Foundations and Trends$^{\circledR}$ in
Signal Processing
Vol. 2, No. 4 (2008) 247–364

DOI: 10.1561/2000000019

Multiple Reference Motion Compensation: A Tutorial Introduction and Survey

By Athanasios Leontaris, Pamela C. Cosman
and Alexis M. Tourapis

Contents

Foundations and Trends® in
Signal Processing
Vol. 2, No. 4 (2008) 247–364

DOI: 10.1561/2000000019

Multiple Reference Motion Compensation: A Tutorial Introduction and Survey

Athanasios Leontaris[1], **Pamela C. Cosman**[2] **and Alexis M. Tourapis**[3]

[1] *Dolby Laboratories, Inc., Burbank, CA 91505-5300, USA, aleon@dolby.com*
[2] *Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407, USA, pcosman@ucsd.edu*
[3] *Dolby Laboratories, Inc., Burbank, CA 91505-5300, USA, atour@dolby.com*

Abstract

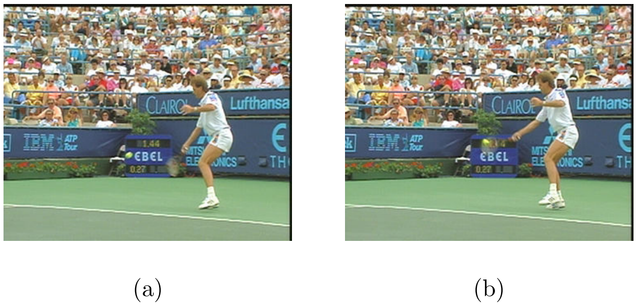
Motion compensation exploits temporal correlation in a video sequence to yield high compression efficiency. Multiple reference frame motion compensation is an extension of motion compensation that exploits temporal correlation over a longer time scale. Devised mainly for increasing compression efficiency, it exhibits useful properties such as enhanced error resilience and error concealment. In this survey, we explore different aspects of multiple reference frame motion compensation, including multihypothesis prediction, global motion prediction, improved error resilience and concealment for multiple references, and algorithms for fast motion estimation in the context of multiple reference frame video encoders.

1

Introduction

Digital video compression has matured greatly over the past two decades. Initially reserved for niche applications such as video-conferencing, it began to spread into everyday life with the introduction of the Video CD and its accompanying Motion Pictures Experts Group MPEG-1 digital video compression standard in 1993 [51]. Home use became widespread in 1996, when the digital video/versatile disk (DVD) with MPEG-2 compression technology was introduced [48, 52]. Digital video compression also facilitated cable and IP-based digital TV broadcast. At the same time, the increase in Internet bandwidth fueled an unprecedented growth in Internet video streaming, while advances in wireless transmission made mobile video streaming possible.

An example video sequence consisting of two frames is shown in Figure 1.1. A frame contains an array of luma samples in monochrome format or an array of luma samples and two corresponding arrays of chroma samples in some pre-determined color sub-sampling format. These samples correspond to pixel locations in the frame. To compress these two frames, one can encode them independently using a still image coder such as the Joint Photographic Experts Group (JPEG) [50] standard. The two frames are similar (*temporally correlated*), hence more



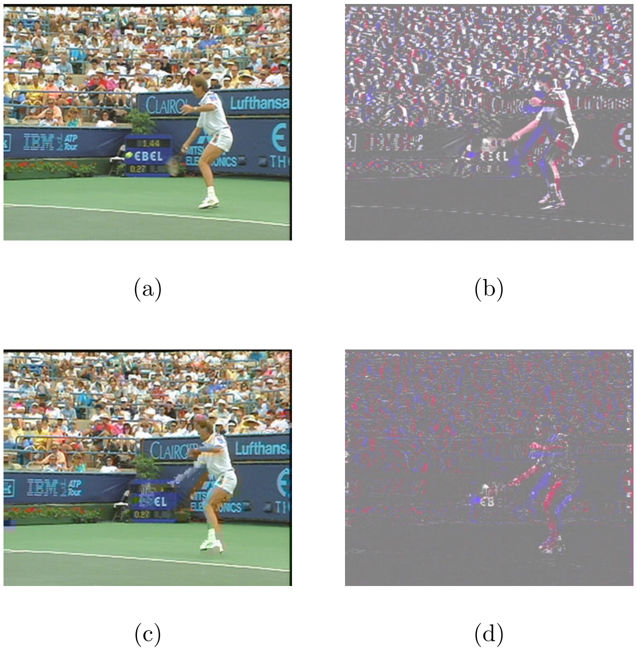
(a) (b)

Fig. 1.1 The previous (a) and the current (b) frame of the video sequence.

compression can be obtained if we use the previous frame to help us compress the current frame. One way to do this is to use the previous frame to *predict* the current frame, and then to encode the difference between the actual current frame and its prediction. The simplest version of this process is to encode the difference between the two frames (i.e., subtract the previous frame from the current frame and encode that difference). In this case, the entire previous frame becomes the *prediction* of the current frame. Let i and j denote the spatial horizontal and vertical coordinates of a pixel in a rectangularly sampled grid in a raster-scan order. Let $f_n(i,j)$ denote the pixel with coordinates (i,j) in frame n. Let $\hat{f}_n(i,j)$ denote the predicted value of this pixel. The prediction value is mathematically expressed as $\hat{f}_n(i,j) = f_{n-1}(i,j)$. This technique is shown in the first row of Figure 1.2. For sequences with little motion such a technique ought to perform well; the difference between two similar frames is very small and is highly compressible. In Figure 1.1(b) for example, most of the bottom part of the tennis court will be highly compressed since the difference for these areas will be close to zero. However, there is considerable motion in terms of the player and camera pan from one frame to the next and the difference will be non-zero. This is likely to require many bits to represent.

1.1 Motion-Compensated Prediction

The key to achieving further compression is to *compensate* for this motion, by forming a better prediction of the current frame from some



(a) (b)

(c) (d)

Fig. 1.2 Motion compensated prediction. The top row shows the prediction which is the unaltered previous frame (a) and the resulting difference image (b) that has to be coded. The bottom row shows the equivalent prediction (c) and difference image (d) for motion compensated prediction. The reduction of the error is apparent.

reference frame. A frame is designated as a reference frame when it can be used for motion-compensated prediction. This prediction of the current frame, and subsequent compression of the difference between the actual and predicted frames, is often called *hybrid coding*. Hybrid coding forms the core of video coding schemes from the early compression standards such as ITU-T H.261 [103] and ISO MPEG-1 to the most recent ISO MPEG-4 Part 2 [53], SMPTE VC-1 [82], China's Audio Video Standard (AVS) [29], ITU-T H.263 [104], and ITU-T H.264/ISO MPEG-4 Part 10 AVC coding standards [1, 84].

When a camera pans or zooms, this causes *global motion*, meaning that all or most of the pixels in the frame are apparently in motion in some related way, differing from the values they had in the previous frame. When the camera is stationary but objects in the scene move, this is called *local motion*. To compensate for local motion, a frame is typically subdivided into smaller rectangular blocks of pixels, in which motion is assumed to consist of uniform translation. The translational motion model assumes that motion within some image region can be represented with a vector of horizontal and vertical spatial displacements. In *block-based* motion-compensated prediction (MCP), for each block b in the current frame, a *motion vector* (MV) can be transmitted to the decoder to indicate which block in a previously coded frame is the best match for the given block in the current frame, and therefore forms the prediction of block b. Let us assume a block size of 8×8 pixels. The MV points from the center of the current block to the center of its best match block in the previously coded frame. MVs are essentially addresses of the best match blocks in the reference frame, in this case the previous frame. Let $\mathbf{v} = (v_x, v_y)$ denote the MV for a block in frame n. For the pixels in that block, the motion-compensated prediction from frame $n-1$ is written as $\hat{f}_n(i,j) = f_{n-1}(i + v_x, j + v_y)$. If the MV is $\mathbf{v} = (0,0)$, then the best match block is the co-located block in the reference frame. As Figure 1.1 shows, parts of the tennis court at the bottom part of the frame appear static, so the best match is found with the $(0,0)$ MV. However, there is substantial motion in the rest of the frame that can only be modeled with *non-zero* MVs.

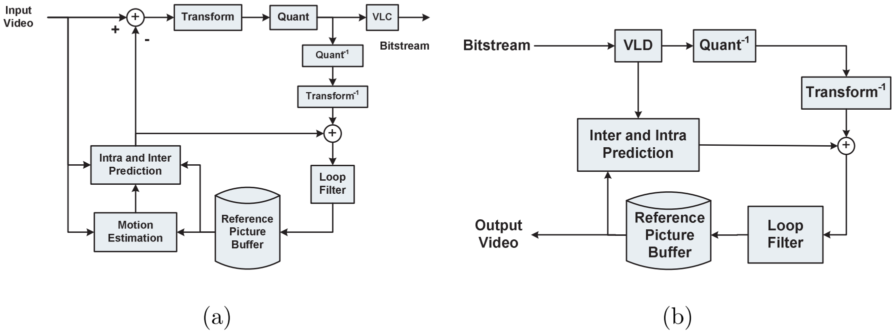
MVs or, in general, *motion parameters* are determined by doing a motion search, a process known as *motion estimation* (ME), in a reference frame. Assuming a search range of $[-16, +16]$ pixels for each spatial (horizontal and vertical) component, $33 \times 33 = 1089$ potential best match blocks can be referenced and have to be evaluated. The MV $\mathbf{v}$ that minimizes either the sum of absolute differences (SAD) or the sum of squared differences (SSD) between the block of pixels f in the current frame n and the block in the previous frame $n - l$ that is referenced by $\mathbf{v} = (v_x, v_y)$ may be selected and transmitted. Let b denote a set that contains the coordinates of all pixels in the block. The SAD and SSD are written as:

$$SAD = \sum_{(i,j)\in b} |f_n(i,j) - f_{n-l}(i + v_x, j + v_y)| \tag{1.1}$$

$$SSD = \sum_{(i,j)\in b} (f_n(i,j) - f_{n-l}(i + v_x, j + v_y))^2 \tag{1.2}$$

To form the MCP of the current frame, the blocks that are addressed through the MVs are copied from their original spatial location, possibly undergoing some type of *spatial filtering* (more on that in Section 3.5), to the location of the blocks in the current frame, as shown in Figure 1.2(c). This prediction frame is subsequently subtracted from the current frame to yield the *motion-compensated difference frame* or, in more general terms, the *prediction residual* in Figure 1.2(d). Obviously, if the MCP frame is very similar to the current frame, then the prediction residual will have most of its values close to zero, and hence require fewer bits to compress, compared to coding each frame with JPEG or subtracting the previous frame from the current one and coding the difference. One trade-off is an increase in complexity since ME is costly. The prediction residual is typically transmitted to the decoder by transforming it using a discrete cosine transform (DCT), rounding off the coefficients to some desired level of precision (a process called quantization) and sending unique variable-length codewords to represent these rounded-off coefficients. Along with this difference information, the MVs are transmitted to the decoder, requiring some additional bit rate of their own. For content with sufficient temporal correlation, the overall bit rate requirements are much less than without the use of MCP.

A diagram of a hybrid codec is illustrated in Figure 1.3. The decoder uses the MVs to obtain the motion compensated prediction blocks from some previously decoded reference frame. Then, the decoded prediction



(a) (b)

Fig. 1.3 Hybrid video (a) encoder and (b) decoder.

residual block is added to the MCP block to yield the current decoded block. This is repeated until the entire frame has been reconstructed. The reconstructed frame at the decoder may not be identical with the original one, because of the quantization used on the residual blocks.

Note that MCP for a block is also known as *inter prediction* since inter-frame redundancy is used to achieve compression. When combined with coding of the prediction residual it is called *inter-frame* coding. When a block is encoded independently of any other frame, this is known as *intra-frame* coding. Usually, intra-frame coding involves some kind of *intra-frame prediction* or *intra prediction*, which is predicting a block using spatial neighbors. This might involve using the DC coefficient of a transform block as a prediction of the DC coefficient of the next transform block in raster-scan order (as in JPEG). Or it might involve prediction of each pixel in a block from spatial neighbors using one of several possible directional extrapolations (as in H.264). In general, inter-frame coding enables higher compression ratios but is not as error resilient as intra-frame coding, since, for inter-frame coding, decoding the current frame depends on the availability of the reference frame. Video frames (or equivalently pictures) that use intra-frame coding exclusively to encode all blocks are called *intra-coded* or *I-coded frames*, while frames that allow the use of either intra-frame or inter-frame coding from some reference frame are known as *P-coded frames*. P-coded frames have been traditionally constrained to reference past frames in display order (as in the early standards H.261, H.263, MPEG-1, MPEG-2, and MPEG-4 Part 2). Finally, *B-coded frames* allow bi-directional prediction from one past and one future frame in display order in addition to intra-frame or inter-frame coding. Note that referencing future frames in display order generally involves transmitting frames out of order. For example, frame 3 can be encoded after frame 1 and then frame 2 can be encoded making reference to both frames 1 and 3. A simple illustration is shown in Figure 1.4. Note that B-coded frames were further extended in H.264/MPEG-4 AVC [1] to provide for a more generic form of bi-prediction without any restrictions in direction. Detailed information on bi-predictive coding is found in Section 4.1.

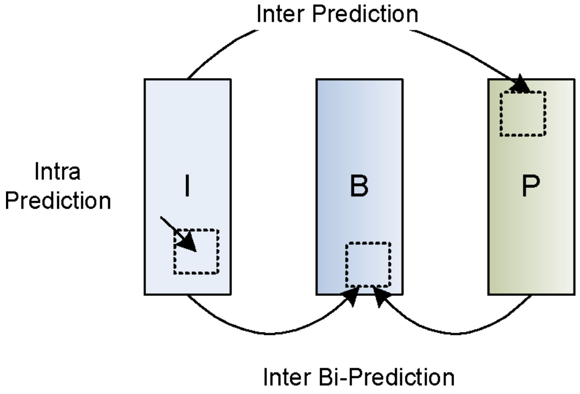


Fig. 1.4 An example of different prediction schemes.

1.2 Outline

Block-based MCP traditionally made use of a single previous frame as a reference frame for motion-compensated prediction, while for B-coded frames a single future frame was used jointly with the previous frame in display order to produce the best prediction for the current frame. However, motion search does not have to be limited to one frame from each prediction direction. Temporal correlation can be often nontrivial for temporally distant frames. In this article, the term multiple-reference frame motion compensation encompasses any method that uses combinations of more than one reference frame to predict the current frame. We also discuss cases where reference frames can be synthesized frames, such as panoramas and mosaics, or even composite frames that are assembled from parts of multiple previously coded frames. Finally, we note that we wish to decouple the term reference frame from that of a decoded frame. While a decoded frame can be a reference frame used for MCP of the current frame, a reference frame is not constrained to be identical to a decoded frame. The first treatise of the then state-of-the-art in multiple-reference frames for MCP is [109]. This work is intended to be somewhat broader and more tutorial. The article is organized as follows. Section 2 describes background, mosaic, and library coding, which preceded the development of modern multiple-reference techniques. Multiple-frame motion compensation is treated in

Section 3, while the almost concurrent development of multihypothesis prediction, often seen as a superset of multiple-reference prediction, is investigated in Section 4. The commercialization and rapid deployment of multiple-reference predictors has been hampered by the increased complexity requirements for motion estimation. Low complexity algorithms for multiple-frame motion search are covered in Section 5. The uses of multiple references for error resilience and error concealment are discussed in Sections 6 and 7, respectively. An experimental evaluation of some of the advances discussed in this work is presented in Section 8. This survey is concluded with Section 9. Appendix A provides the reader with additional information on rate-distortion optimization and Lagrangian minimization. Note that this work disregards the impact of each prediction scheme on decoder complexity.

2

Background, Mosaic, and Library Coding

During the mid-1980s, video compression was primarily aimed at video-conferencing, often at speeds as low as 28 kbps. Given the low rate, researchers were hard pressed to make every transmitted bit count. Because of the application, researchers realized that motion-compensated hybrid video coding could benefit if background information could be stored and re-used for coding subsequent frames. Video-conferencing usually consists of one or more people talking in front of a static background (an office, a blackboard, etc.). Small gestures and movements can temporarily *occlude* parts of the background, but these are often *uncovered* again later on. This is illustrated in Figure 2.1, where the block in frame n (the current frame to be encoded) that contains the tree trunk is unable to find a suitable reference block in frame $n-1$, since the trunk is occluded. A good match, however, is available in frame $n-6$. The lack of a good match in the previous frame could be handled by using not only the previous frame for reference prediction, but by also storing content that is deemed to be static in a secondary background memory, which can be used as an additional prediction reference. In the following sections, we discuss several approaches for accomplishing this.

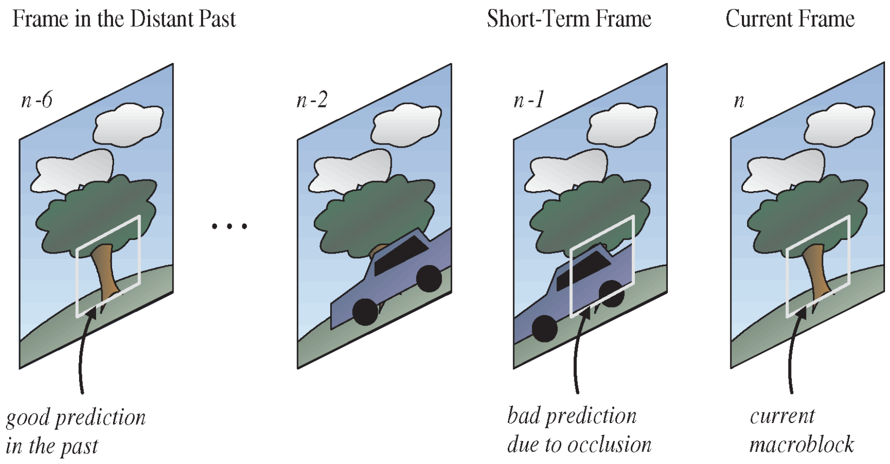

Frame in the Distant Past
Short-Term Frame
Current Frame
n-6
n-2
n-1
n
. . .
good prediction in the past
bad prediction due to occlusion
current macroblock

Fig. 2.1 Multiple frame motion-compensated prediction. The availability of multiple reference frames improves motion compensation by providing reliable matches even when occlusion is temporarily present.

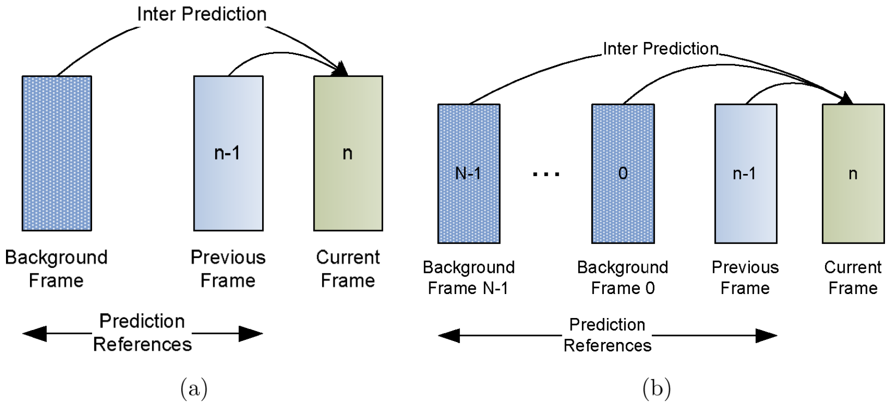


Fig. 2.2 Background updating for inter prediction: (a) single and (b) multiple backgrounds.

2.1 Background Updating and Replenishment

Background updating and replenishment techniques, first introduced in [74], are treated in this section. Most of the schemes presented here employed two references for prediction: the previous frame and the background frame as shown in Figure 2.2(a). More than two reference frames for prediction were used in [116] and [118] as shown in Figure 2.2(b). The *initialization* of the background frame memory is universally accomplished by adopting the first reconstructed frame of the video sequence as the initial background frame. After initialization, parts of the background frame memory are updated as needed. Updating and replenishing are not quite synonymous; a background pixel's old value is *updated* toward a value in the current frame when the weighted average of old and current values assigns more weight to the old value, and the pixel is *replenished* if the background pixel's value becomes very close to the current value because the current value is afforded a larger weight. When speaking of these approaches in general, we will use the term updating to refer to the general case.

This family of techniques comprises three major components:

(1) The specification of the background updating process.
(2) The decision on whether and how to update.
(3) A reference selection mechanism that decides whether to use the background memory or some other previous reference frame for the prediction of each block.

We will discuss each of these three aspects in turn.

Most of the techniques update small portions of the background frame, which can be either macroblocks (four blocks in a 2×2 arrangement), blocks, or even single pixels. The first known work in this genre attempted to identify background/object pixels [74]. If the difference between the pixel in the current frame and the one in the previous frame was below a threshold, the pixel was classified as background, and the corresponding background memory pixel was updated. A more complex approach followed in [10] that made use of an image segmentation algorithm. The current background memory was updated using segmentation information from the previous frame. In [42], each time a pixel is unchanged between two frames, the pixel counter is incremented. If the pixel changes, the counter is reset to zero. When the counter reaches a threshold, that pixel in the background memory is updated. A higher threshold entails more reliable updating, while a lower threshold yields faster response. An evolution of the updating algorithm of [42] appeared in [41]; here the pixel is considered unchanged and its counter is incremented if the sum of absolute differences (SAD) for a square neighborhood around the pixel in the current and previous frame is below a threshold. A similar change detector was later employed in [116].

All of the approaches discussed so far attempt to decide whether some portion of the background needs to be updated based on evaluating how static a pixel or block is. If the value is substantially different, then it is highly likely that there is an object in motion, and the area should not be considered as part of the background; the corresponding area in the background frame memory should be left alone. On the other hand, if the value is similar, then perhaps there is a slight change in illumination or some other small background change, and the corresponding area in the background frame memory should be updated for future reference use.

A new element to this approach appeared in [117], where the authors buffered a map of the quantization parameter (QP) that was used to encode each pixel. By storing the QP values associated with different portions of the background frame memory, the encoder and decoder can both assess the quality of a stored pixel value. If a block is encoded with an all-zero motion vector, the transformed coefficients of the difference are transmitted and the current QPs are lower (hence better quality) than the stored ones, then the background memory and the QP map for that block are updated.

Further work presented in [118] employed a similar background updating scheme, but featured multiple background memories, as shown in Figure 2.2(b). The algorithm first determines whether a frame is a scene change. If the current scene is not matched with any of the stored background frame memories, then a scene change frame is intra-frame coded. A spare memory or, if all have been used, the oldest one is then used to store the scene change frame.

We have so far discussed how the decision to update is made; we now discuss how the updating itself is performed. Mukawa and Kuroda [74] updated the background memory with constant increments/decrements so that background values gradually approach current ones. In [10], the old values are simply replaced by the new values obtained through segmentation. The background memory was updated in [42] on a per-pixel basis by weighting the current reconstructed and the stored background value adaptively according to the counter level. In [41], the stored background values are slightly updated by altering them by −1, 0, or +1 toward the new reconstructed value, similar to [74]. Updating and replenishing, with weighted averages conditioned on the QP maps, was used in [117] and [118].

The final component of these techniques is the decision, when encoding each macroblock, between using the background memory or the previous frame as a prediction reference. The majority of the techniques selected the best prediction as the one minimizing the SAD or the sum of squared differences (SSD). That is, when encoding the current block, both the previous and background frames would be searched for the best match block. The block in any of these frames that was found to minimize the error metric was declared the overall best match. There are a few exceptions to this general approach. In [10], pixels classified as background were encoded referencing the background memory, while object pixels referenced the previous frame. A different approach for the selection of the prediction was proposed in [116]. The scheme uses *three* separate reference frames: the previous frame, the background frame, and a frame with the zero value in all pixels. The use of the third reference is equivalent to switching prediction off altogether (a special case of intra-frame prediction). The input and the three predictions are low-pass filtered and subsampled to produce multi-resolution pyramids. The input pyramid has overlapping frequency content while the prediction pyramids do not due to additional high-pass filtering. The best predictions are then selected for each resolution level from the references. Hence, all three references can in theory be employed to predict a single block.

Background prediction for video coding was also standardized as part of the ITU-T Recommendation H.120 [24], which also bears the distinction of being the first digital video coding standard having been introduced in 1984. The first version of H.120 supported conditional replenishment coding. In conditional replenishment in H.120, a frame memory is updated at a constant rate and differences between the current image and the values stored in the frame memory are used to determine the picture elements in the current image that are deemed to be moving. Only these picture elements were coded using differential pulse-coded modulation (DPCM) followed by variable length coding of the DPCM codewords. The second version of H.120 added motion-compensated prediction and background coding. In fact, the H.120 codec utilized two references for inter-frame prediction: either motion-compensated prediction from the previous frame or background-based prediction. These were supplemented by intra-frame coding. The background frame memory was updated on a pixel-level at a constant rate, provided the difference between the stored values and those in the current frame was similar enough. The updating process was on purpose designed to be slow.

2.2 Mosaics Generated Through Global Motion Models

As background updating and replenishment techniques sprang out of the need for efficient compression of video-conferencing signals, the use of mosaics originated from coding surveillance video signals, long *panoramic* shots, and scenes involving camera pan and zoom. A panorama is typically a rectangular image often spanning the space of at least two frames side-by-side that are stitched together at their common edge. The mosaic is almost identical to the panorama with the difference of allowing pictures to be stacked vertically or diagonally as well. In the remainder of this paper we will use the terms panorama and mosaic interchangeably. As we can see in Figure 2.3, a camera that pans back and forth across a scene, with some sparse object motion every now and then, could greatly increase its compression efficiency if it had access to a panoramic view of the *background* and used it as an additional reference.

