Error-Resilient Video Coding in the ISO MPEG-4 Standard

Raj Talluri, Texas Instruments

ABSTRACT This article describes error resilience aspects of the video coding techniques that are standardized in the ISO MPEG-4 standard. The article begins with a description of the general problems in robust wireless video transmission. The specific tools adopted into the ISO MPEG-4 standard to enable the communication of compressed video data over noisy wireless channels are presented in detail. These techniques include resynchronization strategies, data partitioning, reversible VLCs, and header extension codes. An overview of the evolving ISO MPEG-4 standard and its current status are described.

Recent advances in technology have resulted in a rapid growth in mobile communications. With this explosive growth, the need for reliable transmission of mixed media information — audio, video, text, graphics, and speech data — over wireless links is becoming an increasingly important application requirement. Typically, the bandwidth requirements of raw video data are very high (a 176 x 144-pixel 4:2:0 color video sequence requires over 1 Mb/s). Video compression techniques are used to reduce the bandwidth requirements and enable the transmission of video information over band-limited wireless channels. These techniques typically apply predictive coding, that is, code the information in the current frame of the video signal predictively as a difference signal with respect to the previous frame. They also employ variable-length coding schemes such as Huffman codes to achieve a further degree of compaction [1].

Wireless channels are typically noisy and suffer from a number of channel degradations such as bit errors and burst errors due to fading and multipath reflections [2]. When compressed video data is sent over these channels it is subject to these degradations. The effect of channel errors on compressed video can be very severe. Variable-length coding schemes also render the compressed bitstream very brittle to channel errors. As a result, the video decoder that is decoding the corrupted video bitstream loses synchronization with the encoder. Predictive coding techniques such as motion compensation applied in current video compression standards make matters worse by quickly propagating the effects of channel errors across the video sequence and rapidly degrading video quality. This renders the video sequence totally unusable. Unless the video encoder and decoder take proper remedial steps the video communication system totally breaks down.

Current video compression techniques take a number of steps to enable robust transmission of compressed video data over noisy communication channels. This article describes the methodologies adopted by the International Organization for Standardization (ISO) Motion Picture Experts Group v. 4 (MPEG-4) video compression standard [3].

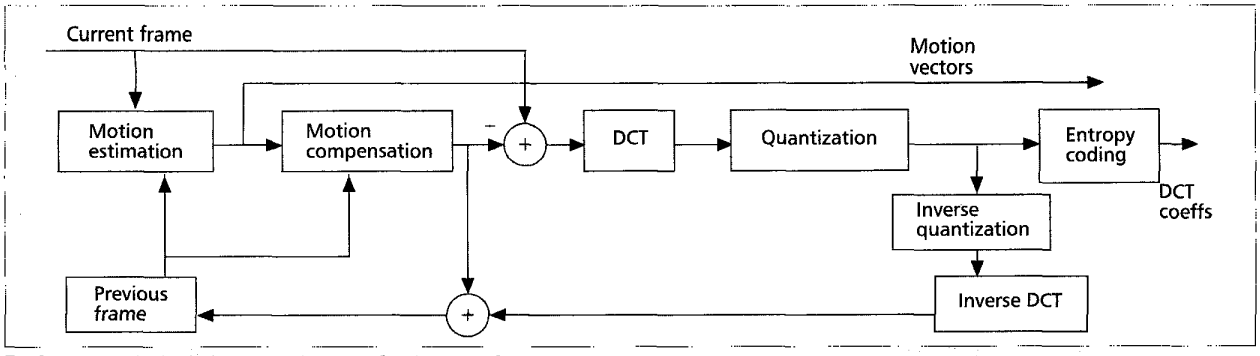
The article is organized as follows. The next section discusses the general problem of wireless video transmission and the various steps that need to be taken by the video encoder and decoder to enable robust video transmission. After that an overview of the ISO MPEG-4 standard is presented, outlining the technologies adopted by the standard, the different functionalities provided by these technologies, and various application areas enabled by these functionalities. This section also describes the timeline of the standard. The following section discusses the error resilience work in the MPEG-4 standard and describes the different tools adopted into the standard to enable robust video communication over noisy wireless channels. The error resilience evaluation criterion used in evaluating the MPEG4 tools is described, and then the part of the MPEG-4 video coding standard that handles arbitrarily shaped video objects and the error resilience aspects of these tools. Finally, the article concludes with a summary and a discussion of future work.

Error-Resilient Video Coding

A block diagram of a typical video compression scheme is shown in Fig. 1. First, each of the current frames of the video sequence is partitioned into rectangular regions of 16 x 16 pixels called *macroblocks*. For each macroblock, the motion estimation stage of the video encoder computes the motion vectors that best represent the location in the previous frame where image pixels of similar intensity values occur. The motion compensation stage applies these motion vectors to the corresponding macroblocks in the previous frame and computes a motion-compensated frame which is an estimate of the current frame based on the previous frame. The intensity differences between this motion-compensated frame and the current frame are then computed. The frame representing these differences is called as the *residual frame*. The residual frame represents the information in the current frame that cannot be predicted from the previous image. This residual frame is then coded by the discrete cosine transform (DCT) stage. The DCT is typically performed on 8 x 8 pixel blocks. The quantized DCT coefficients are then coded using variable-length coding (VLC) schemes such as Huffman coding. Hence, for every frame of the video sequence the video encoder transmits motion vector information, DCT information, and some overhead header information. In order to achieve further coding efficiency, the header and motion vector information are also coded using VLC techniques. These motion-compensated frames of the video sequence are known as *predictive frames* or *inter frames*. Some frames of the video sequence are coded completely with respect to themselves with no motion compensation, and these are known as *intra frames*. Intra frames do not have any motion vector information associated with them.

