CHAPTER

# Communicating Pictures: Delivery Across Networks 11

## CHAPTER OUTLINE

Video compression algorithms rely on spatio-temporal prediction combined with variable-length entropy encoding to achieve high compression ratios but, as a consequence, they produce an encoded bitstream that is inherently sensitive to channel errors. This becomes a major problem when video information is transmitted over unreliable networks, since any errors introduced into the bitstream during transmission will rapidly propagate to other regions in the image sequence.

In order to promote reliable delivery over lossy channels, it is usual to invoke various error detection and correction methods. This chapter firstly introduces the requirements for an effective error-resilient video encoding system and then goes on to explain how errors arise and how they propagate spatially and temporally. We then examine a range of techniques that can be employed to mitigate the effects of errors and error propagation. We initially consider methods that rely on the manipulation of network parameters or the exploitation of network features to achieve this; we then we go on to consider two methods, EREC and PVQ, where the bitstream generated by the codec is inherently robust to errors. Finally we present decoder-only methods that conceal rather than correct bitstream errors. These deliver improved subjective quality without adding transmission overhead.

## 11.1 The operating environment

### 11.1.1 Characteristics of modern networks

#### *The fixed IP network*

The *internet* is essentially a collection of independent packet-switched networks operating under the same protocol and connected via routers. Each packet has a header that identifies its source and destination IP addresses, where packet delivery to the destination is conducted using TCP/IP protocols. The *Internet Protocol* (IP) provides the basis for packet delivery and the *Transmission Control Protocol* (TCP) provides a best-effort delivery mechanism using *Automatic Repeat reQuest* (ARQ) techniques, but with no bounds on delivery time. It is thus an unreliable, best-effort, connectionless packet delivery service with no guarantee of the actual delivery of packets.

For less reliable networks and real-time delay-constrained applications, the *User Datagram Protocol* (UDP) is normally preferred. The *Real-time Transport Protocol* (RTP) operates on top of UDP and contains essential timing and sequence information.

For colour and a higher quality version of Figure 11.8 please refer to the electronic version or the website.

Alongside RTP, the *Real-Time Control Protocol* (RTCP) enables quality of service (QoS) feedback from the receiver to the sender to enable it to adjust its transmission characteristics to better suit the prevailing network conditions. This is often used in conjunction with the *Real-Time Streaming Protocol* (RTSP) for control of streaming media servers and for establishing and controlling media sessions.

### The wireless network

Most modern wireless networks also operate under IP protocols to support seamless integration with the internet backbone. The band-limited and error-prone nature of the wireless environment, however, needs additional mechanisms for adaption. These are normally based around a suite of operating modes combining different error control and modulation schemes (MCS modes) that can be selected according to prevailing channel conditions. They are implemented in the *Physical Layer* (PHY) and *Medium Access Control* (MAC) layers of the modem.

There are two main classes of wireless network and these are converging rapidly under new 4G standards. *Wireless Local Area Networks* (WLANs) are invariably based on IEEE 802.11 standards and offer local high speed data transfer via hotspots. The *digital cellular network* was originally designed to support voice and data traffic but now, under the 3G and 4G standards, provides the primary means of video downloading and conversational video services.

We can see the effect of a typical wireless channel on system performance in Figure 11.1. This shows data logged on a wireless video research platform at the University of Bristol. The data logging traces show how an 802.11 WiFi link adapts as a user moves away from the access point (time: 1–70 s) and then back toward it (time: 70–140 s). This illustrates how the radio environment changes and the effect that this has on packet delays, packet delay variability, linkspeed (modulation and coding modes), and the bit rate at the application layer. Clearly, our source coding

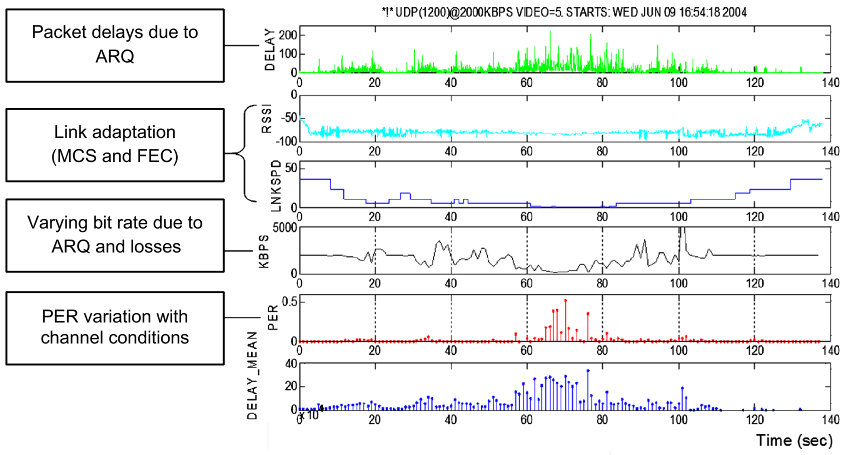


**FIGURE 11.1**

Logged radio link parameter variations with channel conditions for a wireless video trial.

methods need to adapt to these variable bandwidth conditions, while the addition of channel coding or other means of error resilience is needed to cope with the errors introduced during transmission.

### 11.1.2 Transmission types

#### *Downloads and streaming*

Access to a video file or stream can be achieved in many different ways. In the simplest case this can be in the form of a file transfer where data is stored on the local machine and then opened in an appropriate player. *Progressive downloads* are also file transfers, typically using the HTTP protocol, and the files are also stored locally, but they are more intimately linked to the player. The player can start playback before the download is complete but there is generally no interactive support. As the file is downloaded from the server, it will be buffered and when sufficient data is available for continuous display, it will begin to play.

In the case of streaming, the video is not usually stored on the user's device but is played directly from a buffer. A streaming server will interact with the user's player to enable interactive navigation through the video (e.g. FFWD, RWD, etc. via RTSP).

#### *Interactive communication*

With streamed video, playout will commence when the receiver buffer is sufficiently full to ensure sustained playout. Clearly, if conditions dictate that the network cannot support the encoded video bit rate, then the buffer will progressively empty and eventually the video will stutter and freeze. In cases where interaction is required, the demands on delay are much more stringent; typically 200–300 ms is cited as the maximum tolerable for conversational services to avoid unnatural hesitations.

#### *Unicast transmission*

In the case of unicast transmission, one sender is connected to one receiver. This is mandatory for conversational services, but can be inefficient for other services as every client demands its own bandwidth. For the case of a single transmitter unicasting a video at $B$ bps to $N$ receivers, the total bandwidth required is $NB$ bps. The advantage of unicast transmission is that there is an intimate link between the sender and receiver and hence a feedback channel can be established to signal errors and thus support retransmission and error control. Unicast streaming is used by operators such as YouTube and Skype.

#### *Multicast transmission*

In the case of multicast transmission, many receivers join a single stream. This is much more efficient than the unicast case, as our $N$ receivers now only consume $B$ bps. However, all receivers will experience different channel conditions and, if the number of receivers is large, then feedback mechanisms become prohibitive and other error control mechanisms must be invoked. Multicasting thus offers much less service flexibility than unicasting, but provides much better bandwidth flexibility. Multicast transmission is used by operators such as Ustream.

### 11.1.3 Operating constraints

As discussed above, IP streaming, conferencing, and entertainment video have been primary drivers for the growth in multimedia communication services over fixed internet and mobile networks. The channels associated with these networked services can be characterized by the following three attributes:

1. **The bandwidth of the wireless channel is limited:** This necessitates the use of sophisticated source coding techniques to match the bit rate for the signal to the capacity of the channel.
2. **The channel quality for wireless networks can be highly variable:** This is due to signal fading and interference giving rise to erroneous transmission and the need for error protection, error correction, and retransmissions.
3. **Congestion can occur in packet switched networks:** This can lead to dropped packets and variable delays. These demand intelligent buffering and rate control strategies and impose strict bounds on retransmission.

We have seen how attribute (1) can be addressed by exploiting data correlations in a video sequence—using inter-frame prediction (motion estimation), intra-frame prediction, and variable length entropy encoding. However, as a consequence these methods produce an encoded bitstream that is inherently sensitive to channel errors, and this presents a particular problem in the context of attributes (2) and (3).

Clearly we want our encoder to exhibit excellent rate–distortion (or rate–quality) performance, but what the user is really interested in is the quality at the decoder, as this is what he or she experiences. In fact, the rate–distortion performance of the encoder is irrelevant if all packets are lost during transmission. In order to promote reliable delivery over lossy channels, it is usual to invoke various additional error detection and correction methods. In summary, the encoded video bitstream needs to be error resilient as well as efficient in a rate–distortion sense.

### 11.1.4 Error characteristics

#### *Types of errors*

Errors that occur during transmission can be broadly divided into two categories: *random bit errors* and *erasure errors*. The former are normally modeled with a uniform pdf whereas the latter are bursty in nature. Bit errors are normally characterized by their *bit error rate* or BER and can be caused by thermal noise. Erasure errors affect longer segments of data and can be caused by congestion in a packet network or fading, shadowing and drop-out in a wireless network. They will often exceed the correcting capability of any channel coding system and will demand retransmission or lead to the loss of a whole packet (itself an erasure at the *application layer*).

#### *Test data*

In order to model the effects of errors on a video signal, we need to ensure that the encoding method used is evaluated with representative test data. This can be obtained through data logging over prolonged periods using a real system, such as is illustrated

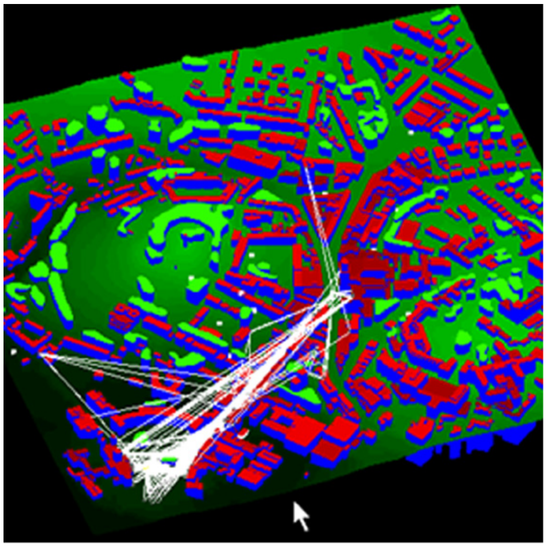


**FIGURE 11.2**

Radiowave propagation modeling using ray tracing. (Courtesy of A. Nix.)

in Figure 11.1, or by emulating system performance using wireless propagation modeling or by statistical channel modeling. We will not discuss Figure 11.1 in detail here, but will return to some of the things it depicts later in the chapter.

In practice, while it is useful to characterize the performance of a system using more generic models, for real-world usefulness it is preferable to use propagation models combined with appropriate network simulation tools to generate realistic bit error patterns at the application layer. Such methods are described in Ref. [1]. An example output from a ray tracing analyzer, that provides accurate representations of multipath effects in a given environment for a specific wireless configuration, is illustrated in Figure 11.2 and is described in Ref. [2].

### *Types of encoding*

The encoding mode will have a significant influence on the way that errors manifest themselves and propagate spatially and temporally. For example, transform coding in combination with variable length coding can cause spatial error propagation within a block if symbol synchronization is lost. Propagation across multiple blocks will occur if an EOB codeword is incorrectly decoded; in this case any errors within a transform block will propagate spatially to adjacent regions of pixels. The situation is similar for wavelet subbands.

Spatial or temporal propagation of errors can also be caused by predictive coding. Errors within any pixel or transform coefficient used as the basis for prediction at the decoder will lead to incorrect reconstructions and these will propagate around the prediction loop, leading to errors in subsequent pixels or transform coefficients. For example, errors in a reference frame used for motion estimation will be propagated temporally by the motion compensation process.

### 11.1.5 The challenges and a solution framework

Up until this point we have attempted to achieve the best rate–distortion performance from our compression system. We have removed as much redundancy as possible from the video signal, without compromising quality. However, we are now faced with the dilemma that, in order to make the compressed bitstream robust in the presence of errors, we need to introduce redundancy back into the bitstream in the form of channel coding, thus offsetting some or possibly all of the gains made during compression. Clearly this is not a good situation!

The challenge of error-resilient coding is therefore to maintain the coding gain obtained through source compression while ensuring that the decoded bitstream retains integrity during transmission, so that it can be faithfully decoded. In order to achieve this we need to:

1. Understand the causes of errors and their effects (Section 11.2).
2. Characterize these from a psychovisual point of view to understand the impact of loss artifacts on the human visual system (Chapter 2 and Section 11.3).
3. Exploit appropriate channel coding and retransmission solutions but use these, where possible, in a manner suited to video content (Sections 11.3 and 11.4).
4. Exploit the adaptive mechanisms that exist in many wireless or internet protocols to work jointly across the application, access control and physical layers to produce joint source–channel coding or cross layer solutions (Sections 11.4 and 11.6).
5. Exploit the structure of the video coding process to make the encoded bitstream inherently more robust to errors (Sections 11.5 and 11.7).
6. Understand where the errors occur in the reconstructed video and take steps to conceal rather than correct them (Section 11.8).

Figure 11.3 shows a generic architecture that captures the essence of this approach. In practice, an error-resilient solution combines aspects of transport layer solutions and

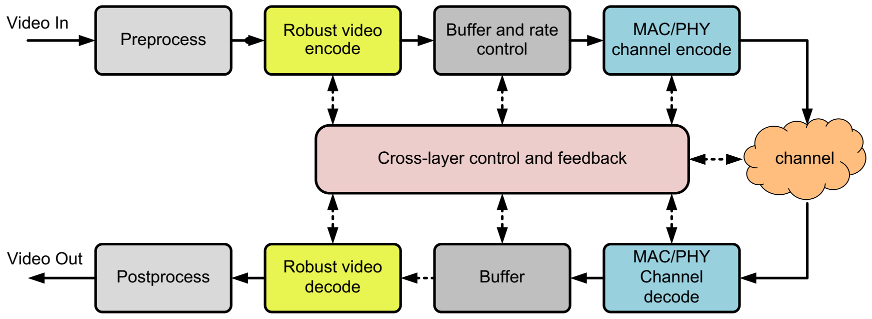


**FIGURE 11.3**

Generic error-resilient video transmission architecture.

application layer solutions in a sympathetic manner. That is, where possible, network parameters should be adapted to suit video requirements and video parameters should be managed to match the network constraints. These issues are discussed in more detail in the following sections.

## 11.2 The effects of loss

### 11.2.1 Synchronization failure

As we have already seen in Chapter 7, a coded video bitstream normally comprises a sequence of variable-length codewords. VLC methods such as Huffman or arithmetic coding are employed in all practical image and video coding standards. Fixed-length codewords (FLC) are rarely used in practice except when symbol probabilities do not justify the use of variable-length coding (e.g. in some header information). A further exception to this is the class of error-resilient methods including Pyramid Vector Quantization (PVQ) which we will study in Section 11.7.2.

As we will see in the following subsections, the extent of the problem caused will be related to the location where the error occurs and the type of encoding employed.