Mosaics are generated with the help of global/camera motion models, of which the affine and perspective models are the most widely used. The affine model has six parameters: a 2D displacement vector, a rotation angle, two scaling factors, and their orientation angle. The perspective model is slightly more complex, requiring eight parameters to fully describe global motion. The perspective global motion model was used in [27], the polynomial in [111, 112], and the affine in [45, 49]. The main motivation behind mosaic frames is to provide the codec with background information that spans a large spatio-temporal space. Traditionally, to generate a mosaic, *warped* frames are obtained with the help

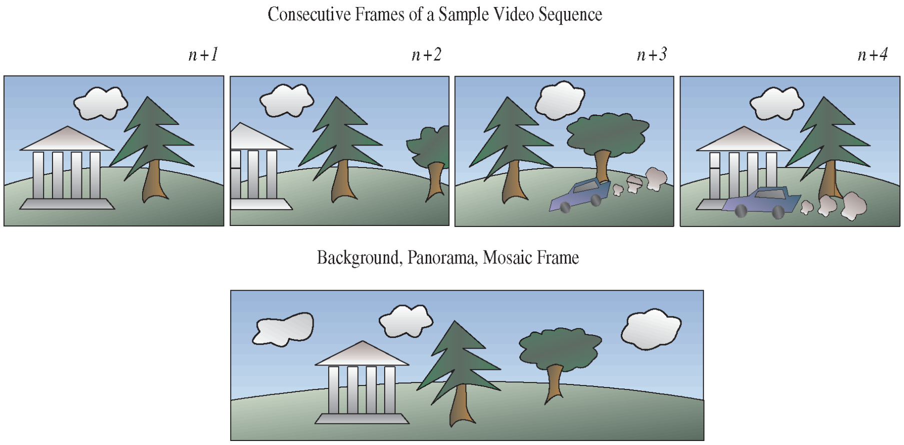

Consecutive Frames of a Sample Video Sequence
n+1
n+2
n+3
n+4
Background, Panorama, Mosaic Frame

Fig. 2.3 Background prediction. The background memory can be a mosaic or panoramic frame obtained through background updating or global motion compensation methods. The availability of the panorama/mosaic to the codec will greatly reduce the bitrate requirements for transmission of this sequence.

of the motion model and are then aligned with respect to a common coordinate system and stitched together (a process termed *temporal integration*) using some kind of weighted temporal median or average.

Mosaics are divided into two major categories: *dynamic* and *static* [49]. Static mosaics can be generated off-line from the entire video sequence when used for storage applications. However, for practical purposes a static mosaic can be generated from the first, say, 100 frames of the video. This fixed mosaic frame can then be used as an additional reference for compression. Since they are generated once and never again updated, static mosaics cannot handle newly appearing objects. Dynamic mosaics (e.g., [27]), in contrast, are progressively updated as new video content is made available to the encoder. New moving objects are incorporated into the dynamic mosaic but may cause artifacts such as “ghosting”. As discussed in [45], dynamic mosaics often cannot handle uncovered background that was previously occluded by moving objects.

Hence the prediction performance of static mosaics suffers when new objects appear, while prediction using dynamic mosaics suffers from ghosting and uncovered background. Two different approaches

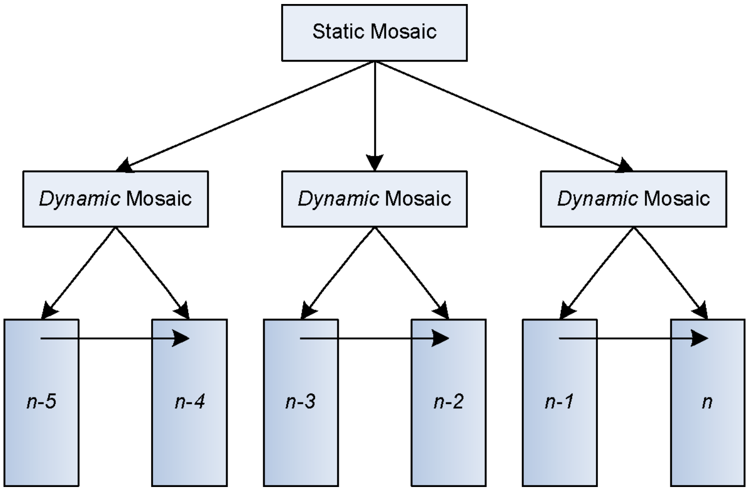


Fig. 2.4 A multiresolution mosaic tree.

were applied to address these weaknesses. One approach consisting of a temporal multi-resolution mosaic *pyramid* was proposed in [45] combining the best of both. Both static *and* dynamic mosaics were employed in a multi-resolution tree where each node represented a mosaic for a different time-scale and each leaf represented an original frame, as shown in Figure 2.4. Interior nodes are mosaics that merge together the visual information of their children. Leaves were predicted from the "left", the dynamic mosaic, or from the "top", the static mosaic. In [27], instead of attempting to compensate for ghosting in dynamic mosaics, steps were taken to stop it from occurring in the first place. Frames were first segmented into foreground and background, and only the background segment of the frame was used to update the dynamic mosaic.

Multi-resolution mosaics [49], composed of either exclusively dynamic or static mosaics, can also be used to provide a better description of the sequence in cases of camera zoom that provide finer resolution of the images.

After the mosaic has been generated at the encoder, it has to be signaled to the decoder so that both the encoder and the decoder have access to the identical mosaic, and *drift* errors (also known as *prediction mismatch*) are avoided. In both [49] and [45], the mosaics had to be encoded and explicitly transmitted to the decoder. In all other techniques mentioned in this section, the decoder can generate the mosaic independently by using the reconstructed frames and the transmitted motion information.

The previous frame or the mosaic frame can be used as a prediction reference. In both [49] and [27], the current frame was predicted from either the previous frame or the mosaic using block-based motion compensation, depending on which reference yielded the smallest error. In the multi-resolution mosaic pyramid of [45], however, each frame is predicted from the dynamic mosaic of its temporal predecessor or from a static mosaic in the higher tree level.

So far we have discussed schemes where successive frames are warped to be registered with respect to a common coordinate system and then stitched together to form a mosaic that serves as a prediction reference. However, one can still warp the frames but not perform the temporal integration step. The warped frames can be employed unaltered [112, 111]. The current frame in [112] is predicted from the previous frame and multiple warped versions of that same previous frame, each requiring a set of polynomial motion parameters. The polynomial motion model employed is an orthonormalized version of the six-parameter affine motion model that enables easier quantization of the motion model's parameters. The notion of using multiple motion parameters for the same frame was motivated by the fact that complex motion cannot be adequately described by a single motion model parameter set. The technique was extended in [111] by making use of multiple regular frames jointly with their warped versions, as shown in Figure 2.5.

2.3 Composite Memories

In this section, we treat another family of techniques that have been designed to compress video sequences with recurring visual elements. No distinction is made here between foreground and background, because recurring elements can belong to *both*. The objective is to select those frame elements that recur over time, selectively buffer them for

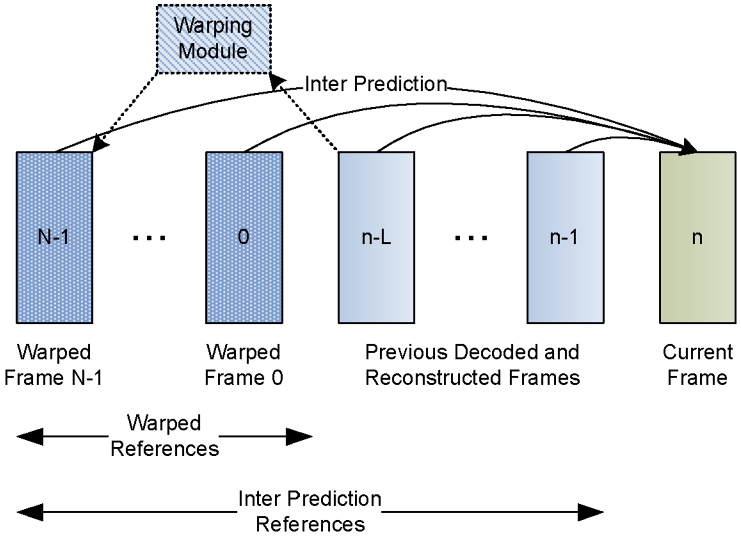


Fig. 2.5 Regular and warped frames for inter prediction. Warping modules are used to warp previously decoded frames and buffer them as additional references.

future use, and then remove them from the buffer when they become obsolete. In these techniques, the resulting *composite* frame was used as an additional prediction reference along with previous frames.

Vector quantization was used to select, buffer, and remove the blocks in [101] within a *library*, which was then used to predict the current frame. This scheme requires the explicit transmission of the composed library for every frame. The overhead was somewhat addressed by selectively transmitting the difference between subsequent libraries.

In contrast, the selection of relevant blocks in [61] was done by satisfying an SAD-based criterion of *frame coherence*. Satisfying the frame coherence criterion is similar to solving a puzzle: the blocks taken from the composite frame and inserted into the composite search area for encoding the current frame have to fit and blend in with their surrounding blocks. The removal of irrelevant blocks from the memory used a "first in first out" (FIFO) scheme where blocks with the lowest priority were removed first. Both the insertion and removal algorithms used reconstructed frames and the decoder replicated the encoder's operation without requiring the transmission of the composite frame as additional overhead. A common property of those schemes that sets them apart from previous methods is that the composite frame or library in [101] and [61] is a compilation of useful blocks with no particular order. They do not have to resemble a frame, although the composite search area that is assembled from the library does tend to resemble a frame or a portion of a frame.

The algorithm by Hidalgo et al. [43] that is discussed next in Section 3.3 selected a reference frame to buffer as a long-term reference for H.264/MPEG-4 AVC codecs using MPEG-7 metadata. The frame selection was made by comparing Euclidean distances of the MPEG-7 descriptors that correspond to each frame. Interestingly, the same algorithm can also be applied on a sub-block basis, each of which can be smaller than a frame. The sub-blocks are selected from the frames buffered in the reference frame memory. The result of assembling these sub-blocks is a composite long-term reference frame, which, however, is no longer H.264/MPEG-4 AVC standard-compliant, as shown in Figure 2.6.

After constructing the composite/library reference frame, the selection of the best match block is done as follows. In [101], all library entries are tested and the best match is transmitted to the decoder by encoding the block index. However, in [61], regular motion estimation is applied by means of copying the blocks, which are stored in

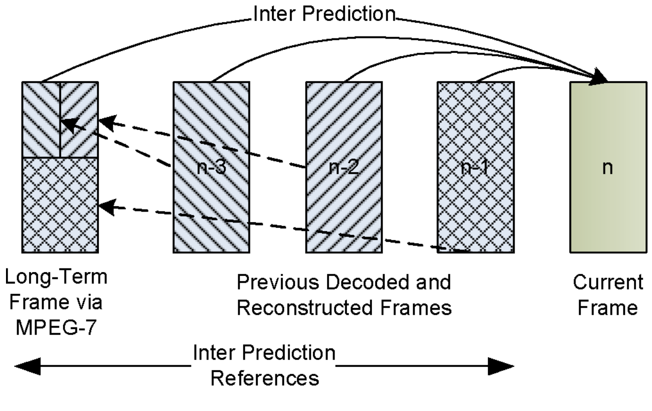

Inter Prediction
n-3
n-2
n-1
n
Long-Term Frame via MPEG-7
Previous Decoded and Reconstructed Frames
Current Frame
Inter Prediction References

Fig. 2.6 A composite long-term reference frame built with the help of MPEG-7 descriptors.

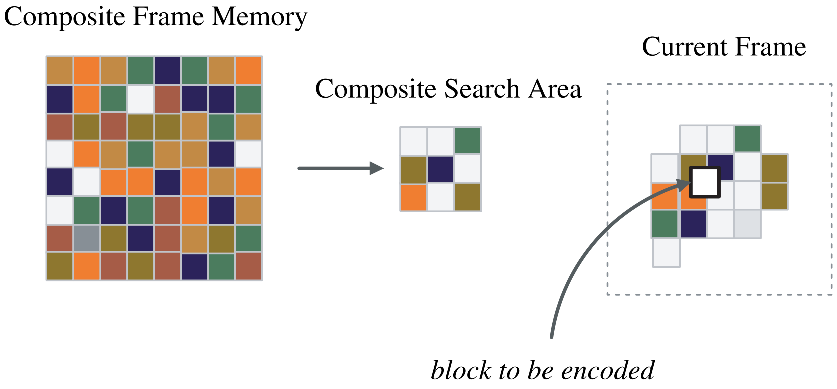


Fig. 2.7 Composite Search Area Generation. The stored composite memory does not necessarily resemble a frame. Most blocks are stored in an FIFO order. However, during motion estimation, stored blocks are used to create search areas (often more than one) to predict the current frame block. These search areas naturally resemble part of the current frame.

the composite memory, into a new prediction buffer with their boundaries matched so that one or more search areas are composed. This is illustrated in Figure 2.7. Last, in [43] the long-term composite frame is treated as any regular reference frame and is considered during motion estimation and compensation.

It is worth noting that a form of composite memories did find its way into a standard specification. The Enhanced Reference Picture Selection mode (Annex U) of H.263 defined syntax for sub-picture removal. The rationale was to be able to reduce the amount of memory needed to store multiple reference frames. A sub-picture removal command to the decoder indicates that specific areas of a picture will not be used as references for prediction of subsequent pictures. This will result in using that memory to store parts of *other* reference pictures, giving rise to composite reference frames. While H.264/MPEG-4 AVC did retain and expand upon a substantial part of the functionality included in Annex U of H.263, the sub-picture removal capability was not adopted into the H.264/MPEG-4 AVC standard.

3

Multiple Reference Frame Motion Compensation

In this section, we consider multiple reference frames for motion-compensated prediction of the current block. Chronologically, the multiple reference prediction (MRP) schemes received attention after the development of background coding methods, which were treated in Section 2. The high computational cost associated with motion estimation and the high procurement cost of computer hardware imposed computational and memory constraints that favored prediction from at most two reference frames; the previous and some mosaic or background frame. The significant strides in semiconductor fabrication in the 1990s allowed more complex prediction schemes that gave rise to multiple reference frame buffers.

3.1 A Brief Historical Perspective

The first documented attempt to use multiple reference frames for MCP dates back to 1993 [40]. The authors used up to eight past frames in display order as references to predict a block in the current frame and showed that the sum of squared differences (SSD) of the prediction error is lower compared to prediction from a single reference frame, which in that case was the previous frame in display order. This work, however, did not consider the impact of such a predictor when incorporated within a video codec. Doing so, the authors disregarded the additional overhead in bits that is required to code the reference indices. In the case of eight reference frames, as studied in [40], one would have to transmit exactly three bits per block to signal the MCP reference index, assuming all indices are equiprobable and that no prediction or entropy coding takes place. Better results are possible when predicting the index from spatial neighbors and entropy coding the prediction residual.

Multiple reference frames for improved MCP were first proposed in the context of a hybrid video codec, in this case H.263, in [11]. Huffman codes were used to signal the reference indices and the selection of the reference index was made by minimizing the motion-compensated prediction error. However, the full potential of multiple reference frame prediction did not become apparent until the submission of a standardization contribution [114] to the ITU-T's Video Coding Experts Group (VCEG) in 1997. A long-term memory frame buffer that comprised up to 50 frames was used for MCP and the reported compression gains were substantial. Rate-constrained motion estimation used Lagrangian minimization that accounted for the increase in bits required to signal the reference frame indices. Furthermore, the authors also provided some highly intuitive and highly efficient approximations of the Lagrangian parameter λ when used for rate-distortion optimization during coding mode decision and motion estimation. The compression gains from the Lagrangian approximation were equally compelling. Additional information on rate-distortion optimization and Lagrangian minimization can be found in Appendix A.

A multiple frame predictor with N reference frames, as shown in Figure 2.1, requires N times the memory complexity of a standard single-frame prediction scheme. While a straightforward brute force motion search would increase the computational complexity of motion search by a factor N, in practice this can be considerably constrained as we discuss in detail in Section 5. Note that for the decoder case, while memory complexity increases, the computational complexity is essentially equal to that of a single-reference frame predictor.

The coding results presented in [11, 114] convinced researchers and industry practitioners in video compression that the coding gains made possible by using multiple reference frames for MCP were nontrivial and motivated people to work toward standardizing the use of multiple reference frames. The result of this effort was the Annex U for “Enhanced Reference Picture Selection” of ITU-T H.263 [104, 107, 119]. The prior Annex N for “Enhanced Reference Picture Selection” of ITU-T H.263 supported multiple reference frames but in a very limited manner. More details on Annexes N and U follow later in this section. Computational and memory complexity was always an issue with multiple reference frames, and people had expressed doubt whether Annex U would be practical in a real-time system. Arguments in favor of wide adoption of multiple reference frames for MCP received a boost when a practical system was demonstrated at the ITU-T VCEG in 2000 [44].

When the ITU-T video coding experts group issued a request for proposals for the successor of H.263, which was then termed H.26L, early proposals [6, 7] were quick to adopt multiple reference frames for prediction. As the H.26L effort morphed into the H.264/MPEG-4 AVC standard of the ITU-T/ISO Joint Video Team [1], multiple reference frames had become and are now an integral part of the standard. Some reasons for the increased compression efficiency are now explained.

3.2 Advantages of Multiple Reference Frames

Often cited [47, 83] reasons for the improved compression performance of multiple-reference over single-reference frame prediction, many of which are also valid for background and mosaic coding, can be summarized as follows:

1. Periodic occurrence of motion. Useful instances of the same object could exist in some past or future frame in display order.
2. Uncovered background. Moving objects occlude parts of the background not available in the previous frame. However, due to motion, that part may be uncovered in some other past coded frame.

3. Alternating camera scenes due to camera (global) periodic movement, such as camera shaking, where a different past coded frame could be a better prediction of the current frame than the previous one.
4. Sampling grid. In past coded frames we often have access to prediction samples for the current frame that correspond to arbitrary fractional pixel displacements, which cannot be obtained by traditional fractional pixel motion compensation.
5. Lighting and shadow changes. Small variations of the local or global luminance can render the previous frame unreliable as a prediction reference. Also moving objects cast shadows on other objects or on the background with the same undesirable effect. With multiple references available for prediction, it is more likely that at least some of the references will have the shadow and illumination conditions matching the current block being encoded.
6. Noise introduced in the signal due to camera distortion, or environmental factors. Again, with multiple references available for prediction, some of them are likely to have the noise or environmental characteristics matching the current block being encoded.

In summary, the *extension of the block codebook alphabet* with additional frames/blocks is the primary reason behind the increased performance of multiple frames, as first pointed out in [40] and [101]. More choices from which to choose can increase the probability for a better match. Still, even if infinite memory and computational resources were available, there will be diminishing returns in terms of prediction improvement vs. the cost in bits. In fact, there can be a point where inefficient entropy coding of the reference indices and the motion vectors will lead to a coding loss.

3.3 Multiple Reference Frame Prediction

Block-based motion compensation with multiple reference frames can be broken down to three major components: (a) the configuration and management of the reference frame buffer, (b) the decision on which decoded frames to store and in which positions of the reference frame buffer, and (c) the decision mechanism that selects the reference frame for MCP, which is uniquely identifiable through a reference index. Next, we base our discussion of multiple reference frames on the above partition of functionalities.

3.3.1 Reference Buffer Configuration and Management

In one category of reference buffer configuration, we consider methods that use more than one reference frame within a causal sliding window of past coded frames, including the previous frame. Such a system with, e.g., three references would predict frame n from frames $n-1$, $n-2$, and $n-3$. An advantage of this arrangement is the simplicity of its implementation with a first-in first-out (FIFO) frame buffer. These are termed *sliding-window* methods and are illustrated in Figure 3.1. The other category includes methods where the reference buffer is divided into two or more segments that may differ in their buffer management. In practice, a buffer may be divided into a short-term and a long-term segment. For example, the long-term segment could consist of past reference frames in display order, say frame $n-N$, where $N > 1$, that are intended to capture long-term video content statistics. These are termed *long-term* methods and require some *signaling* method to store and remove frames from the reference frame buffer.

The operational control of a reference frame buffer is termed reference frame buffer management. In general, buffer management has to perform three major functions: (a) add/store frames into the reference frame buffer, (b) remove frames from the reference frame buffer, and (c) assign reference indices to the references. Sliding-window approaches preceded long-term schemes. In part, sliding-window reference frame buffers benefit from a straightforward implementation of the buffer management. Obviously, functions (a) and (b) are implicitly defined by the FIFO nature of the buffer. For the reference index assignment function, a sliding-window buffer usually assigns smaller (and thus more desirable since fewer bits are needed to code them) indices to the most recently decoded frames. For the case of predicting frame n from

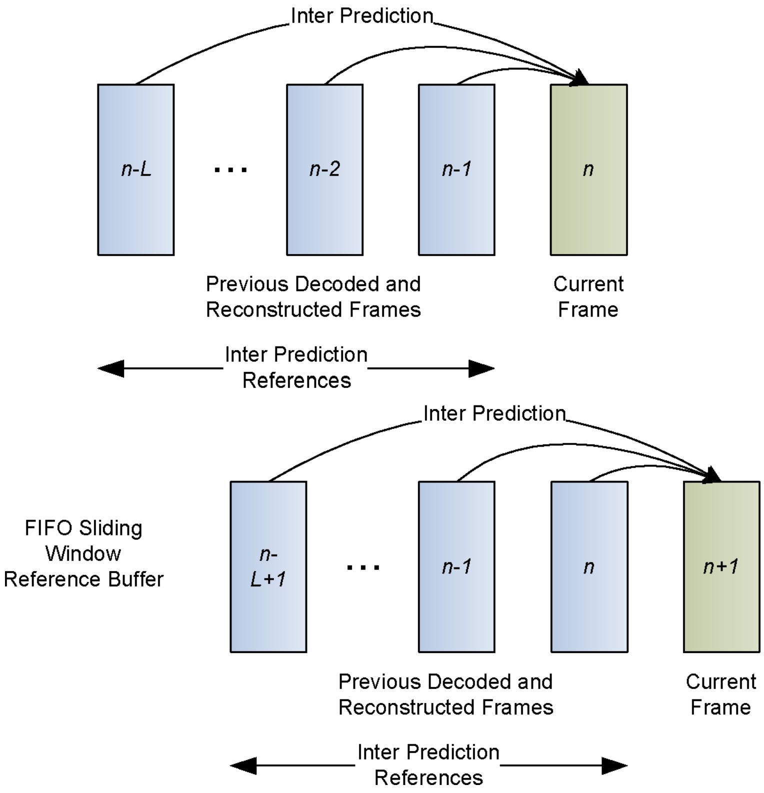

Inter Prediction
n-L
...
n-2
n-1
n
Previous Decoded and Reconstructed Frames
Current Frame
Inter Prediction References
Inter Prediction
FIFO Sliding Window Reference Buffer
n-L+1
...
n-1
n
n+1
Previous Decoded and Reconstructed Frames
Current Frame
Inter Prediction References

Fig. 3.1 An FIFO sliding-window buffer.

reference frames $n-1$, $n-2$, and $n-3$, the reference indices may be set as 0, 1, and 2, respectively. The first study for MRP [40] and the subsequent work [11] on H.263 used up to eight past coded frames in a sliding window. Similarly, in [113, 114] up to 50 past decoded frames in a sliding window were used for block-based MCP within the framework of a modified H.263 video codec.

Long-term methods were initially devised to keep the computational and memory complexity low and enable an easier theoretical formulation and efficiency analysis. The first such example used just two reference frames for prediction [35]. The previous frame in display order was buffered in the short-term frame memory (STFM), and the long-term frame in the long-term frame memory (LTFM), which was meant to serve as a background memory and was periodically updated. Suppose we are currently coding frame n, using the STFM (which contains frame $n-1$), and using the LTFM (which gets updated with every N-th frame and currently contains frame $n-N-1$). After frame n is coded, the LTFM, which has reached its limit of obsolescence, is updated to contain frame $n-1$, and the STFM is updated to the next frame, as is done after encoding every single frame, so that it now contains frame n. Then, frame $n+1$ can be encoded, with the prediction coming from frames n and $n-1$. Since the LTFM only gets updated once every N frames, frame $n-1$ will continue to be buffered in the LTFM until frame $n+N$ is coded, at which point the LTFM will be updated with frame $n+N-1$. Figure 3.2 depicts this scheme. A similar scheme was employed in [65], while in [15] every N-th frame is coded at a somewhat higher bit rate and then buffered as a high-quality long-term reference frame. In both [15, 65], the long-term reference frames are spaced evenly in a periodic fashion. Such techniques require a more comprehensive and flexible reference frame buffer management scheme. Frames that are stored in the reference frame buffer are not necessarily in some fixed pre-determined order. The buffer management has to be flexible enough to store and remove frames in some arbitrary fashion. While different methods were adopted in the research literature, there are actually now some standardized techniques for reference buffer management, which will be discussed in detail in Section 3.4.

3.3.2 Selection of Frames to Buffer as References

For sliding-window methods, the selection is implicit. The selection of the frames to be buffered as long-term was pre-determined as periodic in [15, 65]. However, performance can be further improved by actively seeking to locate and store frames that will benefit performance the most. In [90] the selection of the long-term reference frame became a function of the network conditions as the encoder is assumed to operate in a cognitive radio environment. Cognitive radio transmitters and receivers may change their transmission or reception parameters to ensure reliable communication without interfering with licensed frequency spectrum users. The adaptation of the wireless parameters is

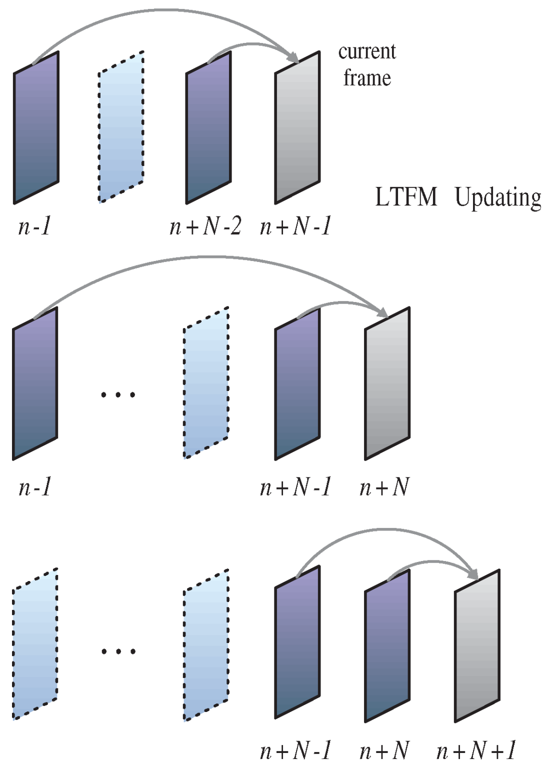


Fig. 3.2 A long-term/short-term frame memory scheme. The darker frames are the long-term and short-term reference frames. The gray frame is the current frame being encoded. The best match block can be found either in the previous frame (short-term reference) or in the long-term frame memory, or can be a linear combination (multihypothesis) of both.

based on monitoring of a series of factors that include the frequency and the state of the wireless channel (e.g., interference) among others. For such a system, it is assumed that extra bandwidth might be available opportunistically for short-time bursts. The authors studied the case where the burst duration is equivalent to one frame. The bandwidth burst was used to code the current frame at high quality and retain it as a long-term reference for a fixed number of frames, unless a new burst happened before the long-term frame was due to be replaced. A scenario with a lookahead buffer was also studied. In that variant the bandwidth burst was not only allocated to the current frame, but also to frames in the lookahead buffer.