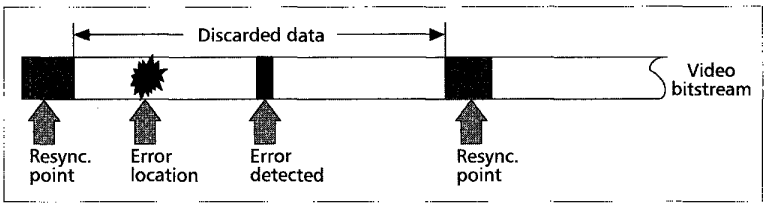
When the compressed video data is transmitted over a wireless communication channel, it is subjected to channel errors in the form of bit errors and burst errors. Typically, in order to make the video codec more resilient to channel

0163-6804/98/$10.00 © 1998 IEEE



■ **Figure 1.** *A block diagram of a typical video encoder.*

degradations, forward error correction (FEC) codes such as Reed-Solomon codes and BCH codes are employed by the encoder to protect the bitstream before transmitting to the decoder [4]. At the decoder, these FEC codes are then used to correct errors in the bitstream due to the channel noise. FEC techniques prove quite effective against random bit errors, but their performance is usually inadequate against longer-duration burst errors. These FEC techniques also come with an increased overhead in terms of the overall bitstream size; hence, some of the coding efficiency gains achieved by the video compression are lost. Typically, we apply FEC to provide a certain level of protection to the compressed bitstream, and the residual errors are handled by the error-resilient video decoder.



■ **Figure 2.** *At the decoder, it is usually not possible to detect the error at the actual error occurrence location; hence, all the data between the two resynchronization points may need to be discarded.*

To handle the residual errors, the following stages are required at the video decoder:

- Error detection and localization
- Resynchronization
- Data recovery
- Error concealment

FEC techniques can also be used to detect errors and pass the location of the errors to the video decoder so that the video decoder can conceal the errors. In addition to FEC, syntactic and semantic error detection techniques are also applied at the video decoder to enable the video decoder to detect when a bitstream is corrupted by channel errors. In a typical block-based video compression technique that uses motion compensation and DCT, the following checks are applied to detect bitstream errors:

- The motion vectors are out of range.
- An invalid VLC table entry is found.
- The DCT coefficient is out of range.
- The number of DCT coefficients in a block exceeds 64.

When the decoder identifies any of these conditions in the process of decoding a video bitstream, it flags an error and jumps to the error-handling procedure. Due to the nature of the video compression algorithms, the location in the bitstream where the decoder detects an error is not the same location where the error has actually occurred but some undetermined distance away from it. This is shown in Fig. 2. Hence, once the decoder detects an error it loses synchronization with the encoder. Resynchronization schemes are then employed for the decoder to fall back into lock step with the encoder. While constructing the bitstream the encoder inserts unique resynchronization words into the bitstream at approximately equally spaced intervals. These resynchronization words are chosen such that they are unique from the valid video bitstream. That is, no valid combination of the video algorithm's VLC tables can produce these words. Upon detection of an error the decoder seeks forward in the bitstream, hunting for this known resynchronization word. Once this word is found, the decoder then falls back in synchronization with the encoder. At this point, the decoder has detected an error, regained synchronization with the encoder, and also isolated the error between the two resynchronization points.

As mentioned above, due to the extensive use of VLC tables, even after error detection typically the decoder can only isolate the error to be somewhere between the resynchronization points, but not pinpoint its exact location. Hence, all of the data that corresponds to the macroblocks between these two resynchronization points needs to be discarded, as shown in Fig. 2. The effects of displaying an image reconstructed from erroneous data can cause highly annoying visual artifacts.

Some data recovery techniques such as reversible decoding enable the decoder to salvage some of the data between the two resynchronization points. These techniques advocate the use of a special kind of VLC table at the encoder in coding the DCTs and motion vector information. These special VLCs have the property that they can be decoded in both the forward and reverse directions. By comparing the forward and reverse decoded data, the exact location of the error in the bit stream can be localized more precisely, and some of the data between the two resynchronization points can be salvaged. The use of these reversible VLCs (RVLCs) is part of the MPEG-4 standard and will be described in greater detail in the following sections.

After data recovery, the effect of the data deemed to be in error needs to be minimized. This is the error concealment stage. One simple error concealment strategy is to simply replace the luminance and chrominance of the erroneous macroblocks with the luminance and chrominance of the corresponding macroblocks in the previous frame of the video sequence. While this technique works fairly well and is simple to implement, more complex technique use some type of estimation strategy to exploit the local correlation that exists within a frame of video data to come up with a better estimate of the missing or erroneous data. These error concealment strategies are essentially post-processing algorithms and

are not mandated by the standard [5–7] However, different implementations of the wireless video systems utilize different kinds of error concealment strategies based on the available computational power and the quality of the channel.

AN MPEG-4 OVERVIEW

MPEG-4 is an ISO/International Electrotechnical Commission (IEC) standard being developed by MPEG. MPEG also developed the Emmy-award-winning standards MPEG-1 and MPEG-2. MPEG-1 is an audiovisual coding standard aimed at addressing the storage and retrieval of multimedia information on a CD-ROM [8]. MPEG-2 followed closely behind MPEG-1 and addresses broadcast TV applications. MPEG-2 is a hugely successful standard with significant acceptance in the marketplace with a number of other applications in addition to broadcast TV [9]. Some of the prominent applications of MPEG-2 include direct broadcast satellite (DBS), digital versatile disk (DVD) and high-definition TV (HDTV). Initially, MPEG-3 was reserved for HDTV applications. However, MPEG-2 was later found to be suitable for HDTV, and it was decided to include HDTV as a separate profile of MPEG-2 and discontinue the MPEG-3 work item.