#### *The effect of a single bit error*

With FLC, any codeword affected by a single bit error would be decoded incorrectly, whereas all other codewords remain unaffected. This means that there is no loss of bitstream or symbol synchronization. In contrast with VLC, single bit errors can cause the decoder to decode a longer or shorter codeword, thus meaning that subsequent symbols, although delivered correctly, will most likely be decoded incorrectly due to the loss of synchronization at the decoder. This loss of synchronization can continue until the next explicit resynchronization point and will often result in the wrong number of symbols being decoded. This might mean that important implicit synchronization symbols such as EOB will be missed. In practice, most Huffman codes will resynchronize themselves, whereas arithmetic codes rarely do. The two scenarios in Example 11.1 illustrate the problems of synchronization loss in VLC, demonstrating the subtle differences between synchronization at bitstream, symbol and coding unit (e.g. transform block) levels.

**Example 11.1 (Loss of VLC synchronization due to a single bit error)**
Given the following set of symbols and their corresponding Huffman codewords,

| **Alphabet** | A | B | C | D | E |
|---|---|---|---|---|---|
| **Huffman code** | 0 | 10 | 110 | 1110 | 1111 |

with the following transmitted message:

| **Message** | B | A | D | E | C | B | A |
|---|---|---|---|---|---|---|---|
| **Encoded bitstream** | 10 | 0 | 1110 | 1111 | 110 | 10 | 0 |

**(a)** Compute and comment on the decoded sequence of symbols if there is a single bit error in the sixth bit.

**(b)** Repeat (a) for the case of a single bit error at the 10th bit position.

**Solution.**

**(a)** The table of decoded bits and symbols is given below with the errored bit in bold font:

| **Encoded message** | B | A | D | E | C | B | A | – |
|---|---|---|---|---|---|---|---|---|
| **Received bitstream** | 10 | 0 | 11**0**0 | 1111 | 110 | 10 | 0 | |
| **Parsed bitstream** | 10 | 0 | 110 | 0 | 1111 | 110 | 10 | 0 |
| **Decoded message** | B | A | C | A | E | C | B | A |

In this case the symbol D is incorrectly decoded as a C followed by an A and then bitstream resynchronization occurs. However, the fact that an extra symbol is decoded means that the subsequent symbols are displaced and hence symbol synchronization is lost.

**(b)** The table of decoded bits and symbols is given below with the errored bit in bold font:

| **Encoded message** | B | A | D | E | C | B | A |
|---|---|---|---|---|---|---|---|
| **Received bitstream** | 10 | 0 | 1110 | 11**0**1 | 110 | 10 | 0 |
| **Parsed bitstream** | 10 | 0 | 1110 | 110 | 1110 | 10 | 0 |
| **Decoded message** | B | A | D | C | D | B | A |

In this case, because of the lengths and encodings of the adjacent symbols, the error is constrained to only two symbols. After this, both bitstream and symbol resynchronization are achieved. Superficially this appears a lot better than the first case. However, consider the situation where the symbols represent {run/size} encodings of transform coefficients. The fact that there are errors in the fourth and fifth symbols means that, after the third symbol, the whole block will be decoded incorrectly because the runs of zeros corresponding to symbols 4 and 5 are incorrect. However, because symbol resynchronization is regained, any subsequent EOB symbol would be correctly detected and block level synchronization will be maintained. See Example 11.2 for more details.

### 11.2.2 Header loss

Errors in header information can cause catastrophic errors or error propagation. If any bit within a header is corrupted, then the result is often a complete failure of the decoder since the basic parameters needed to decode the bitstream are incorrect. There is therefore some justification for header information to be protected more strongly than other data as, without it, little else matters.

### 11.2.3 Spatial error propagation

Errors can propagate spatially during decoding for a number of reasons:

1. **DPCM prediction:** For example as used in the encoding of DC transform coefficients in JPEG. This will normally result in a luminance or chrominance shift in the image blocks decoded after the error.
2. **Intra-prediction:** Modern compression standards employ sophisticated intra-prediction modes that can result in the propagation of errors.
3. **Loss of VLC symbol synchronization within a block:** For example in a DCT-based system, this will result in the appearance of artifacts relating to superpositions of incorrect DCT basis functions.
4. **Loss of block synchronization:** Errors will propagate across blocks if EOB symbols are incorrectly decoded. This normally leads to a loss of positional synchronization.

The effect of a single error on a simple JPEG DCT/Huffman encoded image is illustrated in Figure 11.4. This shows the effects described in Examples 11.1 and 11.2. We can clearly see the block where the error has occurred as the corrupted DCT basis

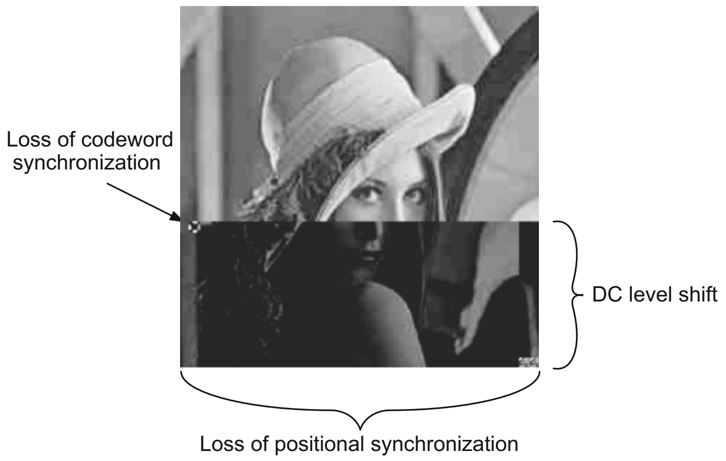

Loss of codeword synchronization
DC level shift
Loss of positional synchronization

**FIGURE 11.4**

The effect of a single bit error on a DCT/Huffman encoded Lena image.

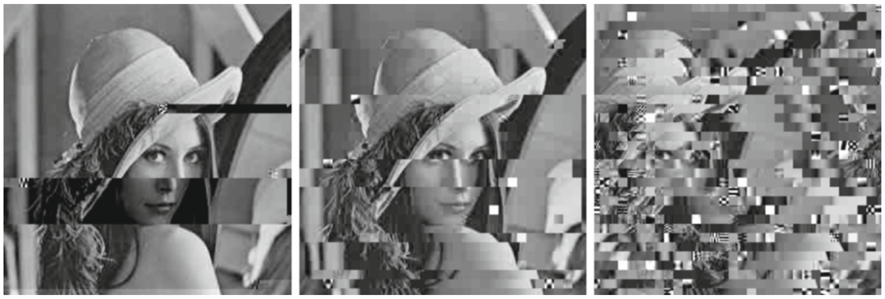


**FIGURE 11.5**

Impact of increased error rates on a DCT image codec. Left: 0.01% BER. Middle: 0.1% BER. Right: 1% BER.

functions are evident. The figure also illustrates the effects of corrupted DC levels propagated by the DPCM method used. Also shown is the effect of a spatial shift in the reconstructed image due to EOB symbols being missed in blocks adjacent to the errored block. Eventually symbol synchronization is regained but the effects of the spatial displacement remain. The effect of higher error rates and the impact these have on subjective image quality are shown in Figure 11.5.

**Example 11.2 (Spatial error propagation due to VLC errors)**

Given the set of symbols and their corresponding Huffman codewords from Example 11.1, let us now assume that the symbols correspond to {run/size} values for an intra-coded DCT block in an image. If in this case, symbol B corresponds to EOB, then comment on the spatial error propagation associated with the following transmitted sequence, when a single error is introduced at bit position 12 in the first block of the image:

| **Message** | A | D | E | C | B | D | A | E | C | B |
|---|---|---|---|---|---|---|---|---|---|---|
| **Encoded bitstream** | 0 | 1110 | 1111 | 110 | 10 | 1110 | 0 | 1111 | 110 | 10 |

**Solution.** The table of decoded bits and symbols is given below with the errored bit in bold font:

| **Message** | A | D | E | C | EOB | D | A | E | C | EOB |
|---|---|---|---|---|---|---|---|---|---|---|
| **Received bits** | 0 | 1110 | 1111 | 11**1** | 10 | 1110 | 0 | 1111 | 110 | 10 |
| **Parsed bits** | 0 | 1110 | 1111 | 1111 | 0 | 1110 | 0 | 1111 | 110 | 10 |
| **Decoded** | A | D | E | E | A | D | A | E | C | EOB |

In this case, symbols C and B are incorrectly decoded as E and A, hence the first EOB symbol is missed. So although bitstream and symbol resynchronization occurs, block synchronization is lost. All subsequently decoded blocks will thus be displaced by one block position to the left. So unless some explicit form of synchronization is introduced, the whole image after the second block will be corrupted.

## 11.2.4 Temporal error propagation

Temporal error propagation occurs when corrupted spatial regions are used as the basis for prediction of future frames. As shown in Figure 11.6, a single corrupted block in the reference frame can propagate to multiple (4 in this case) blocks in the predicted current frame. The spatial spread of the error is dependent on the local distribution of motion vectors and more active regions will tend to spread errors more widely than regions of low motion. Figure 11.7 shows two examples of temporal propagation in practice. Here, the error introduced locally due to spatial data loss still persists and has spread, several frames later.

It should be noted that corrupted motion vectors can also provide a source of errors due to incorrect motion compensation at the decoder.

**Example 11.3 (Temporal propagation of errors due to prediction)**
Consider the same symbol encoding and transmitted sequence as given in Example 11.2. Assume that the symbols represent {run/size} encodings related to the transform coefficients in two DFD blocks and that these are used as the basis for future motion compensated predictions. Comment on the temporal error propagation that would result from this single bit error in bit position 12.

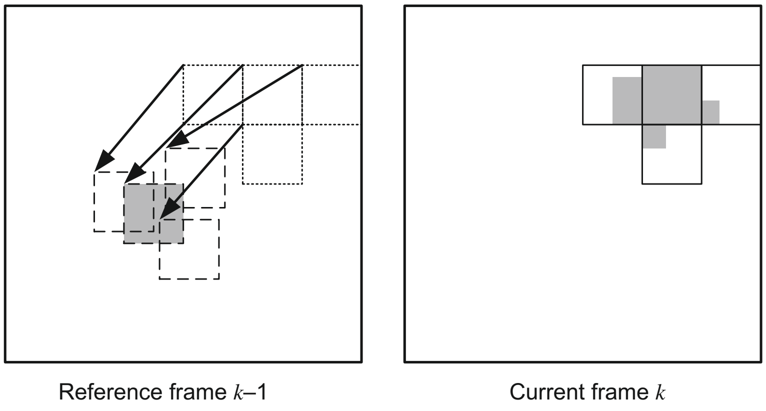


FIGURE 11.6
Temporal error propagation from a single corrupted block in frame $k - 1$ to multiple corrupted blocks in frame $k$.

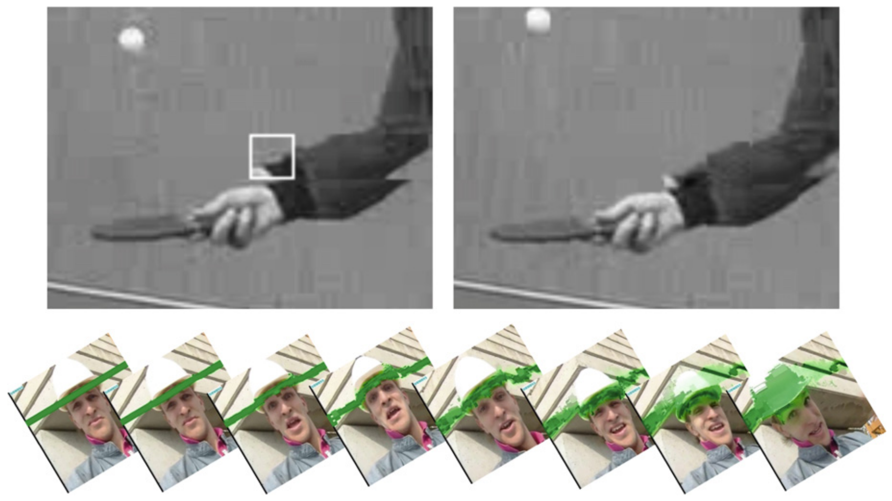


FIGURE 11.7

Example of temporal propagation; Top: across four frames of the table tennis sequence. Bottom: across 190 frames of Foreman.

**Solution.** As with Example 11.2, because the EOB symbol is missed, block synchronization is lost, and all blocks decoded after the errored block will be displaced.

In this case, however, because we are dealing with inter-coded DFD blocks, there is a second issue. The DFD blocks will be used as the basis of temporal prediction at the decoder and, since the decoded version will contain spatial errors, these will propagate temporally to subsequent frames because of motion prediction. Because all blocks after the errored block are displaced and because the errored block occurs early in the frame, if there is no explicit resynchronization point, almost all the data in future predicted frames are likely to be corrupted.

## 11.3 Mitigating the effect of bitstream errors

### 11.3.1 Video is not the same as data!

This might seem like a strange subsection heading but, to a certain extent, it is true. If we consider, for example, the case of a numerical database, then in order to guarantee its effectiveness we would expect it to be transmitted perfectly. If we were told it contained errors (especially if we did not know where they were), we probably would not trust it and avoid using it. However, in the case of an image or video sequence, the situation is, in general, quite different.

On one hand we have seen that a single bit error can be highly destructive, due to error propagation. On the other hand we know that the human visual system is tolerant to certain types of artifact and can accept loss if it is handled correctly. In fact

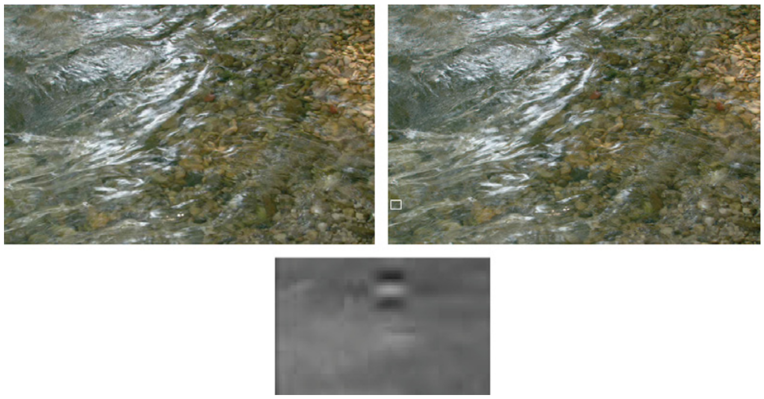


FIGURE 11.8

Subjective appearance of loss. PVQ encoded Riverbed sequence at 30 Mb/s. Left: encoded at 30 Mb/s (intra). Right: reconstructed after 25% PER. Bottom: magnified artifact from box shown in top right image.

the compression process itself is nothing but a managed process of introducing noise into an image and we are normally quite happy with the resulting approximations of the original content. Consider, for example, the video frames shown in Figure 11.8. On the left is the original compressed image (at 30 Mb/s coded using Pyramid Vector Quantization (PVQ)—see Section 11.7.2) while the one on the right has been reconstructed in the presence of 25% *packet error rate* (PER) (i.e. 1/4 of all packets contain uncorrectable errors). Artifacts do exist, as shown in the bottom sub-figure, but are in this case well masked and tolerable.

The process of managing bitstream errors is not straightforward since they are unpredictable. We can however model the impact of loss on the bitstream and take steps to avoid it, correct it, or conceal it and these approaches are all addressed in the following sections.

## 11.3.2 Error-resilient solutions

We have seen that the more redundancy we remove from the source, the more fragile our resulting bitstream becomes. We thus need to trade some rate–distortion performance in a clean channel for improved error resilience in the presence of errors.

