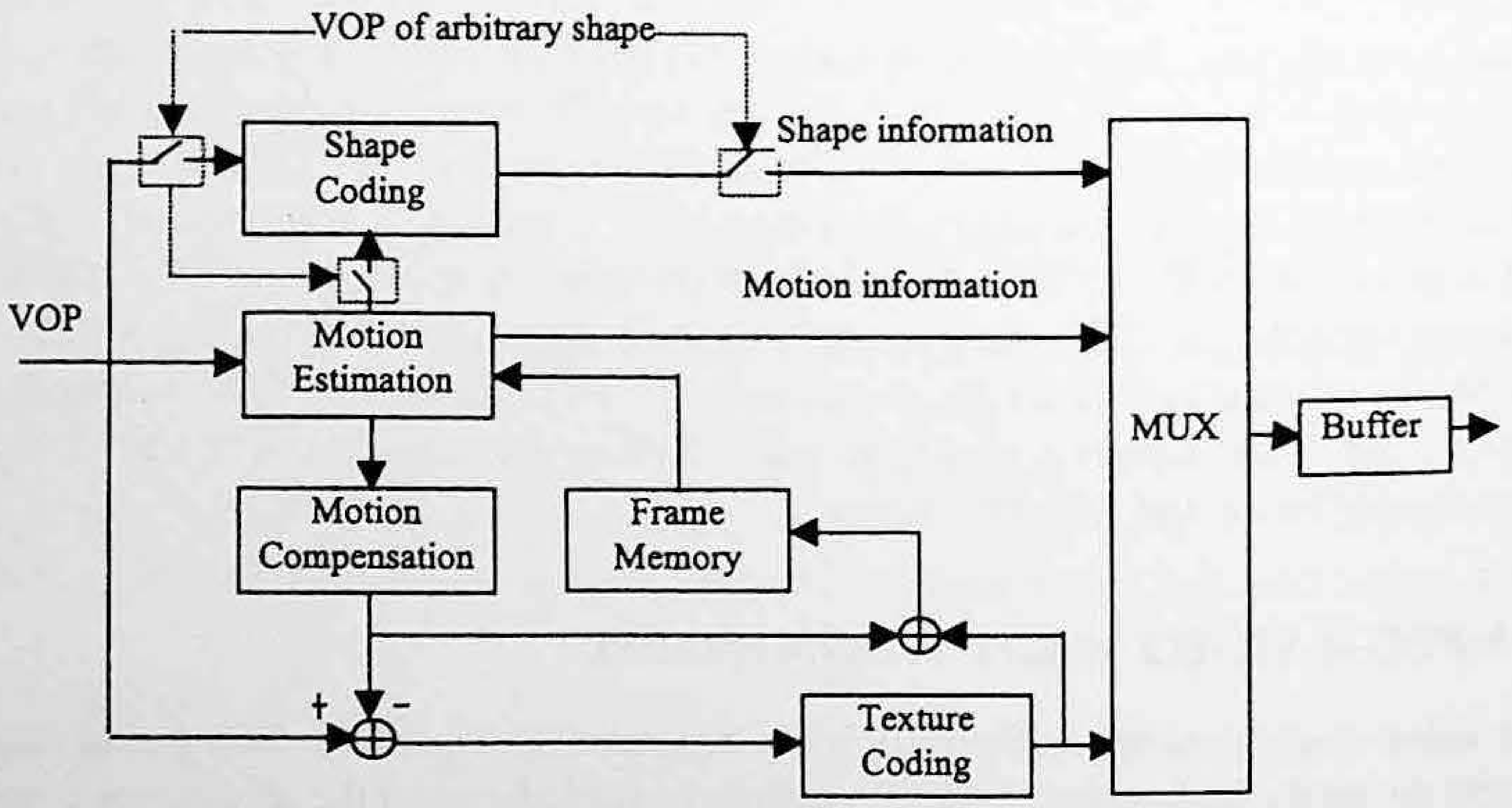


FIGURE 18.14 Block diagram of MPEG-4 video encoder structure.

18.5.2.1 Video Segmentation

Object-based coding is the most important feature of MPEG-4. Therefore, the tool for object boundary detection or segmentation is a key issue in efficiently performing the object-based coding scheme. But the method of decomposing a natural scene to several separate objects is not specified by the standard since it is a preprocessing issue. There are currently two kinds of algorithms for segmentation of video objects. One kind of algorithm is an automatic segmentation algorithm. In the case of real-time applications, the segmentation must be done automatically. Real-time automatic segmentation algorithms are currently not mature. An automatic segmentation algorithm has been proposed in MPEG96/M960 (Colonnese and Russo, 1996). This algorithm separates regions corresponding to moving objects from regions belonging to a static background for each frame of a video sequence. The algorithm is based on a motion analysis for each frame. The motion analysis is performed along several frames to track each pixel of the current frame and to detect whether the pixel belongs to the moving objects.

Another kind of segmentation algorithms is one that is user assisted or "semiautomatic." In non-real-time applications, the semiautomatic segmentation may be used effectively and give better results than the automatic segmentation. In the core experiments of MPEG-4, a semiautomatic segmentation algorithm was proposed in MPEG97/M3147 (Choi et al., 1997). The block diagram of the semiautomatic segmentation is shown in Figure 18.15.