MPEG-4 is the next audiovisual coding standard from ISO after MPEG-1 and MPEG-2. Unlike the previous two standards, which had one clear application in mind when they were developed, MPEG-4 is a much broader umbrella-type standard that has a number of different technologies, which are targeted at different applications. Initially MPEG-4 was aimed primarily at low-bit-rate video communications; however, its scope was later expanded to be much more of a multimedia coding standard. MPEG-4 is efficient across a wide variety of bit rates ranging from a few kilobits per second to tens of megabits per second. In addition to providing improved coding efficiency, MPEG-4 also provides a number of functionalities. These include:

- The ability to efficiently encode mixed media data such as video, graphics, text, images, audio, and speech, called *audiovisual objects* (AVOs)
- The ability to create a compelling multimedia presentation by compositing these mixed media objects by a compositing script
- Error resilience to enable robust transmission of compressed data over noisy communication channels
- The ability to encode arbitrarily shaped video objects
- Multiplexing and synchronization of the data associated with these objects so that they can be transported over network channels providing a quality of service (QoS) appropriate to the nature of the specific objects
- The ability to interact with the audiovisual scene generated at the receiver end

These functionalities supported by the standard will enable many compelling applications ranging from wireless videophones to Internet multimedia presentations, broadcast TV, and DVD. A standard that supports these diverse functionalities and associated applications is fairly complex. This article focuses only on the error resilience aspects of the MPEG-4 video encoder. The reader is directed to the wealth of information available on MPEG at the official MPEG web site http://www.cselt.it/mpeg and in other recent MPEG-4 publications [10–14].

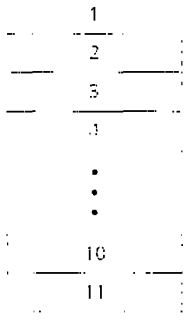


Figure 3. *H.263 GOB numbering for a QCIF (176 x 144) image.*

MPEG-4 reached Committee Draft (CD) standard status in November 1997. MPEG-4 will be released in November 1998, and will be an official International Standard (IS) in January 1999. The MPEG-4 Committee has also taken the approach of versioning to the standard formation process. Version 1 of the standard is now in CD status; version 2 will reach CD status in November 1998. MPEG-4 v. 1 includes a number of useful tools [3], and v. 2 is expected to include some others being developed by the standards body [15]. It is expected that MPEG-4 v. 2 will be backward-compatible with MPEG-4 v. 1.

As a multimedia coding standard, MPEG-4 standardizes tools not only for video coding but also for coding audio, graphics, and text. The standard also includes a systems part, which describes how the audio, video, text, and graphics are synchronized and presented in a compelling manner for various applications. In this article, we limit our discussion to the video error resilience tools that are included in MPEG-4 v. 1. We will describe each of these tools and demonstrate how together they enable robust video transmission over noisy communication channels such as wireless video links.

ERROR RESILIENCE TOOLS IN MPEG-4

A number of tools have been incorporated into the MPEG-4 video encoder to make it more error-resilient. These tools provide various important properties such as resynchronization, error detection, data recovery, and error concealment. There are four tools:

- Video packet resynchronization
- Data partitioning (DP)
- Reversible VLCs (RVLCs)
- Header extension code (HEC)

We describe below each of these tools and its advantages.

RESYNCHRONIZATION

As mentioned earlier, when the compressed video data is transmitted over noisy communication channels, errors are introduced into the bitstream. A video decoder that is decoding this corrupted bitstream will lose synchronization with the encoder (it is unable to identify the precise location in the image where the current data belongs). If remedial measures are not taken, the quality of the decoded video rapidly degrades and quickly becomes totally unusable.

One approach is for the encoder to introduce resynchronization markers in the bitstream at various locations. When the decoder detects an error it can then hunt for this resynchronization marker and regain resynchronization. Previous video coding standards such as H.261 [16] and H.263 [17, 18] logically partition each of the images to be encoded into rows of macroblocks called *groups of blocks* (GOBs). These GOBs correspond to a horizontal row of macroblocks for QCIF images. Figure 3 shows the GOB numbering scheme for H.263 for QCIF resolutions. In the

Figure 4. *Position of the resynchronization markers in the bitstream for a baseline H.263 encoder with GOB headers.*

Figure 5. *Position of the resynchronization markers in the bitstream for an MPEG-4 encoder with video packets.*

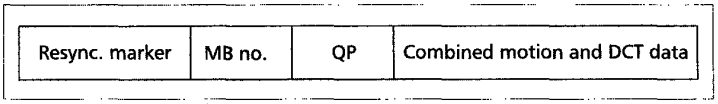


Figure 6. *Organization of the data within a video packet.*

case of CIF images, each row of macroblocks consists of two GOBs. To be able to provide error resilience, H.263 provides the encoder an option of inserting resynchronization markers at the beginning of each GOB (Fig. 4). Hence, for QCIF images these resynchronization markers are allowed to occur only at the left edge of the images. The smallest region to which the error can be isolated and concealed in this case is thus one row of macroblocks. MPEG-2 has a similar optional slice resynchronization scheme.

MPEG-4 provides a similar method of resynchronization with one important difference: the MPEG-4 encoder is not restricted to inserting the resynchronization markers only at the beginning of each row of macroblocks. The encoder has the option of dividing the image into video packets, each made up of an integer number of consecutive macroblocks. These macroblocks can span several rows of macroblocks in the image and can even include partial rows of macroblocks. One suggested mode of operation of the MPEG-4 encoder is for it to insert a resynchronization marker periodically every K bits. Note that when there is significant activity in one part of the image, the macroblocks corresponding to this area generate more bits than other parts of the image. Now, if the MPEG-4 encoder inserts the resynchronization markers at uniformly spaced bit intervals (say every 512 bits), the macroblock interval between the resynchronization markers is a lot closer in the high-activity areas and a lot farther apart in low-activity areas. Thus, in the presence of a short burst of errors, the decoder can quickly localize the error to within a few macroblocks in the important high-activity areas of the image and preserve the image quality in these areas. In the case of baseline H.263, where the resynchronization markers are restricted to be at the beginning of the GOBs, it is only possible for the decoder to isolate the errors to a fixed GOB independent of image content. Hence, effective coverage of the resynchronization marker is reduced from that of the MPEG-4 scheme (Fig. 5). The recommended spacing of the resynchronization markers in MPEG-4 is based on the bit rates. For 24 kb/s it is recommended to insert them at intervals of 480 bits; for bit rates between 25 to 48, every 736 bits. The later version of H.263 [19] adopted a resynchronization scheme similar to MPEG-4 in an additional annex (Annex K, Slice Structure Mode).