Let us first consider the case of transmission of an encoded image over a noisy channel with varying degrees of loss. Table 11.1 compares three cases: (i) a DCT-only

**Table 11.1** FLC vs VLC encoding for various channel conditions. 256 × 256 Lena image at 2 bpp.

| PSNR (dB) | DCT only | DCT + DC pred. | DCT + DC pred. + VLC |
|---|---|---|---|
| **Error-free** | 33.7 | 33.9 | 38.5 |
| **0.1% BER** | 24.6 | 21.0 | 6.4 |

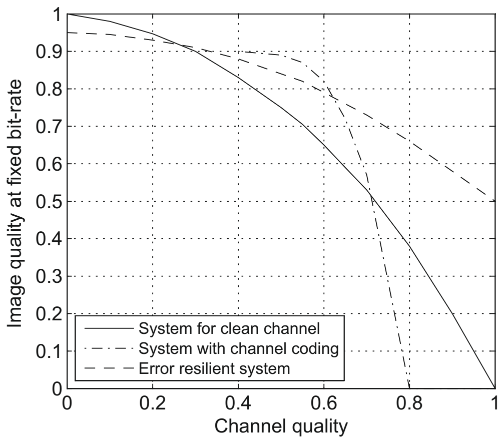


**FIGURE 11.9**

Idealized comparison of performance.

transform with fixed length encoding, (ii) a DCT transform-based system with DC prediction, and (iii) a DCT-based system with DC prediction and Huffman-based VLC. As can be seen, the VLC-based system provides excellent coding performance in a clean channel, yet it falls off badly as channel conditions deteriorate. On the other hand, the FLC scheme provides poor performance in a clean channel yet significantly outperforms the VLC case in lossy channels.

This trade-off between clean channel performance and error resilience is one of the challenges we face in designing our system. Ideally we want the best error resilience without compromising clean channel quality. In practice, however, it is necessary to strike a compromise and sacrifice some clean channel performance for improved error resilience in the presence of loss. These characteristics are illustrated in Figure 11.9 and methods for achieving this error resilience with graceful degradation are described in the following sections.

## 11.4 Transport layer solutions

In this section we use the term transport layer (a little loosely) to describe all aspects of the video transmission process that are not intimately associated with the application layer (i.e. the video codec). In many cases, the error control mechanisms will be implemented at the transport layer in the ISO 7 layer model. However, they may also be implemented in MAC and PHY layers. Many excellent references exist on this well-researched topic, for example Refs. [3–6].

### 11.4.1 Automatic Repeat request (ARQ)

ARQ is an interactive technique that is widely used in many communication systems. It relies on feedback from the decoder to determine whether a message (packet) has been correctly received. The detection of an error is normally performed through the addition of some form of *Cyclic Redundancy Check* (CRC) or through *Forward Error Correction* (FEC) coding. When an error is detected, the decoder returns a NACK (Negative ACKnowledge) signal which instructs the encoder to retransmit the data. There are a number of obvious advantages and problems with this method:

#### Advantages

- ARQ is simple and efficient. The redundancy introduced through retransmission is efficient as it is targeted specifically at lost data.
- It can be effective for coping with packet loss or large burst errors.

#### Problems

- A delay is introduced while the decoder waits for data to be retransmitted and, in real-time video applications, this may imply that frames have to be dropped (video frames have to be rendered at specific time intervals).
- ARQ requires a feedback channel which is not always available, for example when multicasting or broadcasting.

#### Delay-constrained retransmission

In practice, ARQ is a simple yet effective technique for ensuring reliable transmission of data but, in the context of real-time video, it has a usefulness lifetime, limited by the maximum tolerable latency between encoding and decoding. To address this problem, *delay-constrained retransmission* is normally used and this can be implemented at both receiver and transmitter. In the former case, the decoder will only request a retransmission of packet $n$ if the following condition is satisfied:

$$T_c + T_{RT} + T_x < T_d(n) \tag{11.1}$$

where $T_c$ is the current time, $T_{RT}$ is the round trip delay for the network, $T_d(n)$ is the playout time for packet $n$, and $T_x$ is a variable that accounts for other factors such as the tolerance on delay estimates and other uncertainties. Similarly, this decision could be made at the encoder if the following condition is satisfied:

$$T_c + T_{RT}/2 + T_x < T_d'(n) \tag{11.2}$$

where $T_d'(n)$ is in this case the estimated playout time for the packet.

### 11.4.2 FEC channel coding

In FEC methods, additional parity data is appended to the compressed signal, which allows the decoder to correct a certain number of errors. FEC can be combined with layered coding (or data partitioning) to provide unequal error protection, where different parts of the compressed bitstream are protected by different strength codes, according to importance.

The use of FEC increases the total data rate required for transmission and clearly offsets some of the benefits gained from source compression—redundancy can reach

100% for a 1/2 rate code. Furthermore, in general, FEC must be chosen with a particular worst case channel scenario in mind. For channels that have a highly variable quality, this worst case situation may imply the need for a very powerful FEC code which can severely reduce the compression performance. Furthermore, such a system will fail catastrophically whenever this worst case threshold is exceeded. One method that is frequently used to help mitigate the effects of burst errors is to use interleaving. However, this is sub-optimal and can introduce large delays.

### Erasure codes

FEC can be very effective in cases of packet erasures as usually the positions of the lost packets are known. Efficient erasure codes include the Reed Solomon Erasure (RSE) correcting code. With an RSE($n$, $k$) code, $k$ data packets are used to construct $r$ parity packets, where $r = n - k$, resulting in a total of $n$ packets to be transmitted. The $k$ source packets can be reconstructed from any $k$ packets out of the $n$ transmitted packets. This provides for error-free decoding for up to $r$ lost packets out of $n$. The main considerations in choosing values of $r$ and $k$ are:

- **Encoding/decoding delay:** In the event of a packet loss, the decoder has to wait until at least $k$ packets have been received before decoding can be performed. So, in order to minimize decoding delay, $k$ must not be too large.
- **Robustness to burst losses:** A higher value of $k$ means that, for the same amount of redundancy, the FEC will be able to correct a larger number of consecutive lost packets, or bursts. For example, an RSE(4, 2) code and an RSE(6, 4) code can both protect against a burst error length of 2, but the RSE(6, 4) has only 50% overhead whereas RSE(4, 2) has 100% overhead.

### Cross packet FEC

If FEC is applied within a packet or appended to individual packets, in cases where packets are not just erroneous but are lost completely during transmission (e.g. over a congested internet connection), the correction capability of the code is lost. Instead, it is beneficial to apply FEC across a number of packets as shown in Figure 11.10. One

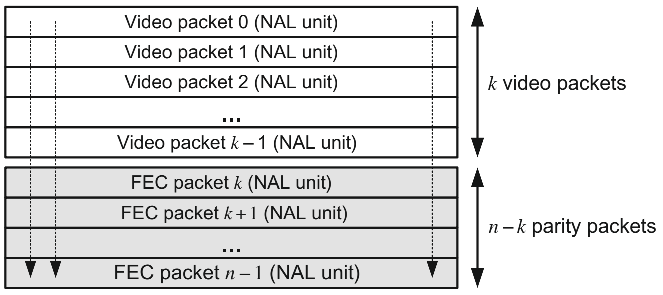

Video packet 0 (NAL unit)
Video packet 1 (NAL unit)
Video packet 2 (NAL unit)
...
Video packet $k-1$ (NAL unit)
FEC packet $k$ (NAL unit)
FEC packet $k+1$ (NAL unit)
...
FEC packet $n-1$ (NAL unit)
$k$ video packets
$n-k$ parity packets

**FIGURE 11.10**

Cross packet FEC.

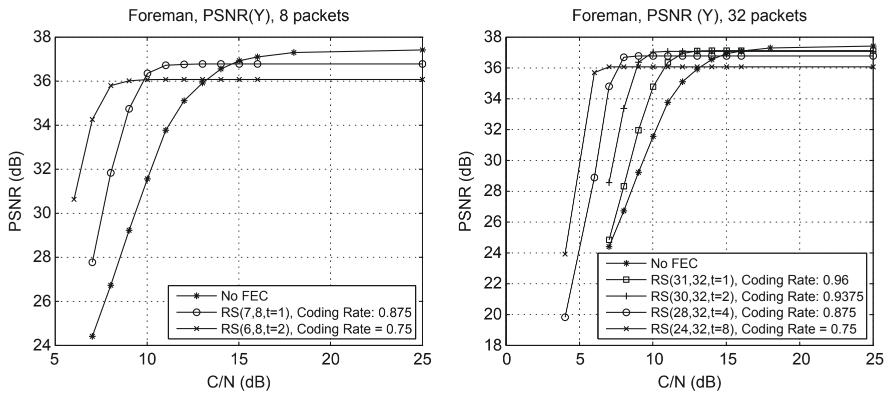


**FIGURE 11.11**

Performance of cross packet FEC for a coding depth of 8 (left) and 32 (right) for various coding rates.

problem with using FEC is that all the $k$ data packets need to be of the same length, which can be an issue if GOB fragmentation is not allowed. The performance of cross packet FEC for the case of different coding depths (8 and 32) is shown in Figure 11.11. This clearly demonstrates the compromise between clean channel performance and error resilience for different coding rates.

### *Unequal error protection and data partitioning*

We saw in Section 11.2 that not all types of video data are equally important. For example, header information is normally most important, as without it little else is possible. Motion information plays a particularly important role, especially because of the relatively small fraction of the total bit rate it occupies (approximately 10%). Faithful reconstruction of the encoded sequence depends heavily on the accuracy of the motion vectors. In order to reflect these priorities, the video data can be partitioned into separate data segments, each being allotted a protection level appropriate to its importance.

Prioritization of data may also be implemented in conjunction with layered coding as described in Section 11.9.

### *Rateless codes*

Raptor codes, proposed by Shokrollahi [7], have gained significant traction as an effective means of channel coding for multimedia information. Raptor codes are a type of fountain code (first introduced by Luby [8] as LT codes) that combine LDPC and LT codes and offer the property of linear, O($k$), time encoding and decoding. Fountain codes encode $k$ symbols into a (potentially) limitless sequence of encoding symbols and the probability that the message can be recovered increases with the number of symbols received. For example, in the case of Raptor codes, the chance of decoding success is 99% once $k$ symbols have been received.

Raptor codes form part of the 3rd Generation Partnership Project (3GPP) for use in mobile cellular multicast. They offer a flexible and adaptive mechanism for dealing with variable wireless channel conditions.

### 11.4.3 Hybrid ARQ (HARQ)

HARQ [9] combines FEC coding with ARQ. It suffers from some of the basic limitations of ARQ, but offers some efficiency gains. In HARQ, the data is encoded with an appropriate FEC code, but the parity bits are not automatically sent with the data. Only when an error is detected at the decoder are these additional parity bits transmitted. If the strength of the FEC code is sufficient to correct the error, then no further action is taken. If however this is not the case, then the system reverts to a full ARQ retransmission. Typically a system will alternate between data and FEC packets during retransmission. HARQ is a compromise between conventional ARQ and conventional FEC. It operates as well as ARQ in a clean channel and as good as FEC in a lossy channel.

In practice, with multiple retransmissions of the same data, a receiver would store all transmissions and use these multiple copies in combination. This is often referred to as *Soft Combining*. This approach has been employed on up- and downlinks for high speed data in the UMTS 3G standard, and in the WIMAX (broadband wireless access—IEEE 802.16) and in 4G LTE.

### 11.4.4 Packetization strategies

The packetization strategy [10] can have an influence on performance in the presence of loss. The basic principles can be summarized as:

- Longer packets lead to improved throughput for clean channels (lower overhead).
- Shorter packets facilitate better error resilience as their loss will have less impact on the video sequence and error concealment will be easier.
- To support error resilience, video packet fragmentation should be avoided. Every NAL Unit (see Section 11.5.1) should thus be transmitted in one and only one UDP frame.

To illustrate the third bullet point, if a video packet containing encoded VLC symbols is split across two UDP packets, then the loss of one of these would render the other useless because of VLC dependencies.

## 11.5 Application layer solutions

### 11.5.1 Network abstraction

Standards since H.264/AVC have adopted the principle of a Network Abstraction Layer (NAL) that provides a common syntax, applicable to a wide range of networks.

A high level parameter set is also used to decouple the information relevant to more than one slice from the media bitstream. H.264/AVC, for example, describes both a Sequence Parameter Set (SPS) and a Picture Parameter Set (PPS). The SPS applies to a whole sequence and the PPS applies to a whole frame. These describe parameters such as frame size, coding modes, and slice structuring. Further details on the structure of standards such as H.264 are provided in Chapter 12.

## 11.5.2 The influence of frame type

### *I-frames, P-frames, and B-frames*

The type of frame employed has a direct influence on the way errors propagate temporally. The main characteristics of I-frames, P-frames, and B-frames are listed below:

- **I-frames:** I-frames are not predicted with reference to any other frame. They therefore provide a barrier to temporal error propagation. They do however, in H.264/AVC and HEVC, exploit intra-prediction and hence can propagate errors spatially. Also I-frames form the basis for prediction P- and B- frames, so any spatial errors occurring in an I-frame can propagate temporally to subsequent frames.
- **P-frames:** P-frames are predicted from other P-frames or from I-frames. They will thus both propagate errors contained in their reference frames and introduce new spatial errors that can propagate spatially and temporally.
- **B-frames:** B-frames are normally not used as the basis for predicting other frames. They therefore do not propagate errors temporally and any spatial errors introduced in a B-frame are constrained to that frame.

### *Intra-refresh*

We have seen that the Group of Pictures (GOP) structure for a video sequence defines the relationship between I-, P-, and B-frames. All encoded sequences will have at least one I-frame and most will have them inserted periodically, for example every 12 or 15 frames. Many codecs support intra-coding of macroblocks in P-frames. This provides a more flexible approach to rate–distortion optimization and helps to limit temporal error propagation. Macroblocks that are intra-coded in this way can be selected according to rate–distortion criteria, or according to their concealment difficulty, or randomly.

Some codecs such as X.264 take this further and support *Periodic Intra-Refresh* instead of key-frames. This effectively spreads the I-frame across several frames using a column of intra-coded blocks that progressively steps across the video frame. In this case, motion vectors are restricted in the same way as in slice structuring, so that blocks on one side of the refresh column do not reference blocks on the other side. This approach has the additional benefit that the peak-to-mean bit rate ratio is constrained (unlike the situation where large I-frames are used in isolation), offering better streaming and lower latency.

### Reference picture selection

In simple motion-compensated hybrid codecs, the most recent previously coded frame is used as a reference for prediction of the current frame. More recent standards (from H.263+ onwards) support the use of multiple reference frames and reference picture selection (RPS) modes. These allow, with some constraints, the choice of any previously coded frame as a reference. This can be beneficial in limiting error propagation in error-prone environments.

In reference picture selection (RPS), both the encoder and the decoder store multiple reference frames. If, through feedback from the decoder, the encoder is instructed that the most recent reference frame has been received erroneously, then it will switch to another (error-free) reference picture for future predictions. The RPS selection mode was originally adopted in H.263+ as Annex N. This is shown in Figure 11.12 where an NACK(4) signal is fed back from the decoder and received during the encoding of frame 5. For frame 6, the encoder switches to use frame 3, which was received error-free, as the new reference frame.