This technique consists of two steps. First, the intraframe segmentation is applied to the first frame, which is considered as a frame that either contains newly appeared objects or a reset frame. Then the interframe segmentation is applied to the consecutive frames. For intraframe, the segmentation is processed by a user manually or semiautomatically. The user uses a graphical user interface (GUI) to draw the boundaries of objects of interest. The user can mask the entire objects all the way around objects using a mouse with a predefined thickness of the line (number of pixels). A marked swath is then achieved by the mouse, and this marked area is assumed to contain the object boundaries. A boundary-detection algorithm is applied to the marked area to create the real object boundaries. For interframe segmentation, an object boundary-tracking algorithm is proposed to obtain the object boundaries of the consecutive frames. At first, the boundary of the previous object is extracted and the motion estimation is performed on the object boundary. The object boundary

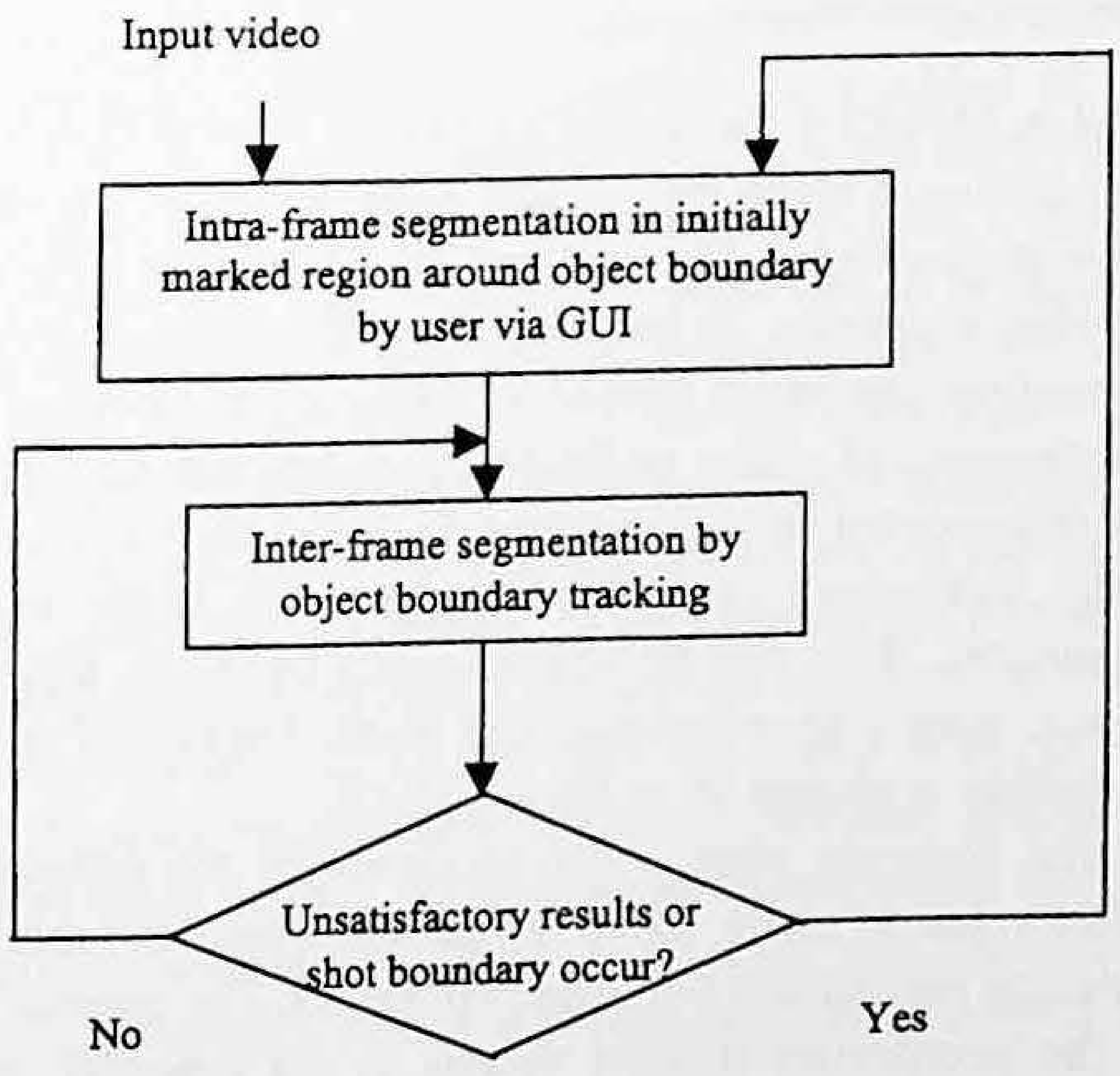

Input video
Intra-frame segmentation in initially marked region around object boundary by user via GUI
Inter-frame segmentation by object boundary tracking
Unsatisfactory results or shot boundary occur?
No
Yes

FIGURE 18.15 Block diagram of a user-assisted VO segmentation method.