Note that in addition to inserting the resynchronization markers at the beginning of each video packet, the encoder also needs to remove all data dependencies that exist between the data belonging to two different video packets within the same image. This is required because, even if one video packet in the current image is corrupted due to errors, the other packets can be decoded and utilized by the decoder. In order to remove these data dependencies, the encoder inserts two additional fields in addition to the resynchronization marker at the beginning of each video packet, as shown in Fig. 6. These are:

- The absolute macroblock number of the first macroblock in the video packet, MB no., which indicates the spatial location of the macroblock in the current image
- The quantization parameter, QP, which denotes the default quantization parameter used to quantize the DCT coefficients in the video packet

The encoder also modifies the predictive encoding method used for coding the motion vectors such that there are no predictions across the video packet boundaries.

DATA PARTITIONING

After detecting an error in the bitstream and resynchronizing to the next resynchronization marker, the decoder has now isolated the data in error to be in the macroblocks between the two resynchronization markers. Typical video decoders discard all these macroblocks as being in error and replace the luminance and chrominance of these macroblocks with the luminance and chrominance from the corresponding macroblocks in the previous frame to conceal the errors. One of the main reasons for this is that between two resynchronization markers, the motion and DCT data for each macroblock are coded all together. Hence, when the decoder detects an error, whether the error occurred in the motion or DCT part, all the data in the video needs to be discarded. Due to the uncertainty of the exact location where the error occurred, the decoder cannot be sure that either the motion or DCT data of any macroblocks in the packet is not erroneous. Figure 6 shows the organization of the video data within a packet for a typical video compression scheme without data partitioning.

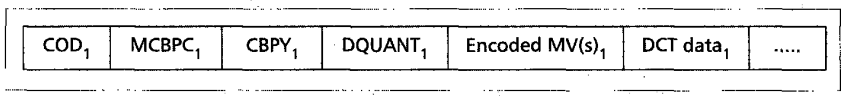


Figure 7. *Bitstream components for each macroblock within the video packet.*

Within the combined motion and DCT data part, each macroblock's (MB's) motion vectors and DCT coefficients are encoded. Note that this part is of variable length and also in general contains a lot more data than the three preceding header fields.

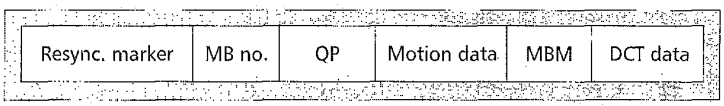
Figure 7 shows the syntactic elements for each MB in the case of a typical video encoder such as H.263. This data is repeated for all macroblocks in the packet. The subscripts indicate the macroblock number. The COD is a 1-bit field used to indicate whether a certain macroblock is coded or not. The MCBPC is a variable-length field used to indicate two things:

- The mode of the macroblock, such as INTRA, INTER, INTER4V (8 x 8 motion vectors), and INTRA+Q (the quantization factor is modified for this NB from the previous MB)
- Which of the two chrominance blocks of the MB is coded

DQUANT is an optional 2-bit fixed-length field used to indicate the incremental modification to the quantization value from the previous macroblock's quantization value. CBPY is a VLC that indicates which of the 4 blocks of the MB are coded. Encoded MVs are the motion vector differences by a VLC. Note that the motion vectors are predictively coded with respect to the neighboring motion vectors; hence, we only code the motion vector differences. DCT data comprises the 64 DCT coefficients actually encoded via zig-zag scanning and run-length-encoded, and then a VLC table [3]

Previous researchers have applied the idea of partitioning the data into higher- and lower-priority data in the context of ATM or other packetized networks to achieve better error resilience properties [20–22].

In MPEG-4, the data partitioning mode partitions the data within a video packet into a motion part and a texture part separated by a unique motion boundary marker (MBM), as



■ **Figure 8.** *Bitstream organization with data partitioning for motion and DCT data.*

shown in Fig. 8. This figure shows the bitstream organization within each video packet with data partitioning. The subscripts indicate the macroblock numbers. Note that compared to Fig. 6, the motion and DCT parts are now separated by an MBM. In order to minimize the differences from the conventional method, we maintain all the same syntactic elements as the conventional method and reorganize them to enable data partitioning. All the syntactic elements that have motion-related information are placed in the motion partition, and all those that relate to DCT data are placed in the DCT partition. Figure 9 shows the bitstream elements after reorganization of the motion part, and Fig. 10 shows bitstream elements of the DCT part. Note that we now place COD, MCBPC, and the MVs in the motion part and relegate the CBPY, DQUANT, and DCTs to the DCT part of the packet.

The MBM marks the end of the motion data and the beginning of the DCT data. The MBM is computed from the motion VLC tables using a search program such that this marker word is Hamming distance 1 from any possible valid combination of the motion VLC tables [23]. This word is uniquely decodable from the motion VLC tables. It indicates to the decoder the end of the motion information and the beginning of the DCT information. The number of macroblocks (NMB) in the video packet is implicitly known after encountering the MBM. Note that the MBM is only computed once based on the VLC tables and is fixed in the standard. Based on the VLC tables in MEPG-4, the MBM is a 17-bit word whose value is 1 1111 0000 0000 0001. When an error is detected in the motion section, the decoder flags an error and replaces all the macroblocks in the current packet with skipped blocks until the next resynchronization marker. Resynchronization occurs at the next successfully read resynchronization marker. If any subsequent video packets are lost before resynchronization, those packets are replaced by skipped macroblocks as well. When an error is detected in the texture section (and no errors are detected in the motion section) the NMB motion vectors are used to perform motion compensation. The texture part of all the macroblocks is discarded and the decoder resynchronizes to the next resynchronization marker.