### Periodic reference frames

In terms of coding efficiency, it is normally more efficient to predict a picture from a frame several frames displaced from it, than to use intra-frame coding. Techniques such as Periodic RPS [11] exploit this by encoding every $N$th frame in a sequence using the $N$th previous frame as reference. All the other frames are coded as usual. This scheme is illustrated in Figure 11.13. Its advantage is that if any errors in a PR frame can be corrected through the use of ARQ or FEC before it is referenced by the next PR frame, then this will effectively limit the maximum temporal error propagation to the number of frames between PR frames.

Since PR frames require far fewer bits to code than intra-frames at the same quality, extra bits can be used to protect the PR frames, thus providing better performance under packet losses than intra-frame coding. Because this technique does not

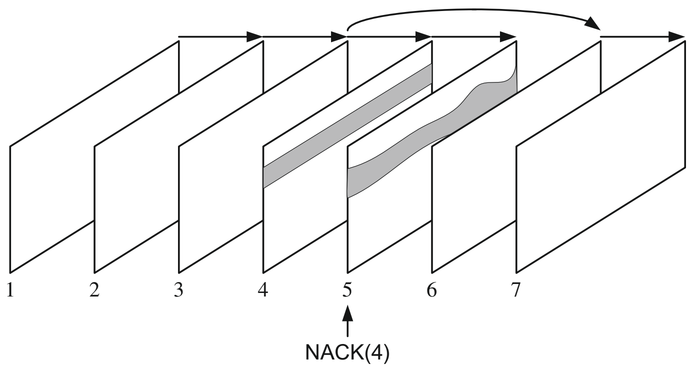


**FIGURE 11.12**

Error resilience based on reference picture selection.

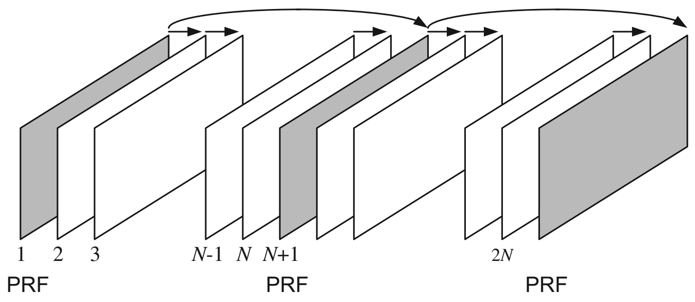


**FIGURE 11.13**

Periodic reference frames.

depend on feedback, it is applicable to broadcast and multicast scenarios. This type of approach is now common in coding standards, for example in H.264/AVC, referred to as *inter-prediction with subsequences*.

## 11.5.3 Synchronization codewords

Periodic resynchronization of a bitstream, and its associated symbols and data structures, can be achieved by the insertion of explicit synchronization codewords. For example, these might be inserted at the end of each row of blocks, slice, or frame. Resynchronization codewords are uniquely identifiable and are distinct from all other codes or concatenations of codes. They offer the following advantages and disadvantages:

### *Advantages*

If an error is encountered, the maximum loss is limited by the distance between synchronization markers. Once a new synchronization marker is encountered, decoding can proceed again correctly.

### *Disadvantages*

Resynchronization codewords incur a significant overhead, especially if used frequently. In the case of a JPEG encoded image, resynchronization markers at the end of each row typically incur a 1–2% overhead, whereas if they were located after each block, they would typically incur over 30% overhead.[1]

## 11.5.4 Reversible VLC

When an error is encountered in conventional VLC, there is a strong likelihood that the error will propagate spatially until the next resynchronization point. In the case of *Reversible VLC* (RVLC), backward decoding as well as forward decoding can be

[1]Overhead will varying according to compression ratio and content.

performed. Hence some of the data lost with conventional VLC can be faithfully decoded with RVLC. RVLC methods initially found favor in H.263+ and MPEG-4, and can be used effectively in combination with synchronization markers to provide enhanced error resilience. They were introduced by Takashima et al. [12], who showed how both symmetric and asymmetric codes can be generated.

RVLC codes [12,13] should exhibit both prefix and suffix properties so that the codewords are decodable in both directions. The types of code that can offer this so-called biprefix property are limited and few are optimal. Codes such as Golomb–Rice codes or Exp-Golomb codes can be mapped into an equally efficient biprefix code, but in most cases the resulting biprefix code is less efficient. Girod [14] introduced an alternative method for generating bidirectionally decodable VLC bitstreams, based on modifications to prefix codewords. This elegant approach results in a coding efficiency equivalent to that of Huffman coding.

It is known that VLC data comprising DCT coefficients or differential motion vectors exhibit a *Generalized Gaussian Distribution*. As discussed in Chapter 7, these can be efficiently entropy coded using Exp-Golomb codes. These can be implemented without look-up tables and reversible versions are easy to design. Exp-Golomb codes form an important part of the H.264 and HEVC standards.

A simple example of RVLC is provided in Example 11.4.

**Example 11.4 (Reversible VLC)**
Consider the symbol–codeword mappings:

$$a \leftrightarrow 0; \quad b \leftrightarrow 11; \quad c \leftrightarrow 101 = \text{EOB}$$

The sequence $\{a, b, c, b, a, a, b, c, b, a\}$ is transmitted followed by a SYNC codeword. If the 9th and 11th bits are corrupted, show how this encoding can minimize errors through bidirectional decoding.

**Solution.**

| **Message** | *a* | *b* | *c* | *b* | *a* | *a* | *b* | *c* | *b* | *a* | SYNC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Rx. bitstream** | 0 | 11 | 101 | 11 | **1** | 0 | **0**1 | 101 | 11 | 0 | SYNC |
| **Fwd. decode** | a | b | c | b | – | – | X | X | X | X | SYNC |
| **Rev. decode** | X | X | X | X | X | X | – | c | b | a | SYNC |

Note that the codewords are symmetrical, giving us a clue that they should be bidirectionally decodable. The table shows the forward and reverse decoding operations. Through bidirectional decoding we have successfully recovered most of the symbols, having lost only three symbols where the errors occur. If we assume that codeword *c* represents an EOB then we can see that the RVLC method recovers two of the three blocks of data transmitted.

## 11.5.5 Slice structuring

Most recent standards support the idea of decomposing a frame into slices prior to coding. A slice comprises a slice header followed by an integer number of macroblocks or coding units. Since all forms of prediction are constrained to within the slice boundary, slice structuring is effective in limiting error propagation. It therefore provides a simple basis for error-resilient coding. Some typical generic slice structures are shown in Figure 11.14.

### *Flexible macroblock ordering (FMO)*

*Flexible macroblock ordering* (FMO) is one slice structuring approach that is particularly attractive for error resilience and for supporting concealment. It forms part of the H.264/AVC standard. The video frame is divided into several independently decodable slice groups (Figure 11.14) and these can assume checkerboard, scattered and interleaved patterns as shown. For example, in the checkerboard pattern, each frame can be represented as two slice groups with macroblocks interleaved as shown in the figure. It is easy to see that if a slice in one group is lost then the remaining one can be employed to provide a basis for the interpolation of the missing slice. Providing that only one of the two slices is lost, then FMO distributes errors across a larger region of the frame.

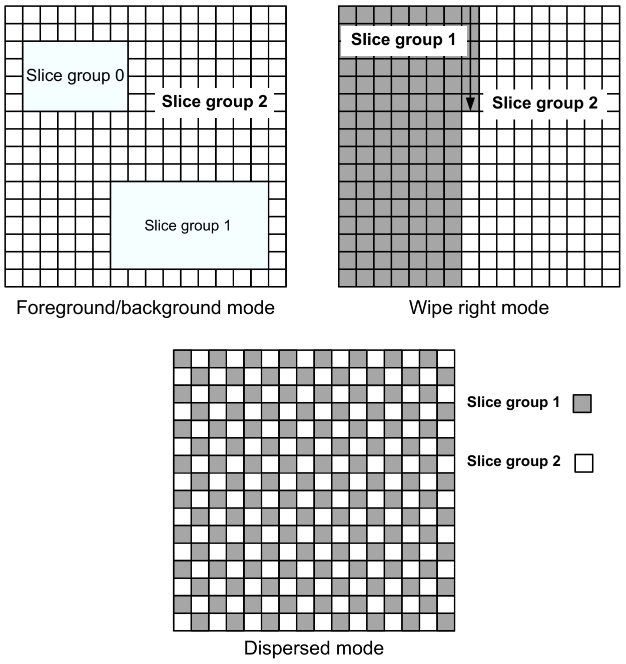


**FIGURE 11.14**

Example slice structures.

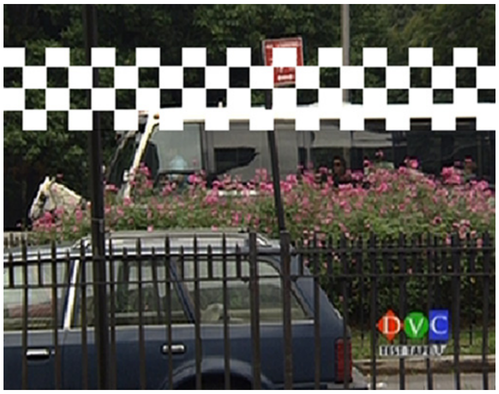


FIGURE 11.15

Use of dispersed mode slice groups as a basis for error concealment.

A disadvantage of slice structuring is that coding gain is reduced. This is especially true in FMO, since neighboring macroblocks cannot be used as a basis for prediction. It has been reported that the efficiency drop or bit rate increase is approximately 5% at QP = 16 and up to 20% for QP = 28 [15].

An example of using dispersed mode slice groups is shown in Figure 11.15. Here, each slice corresponds to two rows of macroblocks and the figure shows the case where slices 2 and 3 are lost from one of the slice groups. It can be clearly seen that data from the remaining slice group is distributed in such a way as to facilitate interpolation of the lost data. This is the basis of the error concealment approach presented in Section 11.8.4.

### *Redundant slices*

Redundant slices (RS) are a feature of standards such as H.264/AVC that supports the inclusion of redundant information for error resilience purposes. The redundant slice can be a simple duplicate of the original or it could be a lower fidelity copy. This copy can then be used in place of the original if it is lost. The question is—which slices should be included as redundancy? This has been addressed by a number of authors. Ferre et al. [16], for example, present a method whereby macroblocks are selected for inclusion on the basis of an end-to-end distortion model aimed at maximizing the reduction in distortion per redundant bit. This shows advantages over alternative methods such as cross packet FEC and Loss-adaptive Rate–Distortion Optimization.

## 11.5.6 Error tracking

This technique also relies on the existence of a feedback channel, not only to indicate that an error has occurred, but also to provide information on the location of the error (e.g. which slices or blocks have been corrupted). If an encoder knows which blocks have been received in error, it can predict how the errors will propagate to the currently encoded frame; it can then take one of two actions:

1. Encode the corresponding blocks in intra-mode.
2. Encode the corresponding blocks based on predictions only from blocks that have been correctly received.

This so-called *selective recovery* technique was proposed by Wada [17]. Steinbach et al. propose an efficient error tracking algorithm in Ref. [18].

## 11.6 Cross layer solutions

We have seen that compressed video (in the context of the human visual system) is tolerant to a certain level of errors, provided the associated artifacts are managed effectively. However, the aim of almost all networks is to deliver perfect data; anything else is deemed useless and is normally discarded by lower network layers without being sent to the application. Resource management and protection mechanisms that are traditionally implemented in the lower layers of the OSI stack (Physical (PHY), Medium Access Control (MAC), and transport) are, by intent, isolated from the application layer. We have seen however how the error characteristics of these lower layers are intimately coupled with their manifestations at the application layer.

There has thus recently been significant interest in performing optimization across the communication layers in order that the demands of the application can be reflected in the parameterization of the network layers and vice versa. This is referred to as *cross layer optimization*. Cross layer optimization differs from *joint source–channel coding* in that the aim of the latter is to optimize channel coding, dependent on the rate constraints of the channel. The former however takes a much more holistic approach, taking account of packetization strategies, retransmission strategies, delay constraints, and the loss tolerance of the data being encoded.

An excellent overview of the various aspects of cross layer optimization is given by van der Scharr et al. [4,19].

### 11.6.1 Link adaptation

One form of cross layer optimization that is realizable in practice is link adaptation, based on end-to-end distortion metrics and modeling. Wireless local area networks (WLANs) such as IEEE 802.11a/g/n support numerous modulation and coding schemes (MCS), each providing different throughputs and reliability levels. Conventional algorithms for adapting MCS attempt to maximize the error-free throughput, based on RF signal measures such as residual signal strength indication (RSSI) and bit or packet error rates (BER or PER). They do not take account of the content of the data stream and they rely heavily on the use of ARQ and FEC to correct erroneous transmissions. In contrast, authors such as Ferre et al. [20] have presented ways of achieving this by minimizing the video distortion of the received sequence.

Ferre et al. use simple, local rate–distortion measures and end-to-end distortion models at the encoder to estimate the received video distortion at the current transmission rate, as well as at the adjacent lower and higher link speeds (MCS modes). This allows the system to select the mode which offers the lowest distortion, adapting to the channel conditions to provide the best video quality. Simulation results, based on H.264/AVC over IEEE 802.11g, show that the proposed system closely follows the optimum theoretic solution. Example results are provided in Figures 11.16–11.19. The first three figures illustrate the significant difference in switching characteristics

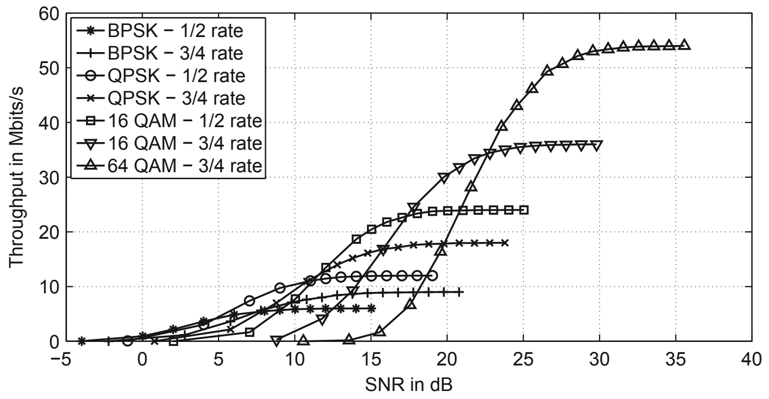


**FIGURE 11.16**

802.11g MCS switching characteristics based on throughput. (Reproduced with permission from Ref. [20]).

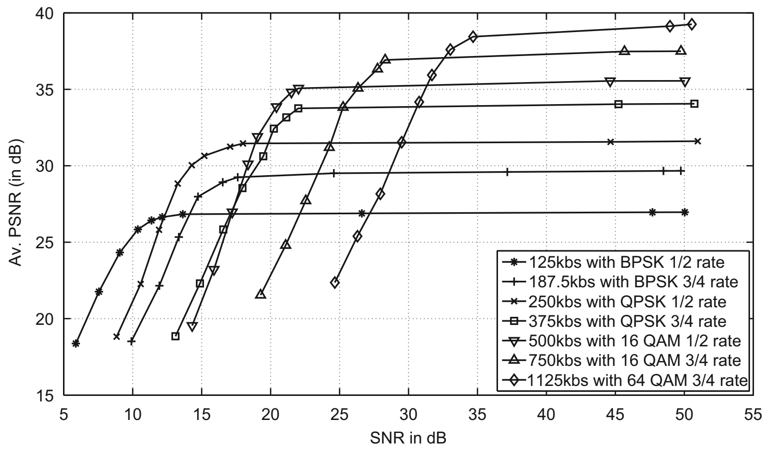


**FIGURE 11.17**

802.11g MCS switching characteristics based on video distortion. (Reproduced with permission from Ref. [20]).