Another approach that dispenses with even/periodic spacing of long-term reference frames was proposed in [91]. Simulated annealing (SA) was applied to solve the problem of long-term reference frame selection for archival video where the sequence is known in advance, or for real-time video with some delay (say, a 2-second broadcast delay) where some frames are known in advance. SA begins with an initial solution that in this case consisted of periodic long-term reference frames. One of the frames is randomly selected and replaced with a neighboring frame with a random temporal distance that is upper bounded by a distance constraint. The solution was accepted as the new one if compression distortion was reduced. An increase in compression distortion would still qualify to be retained as a solution with a certain acceptance probability. This process was repeated for each long-term reference position to complete one iteration of this algorithm. The acceptance probability and the distance constraint were lowered and the process proceeded to its next iteration until the distance constraint was zeroed out.

In [43], the authors proposed the use of MPEG-7 indexing metadata to select long-term reference frames for H.264/MPEG-4 AVC. A long-term frame buffer (LTFB) stores encoded frames. Each frame in the LTFB is divided into sub-blocks. The long-term reference frame buffer (LTRFB) was constructed for each frame being coded by using MPEG-7 indexing metadata to signal the sub-blocks. Each sub-block of the LTRFB is filled with a sub-block drawn from past encoded frames of the LTFB that has the minimum distance to the corresponding sub-block of the frame being coded. The distances between the sub-blocks are calculated as Euclidean distances of the available MPEG-7 color layout descriptors for each sub-block. This approach is H.264/MPEG-4 AVC-compliant when a sub-block corresponds to a whole frame.

3.3.3 Reference Selection for Motion Compensation

Management of the reference frame buffer has to be complemented with a good strategy to select the best match block, and consequently the best reference index. Different approaches were adopted to select the best match block. In [11, 15, 35, 40] the best match was selected by minimizing the SAD metric. In [114, 113], as well as in [112, 111], rate-constrained motion estimation was applied to select the MV and the temporal reference by minimizing the rate-distortion (RD) Lagrangian cost (Appendix A). For block b, spatial vector $\mathbf{v}$, and temporal reference t, the following cost $J(\mathbf{v},t|b)$ was minimized during motion search:

$$J(\mathbf{v},t|b) = D(\mathbf{v},t|b) + \lambda_{motion} \times R(\mathbf{v},t|b). \tag{3.1}$$

A further RD cost minimization is then used to select the coding mode. In [65] RD optimization was used for joint mode and reference frame selection, but not for the spatial MV as in the two-step RD selection scheme of [113]. In conclusion, the rate-constrained motion estimation approach in [114] has shown consistently good performance and has been adopted in numerous encoder implementations.

3.4 Multiple Reference Frames in Standards

In Section 3.1 we gave a brief account of the standardization of multiple reference frames. Standardization of a technology enables wider adoption while preserving the necessary interoperability, since a common and standarized reference is available to practitioners implementing the technology. In the case of multiple reference frames for motion compensation, standardization enabled the jump of this technology from academic and industry laboratories to the living room. Many of the concepts we present in this work can be implemented and tested using widely available standardized coding tools. In this section, we briefly describe standardized coding tools that involve multiple reference frames.

The first standardized coding tool that involved multiple reference prediction was Annex N “Reference Picture Selection” (RPS) of the ITU-T H.263 [104]. Annex N was primarily devised to improve error resilience, and coding gains were only an afterthought. The reference frame buffer adopts the sliding-window FIFO paradigm. Backchannel messages can be sent from the decoder to the encoder to inform the encoder of which parts from each frame have been correctly received. Doing so effectively disqualifies certain reference indices, because the encoder will choose to use only reference frames that are known to have been received correctly, so that temporal propagation of errors is avoided. The size of the reference frame buffer is set by some external messaging mechanism which is not defined by the Annex. While RPS was originally devised to help improve error resilience, it is possible to use it as a means of improving coding efficiency. For applications where error resilience is not an issue, the decoder has access to all frames stored in the reference frame buffer for MCP. The encoder can thus consider all of them during motion search. This scheme, however, is handicapped as the temporal reference index is sent for each group of blocks or slice, and may not be signaled on a block basis.

The RPS mode (Annex N) was later [107, 119] extended and improved to form Annex U "Enhanced Reference Picture Selection" of ITU-T H.263. Annex U is effectively a superset that includes all of the functionality of Annex N. Several new functionalities, primarily targeted toward improving coding efficiency, were added. Unlike Annex N, reference indices may be signaled on a macroblock basis, with substantial benefits for both compression efficiency and error resilience.

There are two distinct implementations of the reference frame buffer configuration and management. One is an FIFO sliding-window buffer control where up to M frames may be used as prediction references. The buffer contains the M most recent preceding decoded frames. The second configuration, termed "Adaptive Memory Control" provides a more flexible framework for managing the contents of the multiple reference frame buffer. These are shown in Figure 3.3. The operations for adaptive memory control allow the encoder to mark which frames or sub-picture areas will be stored in the reference buffer and which will be marked as unused.

Memory management control operations are defined that enable the manipulation of the reference frame buffer. These allow manipulation of both a short-term and a long-term reference buffer. There exist instructions to mark short-term or long-term sub-picture areas as unused. Data from other reference frames may then be stored in those unused areas. One such operation is also used to set up the size of the reference buffer and further divide it into *minimum picture units*.

Annex U not only allows extensive flexibility in modifying the contents of the short-term and long-term reference frame buffers, but it

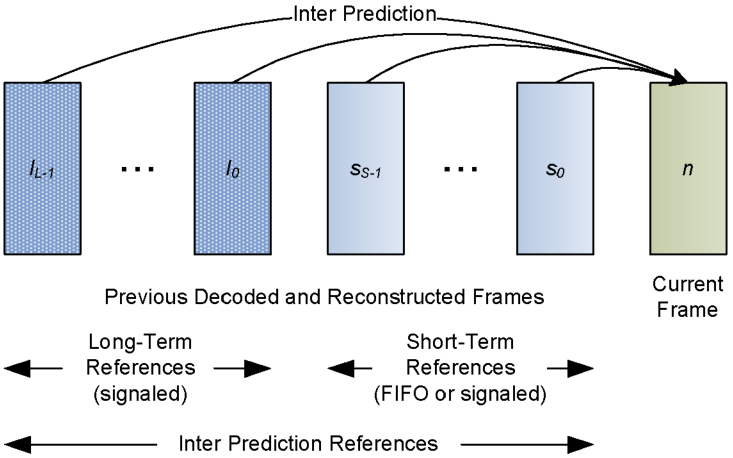

Inter Prediction
l_{L-1}
. . .
l_0
s_{S-1}
. . .
s_0
n
Previous Decoded and Reconstructed Frames
Current Frame
Long-Term References (signaled)
Short-Term References (FIFO or signaled)
Inter Prediction References

Fig. 3.3 A buffer utilizing adaptive memory control.

also allows *modifying the reference indices* of those references. There can exist cases where, e.g., frame $n - 2$ is more correlated to the current frame n than is frame $n - 1$. Traditionally, frame $n - 1$ would have been assigned reference index 0, which requires the least number of bits to signal. The encoder is thus *biased* toward using frame $n - 1$ as a prediction reference. In the case where frame $n - 2$ is more correlated with the current frame n, coding gains are possible if one were to assign reference index 0 to frame $n - 2$. This is made possible with a feature in Annex U that is termed “Remapping of Picture Numbers Indicator”. This feature allows altering the default reference picture ordering by signaling commands that remap the default indices to the intended ones. An example of this process is shown in Figure 3.4.

While multiple reference frame prediction had been standardized as Annexes of the ITU-T H.263 standard, the reality is that both that standard as well as the specific Annexes never really found widespread adoption. Rather, multiple reference frames entered the living room through the ITU-T/ISO H.264/MPEG-4 AVC standard [1]. The multiple reference support in H.264 has several similarities with the concepts adopted in Annex U. Consequently, we will describe multiple references in H.264/MPEG-4 AVC by contrasting the similarities and differences with H.263.

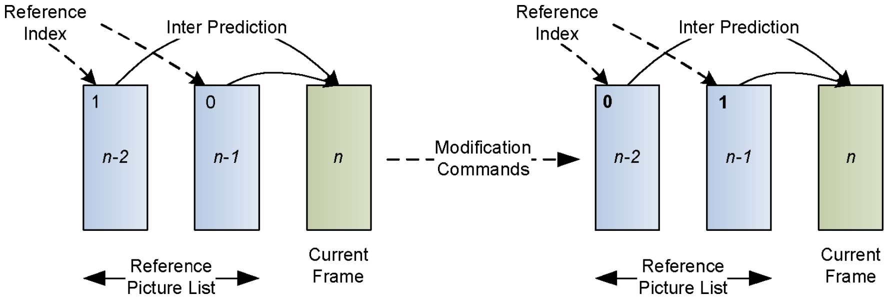


Fig. 3.4 An example of "remapping of picture numbers" as in H.263 or, equivalently, "reference picture list modification" as in H.264.

Similarities include the support for both a sliding-window buffer management model and an adaptive frame memory model, which is controlled by signaling explicit memory management control operations. Again, there is both a short-term and a long-term reference frame memory. The differences though are not trivial: the maximum size of the reference frame buffer that includes both short-term and long-term references is constrained to 16 frames, unlike H.263 where it could grow to be larger. The size of the reference frame buffer is signaled in the sequence header, in contrast to H.263 where it is signaled using a memory management control operation. A pre-determined process constructs a *reference picture list* given in the frames stored in the reference frame buffer. This list contains the reference indices used for MCP. For B-coded frames there are *two* reference picture lists. More details on B-coded frames can be found in Section 4.1.

A groundbreaking difference is the removal of the rigid connection between frame types and references that exists in H.263, where one I or P-coded frame gives rise to one reference frame, and B-coded frames are not used as references. This is no longer true in H.264/MPEG-4 AVC. Any type of coded frame can be used, or not used, as a reference. I- and P-coded frames can be designated as non-reference or "disposable" frames. Another difference involves the Remapping of Picture Numbers Indicator signals in H.263 that are also found in H.264/MPEG-4 AVC as part of the "Reference Picture List Modification." These standardized signals can accomplish all that the H.263 remapping signals did, and at the same time also the *replication* of references. It is possible to assign multiple temporal reference indices, e.g., 0 and 1 to the same reference frame in the reference frame memory. While this may seem counter-intuitive, the reasons for allowing such flexibility will become obvious when describing the implementation of weighted prediction in the standard in Section 3.6.

Last, we note that in the state-of-the-art H.264/MPEG-4 AVC standard, P-coded frames reference any buffered past coded frame regardless of its display order. Hence P-coded frames can also reference future frames in display order. B-coded frames (Section 4.1) allow biprediction (weighted linear combination of two predictions) from any past coded frames. This means that it is possible for the two predictions to originate from the same frame, from a future and a past frame, or from two past frames or from two future frames (in display order).

3.5 Interpolation for Motion Compensated Prediction

In Section 1.1, we mentioned that a motion-compensated block may undergo additional spatial processing before being used as a prediction block. We elaborate on this here and relate spatial processing to multiple reference frames. In early video coding standards and video codecs, motion vectors could only take integer values. The physical meaning of integer-valued motion vectors is that the prediction values are samples from the reference frame that are simply translated according to the motion vector horizontal and vertical components. Early video coding standards such as ITU-T H.120 [24] and ITU-T H.261 [103] employed integer motion vectors. While video capture is done digitally at integer precision, the motion in the video signal does not have to obey an integer model. It can be arbitrary and can correspond to spatial displacements that are not of integer precision. Coding performance can thus be improved by considering motion vectors with fractional-pixel precision [37]. The ISO MPEG-1 standard [51] was the first international video coding standard to adopt fractional pixel motion compensation. Half-pixel motion vectors were used to augment motion compensation. The samples corresponding to the half-pixel positions were interpolated using bilinear interpolation in MPEG-1.

Quarter-pixel motion vectors were first introduced as an optional interpolation mode in ISO MPEG-4 Part 2 [53]. Quarter-pixel motion compensation is also a part of the SMPTE VC-1 [82] and the ISO/ITU-T H.264/MPEG-4 AVC [1] coding standards. In VC-1, quarter-pixel interpolation was realized through either bicubic or bilinear interpolation. In contrast, in H.264/MPEG-4 AVC a two-tiered approach was adopted that first obtained the luminance half-pixel samples using a six-tap Wiener filter with coefficients $[\ 1 \quad -5 \quad 20 \quad 20 \quad -5 \quad 1\ ]$ applied on existing integer-pixel samples. For each integer-pixel sample, three half-pixel samples are derived: one with vertical interpolation from integer-pixel samples, a second with horizontal interpolation from integer-pixel samples, and a third with horizontal interpolation from the half-pixel samples derived at the previous step. The filter output h_w is then rounded and normalized as $h = (h_w + 2^4) >> 5$. In a second step, the quarter-pixel samples are generated by applying bilinear interpolation on existing integer-pixel samples and the already interpolated half-pixel samples. Integer and half-pixel precision samples a and b are processed to obtain the quarter-pixel samples q as: $q = (a + b + 1) >> 1$. Note that interpolation operations in H.264/MPEG-4 AVC always *round away from zero*. For example, bilinear interpolation of a sample from values 1 and 2 will result in 2: $(1 + 2 + 1) >> 1 = 2$.

The coding efficiency gains due to fractional-pixel MCP have been traditionally [37] interpreted as a function of two factors: (a) the increase in the motion vector precision that ensures that the motion vector can better model the true spatial displacement and thus reduce the prediction error and (b) the interpolation filter used to generate the fractional-pixel samples. Fractional-pixel motion compensation can also be interpreted as multiple reference frame prediction. Fractional-pixel prediction requires the interpolation of the otherwise unavailable sample values at the fractional-pixel positions. We note that for a given fractional-pixel motion vector, the samples constituting the prediction block will be generated with a process that is unique to the fractional components of the motion vector. For example, samples indexed by vectors $(1.25, 2.0)$ and $(16.25, 0.0)$ will be generated in the same way as they both *belong* to the subset of samples that correspond to the

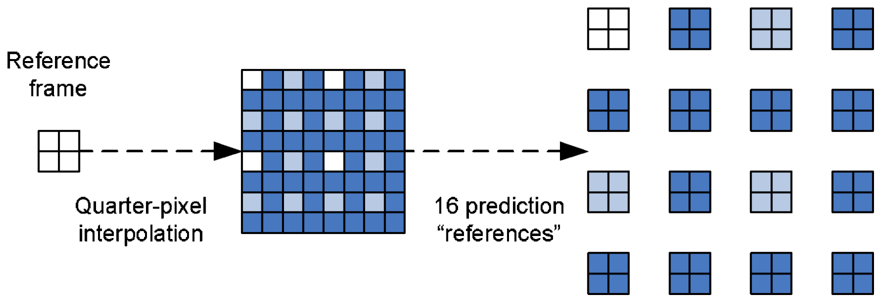


Fig. 3.5 Multiple reference interpretation of interpolation.

$(0.25, 0.0)$ sub-pixel spatial displacement. For quarter-pixel precision, for each integer-pixel sample, one generates an additional $4 \times 4 - 1$ samples. Each of these samples corresponds to a unique fractional component of the motion vector. If one were to gather all samples in the interpolated frame that correspond to a specific sub-pixel displacement, this gives rise to a reference frame that corresponds to those sub-pixel displacements. This is shown in Figure 3.5.

Since there are 16 possible displacements for luma samples in H.264/MPEG-4 AVC, this gives rise to 16 distinct reference frames that are generated from a single decoded frame and are available for MCP. These 16 references are addressable through the fractional component of the motion vectors. One could thus alternatively formulate fractional-pixel motion compensation as a multiple-reference scheme. Such a multiple reference formulation is characterized by a fixed interpolation strategy given the motion vector sub-pixel components.

Better performance however is possible if one were to test multiple interpolation strategies for each sub-pixel displacement. Multiple sets of references could be created, each one corresponding to a different interpolation strategy and each one giving further rise to 16 sets of references corresponding to the 16 fractional-pixel displacements. Certain types of content would benefit from using filters with long support while other types of content would prefer filters with short support. Similar arguments hold for different types of filters such as bicubic vs. bilinear. Such a method that considers additional references derived with multiple interpolation strategies has been proposed in [9].

3.6 Weighted Prediction and Multiple References

Illumination changes can present challenges for MCP. Even if there is little motion from one frame to the next, the increase or decrease in illumination will lead to inefficient prediction of the current block. Such changes can happen during a cross-fade (a fading transition from one scene to the next), a fade-in (a transition that starts from some uniform color, e.g., black, and fades in the start of a scene), a fade-out (a fading transition from a scene to some uniform color), and during a local illumination change such as a camera flash, among others. Weighted prediction was proposed to improve the efficiency of block-based MCP in such cases. H.264/MPEG-4 AVC is the first international coding standard to adopt weighted prediction in its specification.

Weighted prediction in H.264/MPEG-4 AVC may optionally be applied after the derivation of the sub-pixel prediction samples, which was described in Section 3.5. Let $\hat{f}_n(i,j)$ denote the MCP of sample $f_n(i,j)$ after the derivation of the sub-pixel samples and before the application of weighted prediction. Let w and o denote a gain and an offset parameter, respectively. The gain and offset parameters are also known as the weighted prediction parameters. The weighted prediction $\hat{f}_{WP,n}(i,j)$ of sample $f_n(i,j)$ will be written as:

$$\hat{f}_{WP,n}(i,j) = w \times \hat{f}_n(i,j) + o. \tag{3.2}$$

Pictures in H.264/MPEG-4 AVC are coded using independently decodable slices. Slices that are constrained to intra-frame coding are called I slices, while slices that use inter-frame coding in addition to intra-frame coding are called P and B slices. B slices are a superset of P slices that additionally allow bi-prediction: constructing a prediction block as a linear combination of two prediction blocks. More information on bi-prediction follows in Section 4. Weighted prediction is switched on and off at the slice header level. For P slices, weighted prediction in H.264/MPEG-4 AVC is implemented as:

$$\hat{f}_{WP,n}(i,j) = \left\lfloor \frac{w \times \hat{f}_n(i,j) + 2^{\log WD - 1}}{2^{\log WD}} \right\rfloor + o. \tag{3.3}$$

When $\log WD < 1$, weighted prediction takes the form of Equation 3.2. Parameter $\log WD$ is transmitted in the bit stream and controls the precision of weighted prediction since both w and o are defined as integer numbers. Weighted prediction for B slices is defined differently and we will defer describing it until after introducing multihypothesis motion compensated prediction in Section 4.

Weighted prediction in H.264/MPEG-4 AVC has two modes: *implicit* and *explicit*. For the implicit mode, the weighted prediction parameters are not explicitly transmitted in the bit stream but they are rather inferred using bit stream information such as the *picture order count*, which orders frames according to their output order and is usually associated with their display order. For the explicit mode, the weighted prediction parameters are transmitted in the slice header and are not tied to the reference frames in the short-term and long-term reference frame memories. Instead, a set of weighted prediction parameters is transmitted for each reference in the reference picture list for the current slice. Recall that the references in the reference picture list are decoupled from the frames available for use as prediction references in the reference frame buffer and that the reference picture list for the current slice is finalized *after* processing all Reference Picture List Modification commands in the slice header. The default reference picture list would assign a unique index to each frame in the buffer. However, in Section 3.4 we discussed that these commands may be used to assign multiple reference indices to the same reference frame. Since separate gains and offsets are sent for each reference index in the list, one can assign different sets of weighted prediction parameters to each reference. In such an example, references 0 and 1 point to the same reference frame 0 and the former is assigned parameters WP_0 and the latter parameters WP_1. This way one can circumvent the limitation of the H.264 standard that does not allow signaling of the parameters at the MB level through the signaling of different reference indices for each MB (down to an 8×8 partition). Such a strategy may benefit several coding situations, such as predicting local illumination changes, where a single offset and gain for a reference would not be optimal when considering the entire frame. It may also be used to emulate different interpolation filters and to account for the rounding implementation in H.264/MPEG-4 AVC that is biased away from zero, as discussed in Section 3.5. Such a solution for the problem of zero biasing was demonstrated in [66].

3.7 Scalable and Multiple-View Coding

So far we have discussed multiple references for MCP that can be available as a result of: (a) buffering multiple decoded frames, (b) interpolation of decoded frames, and (c) weighted prediction of decoded frames. Multiple references can also result from accessing predictions from a separate layer of the compressed video bit stream. This is possible when considering the paradigm of scalable video coding. Traditional hybrid video compression results in a bit stream that when decoded will reconstruct the source sequence at a given temporal resolution (e.g., frame rate), spatial resolution, and quality level or bit rate. A layer denotes a portion of the bit stream that needs to be decoded to give rise to a specific frame rate, spatial resolution, and bit rate. One of the layers, the *base*, can always be decoded independently of all other layers. In contrast, all other layers, the *enhancement* layers, depend on the base as well as on other previously decoded enhancement layers for correct decoding. A bit stream that contains multiple layers that result in different frame rates, spatial resolutions, or bit rates (i.e., quality) is called a scalable bit stream. A codec that can produce and interpret such a bitstream is called a scalable video codec. Codecs that cannot produce such bit streams are called single-layer codecs.

Scalability is in general a desirable trait in a coding system: instead of coding a bit stream at, e.g., 15 frames per second, and a second one at 30 frames per second, it is more efficient in terms of storage and flexibility to have access to a single bit stream that can give rise to both temporal resolutions. This *temporal* scalability has been available in most modern video codecs such as MPEG-2 and H.264/MPEG-4 AVC (conforming to profiles specified in Annex A), as both codecs support the notion of disposable frames. Disposable frames are frames that are coded and used for display but are not used for motion compensated prediction of subsequent coded frames. If one out of every two frames is coded as disposable, then one can simply discard the disposable half of the frames and still be able to decode the image sequence at half the frame rate. It may also be desirable to extract multiple spatial resolutions from a single bit stream: a low-resolution version for mobile devices and a larger resolution (e.g., twice the resolution in each dimension) for a large display. Such a capability is often referred to as *spatial* scalability. Last, there are applications that would benefit from a bit stream that could give rise to streams with different bit rates. One such application is Internet streaming where network conditions can vary unexpectedly and a server could switch to a lower bit rate stream. While one could encode multiple bitstreams and switch among them, a single bit stream solution is more flexible and can lead to better quality since it can be designed to avoid mismatches when switching from one bitstream to another. This capability is often termed *SNR* (signal-to-noise ratio) or quality scalability.

We note here that SNR scalability, albeit desirable, does not come for free. Compression efficiency suffers: an SNR-scalable bit stream that gives rise to bit rates A and $B > A$ will in general result in lower quality than a single-layer bitstream coded at bit rate B. Similar arguments hold for spatial scalability. Temporal scalability is a functionality of the majority of modern video codecs, and the coding loss it incurs is considered trivial compared to the benefits. Benefits, apart from scalability, also include fast random access, and digital video recorder functionality [96] (e.g., rewind and fast-forward), among others.

Spatial and SNR scalability have been studied in the past and became parts of standards but never found widespread use. Recently, spatial and SNR scalability extensions were adopted as Annex G ("Scalable Video Coding") of the H.264/MPEG-4 AVC video coding standard [1]. For SNR scalability, the inter prediction of the current enhancement layer considers decoded enhancement layer reference frames. To further improve coding efficiency, inter prediction may also include decoded base layer reference frames, termed "base reference pictures". While these frames are not used for display, they can be beneficial as prediction references. In such a case, we predict the enhancement layer from the base layer. This is often termed as *inter-layer* prediction. It is worth noting that the consideration of the base reference pictures for multiple reference MCP was facilitated through simple

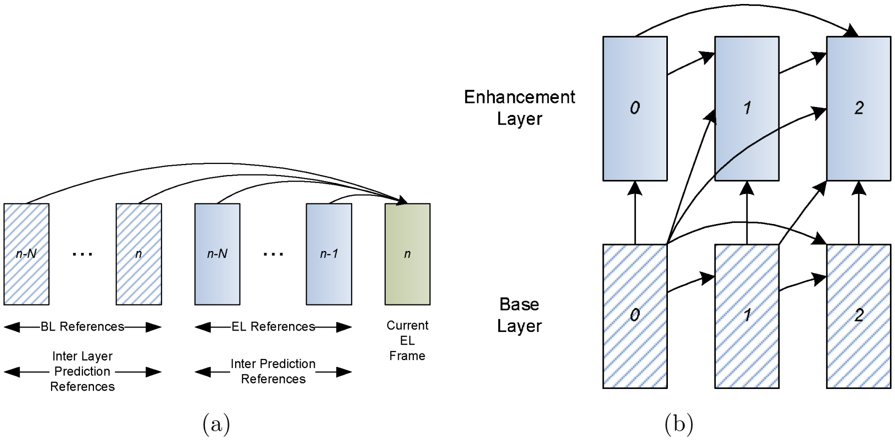


Fig. 3.6 Scalable coding with a base and an enhancement layer. (a) Reference picture list, and (b) prediction structure.

modifications of the H.264/MPEG-4 AVC reference picture list generation process. An example is shown in Figure 3.6.

A similar scheme was applied for the recently finalized "Multiview Video Coding" (Annex H) extension of the H.264/MPEG-4 AVC standard [102]. In such content, there are multiple image sequences, each one corresponding to one camera view. Such an example is stereoscopic 3D content where each time instance gives rise to a left and a right view. One may compress such a multiview image sequence as separate image sequences, e.g., using H.264/MPEG-4 AVC or MPEG-2. However, in practice there is considerable *inter-view* correlation that can be exploited to improve coding efficiency. Multiview coding as implemented in Annex H bears similarities to SNR scalability in Annex G as there is a "base layer" that consists of the frames of a "base" view. Pictures of the rest of the views may be coded with reference to that base view, or to other already decoded views.