If an error is not detected in the motion or texture section of the bitstream, and the resynchronization marker is not found at the end of decoding all the macroblocks of the current packet, an error is flagged. In this case, only the texture part of all the macroblocks in the current packet needs to be discarded. Motion compensation can still be applied for the NMB macroblocks since we have higher confidence in the motion vectors since we got the MBM.

Hence, the advantages of this data partitioning method are twofold. We have a more stringent check on the validity of the motion data since we need to get the MBM at the end of the decoding of motion data in order for us to consider the motion data to be valid. In case we have an undetected error in the motion and texture, but do not end in the correct position for the next resynchronization marker, we do not need to discard all the motion data since we can salvage the motion data as validated by the detection of MBM.

VLC code with Hamming weight of 1	RVLC
1	111
01	1011
001	10011
0001	100011

■ **Table 1.** *An RVLC is formed by adding a fixed prefix and suffix (in this case a 1) to a constant Hamming weight VLC.*

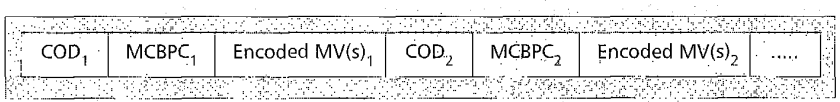
Reversible Variable-Length Codes

As mentioned above, one of the problems with transmitting compressed video over error-prone channels is the use of variable-length codes. During the decoding process, if the decoder detects an error while decoding VLC data, it loses synchronization and hence typically has to discard all the data up to the next resynchronization point. RVLCs alleviate this problem and enable the decoder to better isolate the error location by enabling data recovery in the presence of errors. RVLCs are special VLCs that have the prefix property when decoding them in both the forward and reverse directions. Hence, they can be uniquely decoded in both the forward and reverse directions. The advantage of these code words is that when the decoder detects an error while decoding the bitstream in the forward direction, it jumps to the next resynchronization marker and decodes the bitstream in the backward direction until it encounters an error. Based on the location of the two error locations, the decoder can recover some of the data that would otherwise have been discarded. In Fig. 11, only data in the shaded area is discarded; note that if RVLCs were not used, all the data between two consecutive resynchronization markers would have to be discarded.

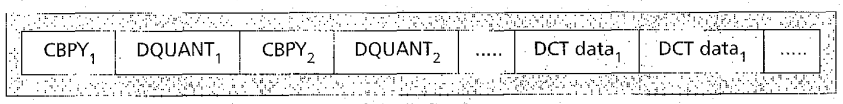
The Hamming weight of a binary sequence is defined as the number of 1s in the sequence. For example, the Hamming weight of 110110 is 4. One way to generate RVLCs is to take a constant Hamming weight code and add a fixed-length prefix and suffix to it. Consider a code of Hamming weight 1 shown in the first column of Table 1. We can add a constant prefix and suffix 1 to it and form an RVLC, as shown in the second column of Table 1. In this table an additional symbol 0 is added to the RVLC to complete the code.

Note that the decoder can forward and backward decode the above RVLC by counting the number of 1s in the code to determine the termination of the code. In the above example the decoder knows it reached the end of a code word when it decodes three 1s. More complex and complete RVLCs can be formed by increasing the length of the fixed-length prefix, and also by including the bit inverses of the RVLCs. In the latter case, the decoder needs to count the number of 0s in the symbol to determine the end of the symbol.

In general, the RVLC property reduces the coding efficiency of the VLC tables [24]. By proper use of training video sequences, the RVLCs are made to match the probability characteristics of the DCT coefficients as closely as possible

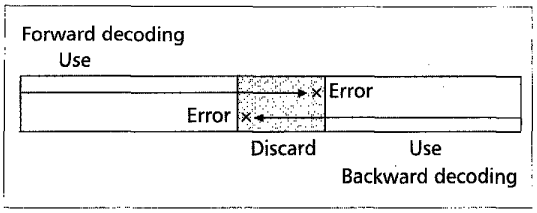


■ **Figure 9.** *Bitstream components of the motion data.*



■ **Figure 10.** *Bitstream components of the DCT data.*

such that the RVLCs still maintain the ability to compactly pack the bitstream while retaining the error resilience properties. MPEG-4 utilizes such efficient and robust RVLC tables for encoding DCT coefficients. Golomb-Rice codes can also be modified to have the reversible property and still maintain good coding efficiency properties [25].



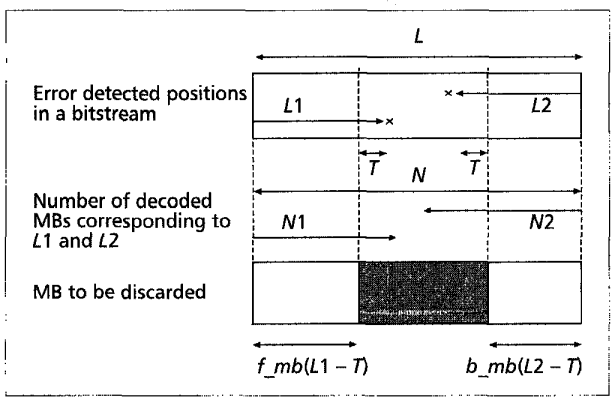
■ **Figure 11.** *The data that needs to be discarded can be reduced by using forward and reverse decoding property of an RVLC.*

Note that the MPEG-4 standard does not standardize the method in which an MPEG-4 decoder has to apply the RVLCs for error resilience. However, one possible strategy that is recommended is shown in Fig. 12. The bitstream is first decoded in the forward direction, and the decoding process is finished if no error is detected. If an error is detected, two-way decode is carried out and the following bit-discarded strategies are used. Let

- L: Total number of bits for DCT coefficients part in a video packet
- N: Total number of MBs in a video packet
- $L1$: Number of bits which can be decoded in forward decoding
- $L2$: Number of bits which can be decoded in backward decoding
- $N1$: Number of MBs which can completely be decoded in a forward decoding
- $N2$: Number of MBs which can completely be decoded in a backward decoding
- $f\_mb(S)$: Number of decoded MBs when S bits can be decoded in a forward direction
- $b\_mb(S)$: Number of decoded MBs when S bits can be decoded in a backward direction
- T: Threshold (90 is the MPEG-4 recommended value)

This strategy is applied in the case that $L1 + L2 < L$ and $N1 + N2 < N$. Here $f\_mb(L1 - T)$ MBs from the beginning and $b\_mb(L2 - T)$ MBs from the end are used. MBs of the dark part are discarded.