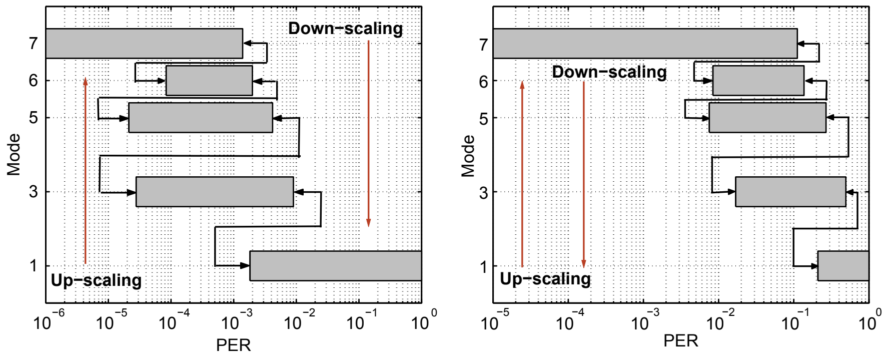


**FIGURE 11.18**

MCS switching characteristic comparison: video quality based (left) vs throughput based (right). (Reproduced with permission from Ref. [20]).

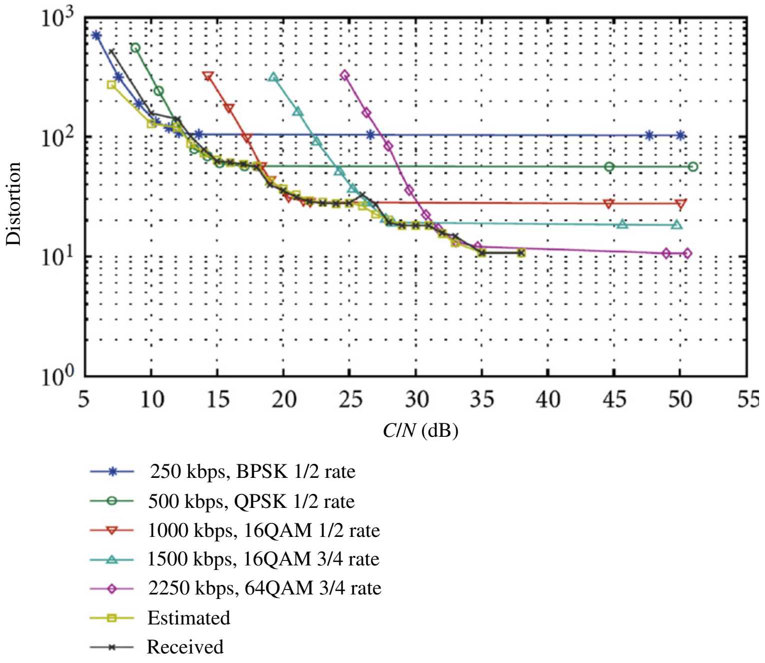


**FIGURE 11.19**

MCS link adaptation results. (Reproduced with permission from Ref. [20]).

between the cases of a throughput-based metric and a video quality-based metric. Figure 11.19 shows the result of the metric applied to the *Foreman* sequence plotted against channel carrier to noise ratio. It can be clearly seen that the switching characteristic closely tracks the optimum trajectory between MCS modes.

## 11.7 Inherently robust coding strategies

We have seen above that one of the contributing causes of error propagation in compressed video is variable length entropy coding. In this section we consider two methods that offer the benefits of fixed length coding (FLC) while preserving the compression performance offered by VLC. The first, *Error-Resilient Entropy Coding* (EREC), is a pseudo fixed length scheme where inherent synchronization is introduced through bitstream restructuring. The second, *Pyramid Vector Quantization* (PVQ), is a true fixed length scheme which exploits subband statistics to create a model that supports FLC.

### 11.7.1 Error-Resilient Entropy Coding (EREC)

#### *Principle of operation*

The Error-Resilient Entropy Code (EREC) [21,22] is a simple yet elegant approach to the coding of variable length blocks of data. It was introduced by Redmill and Kingsbury [21]. Based on a form of bin-packing, using a fixed length slotted structure, it enables the introduction of implicit synchronization points without the overhead of explicit synchronization codewords. The algorithm rearranges the variable length encoded blocks of data (for example, those resulting from block-based transformation and entropy coding) into a structure with dimensions that are predefined to ensure that it has sufficient capacity.

Assume that there are $N$ variable length blocks, each of length $b_i : i = 1 \cdots N$ and we wish to pack these into $M$ slots of length $s_j : j = 1 \cdots M$. The slotted EREC data structure should therefore satisfy the following conditions:

$$\begin{cases} N \geq M \\ T = \sum_{j=1}^{M} s_i \geq \sum_{i=1}^{N} b_i \end{cases} \tag{11.3}$$

where $T$ is the total data size of the slotted structure. The first condition ensures that there are enough slots while the second ensures that the EREC structure is large enough to code all the variable length blocks of data. In most circumstances, for image or video coding, it is appropriate to set $N = M$ and we shall assume this from now on. We can thus compute the slot length for the structure as given in equation (11.4):

$$s_i = \left\lceil \frac{1}{N} \sum_{i=1}^{N} b_i \right\rceil \tag{11.4}$$

An $N$ stage algorithm is then used to pack the blocks into the slotted structure. To enable a systematic encoding process that can be reversed at the decoder, an offset sequence is used. This defines the offset used at each stage in the algorithm. At stage $n$, the algorithm will search slot $k = i + \phi_n(\text{mod}(N))$, where $\phi$ is a pseudo-random offset sequence. In our examples we will simply use the sequence: $\phi = \{0, 1, 2, 3, 4, \ldots, N\}$. At each stage in the process, the EREC algorithm searches the slot at the appropriate offset to see if there is space to pack some or all of the excess

**Algorithm 11.1** EREC encoding.

```
1. Define offset sequence $\phi_n$;
2. INPUT $N$ blocks of data;
3. Compute slot length, $s_i$, using equation (11.4);
4. Assign block $b_i$ to slot $s_i$;
5. FOR $n = 1 \cdots N$
   6. Compute residual data for each slot: $r_i = b_i - s_i$;
   7. FOR $i = 1 \cdots N$
      8. Compute offset: $k = i + \phi_n (\mathrm{mod}(N))$;
      9. IF $r_i > 0$ AND $r_k < 0$ THEN pack slot $k$ with residual from block $i$;
         IF $|r_k| \geq |r_i|$ THEN $r_k \leftarrow r_k + r_i$; $r_i \leftarrow 0$;
         IF $|r_k| < |r_i|$ THEN $r_i \leftarrow r_k + r_i$; $r_k \leftarrow 0$;
   10. END FOR;
   11. IF $\{r_i = 0 : i = 1 \cdots N\}$ THEN END;
12.END FOR.
```

data from overlong blocks—i.e. where $b_i > s_i$. This process is repeated until all excess data from overlong blocks is packed into the structure. The EREC encoding procedure is defined in Algorithm 11.1.

The decoding process is simply the inverse operation, progressively reconstructing, over $N$ decoding stages, each original block until the block is fully decoded (e.g. an end-of-block code is detected). Thus, in the absence of channel errors, the decoder will correctly decode all the data.

A simple example of EREC encoding is shown in Figure 11.20 for six blocks of data. Referring to Figure 11.20: at stage 0, blocks 3, 4, and 6 are completely coded in slots 3, 4, and 6, with space left over. Blocks 1, 2, and 5 however are only partially coded and have data left to be placed in the space left in slots 3, 4, and 6. At stage 1, for an offset of $\phi = 1$, the remaining data from block 2 are coded in slot 3, and some data from block 5 are coded in slot 6. This process continues until, by the end of the sixth stage, all the data are coded. Further worked examples, illustrating both the encoding and decoding processes for a specific set of blocks, are given in Examples 11.5 and 11.6.

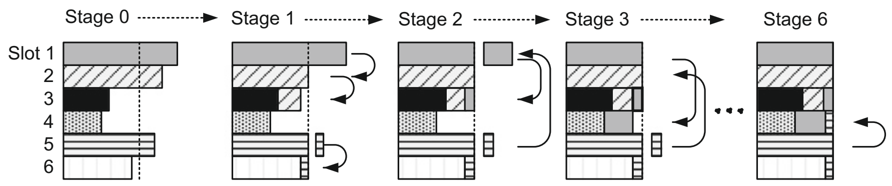


**FIGURE 11.20**

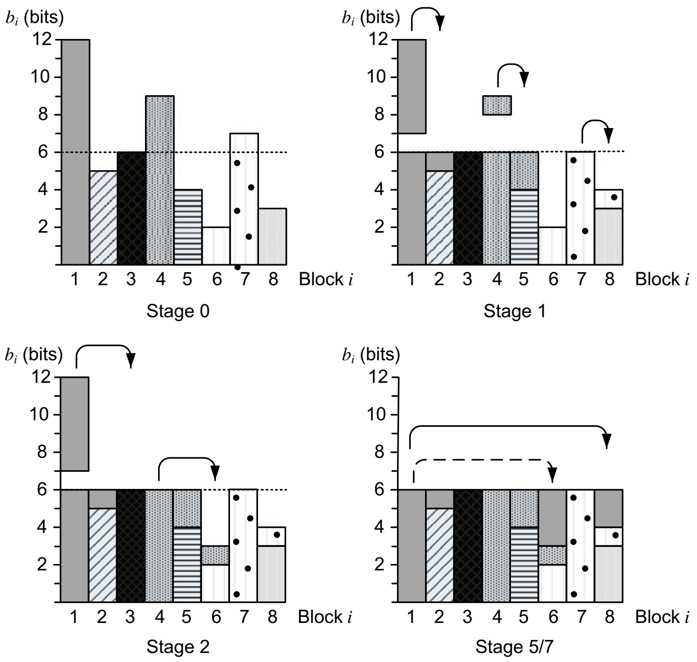
EREC operation.

**Example 11.5 (EREC encoding)**
A DCT/VLC-based encoder has generated a slice of data comprising eight blocks with the following lengths:

| *Block i* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Length $b_i$ | 12 | 5 | 6 | 9 | 4 | 2 | 7 | 3 |

Assuming an offset sequence {0, 1, 2, 3, 4, 5, 6, 7}, show how the EREC algorithm would encode this slice of data.

**Solution.** The total length of the slice is 48 bits, hence the slot length is 6 bits. The encoding process proceeds as follows:



Stages 1 and 2 of the encoding process are shown above. Then, for stages 3 and 4, there is no spare space to fill in slots 4 and 5, hence no action is taken. At stage 5, 3 bits from block 1 are packed into slot 6 and, at stage 7, the remaining 2 bits from this block are packed into slot 8. These two operations are shown combined for convenience above.

**Example 11.6 (EREC decoding)**
Perform EREC decoding on the EREC-encoded blocks of image data from Example 11.5 above. Assume that each block is terminated with an EOB symbol.

**Solution.** For convenience, here we use a more compact, tabular structure to illustrate each stage of the decoding process, where **X** indicates an incomplete block and **C** indicates a completely decoded block.

| Stage | Offset | bk1 | bk2 | bk3 | bk4 | bk5 | bk6 | bk7 | bk8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | **X** | **C** | **C** | **X** | **C** | **C** | **X** | **C** |
| 1 | 1 | **X** | | | **X** | | | **C** | |
| 2 | 2 | **X** | | | **C** | | | | |
| 3 | 3 | **X** | | | | | | | |
| 4 | 4 | **X** | | | | | | | |
| 5 | 5 | **X** | | | | | | | |
| 6 | 6 | **C** | | | | | | | |

At stage 0, we start decoding from the base of the first block and proceed until the end of the slot is reached or an EOB symbol is decoded. In the case of block 1, we reach the end of the slot with no EOB detected. We do the same for all other slots. Hence blocks 2, 3, 5, 6, and 8 are all fully decoded at stage 0. At stage 1 we search (for all incomplete blocks) for additional data at an offset of 1. For example, in the case of block 1 we search slot 2: we detect additional data beyond the EOB for block 2, but still no EOB is detected for block 1 so it remains incomplete. This proceeds until stage 6 in this case, when all blocks are fully decoded.

### *EREC performance*

In a clean channel, the EREC decoder will correctly decode all the data with negligible overhead (typically only the value of $T$ needs to be sent as side information). When channel errors occur, their effect depends on the particular distribution of codewords in the block and how rapidly symbol resynchronization is achieved. If symbol resynchronization is not regained before the end of the block, then this can cause the EOB symbols to be missed or falsely detected. This means that the decoder will incorrectly decode following information which was packed at later stages of the algorithm. This, in turn, implies that data placed in later stages (that toward the ends of long blocks) are more susceptible to error propagation than data placed in earlier stages. For typical compression methods, these propagation effects occur in high frequency data from active regions of the image. The important property of EREC is that, because the decoding of each block commences from a known location, block synchronization is preserved.

Examples of reconstructed images after EREC encoding for 0.1% BER and 1% BER are shown in Figure 11.21. Comparing these with their equivalents in Figure 11.5, the robust properties of EREC encoding can clearly be seen. Further improvements in subjective quality can be obtained if concealment is applied (see Section 11.8). An example of the relative performance of EREC vs a conventional sequential encoding is shown in Figure 11.22 showing up to 10 dB improvement in noisy channels.

Redmill and Bull [22] have shown that EREC can perform extremely well with arithmetic coding systems as well as with conventional VLC methods. Interestingly,

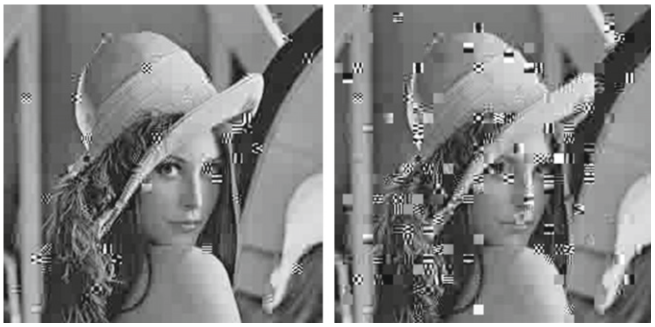


**FIGURE 11.21**

EREC performance examples of reconstructed DCT/Huffman encoded images after erroneous transmission. Left: 0.1% BER. Right: 1% BER. (Courtesy of D. Redmill.)

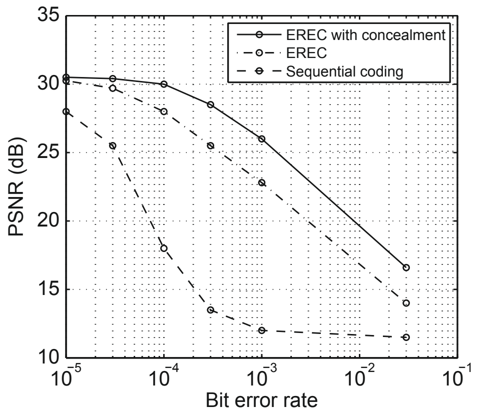


**FIGURE 11.22**

Graph of EREC performance for increasing BER compared to conventional DCT/VLC encoding. (Courtesy of D. Redmill.)

the paper demonstrates that, without EREC, Huffman coding is significantly more robust to errors than arithmetic coding. However, with EREC, significant improvement gains are manifest for both approaches and their performances are similar, with up to 15 dB improvement in PSNR.