Recall that bitstreams conforming to Annex A of H.264/MPEG-4 AVC allowed reference or non-reference frames. Annex G allowed in addition base reference frames for prediction of enhancement layers when coding for SNR scalability. In Annex H, "inter-view reference pictures" and "inter-view only reference pictures" are considered for inclusion in the reference picture list. Previously decoded frames

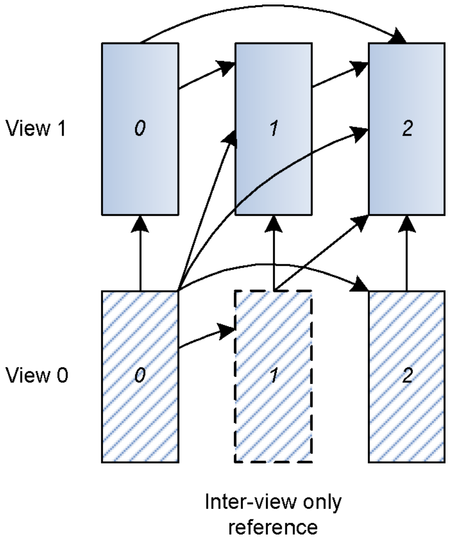


Fig. 3.7 Inter-view prediction. The inter-view only picture is not used for prediction of frames belonging to the same view.

belonging to different views that were marked as reference pictures for inter prediction may be included in the reference picture list as inter-view reference pictures. Furthermore, previously decoded frames belonging to different views that were *not* marked as references for inter prediction may be included in the reference picture list as inter-view only reference pictures. An inter-view only reference picture is used for inter-view prediction for subsequently decoded views of the same time instance, but not for inter prediction of frames from other time instances within the same view. Inter-view reference pictures, on the other hand, may be used both ways: for inter-frame coding of frames from other time instances within the same view, and for inter-view prediction. This is shown in Figure 3.7.

4

Multihypothesis Motion-Compensated Prediction

In multihypothesis (MH) motion-compensated prediction (MCP), the prediction block is a weighted average (superposition) of two or more blocks, known as the *hypotheses*, which can belong to distinct reference frames. Such a scheme is illustrated in Figure 4.1.

A general expression for MHMCP for predicting pixel (i, j) in frame f_n from N hypotheses is given as:

$$\hat{f}_n(i,j) = \frac{1}{\sum_{k=1}^{N} w_k} \left(\sum_{k=1}^{N} w_k \times f_{n-r_k}(i + v_{x,k}, j + v_{y,k}) \right). \tag{4.1}$$

Terms w_k and $n - r_k$ denote the weights and reference frames indices corresponding to each hypothesis k. Terms $v_{x,k}$ and $v_{y,k}$ denote the horizontal and vertical displacements for hypothesis k. MHMCP can be thought of as a general case of multiple reference prediction since the prediction not only originates from multiple reference frames but is also a linear combination of such prediction blocks. Next, we provide insight on the origin, development, and current state-of-the-art in MHMCP.

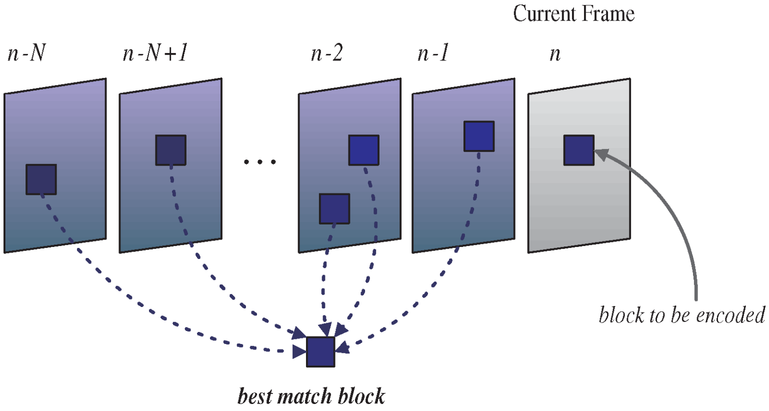


Fig. 4.1 Multihypothesis prediction. The best match block is obtained by superimposing an arbitrary number of blocks from an arbitrary number of (past or future in display order) frames.

4.1 Bi-Directional Prediction and Generalized Bi-Prediction

An early form of two-hypothesis prediction was bi-directional interpolative prediction for skipped frames [75]. It quickly became obvious that bi-predictive interpolation was not only useful for interpolation of missing frames, but also as an additional prediction mode for inter-frame coding, if it is followed by transform, quantization, and entropy coding of the prediction residual. Such an approach was later standardized as bi-directional coding (B-coded frames) in MPEG-1 [80]. In bi-directional coding, a block in the current frame is predicted as the equally weighted superposition of a previous and a future frame block in display order. Let $n - r_1$ and $n - r_2$ denote the temporal indices of the past and future reference frames in display order. Let $(v_{x,1}, v_{y,1})$ and $(v_{x,2}, v_{y,2})$ denote the spatial motion vector components for the past and future reference frames in display order. Let $\hat{f}_{n,m}(i,j) = f_{n-r_m}(i + v_{x,m}, j + v_{y,m})$ denote the prediction value for pixel (i,j) of frame n from reference frame $n - r_m$. The value of pixel (i,j) in a prediction block for frame f_n is estimated as:

$$\hat{f}_n(i,j) = \left\lfloor \frac{\hat{f}_{n,1}(i,j) + \hat{f}_{n,2}(i,j) + 1}{2} \right\rfloor . \tag{4.2}$$

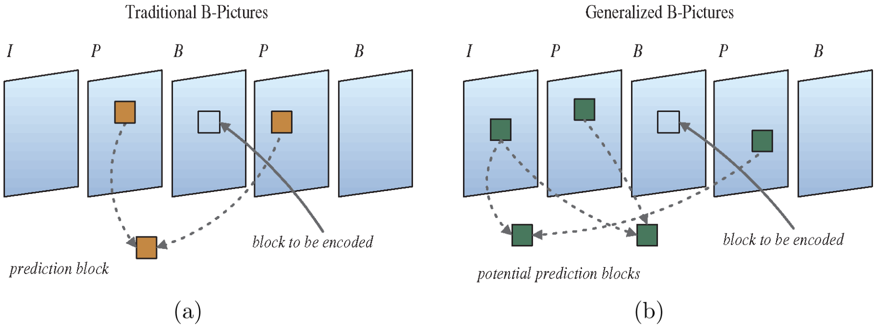


Fig. 4.2 Traditional and generalized B-coded frames. (a) In traditional B-coded frames, the block is predicted using an unweighted average of a block in the previous I/P-coded frame and a block in the subsequent I/P-coded frame. (b) In the generalized version found in H.264/MPEG-4 AVC, the prediction block can be a *weighted* average of any two blocks from any number of reference frames from any direction. Constraining to use blocks from past frames in display order alone eliminates the delay problem that characterizes traditional bi-directional coding.

Traditional bi-directional coding, as implemented in video coding standards such as MPEG-1, MPEG-2, MPEG-4 Part 2, and VC-1 may use the closest past I/P-coded reference frame and the closest future I/P-coded reference frame in display order. An example is illustrated in Figure 4.2(a).

Generalized two-hypothesis prediction as standardized in H.264/MPEG-4 AVC [32] is illustrated in Figure 4.2(b). Generalized B-coded frames allow referencing any two blocks in any of the decoded pictures (I, P, or B), *regardless* of the display order, that have been stored in the short-term or long-term reference frame buffers. It is also possible that the two hypothesis blocks may originate from the same reference frame. Furthermore, since no constraints are imposed on the display order of the reference frames, the term bi-directional prediction is no longer suitable and the term bi-prediction is often used instead. The two hypothesis blocks are linearly combined to yield the final prediction block. In the default prediction configuration, the final prediction block is obtained as in Equation 4.2 but with a critical difference: references $n - r_1$ and $n - r_2$ are now unconstrained. Each hypothesis block may also be assigned an arbitrary gain and offset when weighted prediction is enabled. Again, let $\hat{f}_{n,m}(i,j) = f_{n-r_m}(i + v_{x,m}, j + v_{y,m})$ denote the prediction value for pixel (i,j) of frame n from reference frame $n - r_m$. The weighted bi-predicted block for the case of explicit weighted prediction is written as:

$$\hat{f}_n(i,j) = \left\lfloor \frac{w_0 \times \hat{f}_{n,0}(i,j) + w_1 \times \hat{f}_{n,1}(i,j) + 2^{\log WD}}{2^{\log WD} + 1} \right\rfloor \tag{4.3}$$

$$+ \left\lfloor \frac{o_0 + o_1 + 1}{2} \right\rfloor \tag{4.4}$$

Video coding standards prior to H.264/MPEG-4 AVC were constrained to a very rigid prediction structure. Apart from the obvious IIII and IPPPP structures, where in the former all frames are coded with intra-frame coding, and in the latter all frames apart from the first are coded as P-coded frames, there was little flexibility with respect to B-coded frames. In addition, B-coded frames were constrained to referencing only I and P-coded frames. These two references were also constrained to be preceding and following the B-coded frame in display order. In H.264/MPEG-4 AVC, it was decided to allow any type of coded frames (I, P, or B) to be designated as a reference frame for MCP, and also to remove the rigid concept that bi-prediction should involve one preceding and one following frame. The flexibility that is possible by these two design decisions is substantial and gives rise to a host of possible prediction structures. For example, one could use two-hypothesis prediction for every coded frame or one could use a hierarchical frame coding structure where there are multiple levels of hierarchy. One such coding structure is shown in Figure 4.3. These structures can have many benefits, such as very granular temporal scalability, random access, and also very good compression efficiency, especially for stationary content.

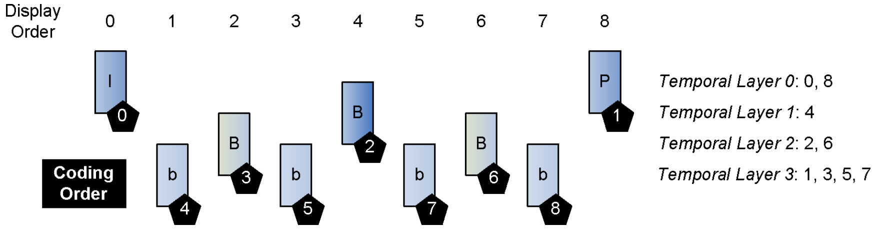


Fig. 4.3 A sample hierarchical prediction structure. The numbers in the black pentagons indicate the coding order.

4.2 Overlapped Block Motion Compensation

MH prediction may involve overlapping blocks from the same reference frame. One such scheme is overlapped block motion compensation (OBMC) [76, 77, 85, 106] that was initially devised to suppress blocking artifacts. The introduction of OBMC was motivated in part by the inability of a single MV to correctly represent motion throughout the block. MVs of neighboring blocks are correlated with the actual motion of pixels in the current block. Hence, in the OBMC paradigm, motion vectors of neighboring blocks contribute to the estimation of pixel values in the current block. To this end, the motion vector of the current block along with the motion vectors of several of its neighbors are used to obtain a number of hypothesis blocks from the previous frame. Weighting coefficients are used to combine those hypotheses and obtain the final prediction block. OBMC can be expressed as a special case of Equation (4.1), where the weights w_k now depend on the pixel coordinates (i,j) and parameter r_k is constrained to be the same irrespective of k.

Optimal OBMC [77, 85] entails estimating all neighboring motion vectors jointly since the effectiveness of the motion vectors is now intertwined. Joint estimation of these hypotheses is very costly as the possible combinations are numerous. Furthermore, optimal estimation of motion vectors for OBMC of the current block entails estimating MVs for neighboring blocks. These MVs however, while efficient for the current block, may not be appropriate for the neighboring blocks. There is thus a causality issue: it is not enough to optimize motion vectors for a current block but one needs to jointly optimize motion vectors in all blocks in an image. Hence, high-performance OBMC increases motion estimation complexity considerably.

However, even if the MVs are not jointly optimized, compression efficiency benefits since OBMC attenuates the error within the signal more efficiently than traditional single-hypothesis motion compensation. OBMC, however, suffers from increased decoder complexity. For an OBMC scheme that considers apart from the current block the top, bottom, left, and right blocks, five hypotheses have to be multiplied with weights, then summed, and finally normalized. The smoothing window

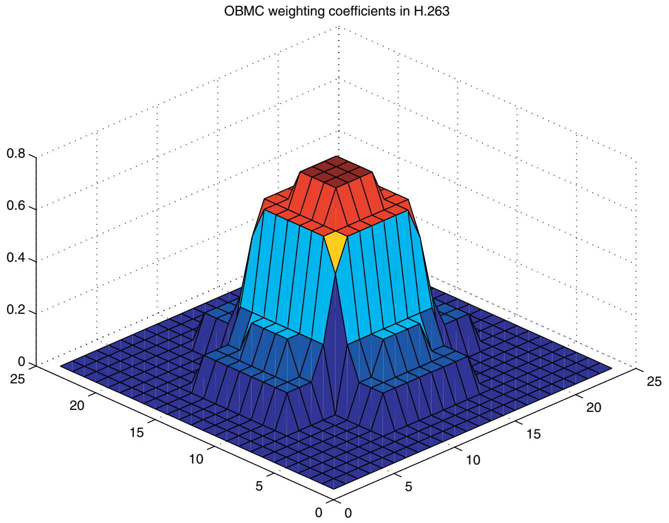


Fig. 4.4 Overlapped block motion compensation smoothing window for H.263. The window is cross-shaped and non-zero for five 8 × 8 prediction blocks, the center and its top, bottom, left, and right prediction blocks. The center block is the *current* 8 × 8 motion-compensated prediction block. The remaining four blocks are the motion compensation prediction blocks that are produced using the MVs of the four neighboring blocks.

standardized for the “Advanced Prediction Mode” of H.263 can be seen in Figure 4.4. OBMC was included in the original specification of the MPEG-4 Part 2 video coding standard, but was never commercially implemented due to the exclusion of OBMC from all MPEG-4 Part 2 profiles. In practice, implementing a standard has become synonymous with implementing a profile of the standard. Quite possibly, the high decoding complexity was instrumental in the exclusion of OBMC from all MPEG-4 Part 2 profiles.

Overlapped motion compensation in Annex F (Advanced Prediction Mode) of H.263 [104] is only applied to the luminance samples. The color component content is usually characterized by such low frequencies that OBMC can do little to improve the prediction performance over single-hypothesis prediction. Each pixel in an 8 × 8 prediction block is estimated as the weighted sum of three prediction values. The three prediction values are obtained through three sets of MVs. One is the motion vector of the current luminance block and the other two are: (a) the MV of the block on the left or right side of the current luminance block, and (b) the MV of the block above or below the current luminance block. For each pixel, the motion vectors of the blocks at the two nearest block borders are used. Hence, to predict the entire block, five sets of MVs are required. The three prediction values that correspond to each motion vector set are further weighted given the nature of the motion vector (current, top/bottom, or left/right) and the position of the pixel in the 8 × 8 prediction block.

4.3 Hypothesis Selection Optimization

In a system such as the one shown in Figure 4.2(b), hypothesis selection can be complex. Hypothesis selection is the process by which the codec decides *how many* hypotheses to superimpose, *how many* and *which frames* to search for each one of them, how to *jointly* optimize their selection, and finally, what *weighting* coefficients to apply during the superposition. To optimally estimate each hypothesis, one has to test every possible combination of the many degrees of freedom. Hence, there is a need for practical and efficient hypothesis selection algorithms.

One of the first known algorithms for joint iterative estimation of forward and backward MVs for bi-directional prediction (two-hypothesis selection) was proposed in [115]. The algorithm consisted of the following basic steps:

(1) The optimal forward and backward MVs are estimated using prediction from the past and future frames, respectively.
(2) The backward MV is fixed and a search is conducted to refine the forward MV using bi-directional interpolative prediction.
(3) The forward MV is fixed and a search is conducted to refine the backward MV using bi-directional interpolative prediction.
(4) Steps (2) and (3) are repeated until the prediction error stops decreasing (converges).

Note that equal weighting was used throughout the above process. The above algorithm can improve prediction performance considerably. Still, there are many cases, such as in [35], where for simplicity the two hypotheses were estimated independently and were averaged to produce the prediction block. That work involved a superposition of two blocks from past frames, one from the short-term frame memory and one from the long-term frame memory.

Hypothesis selection for unconstrained MH motion-compensated prediction was treated in [33], where the algorithms proposed in [77, 115] were extended to handle multiple frames in arbitrary prediction directions. Hypothesis selection was again performed in an iterative fashion. Three components have to be determined: the hypotheses, the superposition weights, and the entropy codewords that signal the selection of each hypothesis. After initialization, three steps are iterated until convergence:

(1) determination of multihypotheses given weights and entropy code;
(2) determination of entropy code given multihypotheses and weights; and
(3) determination of weights given multihypotheses and entropy code.

The hypothesis selection algorithm, which was introduced in that work, tackled Step (1), while the predictor weighting coefficients were solved adopting methodology from the Wiener problem. Linear combinations of up to four hypotheses were evaluated. It was established that the joint determination of the hypotheses and their weighting coefficients is a daunting task for more than two hypotheses.

By constraining the problem to just two hypotheses, selected from multiple past references, the hypothesis selection algorithm was greatly simplified in [34], and was essentially a generalization of the algorithm in [115] for arbitrary hypotheses. It was also shown in [30] that the gain for two jointly optimized hypotheses is close to the theoretical limit, and that the gains from MH and multiple reference prediction add up to *more* than their sum.

Last, the determination of efficient weights for the superposition is a critical part of MH prediction. Adaptive weighting coefficient selection was investigated in [85] and [77]. For the more complex case of [33], a Lagrangian approach was used to obtain the coefficients, and it was found that after extensive training with real data the final weights approximated $\frac{1}{N}$ for N hypotheses, regardless of initialization. Hence in [34] equal weights were applied. However, a Wiener filter was used to obtain the coefficients in the analysis of [31].

4.4 Multihypothesis Prediction in the Frequency Domain

All schemes described above perform MH prediction in the spatial-domain. Spatial-domain techniques deal with the direct manipulation of pixels in an image. In [25] it was proposed to transform the frames employing a redundant discrete wavelet transform (DWT) and then apply MH prediction in the wavelet domain. Traditional DWT applied on a signal yields two subbands, the *low-pass* and the *high-pass*, each of which has half of the samples of the original signal due to *critical sampling*. However, critical sampling results in wavelet coefficients that are highly dependent on their location in the sampling grid. Small shifts in the input frame cause large changes in wavelet coefficients and possibly large changes in reconstructed frames. The DWT is hence *shift-variant*. Compared to the traditional DWT, the redundant DWT (RDWT) forfeits the latter's subsampling operation to produce an overcomplete representation of multiple hypotheses that are diverse in frequency content. Such a scheme is depicted in Figure 4.5. The lack of sub-sampling renders the RDWT shift-invariant since the spatial sampling rate is now fixed across subbands. Shift-invariance facilitates MH prediction. A hierarchical search is used for ME among different phases, minimizing a cross-subband distortion metric, in contrast to the rate-distortion cost functions used in [33, 34]. An inverse RDWT is then performed on the RDWT-domain motion-compensated frame, combining the multiple subbands into a spatial-domain MH prediction. This spatial-domain prediction is subtracted from the current frame and the residual is coded.

4.5 Theoretical Insight

A theoretical analysis of MHMCP was presented in [39]. Several important conclusions were drawn that provide insight into designing

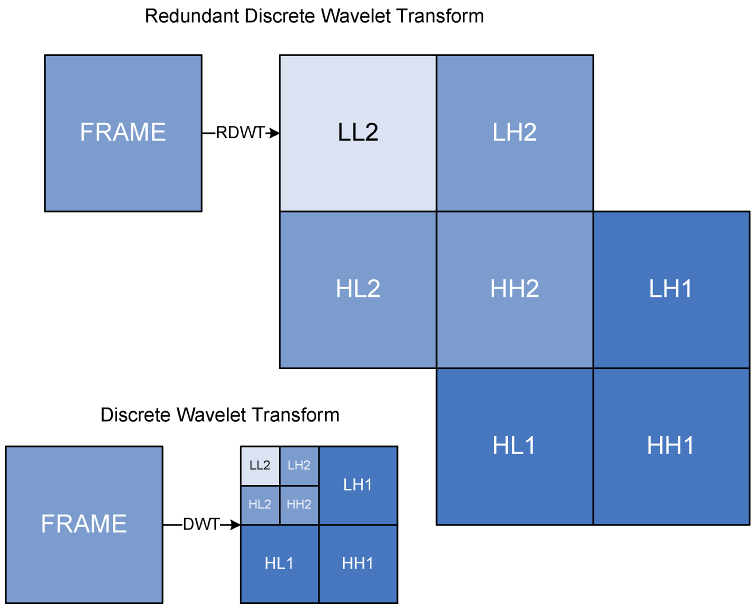

Redundant Discrete Wavelet Transform
FRAME
RDWT
LL2
LH2
HL2
HH2
LH1
HL1
HH1
Discrete Wavelet Transform
FRAME
DWT
LL2
LH2
HL2
HH2
LH1
HL1
HH1

Fig. 4.5 Redundant Discrete Wavelet Transform. Compared to the traditional DWT, we have signal expansion. However, it helps provide multiple hypotheses that view the signal from multiple spatial frequencies.

efficient multihypothesis MCP schemes:

(a) An optimal combination of N hypotheses improves compression efficiency for increasing N. For realistic signals, however, increasing N yields diminishing returns in terms of improved prediction performance vs. the increase in the bit cost necessary to signal the hypotheses.

(b) Doubling the precision of motion compensation, such as going from integer-pixel to half-pixel precision, improves compression efficiency. However, again for realistic signals, doubling the number of hypotheses N is more effective than doubling the precision of motion compensation, i.e., B-coded frames or OBMC provide a *larger* gain than half-pixel MVs.

(c) Fractional-pixel motion compensation becomes less important with MHMCP.

(d) Spatial filtering of the motion compensated candidate signal becomes less important if multiple hypotheses are combined.

The main mechanism behind the improved performance of MHMCP is the reduction of the residual noise level, accomplished by averaging multiple predictions similar to temporal averaging techniques used in imaging sensors to reduce noise. Furthermore, in [31], it was shown that the coding gain of MH over traditional MCP is independent of the motion compensation precision for non-band-limited signals. We note though that the above findings are based on theoretical assumptions and simplifications. In Section 8, a few of the above conclusions are tested experimentally.

5

Fast Multiple-Frame Motion Estimation Algorithms

The most computationally intensive component of a practical video compression system is motion estimation. A straightforward application of motion search on a 33×33 pixel search area, where vectors range from -16 to $+16$ in each dimension, evaluates the sum of absolute differences/sum of squared differences (SAD/SSD) metric for a pair of blocks 1089 times. Such a method is called full search (FS). Even for single frame prediction, there has been a considerable body of research devoted to fast motion vector searches to reduce encoder complexity. Of course the need to reduce complexity is compounded for MRP schemes, because searching naively over N frames in an MRP scheme increases computational complexity by N times. One option is to extend fast algorithms designed for single frame prediction, but this does not exploit temporal correlation. Ideally one wants to constrain complexity in the entire three-dimensional (3D) (spatial and temporal) space. Fast motion search algorithms for multiple reference motion estimation belong to the following main categories:

1. Multiresolution and hierarchical search.
2. Fast search using mathematical inequalities.

3. Motion information re-use and motion composition.
4. Simplex and constrained minimization.
5. Zonal and center-biased algorithms.
6. Fractional-pixel texture shifts.
7. Content-adaptive temporal search range.

Some of these categories were proposed in [3]. Several of the described algorithms that follow overlap more than one category. Rate-constrained motion estimation (ME) is again the preferred algorithm for estimating the MVs. Assuming a translational motion model, together with the spatial MV parameters v_x and v_y, we need to estimate the optimal reference index ref. The distortion for the tested motion parameters is $D(v_x,v_y,ref)$. The bit rate usage R must also account for the bits needed to signal the indicated reference frame, and depend on the same parameters. Lagrangian minimization is then applied by minimizing the cost J to obtain the optimized parameters $(v_x^\star,v_y^\star,ref^\star)$ as $(v_x^\star,v_y^\star,ref^\star) = arg\min_{(v_x,v_y,ref)} J(v_x,v_y,ref)$, where the cost is written as:

$$J(v_x,v_y,ref) = D(v_x,v_y,ref) + \lambda \times R(v_x,v_y,ref). \tag{5.1}$$

5.1 Multiresolution and Hierarchical Search

In multiresolution search, the frame is subsampled to obtain multiple resolution levels. The motion search starts from the lowest resolution level yielding a best match MV for that level. This coarse motion vector is refined in the next higher resolution level by doing a motion search within a reduced search window which is centered around the block corresponding to the best match MV for the previous level. This process continues until the final best match MV in the highest resolution level is determined. Multiresolution search for multiple frames was treated in [21, 22]. In contrast to [22] where the search window was fixed, in [21] the *error surface*, i.e., the spatial error distribution, from the lowest resolution (LR) level was used to adaptively constrain the search window. ME was initially applied on the LR level, and if the SAD of the best motion vector for a particular reference frame for the LR level is much larger than the ones obtained for the other frames, then this frame can be excluded, and its higher resolution levels are not searched. However, the sub-sampling required to obtain the lower resolutions causes aliasing which degrades the ME efficiency. Aliasing creates visible distortion artifacts comprising of jagged edges. The aliasing issue was partly addressed in [23], where the search window location was determined with the help of neighboring blocks, in addition to using the lower resolution level as in [21].

Apart from resolution level hierarchies, there exist block hierarchies. The multiple triangle inequalities employed in [110] facilitated the early termination of the search on smaller block types (e.g., 4 × 4), if conditions were satisfied for blocks in higher levels of the hierarchy (e.g., 16 × 16). A hierarchical search criterion was adopted in [67] as well, where MVs for larger block types were used as predictors for smaller block sizes in the same spatial location.