Similar strategies are recommended for the other cases of $L1$ and $L2$ and for the treatment of the INTRA-coded macroblocks in the bitstream. In the MPEG-4 evaluations, RVLCs have been shown to provide a significant gain in subjective video quality in the presence of channel errors by enabling more data to be recovered. Note that for RVLCs to be most effective, all the data that is coded using the same RVLC tables has to occur together. Hence, in MPEG-4 RVLCs are utilized in the data partitioning mode which modifies the bitstream syntax such that all the DCT coefficients occur together and hence can be coded effectively using the RVLC tables. In MPEG-4 v. 1, RVLCs are used only for the DCT data. Currently, experiments are underway for MPEG-4 v. 2 which advocate the use of RVLCs for coding the motion vector information as well.



■ **Figure 12.** *One decoding strategy when errors are detected in the MPEG-4 bitstream and RVLCs are used to encode the DCT coefficients.*

Header Extension Code (HEC)

Some of the most important information the decoder needs to be able to decode the video bitstream is the header data. This data includes information about the spatial dimensions of the video data, the time stamps associated with the decoding and presentation of this video data, and the mode in which the current video object is encoded (whether predictive or INTRA). If some of this information is corrupted due to channel errors, the decoder has no other recourse but to discard all the information belonging to the current video frame. In order to reduce the sensitivity of this data, a technique called *header extension code* (HEC) is introduced into the MPEG-4 standard. In each video packet, a 1-bit field called the HEC bit is introduced. If this bit is set, the important header information that describes the video frame is repeated in the video packet. By checking this header information in the video packets against the information received at the beginning of the video frame, the decoder can ascertain if the video frame header is received correctly. If the video frame header is corrupted, the decoder can still decode the rest of the data in the video frame using the header information within the video packets.

In MPEG-4 verification tests, it was found that the use of HEC significantly reduced the number of discarded video frames and helped achieve higher overall decoded video quality.

MPEG-4 Error Resilience Evaluation Criterion

All the tools that have been accepted into the MPEG-4 standard are evaluated thoroughly and independently verified by two parties before being accepted into the standard [26]. The techniques are rigorously tested over a wide variety of test sequences, bit rates, and error conditions. A set of test sequences in QCIF format (frame size 144 x 176 pixels, progressive scan, 4:2:0 subsampling format) are coded at 24 and 48 kb/s. The compressed bitstreams are corrupted using random bit errors, packet loss errors, and burst errors. A wide variety of random bit errors — BER of 10^{-2}–10^{-3}, burst errors of durations 1, 10, and 20 ms, and packet loss errors of varying lengths (96–400 bits) and error rates (10^{-2} and 3×10^{-2}) — have been tested [27]. To provide statistically significant results, 50 tests are performed for each of the mentioned error conditions. In each test, errors are inserted in the bitstreams at different locations. This is achieved by changing the seed of the random number generators used to simulate the different error conditions.

For each test, the peak signal-to-noise ratio (PSNR) of the video decoded from the corrupted stream and the original video is computed. Then the average PSNR of the 50 runs is computed for each frame in the sequence. To evaluate the performance of the proposed techniques, the average PSNR values generated by the error-resilient video codec with and without each error resilience tool are compared. The techniques are also compared on the basis of the number of

MPEG-4 test sequence	Average PSNR without data partitioning (dB)	Average PSNR with data partitioning (dB)
Silent	24.81	25.92
Container ship	25.58	28.30
Mother and daughter	27.21	29.17
Foreman	19.34	20.19
Coast Guard	20.74	22.02

■ **Table 2.** *The performance of the MPEG-4 error-resilient video encoder with and without data partitioning for random errors (BER 10^{-3}) on the MPEG-4 test sequences.*

frames discarded due to errors and the number of bits discarded. Before accepting a technique into the standard the additional bit rate overhead incurred due to this technique compared to the baseline is compared to the gains provided by the technique.

Techniques that consistently achieve superior performance independently verified by two different parties are then accepted into the standard. As an example, the data partitioning technique achieves an average of 2 dB performance gain over the wide range of test sequences and error conditions. Table 2 summarizes the performance of the error-resilient video codec with and without data partitioning for the entire set of test sequences on one of the error conditions. Each entry in the table is the average of the PSNR (luminance component only) of all 100 frames across all 50 runs. This is also illustrated in graphical form in Fig. 13, which shows the performance with and without data partitioning. Similar consistently better performance gains were obtained for the other error conditions and test sequences.

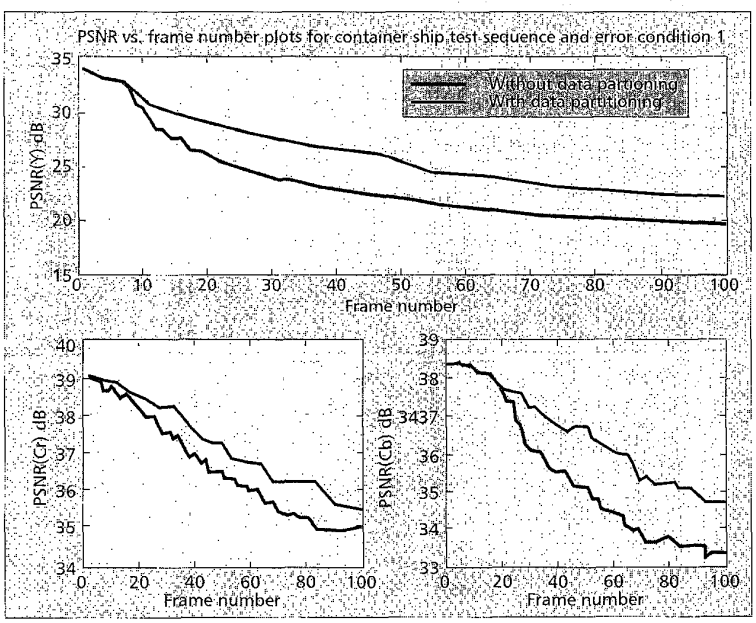
Because of the MBM insertion in the video packet, the data partitioning technique results in a higher overall bit rate. The amount of overhead bit rate depends on the frequency at which the video packets occur. The overhead bit rate resulting from the MPEG-4 test conditions is about 2–3 percent.