EREC has been considered for adoption in standards such as MPEG-4, but has never been included in the final draft. Possible reasons for this are that it requires a number of blocks to be processed before they can be packed, although these delays can be managed in a slice structured environment. More likely, it is because the method does not fit well with conventional processing pipelines. Also, in the case of video, the use of predictive coding to remove inter-frame redundancy implies that an alternative approach to coding motion vectors must be employed. Swann and Kingsbury demonstrated the benefits of EREC for video coding in Ref. [23] showing significant benefits over conventional approaches in the context of MPEG-2.

### 11.7.2 Pyramid Vector Quantization (PVQ)

#### *Principle of operation*

Techniques such as EREC provide enhanced resilience through restructuring a bitstream in such a way that pseudo fixed length coding is achieved. In contrast, Pyramid Vector Quantization (PVQ) prevents error propagation through the use of actual fixed length codewords. PVQ, introduced by Fischer [24], exploits the Laplacian-like probability distributions of transform coefficients to provide an efficient means of compressing them.

PVQ is a form of vector quantization optimized for i.i.d. Laplacian random variables. A vector $\mathbf{x}$ is formed by grouping $L$ such random variables (e.g. transform coefficients). Based on the law of large numbers and the asymptotic equipartition property, it can be shown that almost all vectors will lie on a contour or region of constant probability. This region is a hyperpyramid in $L$-dimensional space with $2L$ vertices and $2L$ faces. Thus, for sufficiently large vector dimensions, there is negligible distortion if $\mathbf{x}$ is approximated by its nearest point on the hyperpyramid. This is the basis of the PVQ encoding algorithm.

The PVQ encoding algorithm consists of two steps. The first finds the closest vector $\hat{\mathbf{x}}$ to the input vector $\mathbf{x}$. The second step generates a bit code for $\hat{\mathbf{x}}$. Since all points on the hyperpyramid are equally probable, FLC is the optimum method for assigning bit codes. The PVQ-encoded bitstream thus comprises a sequence of codebook vector indices.

Chung-How and Bull [25] showed that PVQ is much more resilient to bit errors than either JPEG or H.263 and that wavelet-based PVQ outperformed DCT-based PVQ in terms of compression performance. In Ref. [26], Bokhari et al. showed that intra-mode PVQ is an effective codec for indoor wireless transmission of HD video. Intra-coding was employed since inter-frame encoding can lead to temporal error propagation. Although there is a loss in compression performance due to using an intra-only mode, this was justified by the increased error resilience.

#### *Performance*

Bokhari et al. [26] introduced an efficient rate–distortion (RD) algorithm for PVQ which yields impressive compression performance, comparable to intra-H.264/AVC and Motion JPEG 2000. They also evaluated the error resilience of the codec in the context of HD video using a realistic lossy wireless channel. Up to 11 dB PSNR improvement over H.264/AVC was demonstrated for lossy conditions. The basic compression performance of their approach is shown in Figure 11.23 for the Tennis sequence. It can be seen that intra-PVQ provides compression performance and video quality (43 dB PSNR @25 Mbps) comparable to other codecs in intra-mode. Importantly, it also obviates the need for sophisticated tools such as intra-prediction, variable size transforms, deblocking filters, or entropy coding (CABAC).

The key aspect of PVQ however is its error resilience. This was characterized and compared to other standardized codecs in Ref. [26] and the results of this are also shown in the right-hand graph of Figure 11.23. PVQ can be seen to offer up to 13.3 dB

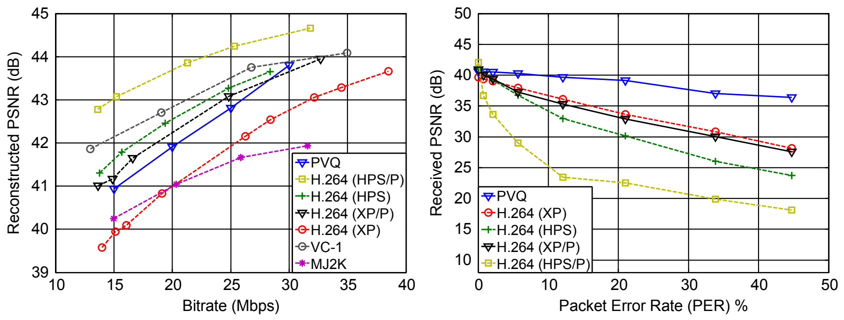


**FIGURE 11.23**
RD optimized PVQ performance for Tennis. Left: RD performance in a clean channel. Right: performance in a lossy 802.11n channel with correlated errors (Reproduced with permission Ref. [26]).

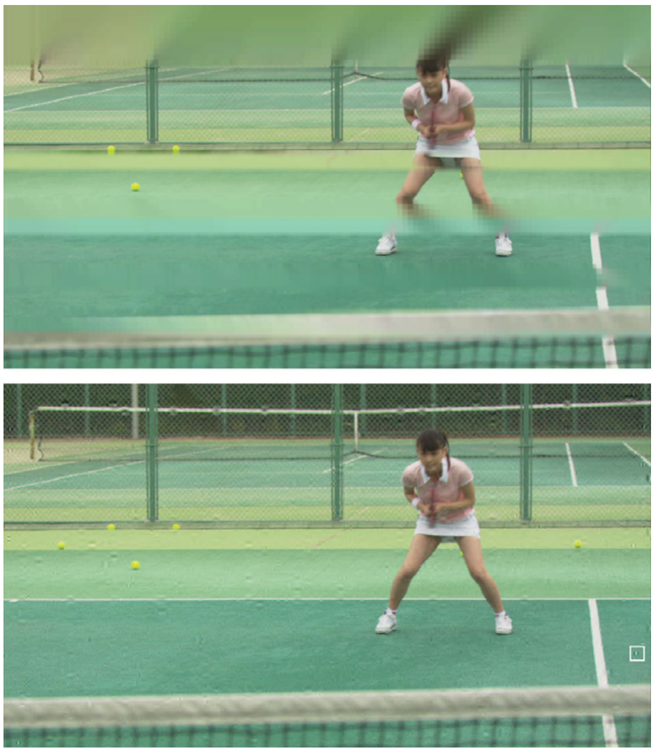


**FIGURE 11.24**
Frame shots for Tennis at 25 Mb/s and 25% BER. Top: H.264/AVC (HPS). Bottom: RD optimized PVQ (Reproduced with permission from Ref. [26]).

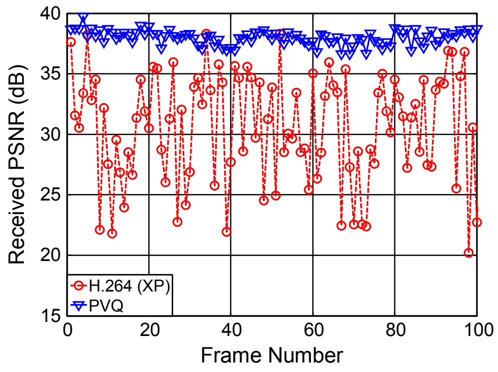


**FIGURE 11.25**

Variability in frame distortions for PVQ and H.264/AVC (Reproduced with permission from Ref. [26]).

PSNR performance gain over the best performing H.264/AVC counterpart in terms of error resilience. Example PVQ encoded frames are compared with H.264/AVC (high profile) in Figure 11.24.

A major attribute of PVQ is the low variance in distortion across corrupted frames compared to H.264/AVC. This is illustrated in Figure 11.25. One of the factors that contributes to PVQ's performance is its ability to cope with corrupted data. In most wireless scenarios, the whole packet is not lost, but instead it has insufficient integrity to benefit from FEC. In such cases, the packet is usually discarded before being sent to the application layer. In PVQ and other robust formats, the codec benefits from the use of corrupted data. For VLC-based schemes, the limiting factor is the PER, whereas for FLC-based schemes it is the actual number of bit errors (in the received packet). This is defined as the residual bit error rate (RBER). This is demonstrated in Figure 11.26, which shows that, although the PER might be high, the corresponding

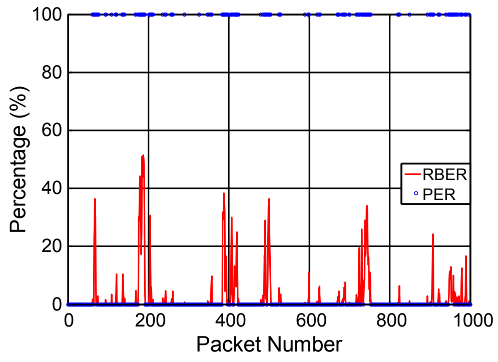


**FIGURE 11.26**

Packet error rate vs residual bit error rate for an 802.11n channel (Reproduced with permission from Ref. [26]).

RBER can be much lower. The figure shows sample packet error traces and their corresponding RBER traces. In bursty or correlated channels, the instantaneous RBER varies considerably. This leads to a temporally localized psychovisual impact. In an uncorrelated channel, the instantaneous RBER varies much less, giving a more consistent loss in quality throughout the sequence.

## 11.8 Error concealment

Error concealment is a decoder-oriented post-processing technique which attempts to conceal the effects of errors by providing a subjectively acceptable approximation of the original data. This exploits the spatial and temporal correlations present in a video sequence to estimate the lost information from previously received data. Lost data is reconstructed using spatial or temporal interpolation in a way that reduces the perception of spatio-temporal error propagation. Error concealment methods can benefit from complimentary error-resilient encoding so that errors are, as far as possible, localized and have local good data upon which estimates can be based.

Concealment methods can offer solutions where other techniques, particularly interactive methods which require a feedback channel, are not possible—for example, in broadcast or multicast systems. Concealment is a post-processing operation and is not mandated in any coding standard. Many algorithms exist and some of these are described below, classified as spatial, temporal, or hybrid methods. The reader is also referred to many excellent papers and reviews on the topic, including Refs. [3,5,6,27]. Reference [28] also contains details of the performance of specific methods based on motion field interpolation and multihypothesis motion estimation. Alternative methods that show some potential are those based on texture synthesis or inpainting. These are not covered further here, but the reader is referred for example to Ref. [30].

### 11.8.1 Detecting missing information

Firstly, before we can apply concealment methods, we need to understand where exactly the errors are located. This can be done in a number of ways and will depend to a degree on the implementation details. Some examples are:

- **Using header information:** For example, the sequence number in a packet header can be used to detect packet loss.
- **Using Forward Error Correction (FEC):** FEC can be used at the application layer as well as at the link layer.
- **Detection of syntax and semantic violations:** For example, illegal decoded codewords, invalid numbers of decoded units, out of range decoded parameters etc.
- **Detection of violations to the general characteristics of video signals:** For example, blocks with highly saturated colors (pink or green), blocks where most decoded pixels need clipping, strong discontinuities at borders of blocks etc.

None of these methods guarantee finding all errors and some can be subject to false detections. So, in practice, combinations are often employed. Let us now consider methods for concealing the errors once they have been detected.

## 11.8.2 Spatial error concealment (SEC)

Spatial error concealment methods exploit the spatial correlations that exist in most video signals. They operate under the assumption of spatial continuity and smoothness. The most common method for spatial interpolation is based on the weighted averaging technique proposed in Ref. [31]. Concealment is ideally based on data that has been correctly received; however, in areas of more extensive loss, a progressive approach can be used where previously concealed blocks or pixels are used as a basis for estimation.

Spatial interpolation, based on weighted averaging, is illustrated in Figure 11.27. After the detection of a lost block (indicated by the thicker lined boundary), each pixel, $s(x, y)$, is interpolated using the following expression:

$$s(x, y) = \frac{d_L s_R(x, y) + d_R s_L(x, y) + d_T s_B(x, y) + d_B s_T(x, y)}{d_L + d_R + d_T + d_B} \tag{11.5}$$

where $s_R(x, y)$ etc. are the border pixels from the spatially adjacent macroblocks located to the right, left, top, and bottom of the missing block. An example of spatial error concealment applied to the *Foreman* sequence is shown in Figure 11.28. The result in this case has an MSE of 46.6 with respect to the original error-free frame.

There are a number of extensions to the basic SEC method that can provide additional benefit, but usually at the cost of additional complexity. For example, *edge continuity* can be enforced. This means that if an edge is detected in a neighboring block or blocks, then this edge should be preserved during concealment. Agrafiotis et al. [32] use an edge preserving method that not only preserves existing edges but also avoids introducing new strong ones. The approach switches between a directional interpolation and a conventional bilinear interpolation, based on the directional entropy of neighboring edges. The proposed strategy exploits the strengths of both methods without compromising the performance of either and offers performance

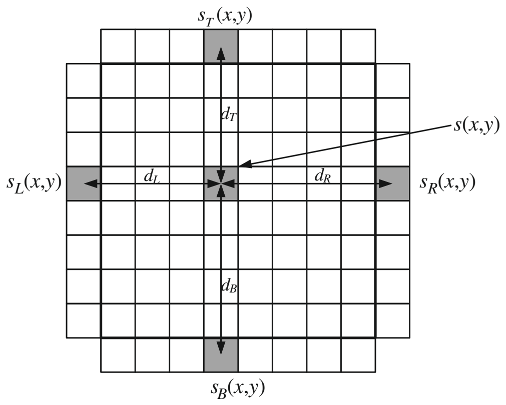


**FIGURE 11.27**

Spatial error concealment using nearest neighboring pixels.

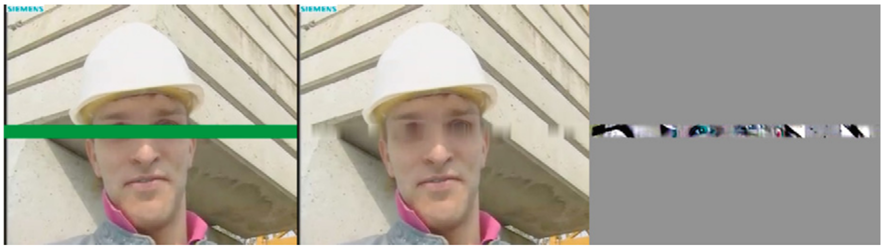


**FIGURE 11.28**

Spatial error concealment applied to a lost row of macroblocks in the Foreman sequence. Left: original with error shown. Middle: result of spatial error concealment. Right: amplified difference signal. MSE = 46.6. (Courtesy of D. Agrafiotis.)

improvements of over 1 dB for some cases. A similar approach was also presented by Ma et al. [33].

**Example 11.7 (Spatial error concealment)**
Consider the missing block of data shown below with the correctly received boundary pixels indicated. By performing SEC on this block, calculate the value of the missing pixel shown encircled in the diagram.

$$\begin{array}{c|cccc|c}
 & 1 & 1 & 2 & 2 & \\
\hline
0 & \mathrm{X} & \mathrm{X} & \mathrm{X} & \mathrm{X} & 2 \\
1 & \mathrm{X} & \mathrm{X} & \mathrm{X} & \mathrm{X} & 4 \\
3 & \mathrm{X} & \textcircled{X} & \mathrm{X} & \mathrm{X} & 7 \\
2 & \mathrm{X} & \mathrm{X} & \mathrm{X} & \mathrm{X} & 11 \\
\hline
 & 2 & 5 & 9 & 8 &
\end{array}$$

**Solution.** Using equation (11.5):

$$s(x, y) = \frac{2 \times 7 + 3 \times 3 + 3 \times 5 + 2 \times 1}{10} = 4$$

All other missing pixel values in the block can be interpolated in a similar manner.