5.2 Fast Search using Mathematical Inequalities

Mathematical inequalities can be used for early termination of the motion search. The *triangle inequality* gives a lower bound on the norm of the difference between two vectors, which in this case are the frame blocks. The norm $d(b)$ is written as: $d(b) = \left(\sum_{(i,j)\in b} |t(i,j)|^p\right)^{\frac{1}{p}}$, where b is a block and so is a set of pixel locations, $p \geq 1$, and $t(i,j)$ is the pixel value at position (i,j). Let $d(a - b)$ denote the norm for the pairwise differences of the pixel locations of sets a and b. If $d(b)$ follows the triangle inequality, then for two blocks b_r and b_c we have: $d(b_r - b_c) \geq d(b_r) - d(b_c)$. In [110], as each point in the search grid is evaluated by calculating the SSD between the referenced block b_r and the current block b_c (the *block pair*), the current minimum SSD is compared with the lower bound calculated with the help of norms of the block pair $d(b_r)$ and $d(b_c)$. If it is greater than that bound, then there is no point in evaluating the actual SSD. Additional speedup is then obtained because the norms $d(b_r)$ and $d(b_c)$ have been *precomputed* and are re-used, since a frame will be used as reference by more than one subsequent frame. For example $d(f_n)$ and $d(f_{n-1})$ were calculated when coding frame n. When then one codes frame $n + 1$, the norm $d(f_{n-1})$ is already available to facilitate fast motion search for frame $n + 1$ from reference frame $n-1$. The norm $d(f_{n+1})$ is calculated once and is then re-used multiple times when performing motion search for subsequent coded frames.

5.3 Motion Information Re-Use and Motion Composition

Previously computed information can be exploited to limit motion search complexity. If the memory buffer has N frames and is being deployed in a sliding window manner, then a given decoded frame is searched N times during ME of the subsequent N frames. Hence, any previous calculation associated with a decoded frame can be re-used N times to speed up ME. Some algorithms use precomputed norms of blocks, as decribed above, and some use precomputed MVs. In [110], precomputed norms of blocks were used twofold: first, jointly with a mathematical inequality for early search termination, and, second, to devise a new search ordering. The order in which blocks are searched is critical, since a small value for the minimum SSD that is determined early in the search will cause the rejection of many blocks using the inequality. The widely used *spiral search* evaluates points in the search grid starting from $(0,0)$ and spiraling out. Since the all-zero motion vector and its adjacent motion vectors (e.g., within ±2 pixels) have a high probability of being the best match, their SSD will often be low, and thus the search on later points in the spiral order can terminate early with a high probability. In the *norm-ordered* search, it is assumed that a reference block r with norm similar to that of the current block c is likely to be a good match. Thus, the motion search is conducted in the order of the absolute differences between the norm of c and the norms of the blocks r.

When precomputed motion vector information is re-used, the approach is termed *motion vector composition* [17, 22, 83]. As with [110], FS is conducted only on the previous frame. Consequently, when encoding frame n, we do an FS of frame $n-1$, and we know that an FS was already done on frame $n-2$ while encoding frame $n-1$, and so forth back in the past. Let us denote with $\mathbf{MV}_j^i$ the motion vector for a block in frame i referencing a block in frame j. The motion vector $\mathbf{MV}_{n-1}^n$ is available through FS. Assuming $N > 1$ reference frames,

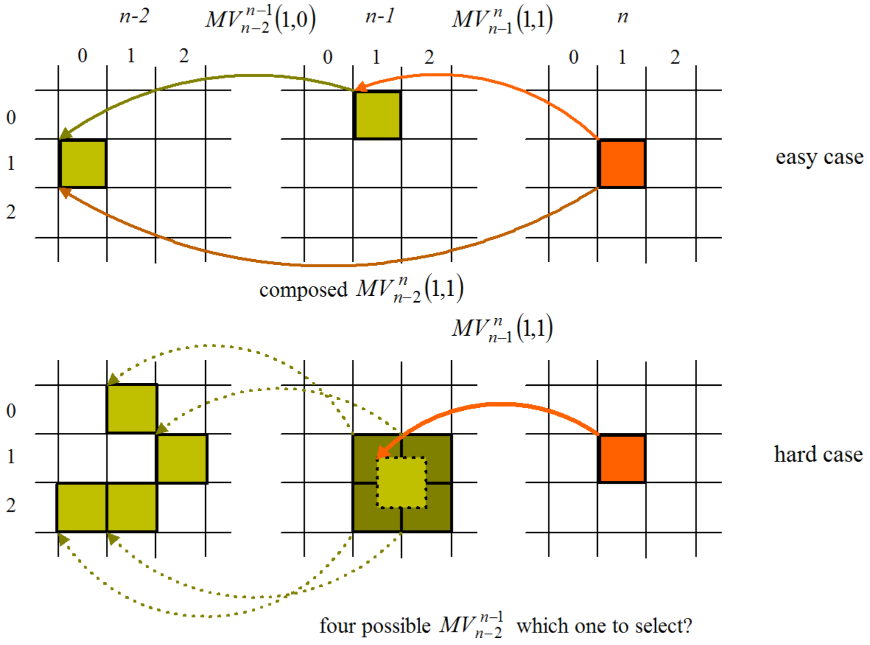


Fig. 5.1 Motion vector composition. When both horizontal and vertical components of the motion vectors are multiples of the block size, it is easy to obtain the composed motion vector. In all other cases, the problem becomes harder to solve.

one has to search frame $n-2$ for a match of the current block in n. To speed up the search, the vector $\mathbf{MV}^{n}_{n-2}$ is approximated with the sum (composition) of the $\mathbf{MV}^{n}_{n-1}$ and $\mathbf{MV}^{n-1}_{n-2}$, *both* of which are available through FS. The scheme is illustrated in Figure 5.1. Due to the use of block-based motion-compensated prediction, the motion vector for a block with spatial coordinates (x,y) is denoted as $\mathbf{MV}^{i}_{j}(x,y)$. The orange block in the current frame, which is aligned with the regular grid, references a block in the previous frame. The $\mathbf{MV}^{n}_{n-1}(1,1)$ might point to a block aligned with the regular grid in frame $n-1$; this is the *easy case*. Otherwise, it will point to a block that overlaps four other regular-grid blocks in frame $n-1$ (*hard case*), shown with a light green color, leading to four possible composed $\mathbf{MV}^{n}_{n-2}(1,1)$ candidates. Again, each of these candidates might overlap with four further blocks in frame $n-2$ (this case is not shown in the figure), if one attempts to compose vector $\mathbf{MV}^{n}_{n-3}(1,1)$.

Each one of these three techniques [17, 22, 83] adopts a different solution to select the final composed MV from the candidates. In [22], it was proposed to select the previous frame block (and hence its MV) with the highest amount of spatial overlap with the block pointed to by MV_{n-1}^{n}, while another variant proposed in the same paper preferred the block with Minimum ME SAD. Spatial overlap was again investigated in [17], where a variant was proposed that made use of the block with maximum energy in the transmitted block DCT coefficients. Both of the previous techniques composed the MV using exclusively vectors available through motion search. In a different approach in [83], a *recursive* motion vector composition was adopted that also considered the *composed* motion vectors generated while encoding the previous frame. Thus, $\mathbf{MV}_{n-3}^{n}$ was composed with the help of $\mathbf{MV}_{n-1}^{n}$ and the composed $\mathbf{MV}_{n-3}^{n-1}$, instead of using $\mathbf{MV}_{n-1}^{n}$, $\mathbf{MV}_{n-2}^{n-1}$, and $\mathbf{MV}_{n-3}^{n-2}$.

In [26], the authors adopt an approach similar to motion vector composition that tracks motion vectors in neighboring frames. MVs that refer to neighboring reference frames, such as $\mathbf{MV}_{m}^{n}$ and $\mathbf{MV}_{m\pm1}^{n}$, were found to be highly correlated, and the assumption is made that a search within a small area around $\mathbf{MV}_{m}^{n}$ is enough to estimate $\mathbf{MV}_{m\pm1}^{n}$. The largest part of the motion search computational budget is allocated to the most recent reference frame, and the rest of the MVs are derived by tracking them in a frame-by-frame manner from the most recent reference frame to the oldest reference frame. For each frame, the MVs are refined using a small search area.

5.4 Simplex and Constrained Minimization

A *simplex* consists of $N+1$ points in an N-dimensional space. In two dimensions, a simplex is a triangle, while in three dimensions a simplex is a tetrahedron (pyramid). Most early fast motion search algorithms assumed a *unimodal* prediction error surface. A unimodal error surface requires that there is a single global minimum corresponding to the best match. If the unimodal assumption holds, the distortion measure (or the "height" of the surface) increases monotonically as one moves further from the global minimum. However, local minima occur in practice. It is also known that multiresolution and hierarchical search algorithms can be trapped in local minima. The simplex minimization search proposed in [3] is resistant to local minima. Formulating the fast ME problem as a two-dimensional constrained minimization problem, simplex minimization was applied to solve it. Three multiple reference variants were proposed: one where 2D simplex minimization is used to individually search every frame; a second where the most recent frame in memory is searched using FS and the rest using 2D simplex minimization; and finally a 3D version of simplex minimization search, which, initialized with four locations as the initial simplex, proved the fastest by taking advantage of the increased dimensionality of the problem.

5.5 Zonal and Center-biased Algorithms

In this family of search algorithms [94], the search area is partitioned into *zones*. Search points in the same zone are assumed to be equiprobable of being the best match. Zones with the highest probability of containing the optimal MV are searched first (usually close to the center), and depending on certain conditions, the search is extended to other zones. Figure 5.2 shows some sample search patterns. A set of MV *predictors* initialize the algorithm, which then continues with the search pattern arrangement. Early stopping criteria are employed to further constrain the search, and the final surviving candidate MV is often refined with a ±1 or ±2 pixel search. The *center-bias* assumption states that the motion vectors that surround a motion vector predictor are the most probable to yield the best match. This assumption has influenced the design of the search patterns. Note that these algorithms can be extended to three dimensions [95], finding application in fast motion search for multiple reference frames.

The choice of predictor MVs is critical for the performance of the search algorithm. The motion vectors consisting of (a) the median of neighboring blocks' MVs, (b) the neighboring blocks' MVs themselves, (c) the $(0,0)$ MV, (d) the MV of the co-located block in the *previous* frame, (e) the MVs of the neighbors of the co-located block in the previous frame, as well as (f) the *accelerator* MV, are some of the predictors employed in [93]. The accelerator MV attempts to

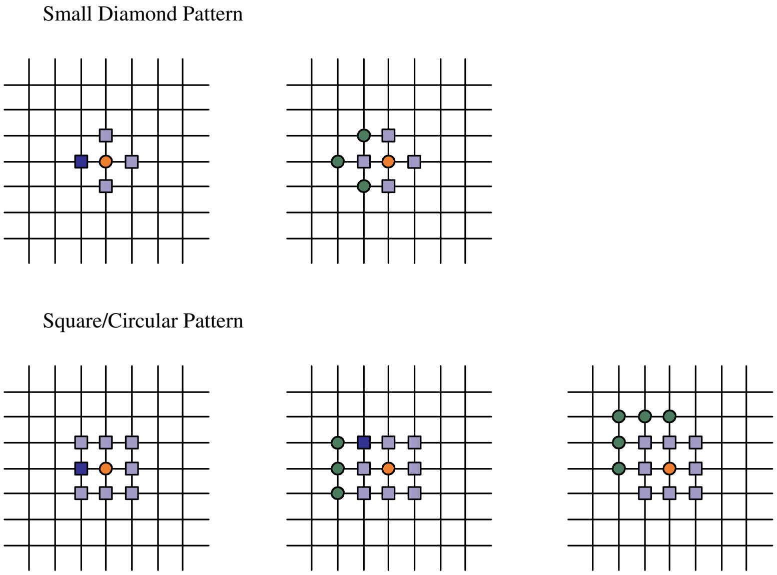


Fig. 5.2 An example of enhanced predictive zonal search. In each row, the leftmost figure shows the initial search points (zone). The point yielding the lowest metric (SAD) is denoted with the darker square. A new search (next figure to the right) is then centered around the dark square. Early stopping criteria are used to terminate it.

predict motion for objects that seem to accelerate, where the predictor can be written as $\mathbf{PMV}_{n-1}^{n} = \mathbf{MV}_{n-2}^{n-1} + (\mathbf{MV}_{n-2}^{n-1} - \mathbf{MV}_{n-3}^{n-2})$. Another motion vector predictor introduced in [97] is a search range-dependent predictor that consists of a grid with motion vector search points equally or logarithmically placed, and a scaled-in-time predictor where $\mathbf{PMV}_{j}^{i} = \frac{\Delta t(i-j)}{\Delta t(k-j)}\mathbf{MV}_{j}^{k}$. A different approach for motion vector prediction was also adopted in [97], where large block MVs (16×16) were used to hierarchically predict MVs of smaller contained blocks (e.g., 4×4). In contrast, no predictors were introduced in [89] and [88], and the $(0,0)$ MV was used as the only predictor. However, the absence of the previous comprehensive MV predictors severely affected the schemes' performance for high motion sequences. In all of these schemes [94, 95, 93, 97, 89, 88], once the predictors are available, zonal search can commence.

A three-dimensional variant termed 3D-EPZS, which accounts for multiple references, was proposed in [93] resembling a diamond, as

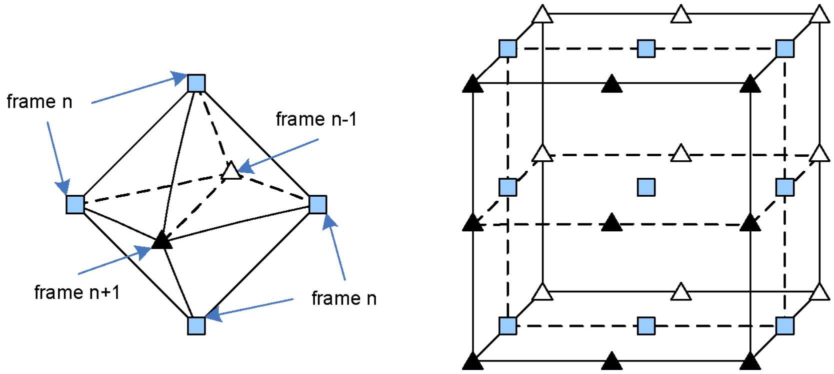


Fig. 5.3 The diamond and EPZS^3 three-dimensional zones.

shown in Figure 5.3. With a square pattern, the variant was termed EPZS^3 since it resembled a cube. A similar frame selection algorithm was proposed in [89], where several cross-shaped and X-shaped patterns were proposed and evaluated. The local minimum found through the zonal search is assumed to correspond to the optimal temporal reference. A further search is then conducted only over that particular frame using FS, in a departure from previous zonal algorithms that avoided FS at any stage. If the frame is in fact the optimal temporal reference frame, then the FS finds the global minimum. We note that any fast single-frame ME method could be applied in place of FS. The cross-diamond pattern was extended in [88] to three dimensions resembling a recent-biased tightening cone (as temporal distance increases) in three dimensions. EPZS also adjusts the search pattern based on predictor dependencies or motion refinement stage. Similarly, in [67] it was proposed to adaptively determine the edge length of the diamond-shaped search area from the MV information.

Further speedup is accomplished through early search termination. In [93, 97] various arrangements of SAD thresholding were used; i.e., the search is terminated if the partial SAD exceeds the minimum SAD of the three neighboring blocks or of certain blocks in the previous frame. Early termination was also affected by the reference frame index. Similar stopping criteria were used in [67]. Early termination for blocks found to be stationary was employed to limit complexity further in [88].

In [97], all predictors and options are enabled for the most recent frame in display order, while for older references complexity is constrained through the elimination of MV predictors and adaptive termination thresholds based on temporal distances. There are also MV predictors that are enabled only for references older than the previous frame, and the sub-pixel motion refinement can be constrained depending on the reference frame index. Furthermore, in both [97] and [67], MV distribution influenced the selection of search patterns.

5.6 Fractional-pixel Texture Shifts or Aliasing

A radically different approach for fast multi-reference frame selection that exploited sub-pixel texture shifts was presented in [14]. Let us consider the simple case of a moving rectangular surface overlayed with a texture. We suppose the resolution of the texture is roughly equal to the resolution of the imaging sensor. If for some frame the sensor sampling grid coincides with the texture grid, then that frame has an integer-pixel location. If, however, they do not coincide, then we have a sub-pixel texture shift, as shown in Figure 5.4. We can have hence the situation where the texture was aligned at integer location at frame $n-2$, moved right by $\frac{1}{2}$-pixel in frame $n-1$, and another $\frac{1}{2}$-pixel right in frame n, so that in the end the shift is one whole pixel to the right

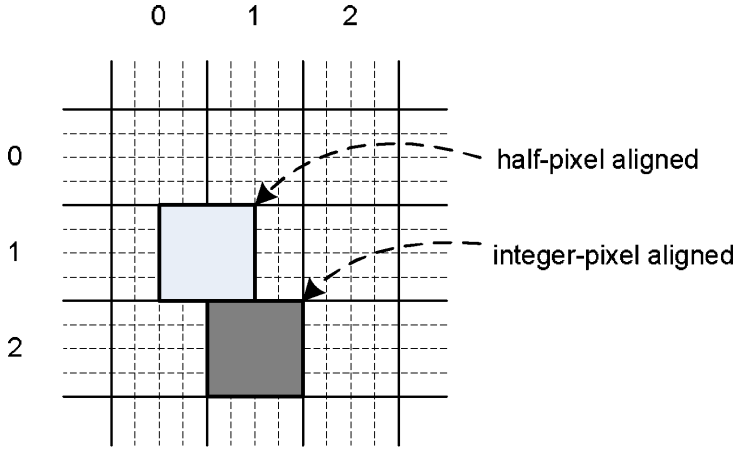


Fig. 5.4 Sub-pixel alignment and shifts.

compared to its position in frame $n - 2$. Frame n can be predicted from $n - 1$ but fractional-pel MCP (Section 3.5) will be required to realign the locations and find a good match. In contrast, by searching in frame $n - 2$, one can locate the best match by using an integer MV that avoids the often unreliable filtering process required by fractional-pixel MCP. This explains the decreasing gain due to sub-pixel motion compensation when multiple reference frames are used, as reported in [39].

Knowledge of the sub-pixel location of the texture of a block can be used to speed up ME. For MCP with quarter-pixel precision, the sub-pixel location of a block in each reference frame can be integer, half-pixel, or quarter-pixel. It is usually initialized as an integer value, and is then successively updated by adding the MV that is calculated during motion search. Hence a block with an MV of $(-1.5, 2.5)$, which was previously classified as integer will now be classified as half-pixel. Assuming block motion search on three reference frames, the sub-pixel locations of the co-located blocks in all three frames are evaluated, and if two or more of the three blocks share the same sub-pixel location, then motion search is only conducted on a single frame out of those frames, which is selected as the closest one in display order. One or two frames are thus skipped, yielding speedup. Still, motion search is always applied on the previous frame. This technique speeds up search exclusively in the temporal domain and can be coupled with any spatial domain fast motion search method.

Another approach that identified the usefulness of MRP for aliased content can be found in [46]. Edge detection was used to classify areas into high-frequency and low-frequency. More reference frames are allocated to code the high-frequency areas. The authors note that spectrum Fourier analysis is the optimal solution for the classification of the frames, but resort to Sobel edge filtering to constrain complexity.

5.7 Content-Adaptive Temporal Search Range

This family of algorithms exploits source or coding statistics such as coding modes and MV distribution to design fast motion search algorithms that adapt the number of evaluated reference frames to the content. These are complemented by a group of algorithms that constrain the temporal search range using rate-distortion (RD) principles.

Experimental results from coding experiments with H.264/MPEG-4 AVC in [47] confirmed that often as much as 80% of MVs point to the previous frame, thus in this group of algorithms, FS is adopted for the previous frame. (a) Quantized coefficients of the motion-compensated prediction residual from the previous frame, (b) particular variable blocking structure (which implies occlusion and object boundaries), and (c) relative performance of the intra mode, texture information, and the MV compactness (similarity of motion vectors for variable-size blocking structures) are all used to constrain the search in the temporal dimension.

Somewhat similarly to [46], in [123], MBs are classified into active and inactive through comparison of co-located pixel differences between consecutive frames. Temporal stationary regions and moving regions with large flat textures are characterized as inactive regions and are usually coded with large block sizes. The authors assume that the prediction gain obtainable from searching many reference frames is mainly from occluded and uncovered objects which occur in active regions. Thus, fewer reference frames are used for inactive regions.

In [70] depending on the number of reference frames and the content, the authors define two states when performing motion search for MRP: *gain aggregation* (GA) where additional references yield meaningful coding gains and *gain saturation* (GS) where additional references have little impact. A so-called reference frame buffer utilization index is used to distinguish between the GA and GS states. This index keeps track of how often references older than the previous frame are used. A high value of this index points to the GA state. The temporal search range is adapted periodically to ensure that the utilization index stays within a desired value range. The range thresholds depend on the QP and the current temporal search range.

A compelling rate-distortion-based formulation of the problem of fast ME for MRP was introduced in [55]. The approach is based on modeling the RD gain with respect to the number of reference frames, which is termed the temporal search range. The authors also model the complexity gain with respect to the number of references, which is found to behave in a linear fashion. In general, an increase in computational/memory complexity results in a gain in RD performance. The concept of *temporal coherence* is introduced to model the above RD-complexity trade-off. Temporal coherence is defined as the root mean square of the difference between a motion vector in the current frame and motion vectors of blocks in a search window of previous frames. High temporal coherence indicates a static region or a uniformly moving object that signifies a need for *fewer* reference frames. The RD coding gain is thus modeled as a function of the number of reference frames and temporal coherence. The algorithm speeds up motion search by reducing the number of references if the *expected* RD gain is low enough. The solution that corresponds to the expected coding gain is chosen at the point (number of references) where the ratio of the RD gain over the complexity gain is maximized. The search in the temporal direction terminates when the actual coding gain reaches the expected coding gain. This is illustrated in Figure 5.5.

An extension of the approach in [55] to the block level can be found in [57]. Here, the number of reference frames for each block varies according to the size of the block. It is assumed that if the block size is smaller, then more reference frames will have to be evaluated. Several possible reference frame collections for each block size are available, and the decision on which to employ during ME is made by jointly minimizing the estimated coding loss and the number of reference frames for the given reference frame collection.

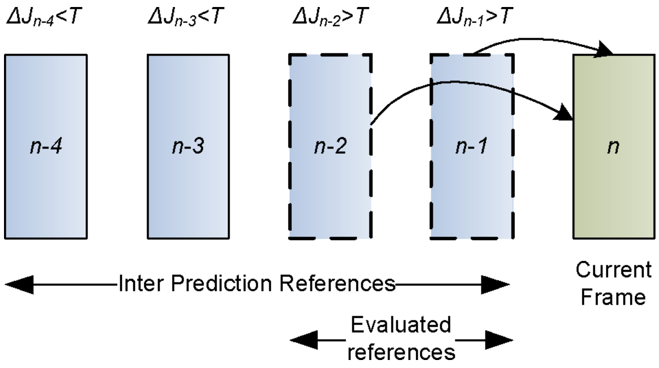


Fig. 5.5 Reduction of references through RD criteria.

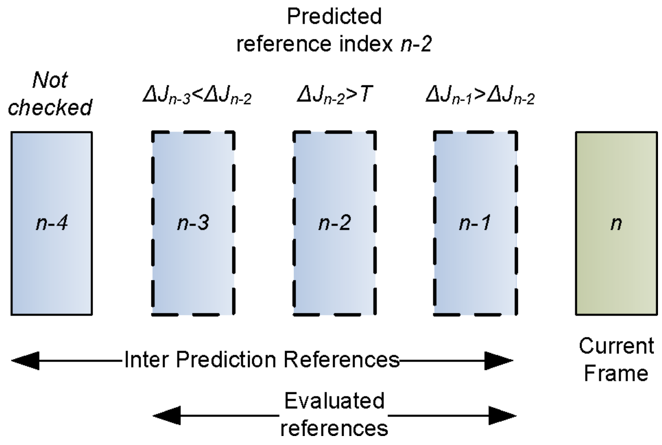


Fig. 5.6 Direction-adaptive search that tracks the largest RD gain.

Another RD-based solution that is conceptually simpler than the approach in [55] was presented in [78]. The premise of the algorithm is the following: usually the Lagrangian cost function J for a macroblock increases as the reference frame index moves away from the optimal reference frame. A reference frame index t is first predicted from the reference indices of coded blocks in the causal neighborhood of the current block. The reference frames corresponding to t, $t-1$, and $t+1$ are evaluated using motion compensation to produce the bit cost and the resulting distortion, which are both combined to derive their Lagrangian cost. By comparing the derived costs, the search terminates if costs at $t-1$ and $t+1$ are higher than the one for t, or, otherwise, the search may evaluate frames before $t-1$ or after $t+1$, extending to the direction that has the lowest cost compared to the one for t. An example can be found in Figure 5.6.

6

Error-Resilient Video Compression

Packets of a video stream can be lost for a variety of reasons, including (a) bit errors that cause subsequent packet-level integrity checks, such as cyclic redundancy checks, to fail; (b) discarded packets at network routers due to buffer overflow and network congestion; and (c) delay caused by inefficient routing so that a packet arrives too late to be useful.

In hybrid video coding, error corruption in one frame can propagate to subsequent frames via motion-compensated prediction. This is illustrated in Figure 6.1. The dark squares denote corrupted blocks. Several techniques have been proposed to address error resilience in MRP schemes. In this section, we address error-resilient source coding. Note that channel coding for error resilience is outside the scope of this survey.