This 2–3 percent overhead in additional bit rate was deemed acceptable for the amount of PNSR and subjective quality gain achieved, which resulted in acceptance of the technique. Similar results were demonstrated by the video packet-based resynchronization tool, HEC, and RVLC.

Arbitrarily Shaped Video Objects

The ISO MPEG-4 video coding standard is the first video coding standard that supports the encoding of not only rectangular video frames but also arbitrarily shaped video objects [3]. For instance, using this standard it is possible to segment the incoming video sequence into semantically meaningful video objects and encode each of these objects separately in the bitstream. However, note that the MPEG-4 standard does not standardize the segmentation methods, only the method for encoding the segment sequences. The encoder can choose to segment the input video sequences into independent video objects by automatic, semi-automatic, or manual means as demanded by the application. Consider a news clip consisting of newscasters, a background scene, a news clip of a ballerina, and a text annotation, as shown in Fig. 14. In this sequence the video objects are segmented using automatic blue-screening techniques. Using the MPEG-4 standard, it is possible to encode each of these as a separate video object and also transmit a compositing script that describes how these video objects need to be composed to make a compelling presentation at the decoder. One of the advantages of this approach is that, depending on the channel bandwidth and the relative importance of each video object, the temporal and spatial resolution of each individual video object can be independently controlled, thereby achieving more efficient utilization of the available channel bandwidth. For example, in this scene the fast-moving ballerina in the news clip can be coded at a much higher temporal update rate (frame rate) than the slow-moving newscasters. Note that these individual video objects can be graphics or text objects. Another advantage of this object-based coding approach is that the encoder can now make a compelling multimedia presentation using mixed media types such as video, graphic, and text objects, and encode each object efficiently using the encoder best suited for that data. Other advantages include the ability to edit the compressed bitstream at the decoder by modifying the compositing script to make another presentation without having to decode and recode the video data. Video games, Internet streaming media, and multimedia presentations are expected to benefit from this object-based coding functionality supported by MPEG-4.

However, in order to code arbitrarily shaped video objects, it is no longer sufficient for the video encoder to encode just the motion vectors and DCT coefficients; the shape of the video object now needs to be encoded efficiently at each instant of time. MPEG-4 has standardized a very efficient shape coding method that codes both binary and grayscale shapes, from lossy to lossless. In MPEG-4, to make the transmission of the video data corresponding to arbitrarily shaped video objects, all the above error resilience approaches are extended to handle not only the rectangular video frames but



■ **Figure 13.** *Performance of the MPEG-4 error-resilient video encoder with and without data partitioning for random errors (BER 10^{-3}) on one of the MPEG-4 test sequences.*

also the arbitrarily shaped video objects. For example, in data partitioning mode the data within each video packet is now partitioned into three partitions corresponding to shape, motion, and texture. The resynchronization strategy, HEC, and RVLCs also extend to the arbitrarily shaped object mode of MPEG-4. MPEG-4 thus has the ability to transmit object-based coded media robustly over noisy communication channels.

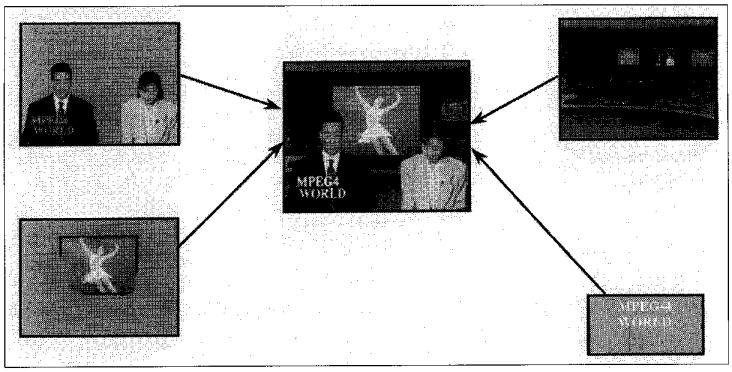
CONCLUSIONS

In this article we present the error resilience aspects of the ISO MPEG-4 video standard. A number of tools have been adopted into the ISO MPEG-4 video standard which enable robust transmission of compressed video over noisy communication channels such as wireless links. We describe these tools in detail and highlight their relative advantages. These are tools that mitigate the effects of residual errors in the video decoder.

There are, however, a number of other methods that further improve the performance of the wireless video codec which the standard does not specify. If the encoder and decoder are aware of the limitations imposed by the communication channel, they can further improve the video quality using these methods. They include encoding methods such as rate control to optimize the allocation of the effective channel bit rate between various parts of video to be transmitted, and intelligent decisions on when and where to place INTRA refresh macroblocks to limit the error propagation. Decoding methods such as superior error concealment strategies which further conceal the effects of erroneous macroblocks by estimating them from correctly decoded macroblocks in the spatiotemporal neighborhood can also significantly improve the effective video quality.

This article mainly focuses on the error resilience aspects of the video layer. There are a number of error detection and correction strategie,s such as forward error correction (FEC), that add on top of these techniques to further improve the reliability of the transmitted video data. These FEC codes are typically provided in the systems layer [28] and the underlying network layers. Some networks may also provide a back channel, which can effectively be utilized in retransmitting some of the corrupted video data — if the ensuing network delay is tolerable by the application.