### 11.8.3 Temporal error concealment

Temporal error concealment methods operate under the assumption of temporal continuity and smoothness of the motion field [28,29]. They exploit the highly correlated nature of most natural sequences and conceal corrupted or lost pixels based on predictions from surrounding regions. They work well on inter-coded blocks, but are often less successful on intra-coded blocks. Two basic techniques are considered below.

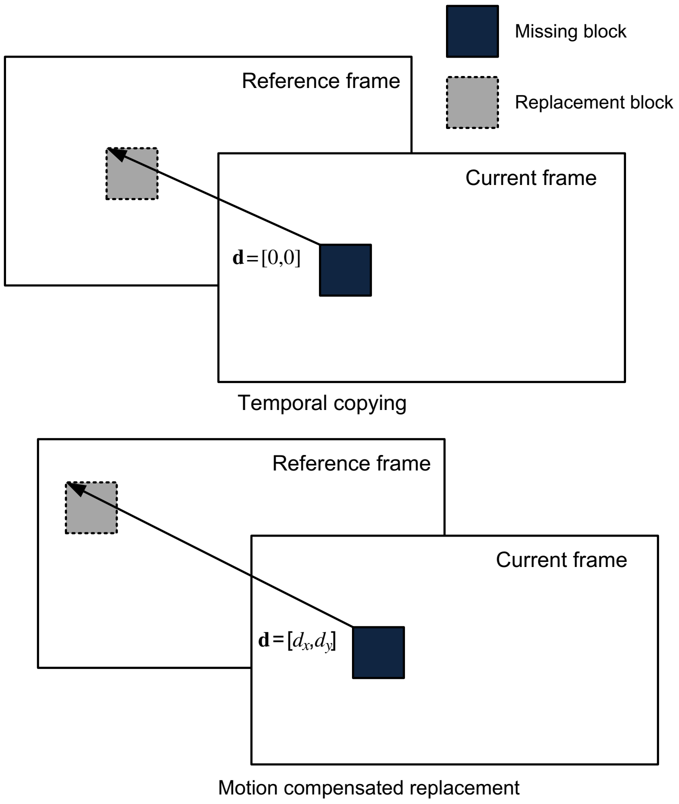


**FIGURE 11.29**

Temporal error concealment based on temporal copying (top) and motion compensated prediction (bottom).

### Temporal copying (TEC_TC)

The simplest form of temporal error concealment is *temporal copying*. In this case, as shown in Figure 11.29, the underlying model is one of zero motion. This method is extremely easy to implement, but the underlying motion model is rarely sufficient in practice. Results for a single lost packet containing one slice of the Foreman sequence are shown in Figure 11.30. Although this result looks reasonable, the error is actually quite large due to the person's head moving. This displacement is particularly evident if you inspect the area on the left-hand edge of his face.

### Motion compensated temporal replacement (TEC_MCTR)

As illustrated in Figure 11.29 (bottom), this method replaces the missing block with the best candidate region from the reference frame or frames. This is significantly more complex than the previous approach as it requires producing a list of candidate motion vectors that can be evaluated in order to choose the best replacement block. The method employs a two-stage process:

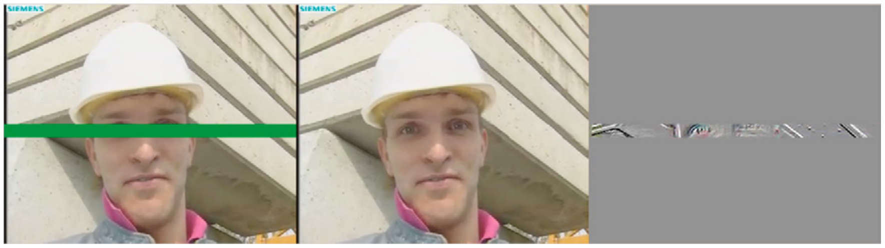


FIGURE 11.30
Temporal error concealment results for Foreman based on temporal copying. MSE = 19.86. (Courtesy of D. Agrafiotis).

1. Estimation of concealment displacement.
2. Displacement evaluation and replacement selection.

The first stage raises the question: How do we select the candidate motion vectors for TEC_MCTR? Assuming that the motion vector for the missing block is also lost, there are several likely candidates for replacement. These include:

- Motion vectors from spatially neighboring blocks.
- A rank-order statistic or average of the motion vectors from neighboring blocks.
- Motion vectors from temporally neighboring blocks.

We then need to evaluate these candidate motion vectors and select the one that produces the best match. So the second question is—What matching metric do we use to decide which of the candidate vectors is the best? This is normally done using a form of *Boundary Matching Error* (BME). Options for this metric are shown in Figure 11.31. The first method shown, referred to simply as BME, evaluates all candidate replacement blocks by computing the SAD value between the boundary pixels of the matched

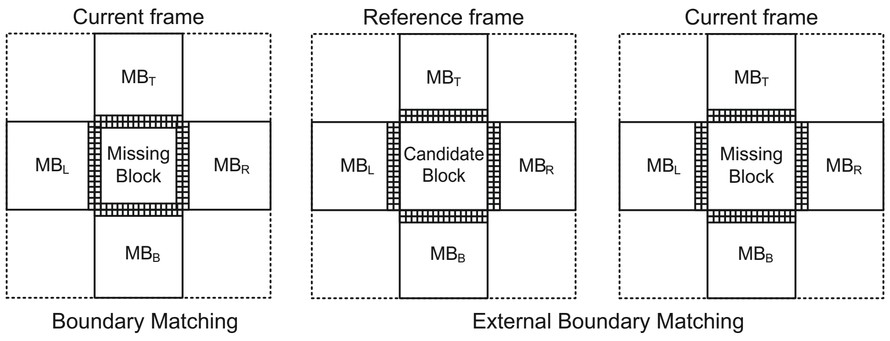


FIGURE 11.31
Boundary Matching Error measures. Left: Boundary Matching Error (BME). Right: External Boundary Matching Error (EBME).

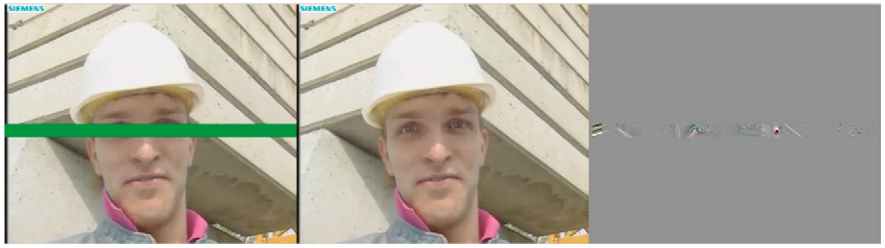


**FIGURE 11.32**
Temporal error concealment results for *Foreman* based on motion compensated replacement. MSE = 14.61. (Courtesy of D. Agrafiotis.)

block and the adjacent pixels from the surrounding blocks. The second method, referred to as an *External Boundary Matching Error* (EBME), again computes an SAD value but this time between the row(s) of pixels surrounding the missing block in the current frame and those surrounding the candidate block in the reference frame. EBME is a superior measure because it fits with the assumption that temporal concealment is preserving temporal smoothness. BME on the other hand is inconsistent with the model, as it uses a spatial continuity metric to assess temporal smoothness.

Temporal error concealment results for the *Foreman* sequence based on motion compensated replacement are shown in Figure 11.32. Because the underlying model is one of smooth translational motion, this technique generally performs significantly better than temporal copying. It can be observed from Figure 11.32 that the residual error is lower than in Figure 11.30 and that the artifacts at the side of the foreman's face are significantly reduced.

**Example 11.8 (Temporal error concealment)**
Consider the region of a current frame and the co-located region of an adjacent reference frame, as shown below. Assuming that the $2 \times 2$ block shown in gray has been lost during transmission, but all other data shown has been received correctly, compute the TEC_MCTR block using a BME metric.

Reference frame

| | | | | | |
|---|---|---|---|---|---|
| 8 | 8 | 8 | 9 | 7 | 9 |
| 8 | 7 | 7 | 6 | 7 | 6 |
| 6 | 5 | 8 | 6 | 4 | 4 |
| 7 | 4 | 7 | 4 | 7 | 8 |
| 5 | 6 | 8 | 6 | 7 | 7 |
| 4 | 6 | 7 | 4 | 4 | 8 |

Current frame

| | | | | | |
|---|---|---|---|---|---|
| 7 | 9 | 7 | 5 | 8 | 9 |
| 7 | 7 | 6 | 4 | 7 | 8 |
| 6 | 5 | 5 | 7 | 7 | 8 |
| 7 | 4 | 3 | 6 | 7 | 7 |
| 5 | 5 | 5 | 7 | 7 | 8 |
| 3 | 4 | 4 | 8 | 4 | 7 |

The motion vector values for adjacent blocks in the current frame are given as:

$$\begin{aligned}\mathbf{d}_L &= [0,0]\\ \mathbf{d}_R &= [-1,1]\\ \mathbf{d}_T &= [-2,2]\\ \mathbf{d}_B &= [2,0]\end{aligned}$$

**Solution.** We can use these four motion vectors to select candidate blocks for replacement. In each case we compute the BME value as follows:

$$\begin{aligned}d &= [0,0] : BME = 19\\ d &= [-1,1] : BME = 11\\ d &= [-2,2] : BME = 7\\ d &= [2,0] : BME = 13\end{aligned}$$

The best candidate block according to the BME metric is that associated with the $[-2,2]$ motion vector. Hence the replacement pixels in this case are:

$$\begin{bmatrix}5 & 6\\ 4 & 6\end{bmatrix}$$

### 11.8.4 Hybrid methods with mode selection

The choice of using either spatial or temporal concealment is not always straightforward. In most cases temporal error concealment provides the best results, and many simple systems use only a temporal copying approach, which works well if there is no motion! In practice, there are obvious situations when one method will be preferable over the other. For example, in the case of a shot cut or fade, spatial concealment is normally superior. In the case of a static scene, temporal block copying will produce good results and in the case of camera or object motion, it is most likely that motion compensated temporal replacement will produce the best results. However, if the scene motion does not fit well with the motion model used to find the candidate vectors then reverting to spatial concealment may be better.

We saw in Section 11.5.5 that slice structured modes can be beneficial in terms of error resilience. In particular, techniques such as *flexible macroblock ordering* (FMO) can help to ensure that, if a slice is lost, some good data is retained in an adjacent slice upon which an interpolation of the lost data can be based. This type of error-resilient encoding can provide significant benefits when combined with concealment. The question that arises when implementing a hybrid system is: How do we select the mode of operation—temporal or spatial? This can be done using a mode selection algorithm.

For example, Figure 11.33 shows the architecture for the EECMS algorithm as proposed by Agrafiotis et al. [27]. This is more formally defined in Algorithm 11.2. EECMS performs mode selection based on a comparison of the motion compensated

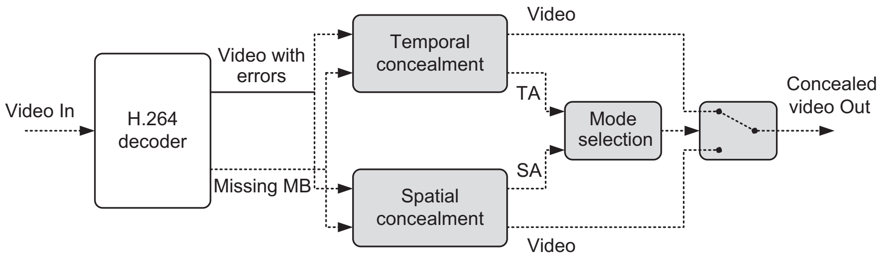


**FIGURE 11.33**

Enhanced error concealment with mode selection [27].

**Algorithm 11.2** EECMS.

```
1. INPUT video with lost block;
2. REPEAT for each corrupted block:
    3. Perform TEC: Use EBME for matching;
    4. Compute spatial activity: $SA = \mathrm{E}\left\{(s_c - \mu)^2\right\}$;
    5. Compute temporal activity: $TA = \mathrm{E}\left\{(s_c - s_r)^2\right\}$;
    6. IF ($TA < SA$) OR ($TA <$ Threshold) THEN (use TEC, GOTO (8));
    7. Perform SEC; use directional interpolation;
8. UNTIL all missing blocks concealed;
9. END.
```

*temporal activity* and the *spatial activity* in the vicinity of the lost block. Spatial activity (SA) is computed as the variance of the surrounding MBs in the current frame whereas temporal activity (TA) measures motion uniformity and is based on the mean squared error between pixels ($s_c$) in the MBs surrounding the missing block in the current frame and those ($s_r$) surrounding the replacement MB in the reference frame. SEC is invoked only if SA is lower than SA, and SA is above a specified threshold. Selected results from the EECMS method are summarized in Table 11.2 and compared to the concealment algorithm used in the H.264/AVC joint model (JM). Across all sequences evaluated in Ref. [27], performance improvements up to 9 dB PSNR were reported.

**Table 11.2** Selected performance results for EECMS (from Ref. [27]).

| PSNR (dB) | 0% PER | 1% PER | | 20% PER | |
|---|---|---|---|---|---|
| | | JM | EECMS | JM | EECMS |
| Foreman | 40.14 | 38.11 | 39.44 | 27.33 | 31.10 |
| Bus | 37.20 | 35.08 | 36.33 | 23.79 | 26.99 |
| Football | 31.27 | 30.14 | 30.56 | 22.05 | 23.22 |

## 11.9 Congestion management and scalable video coding

In cases where streams are encoded on-line, it is possible to control the rate of the video to match that of the channel. However, in many cases, channel conditions can be highly variable and detailed knowledge (especially for multicast applications) may not be available. Also, with pre-encoded streams, it is not possible to adapt the rate unless multiple versions of the same content are encoded. A solution to this problem has been to represent the bitstream in multiple hierarchical layers or in terms of multiple independent descriptions. These approaches are briefly described below.

### *Scalable video encoding*

It is common in video transmission for the signal to be encoded and transmitted without explicit knowledge of the downstream network conditions. In cases where network congestion exists in a packet switched network such as the internet, routers will discard packets that they are unable to forward due to congestion. In some cases it is possible for packets within a stream to be prioritized or embedded such that the least important information is discarded before the more important data. This mechanism lends itself to scalable or layered encoding, where a video signal is composed of a hierarchical layering in terms of spatial resolution, temporal resolution or SNR. This enables devices to transmit and receive multilayered video streams where a base level of quality can be improved through the use of optional additional layers that enhance resolution, frame rate, and/or quality.

The first standardized codec to include optional scalable modes was MPEG-2. However, these were rarely used in practice. More recently, H.264/AVC has introduced a comprehensive set of scalable modes in its Scalable Video Coding (SVC) Annex G extension of H.264/MPEG-4 AVC. SVC has many attractive features, not least that it is compatible with the conventional H.264/AVC bitstream. As we have already seen in Chapter 6, scalable methods have also found favor in still image coding standards such as JPEG2000. Scalable coding methods are now being adopted for some surveillance and video conferencing applications.