6.1 Multiple Frames

The codecs described in this section use two or more frames as prediction references, and error resilience is primarily improved by appropriate reference picture selection (RPS). RPS is facilitated by a host of

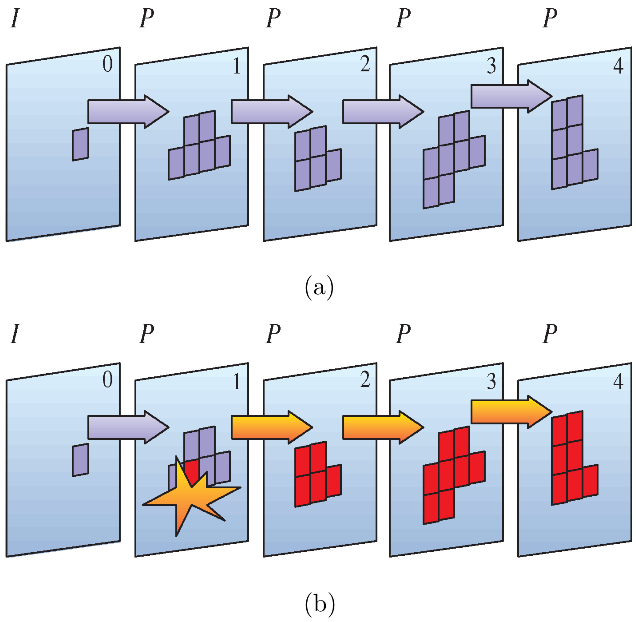


Fig. 6.1 Error propagation for a motion-compensated video codec. The large arrows denote motion-compensated prediction. Blocks on the left of an arrow are prediction references of blocks on the right. (a) Due to motion-compensated prediction, the seven blocks depicted in the current frame (frame 4) will be decoded correctly only if all of the outlined blocks across frames 0, 1, 2, and 3 survive. (b) The loss of a single block in frame 1 corrupts all blocks referencing it in frame 2 and the error propagates until everything in its path has been corrupted.

schemes, which we describe next: (a) distortion estimation, (b) probabilistic selection, (c) feedback, (d) fixed selection, and (e) decoder-based selection. At the end of the section, some theoretical insight into the efficiency of these systems is briefly presented.

6.1.1 RPS through Distortion Estimation

These techniques estimate the distortion due to error propagation by modeling the decoder reconstructed samples as *random variables*. Multiple ($N > 2$) frames were used as references in [36, 68, 92, 108], while only two frames were used in [65]. The selection of the reference frame considers the impact on error resilience. The decision is a trade-off between error resilience and compression efficiency, and is solved with rate-distortion (RD) techniques in [65, 68, 108]. The distortion calculation becomes now an *estimation* as potential corruption is modeled probabilistically to obtain an estimate of the resulting distortion. Different approaches have been proposed on how to estimate the distortion due to errors.

Branches of an event tree were employed to model error propagation in [108, 68]. For the simple case of modeling the possible outcomes of decoding the second of two consecutive frames, there are two binary splits of the tree, with the first level representing the outcome of decoding the first frame and the second level representing the outcome of decoding the second frame. One leaf represents the event "first and second frames received intact", the second leaf the event "first frame received and second frame lost", the third leaf the event "first frame lost and second frame received", and finally the last leaf the event "both frames lost". To limit complexity, since modeling the n-th frame's distortion requires a 2^n-leaf tree, the tree was pruned in [108]. However, if the status of a transmitted frame is known, the event tree can be reinitialized with that decoded frame as its root. Hence assuming a feedback delay of $d \ll n$ frames, the tree will now have up to 2^d leaves. In [68] the tree was not pruned and its size was kept in check with the help of feedback. A recursive per-pixel distortion estimate [120] was used in [65]. Each pixel is estimated as the weighted average of its error-free reconstructed value and its potentially distorted value due to corruption, and the estimate is recursively updated.

The way the distortion estimates are incorporated in the RD framework differs significantly among schemes. Let $\tilde{D}$ denote the distortion estimate that considers the impact of error propagation. Let b denote the current block, which is predicted with motion vector $\mathbf{v}$ from reference frame t using coding mode M. We assume that the prediction residual is coded using a quantization parameter Q. In [108, 113], first the reference frame is determined per block with rate-constrained motion estimation, as $arg\min_{\mathbf{v},t} J(\mathbf{v},t|b)$ where $J(\mathbf{v},t|b) = \tilde{D}(\mathbf{v},t|b) + \lambda_{motion} \times R(\mathbf{v},t|b)$. Then the coding mode is determined with another RD decision, as $arg\min_M J(M|b,Q)$ where $J(M|b,Q) = \tilde{D}(M|b,Q) + \lambda_{mode} \times R(M|b,Q)$. However, in [65], the spatial motion vector is found by minimizing the SAD for all reference frames as $arg\min_{\mathbf{v}} \tilde{D}(\mathbf{v}|b,t)$, and subsequently used in an RD decision that jointly decides both mode and reference frame per block as $arg\min_M J(M,t|b,\mathbf{v},Q)$ where $J(M,t|b,\mathbf{v},Q) = \tilde{D}(M,t|b,\mathbf{v},Q) + \lambda_{mode} \times R(M,t|b,\mathbf{v},Q)$. In contrast to [108], the reference frame is selected on a frame basis in [68].

A different approach made use of so-called periodic macroblocks in [121]. Periodic macroblocks use only the previous n-th frame as the reference frame. Furthermore, a periodic macroblock is not necessarily used as a reference for another periodic macroblock. The selection of the periodic macroblock was based on the expected distortion: the macroblocks with the highest expected distortion were coded as periodic macroblocks. To intuitively understand why periodic macroblocks can be useful in an error-prone environment, consider the limiting case where all frames are predicted from frames that are N frames apart. This means that if one frame is damaged, only $1/N$-th of the total frames will be corrupted in the end, while with traditional prediction from the previous frame, all of them will be corrupted. Of course compression efficiency would suffer substantially in this case, so some balance between error resilience and compression efficiency in the choice of periodic macroblocks is needed.

6.1.2 Probabilistic RPS

Another family of techniques [12, 13] that adopt a sliding-window frame buffer with multiple references used probabilistic analysis to estimate and decrease the block error *probability*. This is fairly different from the schemes in Section 6.1.1 that use probabilistic tools to determine the expected distortion at the decoder. It was determined that the error probability has a direct relationship with the probability of selecting the most recent reference frame in display order for motion compensation. Markov chain analysis was applied to improve the error resilience of the bit stream. The objective is to decrease the probability of using the most recent reference frame in display order while at the same time control the increase in bit rate. The final algorithm adapts the reference selection to ensure that all reference frames in the frame buffer are selected as prediction references with the same frequency for a given spatial position (block) when considering the entire sequence.

6.1.3 RPS through Feedback

Reference selection for error resilience can greatly benefit from the availability of feedback from the decoder. Such strategies were presented in [36] and [92], where multiple frames were buffered and the reference selection at the encoder was assisted by decoder feedback signals.

One traditional method for containing error propagation is intra-refreshing (coding an entire frame or specific blocks as intra) when a negative-acknowledgment (NACK) feedback signal is received. In [36] it was proposed to use feedback to dynamically replace the reference picture. If a loss is indicated, then the reference frame buffer stays unchanged containing the last correctly decoded frame. Two schemes are proposed: the “ACK” and the “NACK” scheme. The “ACK” scheme updates the reference frame buffer only when receiving an ACK signal from the decoder and is effective at high error rates, while the “NACK” scheme updates the reference buffer with the coded picture as long as no NACK signal has been received, being effective at low-error rates. Both schemes were extended in [92] to work in mobile devices characterized by *limited* frame memories, where reference memory overflow can occur. An example is shown in Figure 6.2. The reference used for motion compensation of a block in frame n is fixed to the one found in the reference frame buffer. For an error-free environment, that would be frame $n - 1$. Otherwise the buffered reference is a result

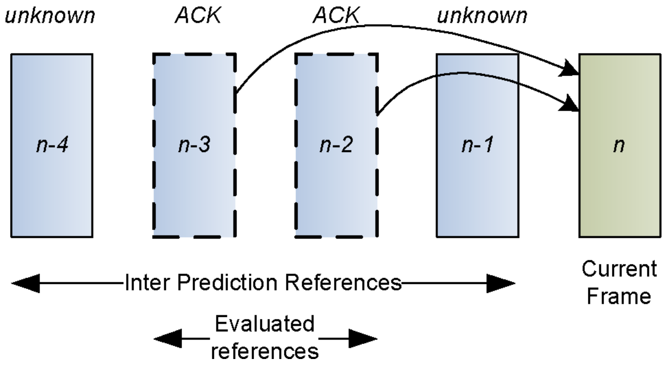


Fig. 6.2 ACKed frames are used as prediction references.

of the received feedback signals. An extension of [36], the feedback-based approach of [99] has the *fixed distance* of the reference frame determined as a function of the round-trip-time and thus feedback is always available for the reference frame.

An alternative use of feedback is discussed in [108]. Using the latest feedback information, the error estimates obtained through the distortion model are re-initialized at the encoder and become more accurate. Compared to the schemes in [36] that constrained reference selection to only consider the most recent frame believed to be correct, reference selection was generalized by allowing use of any frame in the reference buffer. The updated distortion estimates were used with Lagrangian minimization to select the reference frame. A similar feedback scheme was later employed in [65] tailored to a short-term/long-term frame scheme. The LT frame was constrained to always be one for which feedback is available.

The ACK and NACK schemes proposed in [36] provided the basis for Annex N (Reference Picture Selection) of the ITU-T Recommendation H.263 [104]. Annex N enables the use of backchannel messages from the decoder to the encoder that inform the encoder which parts of which pictures have been correctly received. Given this information, the encoder can signal the motion-compensated prediction (MCP) reference picture to the decoder so that temporal error propagation is avoided. The temporal reference index is sent for each group of blocks or a slice. Both the encoder and the decoder have to provision for additional picture memories to store multiple decoded pictures. This is usually handled by some external messaging mechanism, such as ITU-T Recommendation H.245. If the indexed picture is not available at the decoder, then the decoder can request a forced intra update where the current picture is re-encoded and re-sent by the encoder as intra.

The backchannel feedback messages adopt one of four possible strategies: (a) No feedback messages are provided. (b) ACK signals are sent for each video picture segment (GOB/slice) that is correctly received. (c) NACK signals are sent when a decoder fails to decode a video picture segment. (d) Both ACK and NACK signals are sent. Unless externally negotiated, decoded picture segments are stored and flushed from the multiple pictures memory on a first-in first-out basis.

Similar functionality also exists in Annex U of H.263, which is essentially a superset that includes all the functions of Annex N.

The schemes in [36] require feedback signals during transmission to generate the video bit stream. A feedback-based approach that uses bit streams generated *in advance* was presented in [122]. In H.264/ MPEG-4 AVC, SP-slices enable efficient switching between different bit streams through the transmission of a "switching" SP-slice in place of a slice that was coded as SP in these bit streams. As an example of the [122] approach, for five reference frames, four H.264/MPEG-4 AVC switching SP-slice macroblocks can be generated for each of the remaining reference frames along with one default version (the best match block and its corresponding reference frame). This is done in an offline stage. These alternative predictions are expected to match the block well since they provide the best match in each reference frame. During streaming, and depending on the losses, the bit stream segments that are transmitted are selected from the encoded versions so that losses are minimized at the decoder. Encoder–decoder mismatch is avoided thanks to the properties of switching SP-slices. Corrupted regions are avoided as references by transmitting bit streams that use more reliable references.

Techniques such as those introduced first in [120], and then later adapted for multiple reference prediction (MRP) in [65], work by coding certain blocks as intra by taking into account their expected distortion due to packet losses. These are also known as intra refresh blocks. In [100] it was proposed to address error resilience by using reliable reference frames (RRFs) to code *inter*-refresh blocks. The motivation was to counter the coding loss introduced by the intra-refresh mechanism. A long-term frame buffer is maintained and frames are stored there in a periodic fashion as with [65]. The difference though now is that the buffer holds multiple long-term references and only those marked as RRFs are used as references. Furthermore, a frame marked as reliable is released from the long-term buffer only after another frame that has been identified as being reliable takes its place. To identify RRFs in the buffer, the inter-frame dependencies are tracked. This information is used along with frame reception information inferred through feedback to identify frames that are free of any significant propagation of error. The per-pixel dependency count is a measure of how important this pixel is for future frames through MCP. A frame is deemed reliable if that frame and a number of previous frames proportional to the dependency count are known to have been correctly transmitted.

6.1.4 Fixed RPS

The approach presented in [105] sought to render the prediction structure robust to errors. Periodic key frames (either I- or P-coded frames) were afforded forward error correction (FEC) to selectively increase their reliability, and it was proposed that periodic key P-coded frames employ long-term MCP to predict exclusively from previous key pictures. Regular P-coded frames (without FEC) were constrained to use only reference frames as old as their past key picture, creating thus a separate prediction path that increased error resilience. This structure is illustrated in Figure 6.3.

6.1.5 RPS at the Decoder

Schemes discussed so far selected the reference frame at the encoder and signaled this selection in the coded bit stream, with or without assistance from decoder feedback. In [87], however, *pre-emptive* transmission of several key and periodic frames increased error resilience. For example, frames 50, 100, 150, and 200 might be transmitted substantially before their regular position for encoding (they might even be sent before frame 1 is encoded). During decoding, the decoder selects among the pre-transmitted or the regular-transmitted key or periodic frames depending on the quantization parameter and on the error corruption.

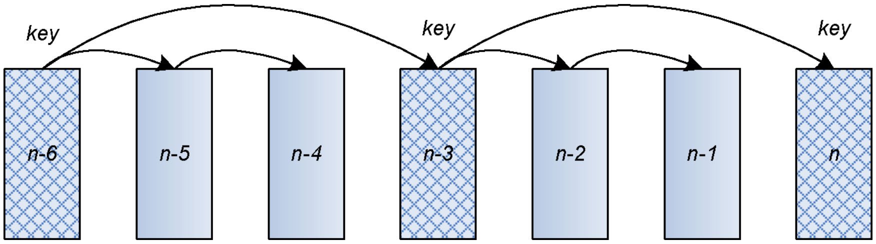
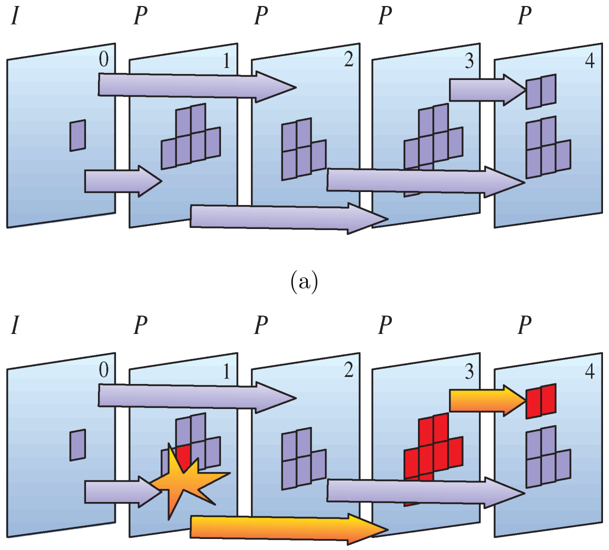


Fig. 6.3 Periodic key pictures.

For example, the version of frame 50 that is encoded in its regular position (after frame 49) may or may not be the one used for display or used as a motion compensation reference, depending on how it compares with the pre-transmitted frame 50. The motion-compensated reference frame used during decoding was not constrained to be identical with the one used during encoding.

6.1.6 Theoretical Work

Valuable theoretical insight into the error resilience of multiple-frame prediction schemes is given in [13], which proved using Markov chains that MRP reduces the error probability for a decoded frame transmitted over an error-prone channel. The error probability for MRP is strictly lower than that of traditional MCP if periodic intra-refresh is used. Intuition, as in Figure 6.4, pointed to that conclusion.



(a)

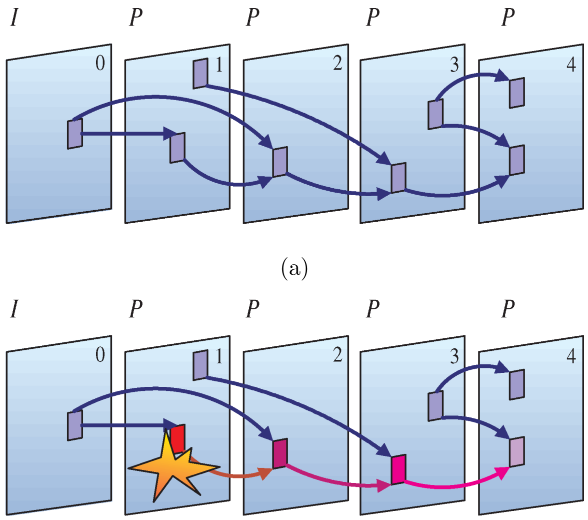
(b)

Fig. 6.4 Error resilience for a multiple-frame motion-compensated video codec. (a) Multiple arbitrary prediction paths are created since there is no restriction as to which frame each block references. (b) The loss of a single block again causes error propagation, but in this case it is easier to contain it, and blocks in frames 2 and 4 survive unharmed. The "domino effect" we witnessed in the case of traditional single-frame MCP is now avoided.

An analysis of the memory requirements of an STFM/LTFM scheme with and without feedback was presented in [65].

6.2 Multihypothesis Prediction (MHP)

Superposition of multiple hypotheses improves error resilience due to an inherent *error attenuating* property. When averaging two hypotheses, one of which is corrupted, the final weighted average will be *partly* corrupted. As it is successively averaged with other healthy hypotheses, the prediction mismatch error will be attenuated over time. This property is illustrated in Figure 6.5. The initially dark corrupted block is superimposed repeatedly with healthy blocks and the error is attenuated (shown as becoming a lighter shade) with time. Multihypothesis prediction for error resilience comprises establishing the hypotheses blocks and the respective weighting coefficients. We discuss methods for doing so next. This section is concluded with a review of theoretical results.



(a)

(b)

Fig. 6.5 Error resilience for a multihypothesis motion-compensated video codec. The arrows here denote MCP. When more than two arrows converge to a single block this represents multihypothesis prediction of that block from the referenced blocks. (a) Error-free transmission. (b) Error and subsequent corruption due to propagation.

Hypothesis selection was constrained in [56], where each block in the current frame was predicted using the weighted superposition of two blocks, one from each of the two previous frames in display order, frames $n - 1$ and $n - 2$, using two MVs. It was assumed that it is very unlikely for both of the hypotheses to be erroneous simultaneously, which could not be said for unconstrained MHP, where the hypotheses could have come from the same frame, and would thus be vulnerable to burst errors. MHP was applied on scalable video coding in [59], drawing heavily from [56], and employing two hypotheses for the base layer (which can be decoded only if received in its entirety), and three hypotheses for the enhancement layer (where granular decoding is feasible). Unconstrained multihypothesis prediction for error resilience was employed in [60], superimposing two hypotheses selected from multiple reference frames.

The MH schemes discussed above form the multihypothesis by linearly combining two or more hypothesis blocks, which were obtained with MCP. In [81], however, N past *frames* were averaged to generate the superimposed reference frame. Following that, motion estimation was applied on this newly generated frame to obtain the prediction block. Instead of MHP from reference frames, the block is now predicted from a multihypothesis frame, as shown in Figure 6.6.

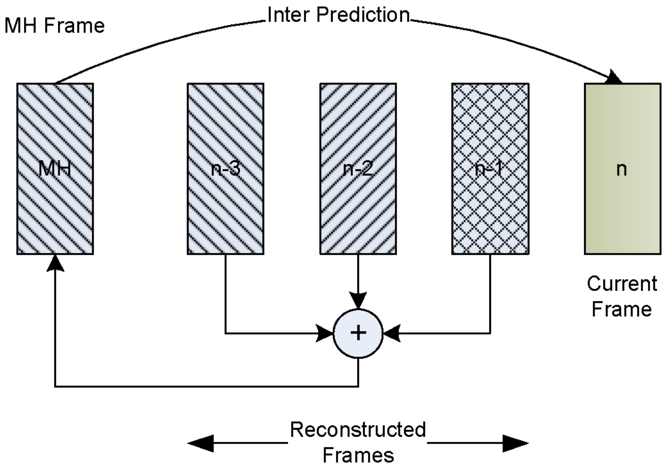


Fig. 6.6 A multihypothesis reference frame.

An RD optimal scheme was used to combine the hypotheses in [60], making use of estimated distortion that modeled the attenuation property of MHP. The bit rate decrease due to additional hypotheses and the bit rate increase due to added motion information were modeled as well. The experimental evaluation established that intra refresh converges more slowly than multihypothesis MCP, but it does converge to *zero*. As a result, MHP yielded very good error attenuation for the short-term, while in the long-term, intra refresh proved better. In addition, intra refresh was found more suitable for high bit rates, while MHP was better for low bit rates, for a small number of hypotheses, and for high loss rates. RD optimization was used to determine the multihypotheses in [59] as well, while standard sum of absolute differences (SAD) minimization and thresholding was used to select the prediction in [81] and [56].

In [98] the authors propose storing the motion vectors and reference picture indices for a given block in *multiple* frames. In H.264/MPEG-4 AVC these motion parameters, which for biprediction consist of two sets, one for each reference picture list, are stored with the header and the transform coefficients of the macroblock they correspond to. Here, one set of the parameters is stored with the current block and the other is stored in the co-located block of a previous frame in display order. This results in retaining half of the motion parameters when a frame is lost, which is beneficial for error concealment. Furthermore, to mitigate the effect of burst losses that affect two consecutive frames, the authors fixed the RPS to consider frames $n - 1$ and $n - 3$ for two-hypothesis prediction. Note that the motion parameter separation results in a codec that is no longer standard-compliant. Last, the prediction formation in the decoder is also modified: When one of the hypotheses has been lost, the prediction block in the decoder is formed by only considering the surviving hypothesis; the concealed version of the lost hypothesis is not used for MCP but only for display purposes.

Theoretical results in [56] showed that MHP exhibits increased error robustness compared to single-frame schemes. Two more theoretical analyses concentrated on the effect of the predictor *coefficients* and the *number of hypotheses*. While studying the error resilience property of MHP in [72], and assuming two hypotheses, it was found that the predictor coefficients have a heavy influence on the error suppression property. The smaller the coefficient for the temporally closest frame, the smaller the value that the attenuated error converges to. This reflects findings for multiple references in [13]. The additional attenuation due to fractional-pixel MCP was also taken into account. The theoretical approach presented in [60] showed that the propagated error decreases as the number of hypotheses increases, however, it converges to a *non-zero* value. An analysis of the error robustness of a multihypothesis scalable coding scheme is also found in [59], where bounds for drift errors were derived.

6.3 Reference Selection for Multiple Paths

We now consider reference frame selection for video transmission over multiple network paths. The reference selection problem becomes difficult to solve, as different network paths exhibit different error characteristics, such as burst duration and packet loss probability, and can be time varying. Two separate transmission paths were investigated in [69, 71, 19]. Frames transmitted along one path could reference frames transmitted along the other in [19, 71]. In [69], however, the predictor was constrained to only reference frames sent along the same path. The encoder-transmitter has to make two important decisions: from which *reference* frame to predict the current frame, and along which *path* to send it.

In all schemes, decisions for reference frames are made on a frame level. Both [69] and [71] make use of feedback, in contrast to [19]. The RPS scheme proposed in [68] is extended in [69] to employ path diversity for streaming applications, adopting the former's RD-optimized decision scheme as well as a distortion estimator. The use of feedback in [71] allows prediction from frames in different paths. The strategy employed was to use the last frame that is *believed* to have been reliably transmitted, taking into account current feedback (having recently received only ACKs), or if no negative ACKs were received during encoding, and the last received message was an ACK, making the assumption that the channel is "good".

A technique based on dynamic programming was adopted in [19] to select reference pictures by modeling the selection process with graphs that described potential reference frame selections. The author did *not* seek to minimize the expected distortion of the reconstructed frames, rather the goal was to optimize (maximize) the *number of correctly received frames*. Parameters taken into account were the rate used and the probabilities of success for individual frames. These probabilities depended on the rate expended. Assuming constant *bit* error rate, the more bits are allocated to a frame the likelier it is that one of those bits will be flipped. The complexity of this scheme is considerable since the rate required to encode one frame given another as a reference has to be calculated for all possible combinations.

In [20] the algorithm of [19] was extended by modeling the impact on error concealment of a particular correctly decoded frame. The resulting algorithm was two-tiered: first the problem is solved without loss compensation and then it is locally refined by considering the impact of error concealment. With respect to the decision to use dynamic programming vs. Lagrangian optimization, the authors remarked that the application of traditional Lagrangian techniques to this optimization problem suffers from either bounded worst case error and high complexity, or low complexity and undetermined worst case error.

Several different approaches were applied for path selection. In [69], the next packet is always sent over the path from which the most recent ACK was received. If both of the paths are in the same state, either good or bad, packets are distributed evenly in an alternating manner. Path selection was avoided altogether in [71], since one path consisted of the even frames and the other of the odd ones. Reference and path were finally jointly selected in [19], instead of independently in [69], and rate constraints for the paths were considered during optimization. In conclusion, we should note that both schemes in [69] and [71] heavily depend on the feedback delay being reasonably small.

7

Error Concealment from Multiple Reference Frames

When poor channels lead to irrevocable loss of information, the decoder can attempt to minimize the perceptual effect of the loss by concealing the corrupted portions. Error concealment (EC) techniques are divided into spatial and temporal methods. Spatial methods rely on information from correctly decoded areas of the damaged frame, while temporal methods rely on information from correctly received and decoded frames. In the context of multiple reference frames, we are interested in methods that rely on temporal information. All methods assume that the decoder can always detect the loss. While for packet losses, detection is straightforward, the same is not true when a few bits are corrupted during, say, wireless transmission of the packets. In the latter case it is the responsibility of the physical layer to detect these errors and drop the packets through, e.g., some checksum mechanism.

7.1 Temporal Error Concealment

The temporal concealment of a lost block has similarities with the problem of finding the best match block for motion compensation. In motion estimation, a prediction for a block B is generally obtained by comparing that block with blocks in some reference frame. Here, the original block B is no longer available. Temporal EC algorithms seek to estimate the motion parameters for a lost block with respect to some reference frame to produce a concealment block. Provided the motion information is retrieved, these methods can be applied on both inter-frame and intra-frame coded blocks. Assuming a translational model, these methods seek to estimate the missing motion vectors, which may consist of the following components: the horizontal and vertical spatial components, weighted prediction parameters for illumination compensation, the motion compensation block size, and the temporal reference component. The estimated concealment motion vector (CMV) is then used to refer to a concealment block in some previous correctly decoded reference frame.