Given all these advances in video coding technology, coupled with the technological advances in processor technology, memory devices, and communication systems, wireless video communications is fast becoming a very compelling application.



■ **Figure 14.** *MPEG-4 provides the technology to encode arbitrarily shaped video objects.*

REFERENCES

[1] K. R. Rao and J. J. Hwang, *Techniques and Standards For Image, Video and, Audio Coding*, Upper Saddle River, NJ: Prentice Hall, 1996.

[2] B. Sklar, "Raleigh Fading Channels in Mobile Digital Communication Systems, Part I: Characterization," *IEEE Commun. Mag.*, vol. 35, pp. 90–100, Sept. 1997.

[3] ISO/IEC 144962-2, "Information Technology–Coding of Audio-Visual Objects: Visual," Committee draft, Oct. 1997.

[4] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Englewood Cliffs, NJ: Prentice Hall, 1994.

[5] H. Sun *et al.*, *Error Concealment for Robust Decoding of MPEG Compressed Video, Signal Processing: Image Communication*, vol. 10, pp. 249–68, 1997.

[6] P. Salama, N. Shroff, and E. Delp, "A Fast Suboptimal Approach to Error Concealment in Encoded Video Streams," *Proc. ICIP '97*, Santa Barbara, CA, Oct. 1997.

[7] W. Kwok and H. Sun, "Multidimensional Interpolation for Spatial Error Concealment, *IEEE Trans. Consumer Elect.*, vol. 39, no. 3, pp. 455–60, Aug. 1993.

[8] ISO/IEC 11172-2, "Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at Up to about 1.5 Mb/s: Part 2: Video," Aug. 1993.

[9] ISO/IEC 13818-2, "Information Technology – Generic Coding of Moving Pictures and Associated Audio Information: Video," 1995.

[10] R. Talluri, "MPEG-4 Status and Directions," Proc. SPIE Critical Revs. of Standards and Common Interfaces for Video Info. Sys., Philadelphia, PA, Oct. 1997, pp. 252–62.

[11] F. Pereira, "MPEG-4: A New Challenge for the Representation of Audio-Visual Information," *Proc. PCS '96*, Mar. 1996, Melbourne, Australia, pp. 7–16.

[12] T. Sikora, "The MPEG-4 Video Standard Verification Model", *IEEE Trans. CSVT*, vol. 7, no. 1, Feb. 1997.

[13] I. Moccagatta and R. Talluri, "MPEG-4 Video Verification Model: Status and Directions," *J. Imaging Tech.*, vol. 8, 1997, pp. 468–79.

[14] T.Sikora and L.Chiariglione, "The MPEG-4 Video Standard and Its Potential for Future Multimedia Applications," *Proc. IEEE ISCAS Conf.*, Hong Kong, June 1997.

[15] ISO/IEC JTC1/SC29/WG11, "Video Group, MPEG-4 Video Verification Model Version 9.0," N1869, Fribourg, Oct. 1997.

[16] ITU-T Rec. H.320, "Narrowband Visual Telephone Systems and Terminal Equipment," Mar. 1996.

[17] ITU-T Rec. H.263, "Video Coding for Low Bitrate Communication," ITU-T SG 15, Oct. 1995.

[18] K. Rijkse, "ITU Standardization of Very Low Bitrate Video Coding Algorithms," *Sig. Processing: Image Commun.*, vol. 7, nos. 4–6, Nov. 1995, pp. 553–65.

[19] ITU-T Rec. H.263, "Version 2, Video Coding for Low bitrate Communication," Jan. 1998.

[20] M. Ghanbari, "Two-Layer Coding of Video Signals for VBR Networks," *IEEE JSAC*, vol. 7, June 1989, pp. 771–81.

[21] M. Nomura, T. Fujii, and N. Ohta, "Layered Packet-Loss Protection for Variable Rate Video Coding Using DCT," *Proc. Int'l. Wksp. Packet Video*, Sept. 1988.

[22] P. Bahl and I. Chlamtac, "H.263 Based Video Codec for Real-Time Visual Communications over Wireless Radio Networks," *Proc. IEEE ICUPC*, San Diego, CA, pp. 773–79, Oct. 1997.

[23] R. Talluri *et al.*, "Error Concealment by Data Partitioning," to appear in *Sig. Processing: Image Commun.*, 1998.

[24] Takashima, Wada, and Murakami, "Reversible Variable Length Codes," *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, Feb./Mar./Apr. 1995, pp. 158–62.

[25] G. Wen and J. Villasenor, "A Class of Reversible Variable Length Codes for Robust Image and Video Coding," *Proc. ICIP '97*, Santa Barbara, CA, vol. 2, Oct. 1997, pp. 65–68.

[26] ISO/IEC JTC1/SC29/WG11, "Adhoc Group on Core Experiments on Error Resilience aspects of MPEG-4 Video, Description of Error Resilience Core Experiments," N1473, Maceio, Brazil, Nov. 1996.

[27] T. Miki *et al.*, "Revised Error Pattern Generation Programs for Core Experiments on Error Resilience," ISO/IEC JTC1/SC29/WG11 MPEG96/1492, Maceio, Brazil, Nov. 1996.

[28] ITU-T Rec. H.223, "Multimedia Protocol for Low Bitrate Mobile Communications," Mar. 1996.

BIOGRAPHY

RAJ TALLURI [M] (talluri@ti.com) received his Ph.D. in electrical engineering from the University of Texas at Austin. He is currently with Texas Instruments Semiconductor Research Laboratories where he manages research activities in various aspects of video technology. He has been an active participant at MPEG meetings and has represented Texas Instruments at MPEG4 for the past three years. He has made a number of technical contributions to the MPEG4 standard and chaired a number of subgroups at MPEG4. He is co-program chair of the IEEE Workshop on Computer Vision Applications, WACV '96. He has authored a number of journal papers and conference articles in the area of video coding and computer vision. His current research interests include signal and image processing, wireless video coding, and computer vision. He is a member of the IEEE Signal Processing Society.