In scalable codecs, the aim is to create a hierarchical bitstream where a meaningful representation of the video can be reconstructed at the decoder even if some of the information is lost during transmission or if the terminal is incapable of decoding it. The basic architecture of a scalable video encoder and decoder pair is shown in Figure 11.34. The figure shows the case for spatial or temporal scalability. The other case—SNR scalability—is a subset of this where the up- and downsampling operators are omitted.[2]

Consider the case of spatial scalability as shown in Figures 11.34 and 11.35. The raw video input is first downsampled to the resolution of the base layer and is then transformed and quantized as usual. This layer is entropy coded and transmitted as the base layer. The base layer is then subject to inverse transformation and inverse

[2]For the case of SNR scalability it is normal for the encoder in Figure 11.34 to be modified so that only the quantization and inverse quantization operations are included in the upscaling path.

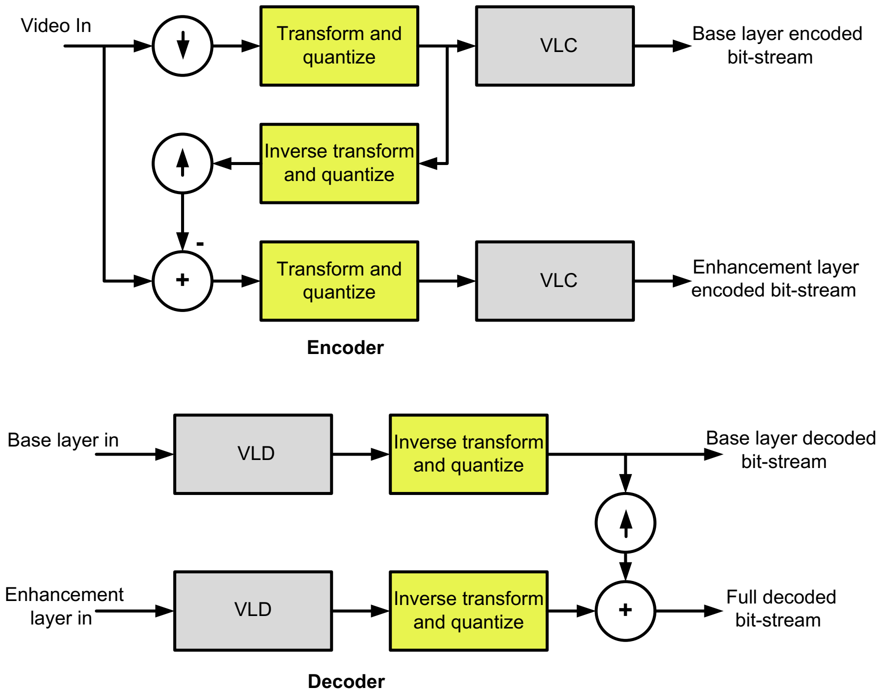

Video In
Transform and quantize
VLC
Base layer encoded bit-stream
Inverse transform and quantize
-
+
Transform and quantize
VLC
Enhancement layer encoded bit-stream
Encoder
Base layer in
VLD
Inverse transform and quantize
Base layer decoded bit-stream
Enhancement layer in
VLD
Inverse transform and quantize
+
Full decoded bit-stream
Decoder

**FIGURE 11.34**

Scalable (spatial or temporal) video codec architecture.

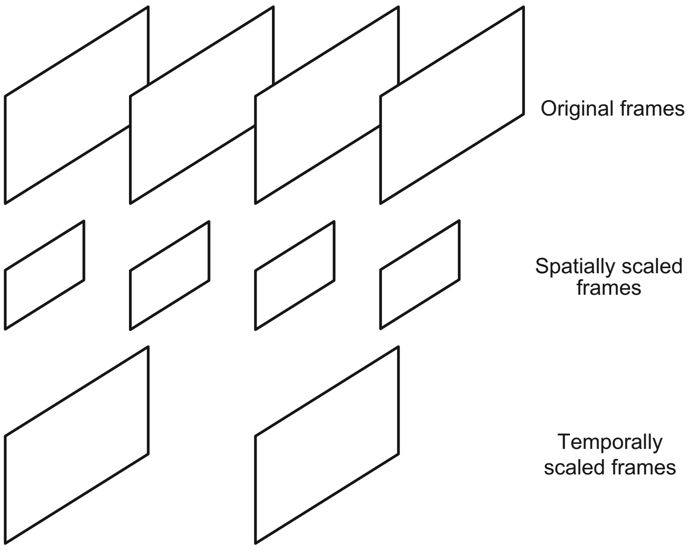


**FIGURE 11.35**

Frame types for scalable encoding.

quantization before being upsampled to its original resolution. The reconstructed base layer is then subtracted from the original video to form a residual signal which is encoded as the enhancement layer. Clearly the base layer must have higher priority than its enhancement layers, thus justifying the use of unequal error protection.

At the decoder, the base layer is decoded as usual but, if the enhancement layer is available, the base layer can be upsampled and added to the decoded enhancement layer to form a full quality enhanced output. This process is similar to that for temporal and SNR scalability. For further information the reader is referred to Refs. [4,34,35].

### *Multiple description coding (MDC)*

Multiple description coding addresses the problem of congestion management in a different way to scalable encoding. Instead of creating a hierarchy of dependent information, where the decoding of each layer depends on the existence of the next most coarse layer, MDC creates a number of independent encodings (descriptions). These can be transmitted independently and may take different paths from source to destination. This path diversity is exploited and MDC ensures that, if any combination of descriptions are received correctly, then a meaningful reconstruction can be achieved at the decoder. If all descriptions are received then the full quality reconstruction is possible. MDC is finding application in methods such as live streaming and peer to peer distribution. This approach is shown diagrammatically in Figure 11.36.

Formation of efficient independent descriptions for video, where coding gain is predicated on the use of predictive encoding, is not straightforward. Many approaches have been proposed and a review of these and their performance is provided by Aposolopoulos et al. [36] and by Wang et al. [37]. For example, the use of FMO slice structuring methods, as described in Section 11.5.5, has been exploited by authors such as Wang et al. [38]. This approach modifies H.264/AVC to use three motion compensation loops and two slice groups to form two independent descriptions. The central encoder is identical to a single description H.264 encoder while the side encoders each process one of the slice groups.

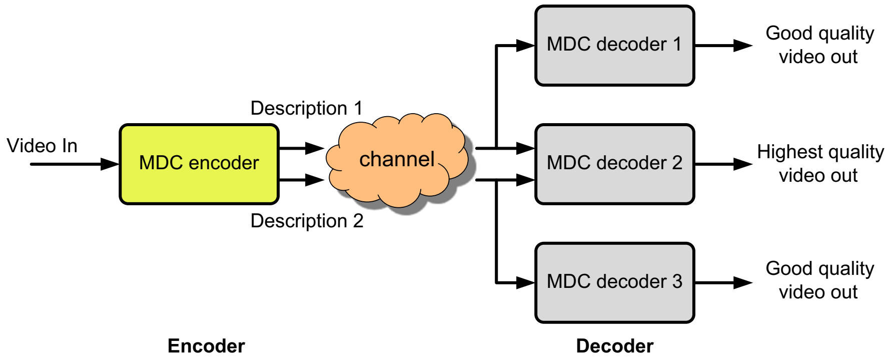


**FIGURE 11.36**

Multiple description codec architecture.

Tesanovic et. al. [39] extend the concept of path diversity to the case of multiple-input-multiple-output (MIMO) wireless technology based on spatial multiplexing. Their paper proposes the use of multiple description video coding (MDC) as a means of emulating the spatial diversity lacking in conventional spatially multiplexed (SM) systems. Singular value decomposition (SVD) is employed to create orthogonal subchannels which provide an efficient means of mapping video content to the wireless channels. Results indicate improvements in average PSNR of the decoded test sequences of around 5–7 dB, compared to standard, single-description video transmission. This is further enhanced by an additional 2–3 dB in the case of channels with low SNR values through the use of unequal power allocation.

## 11.10 Summary

We have examined in this chapter a number of methods that can be employed to improve the reliability of video delivery over lossy channels. We considered why errors propagate spatially and temporally under the influence of variable length and predictive coding regimes and saw examples which clearly demonstrate that the basic coding methods, considered earlier in this book, need to be adapted in a way that preserves rate–distortion performance while improving bitstream resilience to errors. We saw how network techniques such as FEC and ARQ can be exploited to provide better performance, but also how the source coding itself can be modified to deliver improved error resilience with less overhead. It is clear that methods such as PVQ provide greater tolerance to corrupted data while obviating the need for complex encoding tools. Finally we saw how error concealment, possibly in combination with other error resilience methods, can deliver significant improvements in subjective quality with little or no transmission overhead.

## References

[1] C. Chong, C.-M. Tan, D. Laurenson, S. McLaughlin, M. Beach, A. Nix, A new statistical wideband spatio-temporal channel model for 5-GHz band WLAN systems, IEEE Journal on Selected Areas in Communications 21 (2) (2003) 139–150.

[2] G. Athanasiadou, A. Nix, J. McGeehan, A microcellular ray-tracing propagation model and evaluation of its narrow-band and wide-band predictions, IEEE Journal on Selected Areas in Communications 18 (3) (2000) 322–335.

[3] T. Stockhammer, M. Hannuksela, T. Wiegand, H.264/AVC in wireless environments, IEEE Transactions on Circuits and Systems for Video Technology 13 (7) (2003) 657–673.

[4] M. van der Schaar, P. Chou (Eds.), Multimedia over IP and Wireless Networks: Compression, Networking, and Systems, Academic Press, 2011.

[5] Y. Wang, S. Wenger, J. Wen, A. Katsaggelos, Review of error resilient coding techniques for real-time video communications, IEEE Signal Processing Magazine 17 (4) (2000) 61–82.

[6] Y. Wang, Q. Zhu, Error control and concealment for video communications: a review, in: Proceedings of the IEEE, 1998, pp. 974–997 (Special issue on Multimedia Signal Processing).

[7] A. Shokrollahi, Raptor codes, IEEE Transactions on Information Theory 52 (6) (2006) 2551–2567.

[8] M. Luby, LT-codes, in: Proceedings of 43rd Annual IEEE Symposium Foundations of Computer Science (FOCS), 2002, pp. 271–280.

[9] S. Lin, P. Yu, A hybrid ARQ scheme with parity retransmission for error control of satellite channels, IEEE Transactions on Communications 30 (I) (1982) 1701–1719.

[10] P. Ferre, A. Doufexi, J. Chung-How, A. Nix, Robust video transmission over wireless LANs, IEEE Transactions on Vehicular Technology 57 (4) (2008) 2596–2602.

[11] J. Chung-How, D. Bull, Loss resilient H.263 video over the internet, Signal Processing: Image Communication 16 (2001) 891–908.

[12] Y. Takashima, M. Wada, H. Murakami, Reversible variable length codes, IEEE Transactions on Communications 43 (1995) 158–162.

[13] A. Fraenkel, S. Klein, Bidirectional Huffman coding, Computer Journal 33 (4) (1990) 297–307.

[14] B. Girod, Bidirectionally decodable streams of prefix code-words, IEEE Communications Letters 3 (8) (1999) 245–247.

[15] JVT-B027.doc, Scattered Slices: A New Error Resilience Tool for H.26L, Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, February 2002.

[16] P. Ferre, D. Agrafiotis, D. Bull, A video error resilience redundant slices algorithm and its performance relative to other fixed redundancy schemes, Signal Processing: Image Communication 25 (2010) 163–178.

[17] M. Wada, Selective recovery of video packet loss using error concealment, IEEE Journal on Selected Areas in Communications 7 (5) (1989) 807–814.

[18] E. Steinbach, N. Farber, B. Girod, Standard compatible extension of H.263 for robust video transmission in mobile environments, IEEE Transactions on Circuits and Systems for Video Technology 7 (6) (1997) 872–881.

[19] M. van der Scharr, S. Krishnamachari, S. Choi, X. Xu, Adaptive cross layer protection strategies for robust scalable multimedia video transmission, IEEE Journal on Selected Topics in Communications 21 (10) (2003) 1752–1763.

[20] P. Ferre, J. Chung-How, A. Nix, D. Bull, Distortion-based link adaptation for wireless video transmission, EURASIP Journal on Advances in Signal Processing (2008) 17, (Article ID 253706).

[21] D. Redmill, N. Kingsbury, The EREC: an error resilient technique for coding variable length blocks of data, IEEE Transactions on Image Processing 5 (4) (1996) 565–574.

[22] D. Redmill, D. Bull, Error resilient arithmetic coding of still images, in: IEEE International Conference on Image Processing, 1996, pp. 109–112.

[23] R. Swann, N. Kingsbury, Transcoding of MPEG-2 for enhanced resilience to transmission errors, in: IEEE International Conference on Image Processing, 1996, pp. 813–816.

[24] T. Fischer, A pyramid vector quantizer, IEEE Transactions on Information Theory 32 (4) (1986) 568–583.

[25] J. Chung-How, D. Bull, Robust image and video coding with pyramid vector quantisation, in: IEEE International Symposium on Circuits and Systems, vol. 4, 1999, pp. 332–335.

[26] S. Bokhari, A. Nix, D. Bull, Rate–distortion-optimized video transmission using Pyramid Vector Quantization, IEEE Transactions on Image Processing 21 (8) (2012) 3560–3572.

[27] D. Agrafiotis, D. Bull, C. Canagarajah, Enhanced error concealment with mode selection, IEEE Transactions on Circuits and Systems for Video Technology 16 (8) (2006) 960–973.

[28] M. Al-Mualla, C. Canagarajah, D. Bull, Video Coding for Mobile Communications, Academic Press, 2002.

[29] M. Al-Mualla, C. Canagarajah, D. Bull, Temporal error concealment using motion field interpolation, Electronics Letters 35 (3) (1999) 215–217.

[30] H. Lakshman, M. Koppel, P. Ndjiki-Nya, T. Wiegand, Image recovery using sparse reconstruction based texture refinement, in: IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), 2010, pp. 786–789.

[31] P. Salama, E. Shroff, E. Coyle, E. Delp, Error concealment techniques for encoded video streams, in: IEEE International Conference on Image Processing, vol. 1, 1995, pp. 9–12.

[32] D. Agrafiotis, D. Bull, C. Canagarajah, Spatial error concealment with edge related perceptual considerations, Signal Processing: Image Communication 21 (2006) 130–142.

[33] M. Ma, O. Au, G. Chan, M.-T. Sun, Edge-directed error concealment, IEEE Transactions on Circuits and Systems for Video Technology 20 (3) (2010) 382–395.

[34] Y. Wang, J. Ostermann, Y.-Q. Zhang, Video Processing and Communications, Prentice Hall, 2002.

[35] H. Schwarz, D. Marpe, T. Wiegand, Overview of the scalable video coding extension of the H.264/AVC standard, IEEE Transactions on Circuits and Systems for Video Technology 17 (9) (2007) 1103–1120.

[36] J. Aposolopoulos, M. Trott, W.-T. Tan, Path diversity for multimedia streaming, in: M. van der Schaar, P. Chou (Eds.), Multimedia over IP and Wireless Networks: Compression, Networking, and Systems, Academic Press, 2011.

[37] Y. Wang, A. Reibmann, S. Lin, Multiple description coding for video communications, Proceedings of the IEEE 93 (1) (2005) 57–70.

[38] D. Wang, N. Canagarajah, D. Bull, Slice group based multiple description video coding with three motion compensation loops, in: IEEE International Symposium on Circuits and Systems, vol. 2, 2005, pp. 960–963.

[39] M. Tesanovic, D. Bull, A. Doufexi, A. Nix, Enhanced MIMO wireless video communication using multiple-description coding, Signal Processing: Image Communication 23 (2008) 325–336.