An overview of widely used algorithms for MV component recovery was presented in [2]. We extend this discussion further to also comment on the feasibility of the algorithms in cases where entire frames are lost. These algorithms are now briefly described:

1. Zero or Temporal Replacement, where the lost block is replaced by the co-located block in the previous frame. In this case the CMV is $(0,0)$. We note that this technique is applicable both to whole and partial frame losses as it is not dependent on any neighborhood statistics. Let $\bar{f}_n(i,j)$ denote the concealment value for pixel $f_n(i,j)$. For zero replacement we have $\bar{f}_n(i,j) = f_{n-1}(i,j)$.
2. Motion-compensated EC (MCEC), where the recovered motion component is the average or median or some other function of the corresponding components of a set of MVs of neighboring blocks that have been correctly received and decoded (for example they may have been transmitted in a different packet). Such a technique requires availability of neighborhood information and is thus more suitable for partial frame losses. Let $(\bar{v}_{x,k}, \bar{v}_{y,k})$ denote the recovered MV corresponding to reference $n-k$. The EC value is given as: $\bar{f}_n(i,j) = f_{n-k}(i + \bar{v}_{x,k}, j + \bar{v}_{y,k})$.

3. Boundary Matching (BM) [62]. In contrast to MCEC, the CMV is not estimated from available neighboring motion information. Here, a *search* is performed for a best match block for missing block b on a previously correctly decoded reference frame. Assuming one has access to the neighborhood boundary $\check{b}$, one can calculate some measure of disparity between the available boundary and the respective boundaries of candidate blocks in some reference frame(s). The best concealment block (CMV $(\check{v}_{x,k}, \check{v}_{y,k})$) is selected by minimizing a block boundary $\check{b}$ match criterion, such as the sum of absolute differences $(\check{v}_{x,k}, \check{v}_{y,k}) = arg \min_{v_x, v_y} \sum_{(i,j)\in\check{b}} |f_n(i,j) - f_{n-k}(i + v_{x,k}, j + v_{y,k})|$. This technique requires access to neighborhood information and is thus not suitable for whole-frame loss concealment. For whole-frame loss, some kind of pre-processing is needed to derive some information that can play the role of a neighborhood. A sample neighborhood is shown in Figure 7.1. Note that BM is characterized by high computational complexity as it requires a motion search.
4. Optical Flow (OF) estimation to derive the missing CMVs [5]. Optical flow estimation makes the assumption that intensity remains constant along a motion trajectory. Let $I(x,y,t)$ denote the intensity of a pixel in an image at spatial coordinates (x,y) at time t. The optical flow constraint

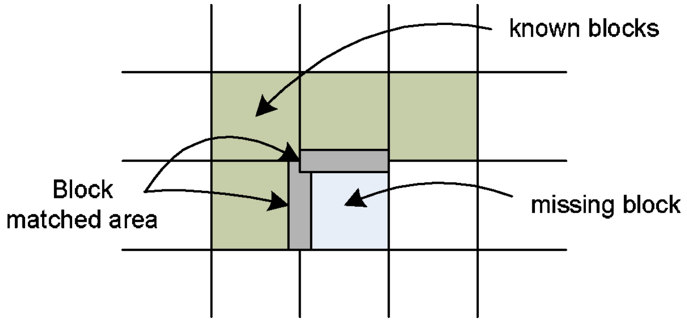


Fig. 7.1 Neighborhood for block matching.

equation $\frac{dI}{dt} = 0$ is written as:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0. \tag{7.1}$$

Terms $\frac{dx}{dt}$ and $\frac{dy}{dt}$ can be seen as the object speed across each direction and can be estimated once the partial derivatives are known. Using multiple frames one may estimate the motion in the missing areas and then use this motion information to conceal the missing areas using some correctly received and decoded frame. This technique is also applicable to whole-frame loss EC.

Performance may be severely impacted by the choice of block size for motion compensation used to generate the concealment block. In H.264/MPEG-4 AVC, block sizes vary from as small as 4 × 4 pixels to as large as 16 × 16 pixels. The block size may be fixed or inferred from available neighborhood information. In one other approach, termed Motion Field Interpolation (MFI), one vector *per-pixel* is recovered by interpolating neighboring MVs in the top, bottom, left, and right blocks. In the case of bilinear interpolation and assuming 8 × 8 blocks and that the horizontal component is 1, 2, 3, and 4, respectively, for each of the neighboring blocks in Figure 7.2, the recovered horizontal component for pixel (7,0) will be $1 \times \frac{7}{8} + 2 \times \frac{1}{8} + 3 \times \frac{1}{8} + 4 \times \frac{7}{8}$. Such a scheme can result in better subjective performance since the per-pixel motion-compensated prediction will produce fewer blocking artifacts compared to a block-based approach. We note, however, that this technique is highly dependent on the availability of neighborhood information. Furthermore, per-pixel motion-compensated prediction can be prohibitively complex.

7.2 Concealment from Multiple Frames

Several techniques have been proposed for EC with multiple frames, all of which exploit information from multiple buffered frames to improve the efficiency of error concealment. These techniques can be divided into two families: (a) techniques where the concealment block is a linear combination of blocks selected from multiple correctly

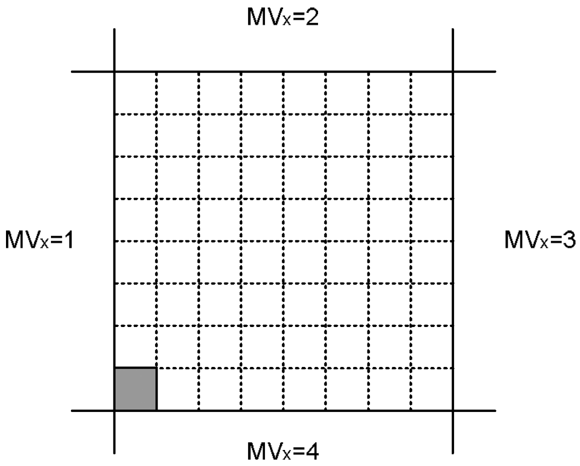


Fig. 7.2 Motion field interpolation. The per-pixel motion vector that is used to predict the highlighted pixel is estimated using bilinear interpolation from the motion vectors of the four neighboring blocks. In this example, for each of the four neighboring blocks, the horizontal component of the motion vector is shown.

decoded pictures, and (b) techniques where information from multiple buffered frames is used to derive a concealment block from a single pre-determined correctly decoded frame.

7.2.1 Concealment Blocks from Multiple References

In a video codec that supports long-term motion-compensated prediction, three parameters have to be estimated to obtain a CMV: the spatial MV horizontal and vertical components, and the temporal reference index. In [2] it was proposed to use MFI to recover the spatial component for the lost block. The per-pixel spatial components of the CMV are then combined with the temporal reference indices of the four neighboring blocks to produce four concealment blocks. In one variant, the block that minimizes the BM metric is selected as the final concealment block. In another variant, the four concealment blocks are *averaged* to produce the final *multihypothesis* concealment block. A generic illustration of concealment from multiple references is shown in Figure 7.3.

BM was also adopted in [79]. For each of the N reference frames, a search is conducted to obtain the best boundary matched block.

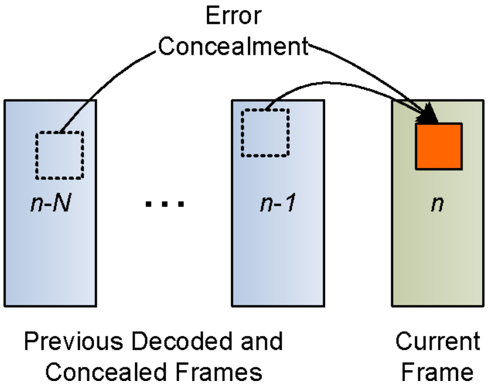


Fig. 7.3 Concealment from multiple references.

This results in the derivation of N concealment *hypothesis* blocks. Similar to the previous work in [2], the problem is now reformulated to that of obtaining the final concealment block given the N hypothesis blocks. Three methods are proposed for the derivation of the final concealment block:

(a) The *constant weight* method averages all hypotheses.

(b) The *adaptive weight* method uses only the two temporally closest hypotheses. The boundary sum of squared difference (SSD) measures of the two hypotheses are used to obtain adaptive weights. A larger weight is afforded to the hypothesis with the lower SSD measure.

(c) An *iterative motion search* method refines the CMVs by keeping one of them constant until a cost converges, in a process that is similar to the iterative algorithm for joint bi-predictive motion estimation in [115].

A different approach for the derivation of the candidate concealment blocks was presented in [54]. The temporal activity of error-free neighboring pixels is first used to classify a block into foreground or background. If the corrupted block is in the background, then *temporal replacement* with the co-located block in the previous frame is used. If it is in the foreground, then candidate concealment blocks in each of the multiple frames are obtained similar to [79], but this time by minimizing a BM metric variant, which is evaluated on the four closest rows/columns of the four adjacent error-free blocks. The three best match blocks are subsequently sorted according to that metric, which is the main contribution over [79] that considered each one as equally important. According to the relative metric difference, either the best one or the *average* of the best two is selected as the concealment block.

Another family of EC algorithms uses "bi-directional" EC [18, 28] to conceal entire lost frames. Let us consider the case where frame n has been lost and frames $n-1$ and $n+1$ have been correctly received. Frame $n-1$ has been decoded and can be used as an EC reference. However, unless frame $n+1$ is an intra-frame coded frame or it was coded by using a prediction reference other than frame n, then we cannot decode it in full. The best one can do is decode the intra-frame coded blocks and inter-frame coded blocks that refer to frames other than frame n. This double hypothesis EC uses the same reference frame (the last correctly decoded frame $n-1$) but two sets of motion vectors to yield the two blocks: one with pixel-level MVs (essentially MFI) for forward prediction and a second one with pixel-level MVs for backward prediction as shown in Figure 7.4. The final concealment block was obtained by averaging the above two blocks.

A different approach for the derivation of the CMV is presented in [16] where the decision on the EC of the current block is made using

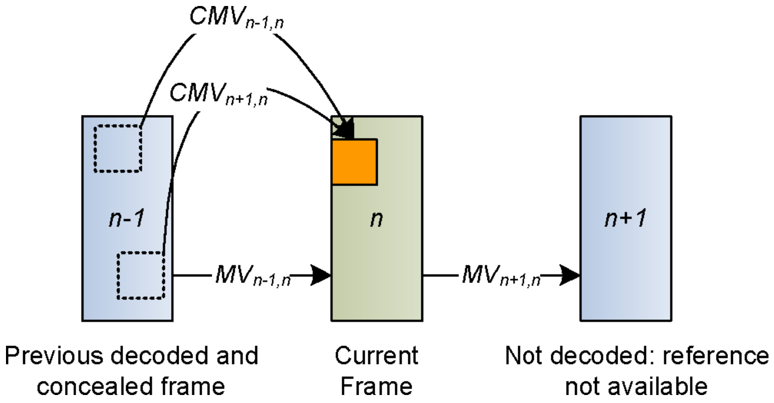

$CMV_{n-1,n}$
$CMV_{n+1,n}$
n-1
n
n+1
$MV_{n-1,n}$
$MV_{n+1,n}$
Previous decoded and concealed frame
Current Frame
Not decoded: reference not available

Fig. 7.4 A bi-directional error concealment algorithm.

a classification tree that has been trained on collected training data. Two reference frames are available for motion compensation: the short-term (ST) and the long-term (LT) frame. The lost block is concealed as either the co-located or a motion-compensated block (using the median MV predictor from neighboring blocks) from the ST or LT frames. The tree selects one of the above four concealment blocks using as input parameters: (a) the reference frame indices of the neighboring blocks, (b) a binary index that indicates whether a neighboring block would have been better concealed by either the ST or LT frame, and (c) the distortion between each neighboring MB and all its possible four concealment blocks. Apart from efficient error concealment, this method also results in substantial computational savings since one can avoid the costly BM operations.

The issue of computational complexity for EC was treated again in [63]. The authors proposed using the similarities in the block matching metrics to constrain the number of evaluated reference frames.

While temporal EC can be quite efficient, there are cases where spatial EC can perform better. A scene change for example will render temporal information useless. Newly appeared objects with no temporal correspondence will also benefit from spatial concealment, provided portions of the objects survive the transmission process. Spatial concealment will also be beneficial if the missing area consists of a regular texture. In [58], a hybrid approach that combined the algorithms of [79] and [54] jointly with spatial error concealment was proposed. For blocks that were coded as intra it was proposed to apply spatial error concealment.

7.2.2 Concealment Block from a Single Reference

Concealment of whole-frame losses is more challenging since no neighborhood context is available. Motion vectors cannot be estimated from motion vectors of neighboring blocks. In [5], it is proposed to address whole-frame losses with an approximate solution of the optical flow equation. The intuitive meaning of the OF equation is that intensity remains constant along a motion trajectory. The OF equation is solved through multiple reference frames stored in the reference frame buffer of an H.264/MPEG-4 AVC decoder. The MVs are first estimated and are then used to recover the missing intensity information (the missing frame). To simplify the solution, the MV field for a given frame is assumed to depend only on previous intensities.

In [4], the authors address again the issue of whole-frame loss concealment. The proposed algorithm uses OF estimation as in [5] to first estimate the missing MVs and then through those to derive the missing areas from correctly decoded frames. For picture regions that cannot be retrieved with this first step, BM is used to estimate those pixels from multiple past decoded frames. The use of boundary matching is made possible by the fact that the first optical flow-based step concealed areas large enough that they can subsequently serve as boundaries to help drive a BM algorithm. Even though this algorithm may use multiple frames for BM, its most critical part, the first step, conceals portions of the frame using a single reference. This algorithm is illustrated in Figure 7.5.

Multiple frames were used in [64] to select the best concealment block in the last correctly decoded frame. The matching criterion was based on boundary matching. In contrast to previous work, it was extended to consider the impact of the concealment on multiple subsequent frames. It was conjectured that a good concealment block ought to minimize block boundary variance as motion propagates to subsequently coded frames. Such an algorithm heavily depends on neighborhood information not only in the current damaged frame but also in the required subsequent frames. Consequently, it does not lend itself to concealment of whole-frame losses. In [73], the authors extended this algorithm to properly consider whole-frame losses. The subsequent frames are initially partially decoded for blocks using intra-frame coding modes and inter-frame coding modes that refer to frames prior to the damaged frame t. In the next step, the missing blocks in the subsequent frames are assigned a priority index that considers the availability of neighbors that may be used to reconstruct them. The missing blocks of frame $t + 1$ with the MVs decoded from the bitstream are used to derive their reference blocks in frame t, which are assigned the priority indices from their counterparts in frame $t + 1$. The MBs in t are then concealed in this order while also taking into account future

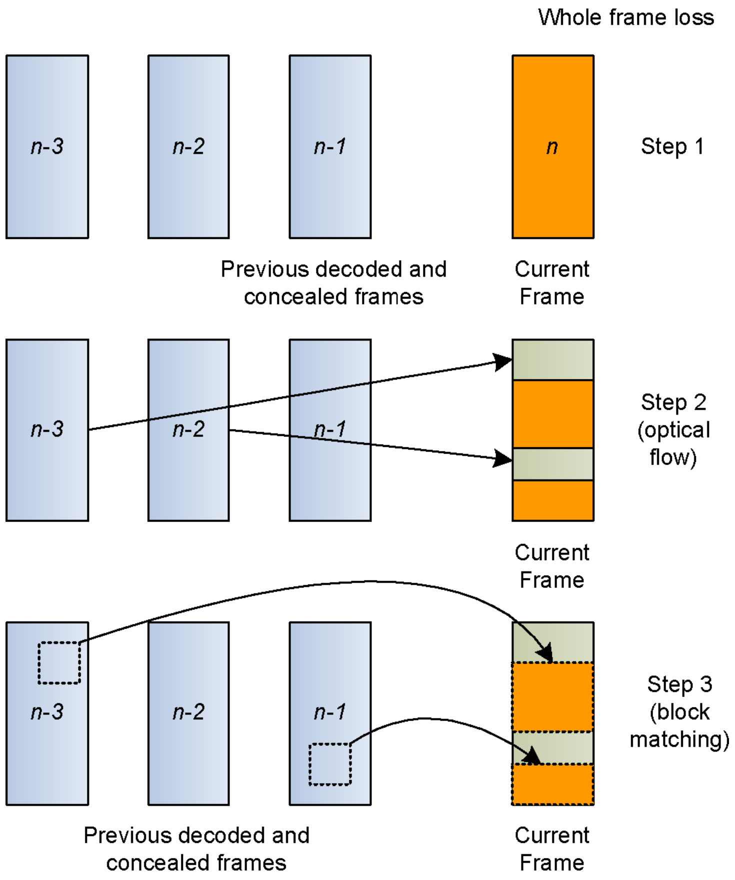


Fig. 7.5 A hybrid error concealment algorithm that combines optical flow estimation and block matching.

impact as in [64]. This is made possible as the order with which this operation is done ensures that there are boundaries to evaluate in the subsequent frames. After the concealment of each block in frame t, the dependent respective blocks in $t+1,\ldots,t+L$, $L>1$ are also decoded. This process is then iterated until all blocks have been concealed.

8

Experimental Investigation

In this last section, we attempt to quantify the performance of some widely used algorithms that utilize multiple reference frames for motion-compensated prediction. We also explore the effect of the motion parameters overhead in a compressed video bit stream.

8.1 Experimental Setup

For our experiments, we used version 12.4 of the Joint Model (JM) reference software of the state-of-the-art H.264/MPEG-4 AVC video coding standard [1, 84]. Six image sequences were used, all at a spatial resolution of 352×288 pixels and at 30 frames per second. The sequences are: (a) “carphone”, (b) “container”, (c) “foreman”, (d) “mobile-calendar”, (e) “stefan”, and (f) “table tennis”.

A quick look-up table of our coding configuration is in Table 8.1. The detailed explanation follows. The encoder was configured to use the context-adaptive binary arithmetic coding (CABAC) method for entropy coding of the motion vectors and the intra-frame and inter-frame prediction residuals. The Main Profile of the standard was used that allows the use of generalized B-coded frames and weighted prediction for motion compensation. The group of frames (GOP) size

Table 8.1. Coding configuration for the experimental evaluation.

Coding Tool	Value
Entropy coding	CABAC
Profile	Main (77)
GOP	Infinite
QP allocation	Fixed — no rate control
Fast motion search	EPZS
Motion search range	64
Integer-pel ME metric	SAD
Half-pel ME metric	SAD
Quarter-pel ME metric	SATD (Hadamard)
Slice structure	One per frame
Lagrangian λ	Default values
Quantization offsets	Default values
Adaptive rounding	Enabled
Joint bi-pred. ME	Enabled
Bi-pred. iterations	4
Bi-pred. search range	64

was set to infinite, which resulted in a single intra-coded frame at the beginning of the coded sequence. The quantization parameter (QP) was kept the same for both luma and chroma blocks. Rate control was not used. The fixed QP allocations are described in the following sections as we discuss and evaluate prediction structures. The motion search method used was EPZS and the motion search range for the EPZS algorithm was set to 64 pixels in each direction. Integer-pixel and half-pixel motion estimation used the SAD metric for distortion calculation, while quarter-pixel motion estimation adopted the sum of absolute transformed differences (SATD) using the Hadamard transform. A single slice was used to encode each picture. The default Lagrangian λ parameters were used for all instances of Lagrangian optimization in the JM reference software. The default quantization offsets matrix was used, while adaptive rounding was also enabled to improve the forward quantization process. When hierarchical frames and B-coded frames were used, the following features of the reference software were enabled: (a) joint bi-predictive motion estimation for 16 × 16 blocks was used unless noted, (b) up to four iterations were conducted during joint bi-predictive motion estimation for 16 × 16 blocks, and (c) the search range was constrained to 64 pixels.

8.2 Coding Efficiency Analysis

The results of our experiments are illustrated in Tables 8.2–8.4. The tables study rate-distortion performance with the help of the Bjontegaard metric [8]. The following QPs were evaluated: 12, 16, 20, 24, 28, 32, 36, 40. The Bjontegaard PSNR gain yields the average gain in PSNR (dB) for the same number of bits, while the Bjontegaard bit savings yield the average savings in bits for the same resulting PSNR. All results in the tables are given as a performance difference with respect to the basic IPPPP coding uni-predictive structure with a single reference picture for motion-compensated prediction. In that structure all frames were allocated the same QP.

8.2.1 Multiple Reference Frames

First, we evaluate the performance of multiple reference frames for motion-compensated prediction as discussed in Section 3.

Table 8.2. Performance of various coding configurations for "Carphone" and "Container".

	Carphone		Container	
Coding configuration	Bit savings	PSNR gain	Bit savings	PSNR gain
2 reference frames	−4.72%	0.22	−3.37 %	0.13
4 reference frames	−8.30%	0.39	−4.43 %	0.18
8 reference frames	−9.12%	0.44	−4.99 %	0.21
16 reference frames	−9.07%	0.43	−4.87 %	0.20
16 × 16	12.17%	−0.52	8.35%	−0.34
8 × 8–16 × 16	2.45%	−0.11	1.69%	−0.07
integer MVs	26.29%	−1.06	78.79%	−2.21
half-pel MVs	11.38%	−0.49	30.13%	−1.13
IBBB	−5.61%	0.27	−2.57%	0.11
IBBB w/o WP	−4.58%	0.23	−2.35%	−0.07
IBBB w/o Bipred.	−4.85%	0.23	−0.16%	−0.01
IBBB w/o WP, Bipred.	−3.84%	0.19	−4.29%	−0.16
IbbP	−16.70%	0.83	−32.96%	1.60
IbbP w/o WP	−16.37%	0.81	−28.03%	1.34
IbbP w/o Bipred.	−15.50%	0.76	−32.81%	1.59
IbbP w/o WP, Bipred.	−15.34%	0.75	−27.61%	1.32
IbBbP	−20.30%	1.00	−37.65%	1.93
IbBbP w/o WP	−20.46%	1.02	−35.71%	1.83
IbBbP w/o Bipred.	−19.13%	0.94	−37.39%	1.91
IbBbP w/o WP, Bipred.	−19.31%	0.94	−35.50%	1.81

Table 8.3. Performance of various coding configurations for “Foreman” and “Mobile”.

	Foreman		Mobile	
Coding configuration	Bit savings	PSNR gain	Bit savings	PSNR gain
2 reference frames	−4.24%	0.22	−7.39%	0.47
4 reference frames	−7.49%	0.39	−13.60%	0.93
8 reference frames	−8.25%	0.43	−15.00%	1.04
16 reference frames	−8.21%	0.43	−14.98%	1.04
16 × 16	14.71%	−0.66	9.30%	−0.58
8 × 8–16 × 16	3.43%	−0.17	2.94%	−0.20
integer MVs	64.77%	−2.32	107.18%	−4.46
half-pel MVs	19.68%	−0.86	34.12%	−1.85
IBBB	−4.51%	0.26	−11.83%	0.85
IBBB w/o WP	−3.76%	0.22	−10.80%	0.79
IBBB w/o Bipred.	−3.18%	0.19	−10.82%	0.76
IBBB w/o WP, Bipred.	−2.44%	0.16	−9.98%	0.72
IbbP	−19.56%	1.07	−30.71%	2.26
IbbP w/o WP	−18.73%	1.02	−29.83%	2.18
IbbP w/o Bipred.	−18.47%	1.00	−28.91%	2.09
IbbP w/o WP, Bipred.	−17.84%	0.96	−28.57%	2.06
IbBbP	−22.80%	1.27	−35.07%	2.69
IbBbP w/o WP	−22.68%	1.26	−35.15%	2.71
IbBbP w/o Bipred.	−21.73%	1.19	−33.31%	2.51
IbBbP w/o WP, Bipred.	−21.70%	1.19	−33.36%	2.52

Table 8.4. Performance of various coding configurations for “Stefan” and “Table”.

	Stefan		Table	
Coding configuration	Bit savings	PSNR gain	Bit savings	PSNR gain
2 reference frames	−2.37%	0.15	−1.95%	0.10
4 reference frames	−4.42%	0.29	−3.41%	0.18
8 reference frames	−4.88%	0.32	−3.90%	0.20
16 reference frames	−4.86%	0.32	−3.86%	0.20
16 × 16	12.91%	−0.72	13.26%	−0.62
8 × 8–16 × 16	3.83%	−0.22	2.72%	−0.14
integer MVs	75.86%	−3.19	32.73%	−1.37
half-pel MVs	24.08%	−1.25	11.03%	−0.51
IBBB	−4.56%	0.33	−6.22%	0.36
IBBB w/o WP	−4.49%	0.32	−5.73%	0.34
IBBB w/o Bipred.	−3.92%	0.28	−5.62%	0.32
IBBB w/o WP, Bipred.	−3.81%	0.28	−5.11%	0.30
IbbP	−10.65%	0.70	−15.75%	0.81
IbbP w/o WP	−9.84%	0.63	−13.68%	0.71
IbbP w/o Bipred.	−9.49%	0.60	−14.82%	0.75
IbbP w/o WP, Bipred.	−8.96%	0.56	−12.92%	0.66
IbBbP	−13.63%	0.93	−21.43%	1.15
IbBbP w/o WP	−13.27%	0.91	−20.93%	1.13
IbBbP w/o Bipred.	−12.44%	0.82	−20.37%	1.08
IbBbP w/o WP, Bipred.	−12.09%	0.80	−19.94%	1.07

Configurations using 2, 4, 8, and 16 reference pictures were tested. For all tested sequences we observe in both tables that RD performance generally improves with more reference pictures. However, these gains prove negligible or slightly negative when going from 8 to 16 reference frames. Depending on the content, a larger number of reference frames may not necessarily guarantee better compression efficiency. This is in part explained by the fact that indexing a large number of reference frames increases the reference index overhead and implicitly helps reduce the efficiency of motion vector prediction and coding. High motion sequences such as “stefan” and “table” benefit the least, while “mobile” benefits the most due to the presence of spatial aliasing, which was also pointed out in [14].

8.2.2 Block Sizes for Motion Compensation

Second, we evaluate the efficiency of motion-compensated prediction with different sets of block sizes for motion compensation. The reference results were obtained using all block sizes supported by the standard. We then constrain motion compensation to use (a) only 16×16 blocks and (b) only 16×16, 16×8, 8×16, and 8×8 blocks. Performance is degraded for both constrained configurations, and is worse for set (a). As expected, the largest performance drop is registered for high motion sequences, such as “table” and “stefan”. Furthermore, we note that constraining block sizes to those of set (b) degrades the RD performance only slightly (2–4%). These conclusions are highly tied to the spatial resolution of the video content: it is known in general that smaller block sizes become less significant when considering, e.g., high-definition content (1920×1080 pixels).

8.2.3 Precision of Motion Compensation

Third, we evaluate motion compensation when constrained to use only integer-pixel and half-pixel motion vectors [39]. To simulate these two cases in a realistic manner, we constrained the motion estimation and also modified the entropy coding and decoding of the motion vectors so that motion vectors are normalized to the specific highest precision level. We observe that performance drops dramatically with the decrease in the precision of motion compensation. Furthermore, the degradation is more severe in highly textured and aliased content such as that in “container” and “mobile”.

8.2.4 Multihypothesis Motion-Compensated Prediction

Fourth, we evaluate multihypothesis prediction with two hypotheses as discussed in Section 4. Let upper-case letters be used to denote reference frames and lower-case letters be used to denote non-reference/disposable frames. The first coding configuration IBBBB employs reference B-coded frames that use as references previously coded B-coded frames. This is a low-delay coding configuration since the references are frames that are in the past in display order. All frames are assigned the same QP and a single reference was used in each prediction list. The second coding configuration IbbP uses disposable B-coded frames that are assigned coarser QPs (incremented by two) compared to the I- and P-coded frames. Such a configuration codes frame 0 as I, then frame 3 as P, and then uses bi-prediction to code frames 1 and 2. This is then repeated periodically by coding one frame out of three as a P-coded frame. The third coding configuration IbBbP uses hierarchical pictures and hierarchically adapted QPs. These QPs of the prediction structure are adapted according to the frame’s significance: The QP of the B frame is incremented by one while the QP of the b frames is incremented by two over the QP of the I and P frames. The coding order is as follows: frame 0 is coded as I, frame 4 is coded as P, frame 2 is then coded as a B-coded frame that is retained as a reference, and frames 1 and 3 are finally coded as non-reference B-coded frames. In general, the third configuration performs the best. The large difference in performance between the second and third configurations vs. the first is attributed to the use of disposable B-coded frames, bi-prediction, and of a hierarchical prediction structure. The above structures are illustrated in Figure 8.1.

We note that the conclusions from the above two paragraphs are different from the conclusions of (b) and (c) in Section 4.5. Increasing the motion compensation fractional-pixel precision seems in general more efficient compared to increasing the number of hypotheses. We note

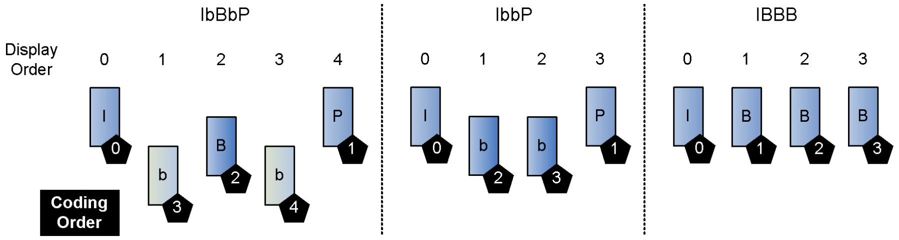


Fig. 8.1 The evaluated prediction structures.

though that MHP in Section 4.5 was generalized, and it disregarded the effect of the bit cost incurred to index the additional hypotheses, while our experiment here is constrained in terms of bits used to code MVs and hypothesis indices. Our experiment is a reliable indication of real-world performance of the evaluated prediction structures.

8.2.5 Joint Bi-Predictive Motion Estimation

Fifth, we evaluate the performance gain of joint bi-predictive motion estimation [115], which is found to lead to savings of approximately 1–2% in bits for the majority of configurations. We note that these numbers reflect all types of frames. Since this tool only benefits B-coded frames, the bi-predictive coding gain will in fact be much higher since B-coded frames require fewer bits to code than the I- and P-coded frames in the sequence. Furthermore, our sequences did not include the type of content (e.g., cross-fades: fading sequences from one scene to a new scene) where iterative motion estimation would have been more beneficial. Furthermore, the algorithm implemented in this version of the JM software only supports 16 × 16 blocks and is therefore suboptimal.

8.2.6 Implicit vs. Explicit Weighted Prediction for Motion Compensation

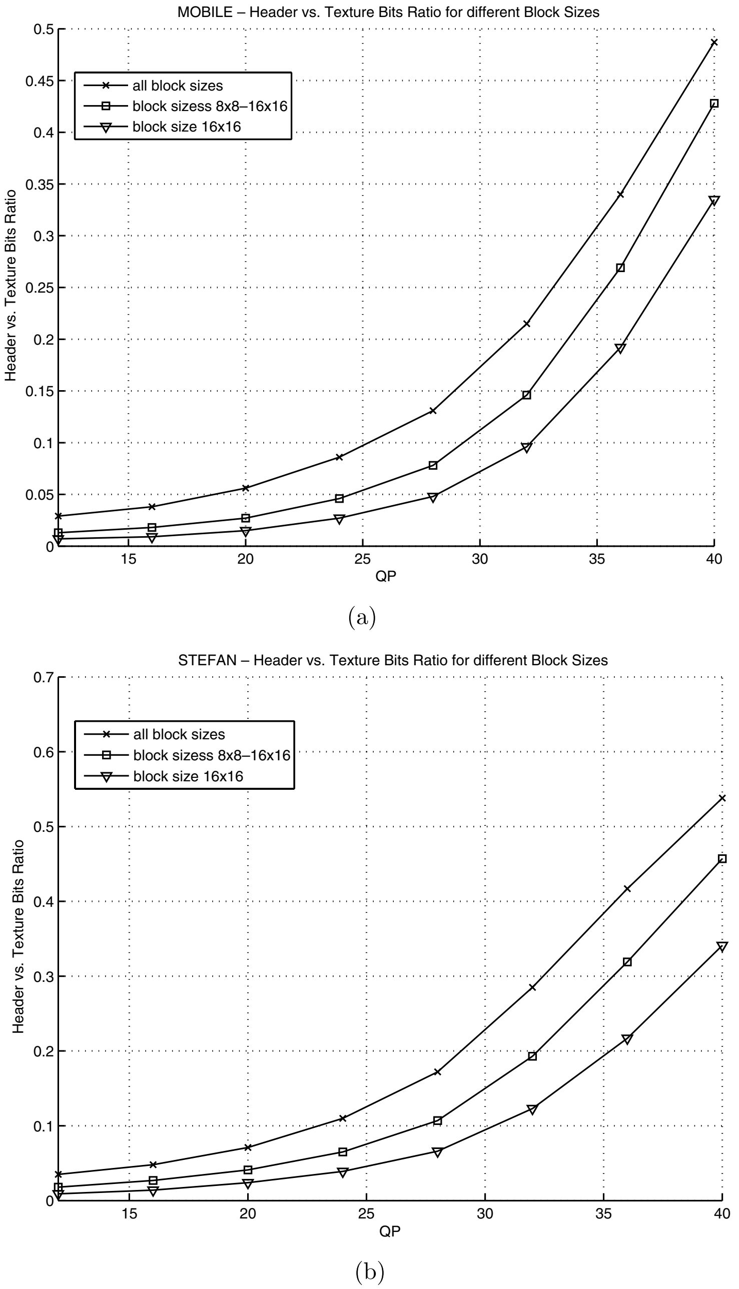
Sixth, we evaluate the performance gain of implicit and explicit weighted prediction for motion compensation in H.264/MPEG-4 AVC. This experiment involved enabling weighted prediction (WP) for P- and B-coded frames, and generating multiple picture parameter sets that enabled switching among no WP, implicit WP, and explicit WP, by satisfying a rate-distortion optimality criterion. The gain is close to 1–2% for most sequences, but it is noteworthy that it is close to 4% for "container". In addition, for the IBBBB configuration, the difference is more than 5%, and disabling weighted prediction leads to an increase in bit rate for this particular image sequence. Note that none of the above content contained large local or global illumination changes such as *fades*. We thus evaluated the efficiency of weighted prediction in an *artificially* created fade sequence based on 30 frames from each one of the "silent" and "tempete" image sequences. The bit rate savings with weighted prediction range from 30% at high bit rates to 60% at low bit rates.

8.3 Motion Parameters Overhead Analysis

In Figures 8.2–8.4, we plot the ratio of header bits vs. the entire bit budget for varying QP values. The header bits include bits spent for header information, reference indices, motion vectors, weighted prediction parameters, sequence parameter sets, etc. The remaining bits are primarily bits used to code the transform coefficients (the texture information) of the coded bit stream. The goal of this evaluation is to study the impact of coding tools, and the evolution of those coding tools, on the ratio of header vs. texture bits in the compressed video bit stream. The IPPPP prediction structure was used in all of the following experiments.

In Figure 8.2(a) and (b) we illustrate the header bit ratio for different sets of blocks sizes that are used during motion-compensated prediction. A single reference frame and quarter-pixel precision were adopted for motion compensation. In general, the ratio increases as the QP increases or equivalently the bit rate decreases. Hence, for low bit rates, a large part of the bit stream consists of header/motion information. For high bit rates, the texture information outweighs header/motion information by a significant margin. We thus observe that as more block sizes are used, there is an increase in the ratio of header bits to texture bits.

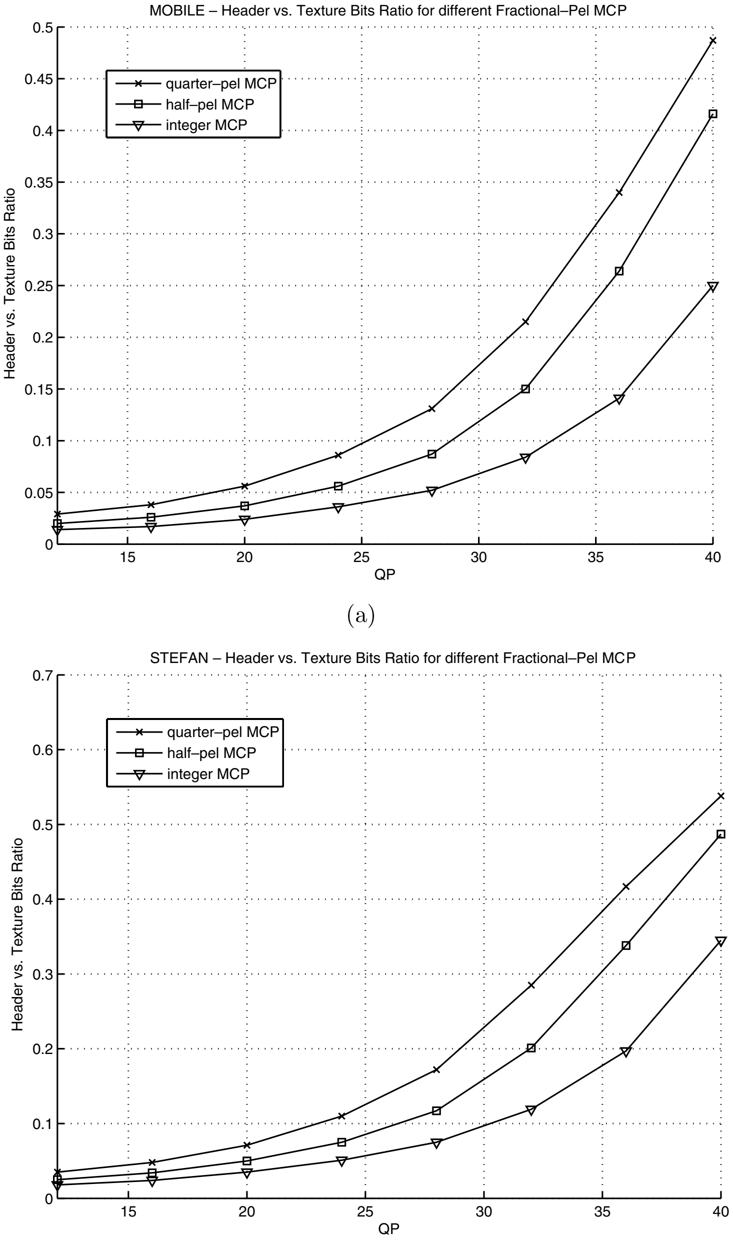
In Figure 8.3(a) and (b), we illustrate the header bit ratio for different cases of fractional-pixel precision for motion-compensated



(a)

(b)

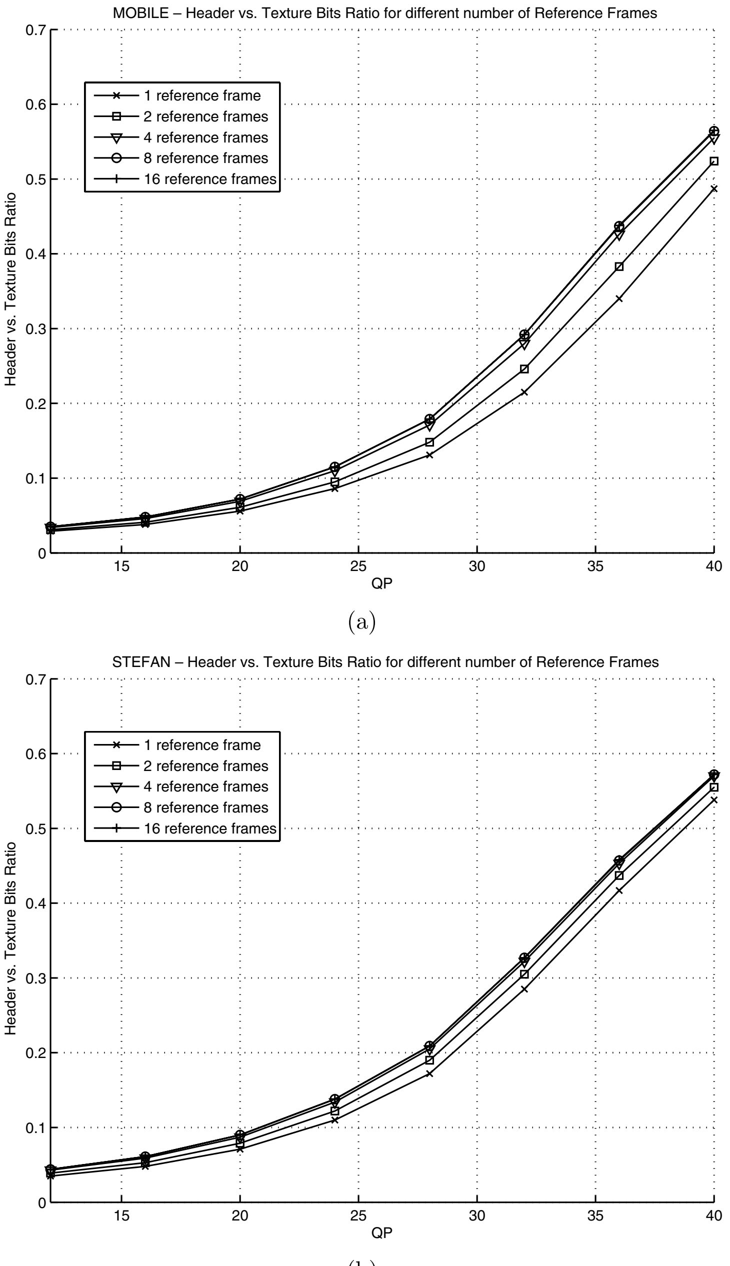
Fig. 8.2 Header vs. texture bit ratios. (a) Varying block sizes for motion compensation. "Mobile". (b) "Stefan".



(a)

(b)

Fig. 8.3 Header vs. texture bit ratios. (a) Varying fractional-pel precision for motion compensation. “Mobile”. (b) “Stefan”.



(a)

(b)

Fig. 8.4 Header vs. texture bit ratios. (a) Varying number of references for motion compensation. “Mobile”. (b) “Stefan”.

prediction. We evaluate integer-, half-, and quarter-pixel motion-compensated prediction. A single reference frame and all possible block sizes were enabled for motion compensation. We observe that as the precision of the motion-compensated prediction becomes finer, there is an increase in the bit overhead as a ratio of the total bits required to code the sequence. Still, this translates to substantial coding gains as witnessed in Tables 8.2–8.4.

In Figure 8.4(a) and (b) we illustrate the header bit ratio for different numbers of reference frames for multiple-reference motion-compensated prediction. All possible block sizes and quarter-pixel precision were adopted for motion compensation. The ratio increases with the number of references, however, the rate of increase is smaller than the corresponding increase for finer fractional-pixel prediction (Figure 8.3(a) and (b)) and different sets of block sizes (Figure 8.2(a) and (b)).

9

Conclusions

In this survey we described the development of methods that adopt multiple reference frames for motion-compensated prediction. Apart from improving compression efficiency, multiple frame prediction schemes can improve the error resilience of a video codec and can also be employed for more efficient error concealment. When including the use of multiple reference frames in a coder, one is adding a coding option that presents a new trade-off among rate, distortion, error resilience, and complexity, among other factors. In this survey, we have focussed on this specific feature, and obviously there are many other features of a codec which offer different trade-offs among the various parameters of the system. Over time, as complexity costs continue to decrease, one expects to see more widespread adoption of multiple reference frames.

A

Rate-Distortion Optimization

Hybrid codecs may compress a block of pixels with intra-frame or inter-frame coding modes, the latter of which require motion estimation to estimate the motion vectors that drive the motion compensation process. A modern video codec supports multiple such coding modes (*coding tools*), which can be signaled on a block basis. It is thus critical to establish criteria for the selection of the best possible coding modes and the estimation of good motion vectors. A straightforward criterion would be to select the mode or motion vector that minimizes some fidelity or distortion criterion, such as the sum of absolute differences (SAD) or sum of squared differences (SSD) as noted in Section 1.1. Such a strategy, however, fails to account for the number of bits required to code the block. The problem of coding optimization in the context of video compression can be solved using rate-distortion optimization principles. Rate-distortion optimization is facilitated through Lagrangian minimization that provides a solution that attains high coding efficiency. An excellent treatment on the subject of rate-distortion optimization can be found in [86]. We next briefly describe the basic rate-distortion optimization principles.

Let $D(b, M, Q)$ denote the distortion measure (e.g., SAD or SSD, among others) for the current block b, when it is predicted using coding mode M, and the prediction residual is coded using a quantization parameter (QP) Q. In general, a low QP value means the information is rounded off with great precision, whereas a high QP value means that a coarse approximation was done in the compression of that information. Fine quantization yields high quality with modest compression, while coarse quantization yields lower quality video with high compression ratios. Note that $D(b, M, Q)$ is not merely the prediction distortion but the end-to-end distortion after taking into account the quantization process. Let also $R(b, M, Q)$ denote the sum of the bits used to signal the selected coding mode M for the current block b to the decoder and the bits used to signal the quantized prediction residual and related header information. Rate-distortion optimization principles can be applied to solve the problem of *coding mode selection* by minimizing the Lagrangian cost function $J(M|b, Q)$:

$$J(M|b, Q) = D(M|b, Q) + \lambda_{mode} \times R(M|b, Q). \tag{A.1}$$

The performance of Lagrangian minimization for coding mode selection depends on the selection of the Lagrangian parameter λ_{mode}, good values for which can be found in [86]. Lagrangian minimization provides significant coding benefits compared to legacy approaches that simply sought to minimize the distortion alone. However, we caution the reader that such an approach should not be considered *optimal*. First, there are no spatial dependency considerations: subsequently coded blocks may depend on previously coded blocks both for prediction of the motion model parameters and also for sample prediction when intra-frame coding modes are used. The consideration of the resulting distortion only for the current block leads to good coding efficiency for that specific block, but at the same time ignores the impact of this coding decision on subsequent blocks. Second, similar arguments hold for inter-frame prediction. During coding of frames used as prediction references for subsequently coded frames, one should preferably account for this future impact. In practice though, computational and memory resources are constrained, and it can be excessively costly to estimate the future impact of each individual coding decision.

Lagrangian minimization is also useful when estimating motion vectors during motion search. While a naive approach that selects the motion vector that results in the minimum error may seem reasonable, rate-distortion theory shows that the number of bits used to signal the motion vector should be considered as well [38]. Assuming that motion vector $\mathbf{v}$ is used for motion-compensated prediction of the current block b, we write down the resulting prediction distortion and bit cost values as $D(\mathbf{v}|b)$ and $R(\mathbf{v}|b)$. The motion vectors are estimated by minimizing the Lagrangian cost $J(\mathbf{v}|b)$:

$$J(\mathbf{v}|b) = D(\mathbf{v}|b) + \lambda_{motion} \times R(\mathbf{v}|b). \tag{A.2}$$

Note that in practice both Lagrangian parameters λ_{motion} and λ_{mode} are often conditioned with respect to the quantization parameter Q. Furthermore, the coding efficiency of a system employing Lagrangian minimization depends heavily on a good selection of the λ parameters. The above motion estimation process can be further enhanced when considering the resulting end-to-end distortion that also includes the bits used to code the residuals. Such an approach is though rarely practiced during motion estimation due to its very high computational requirements. In practice, a metric (SAD or SSD) of the block prediction distortion suffices as the distortion value, while the rate usage includes the number of bits required to code the motion parameters.

Acknowledgments

We are grateful to the anonymous reviewers for the comments and guidance they provided. This work was supported in part by the National Science Foundation, the Center for Wireless Communications at the University of California, San Diego, the Office of Naval Research, the UC Discovery Grant Program, and Dolby Laboratories, Inc.

References

[1] Advanced video coding for generic audiovisual services. ITU-T Recommendation H.264, 2005.

[2] M. E. Al-Mualla, C. N. Canagarajah, and D. R. Bull, "On the performance of temporal error concealment for long-term motion-compensated prediction," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 376–379, September 2000.

[3] M. E. Al-Mualla, C. N. Canagarajah, and D. R. Bull, "Simplex minimization for single- and multiple-reference motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 12, pp. 1209–1220, December 2001.

[4] S. M. Amiri and I. V. Bajic, "A novel noncausal whole-frame concealment algorithm for video streaming," in *Proceedings of Tenth IEEE International Symposium on Multimedia*, pp. 154–159, February 2008.

[5] S. Belfiore, M. Grangetto, E. Magli, and G. Olmo, "Concealment of whole-frame losses for wireless low bit-rate video based on multiframe optical flow estimation," *IEEE Transactions on Multimedia*, vol. 7, no. 2, pp. 316–329, April 2005.

[6] G. Bjontegaard. Response to Call for Proposals for H.26L. ITU-T SG16/Q15, VCEG, Document Q15-F-11, November 1998.

[7] G. Bjontegaard. Enhancement of the Telenor proposal for H.26L. ITU-T SG16/Q15, VCEG, Document Q15-G-25, February 1999.

[8] G. Bjontegaard. Calculation of Average PSNR Differences between RD Curves. ITU-T SG16/Q6, VCEG-M33, April 2001.

[9] F. Bossen and A. Tourapis, "Method and apparatus for video encoding and decoding using adaptive interpolation," U.S. Patent Application, US2006/0294171A1, December 2006.

[10] S. Brofferio and V. Corradi, "Videophone coding using background prediction," in *Proceedings of European Signal Processing Conference*, vol. 2, pp. 813–816, 1986.

[11] M. Budagavi and J. D. Gibson, "Multiframe block motion compensated video coding for wireless channels," in *Proceedings of Asilomar Conference on Signals, Systems and Computers*, pp. 953–957, 1996.

[12] M. Budagavi and J. D. Gibson, "Random lag selection in multiframe motion compensation," in *Proceedings of IEEE International Symposium on Information Theory*, p. 410, August 1998.

[13] M. Budagavi and J. D. Gibson, "Multiframe video coding for improved performance over wireless channels," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 252–265, February 2001.

[14] A. Chang, O. C. Au, and Y. M. Yeung, "A novel approach to fast multiframe selection for H.264 video coding," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, pp. 413–416, May 2003.

[15] V. Chellappa, P. C. Cosman, and G. M. Voelker, "Dual frame motion compensation with uneven quality assignment," in *Proceedings of IEEE Data Compression Conference*, pp. 262–271, March 2004.

[16] V. Chellappa, P. C. Cosman, and G. M. Voelker, "Error concealment for dual frame video coding with uneven quality," in *Proceedings of IEEE Data Compression Conference*, pp. 319–328, March 2005.

[17] M.-J. Chen, Y.-Y. Chiang, H.-J. Li, and M.-C. Chi, "Efficient multi-frame motion estimation algorithms for MPEG-4 AVC/JVT/H.264," in *Proceedings of IEEE International Symposium on Circuits and Systems*, May 2004.

[18] Y. Chen, K. Yu, J. Li, and S. Li, "An error concealment algorithm for entire frame loss in video transmission," in *Proceedings of Picture Coding Symposium*, December 2004.

[19] G. Cheung, "Near-optimal multipath streaming of H.264 using reference frame selection," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 653–656, September 2003.

[20] G. Cheung and W. Tan, "Loss-compensated reference frame optimization for multi-path video streaming," in *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 844–847, July 2005.

[21] H. Chung and A. Ortega, "Low complexity motion estimation algorithm by multiresolution search for long-term memory motion compensation," in *Proceedings of IEEE International Conference on Image Processing*, vol. 2, pp. 261–264, September 2002.

[22] H. Chung, A. Ortega, and A. Sawchuk, "Low complexity motion estimation for long term memory motion compensation," in *Proceedings of SPIE Visual Communication and Image Processing*, January 2002.

[23] H. Chung, D. Romacho, and A. Ortega, "Fast long-term motion estimation for H.264 using multiresolution search," in *Proceedings of IEEE International Conference on Image Processing*, vol. 1, pp. 905–908, September 2003.

[24] Codecs for videoconferencing using primary digital group transmission. ITU-T Recommendation H.120, 1993.

[25] S. Cui, Y. Wang, and J. E. Fowler, "Multihypothesis motion compensation in the redundant wavelet domain," in *Proceedings of IEEE International Conference on Image Processing*, vol. 2, pp. 53–56, September 2003.

[26] C. Duanmu, M. O. Ahmad, and M. N. S. Swamy, "A continuous tracking algorithm for long-term memory motion estimation," in *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 356–359, May 2003.

[27] F. Dufaux and F. Moscheni, "Background mosaicking for low bit rate video coding," in *Proceedings of IEEE International Conference on Image Processsing*, vol. 1, pp. 673–676, September 1996.

[28] J. Fan, Z. Zhang, and Y. Chen, "A new error concealment scheme for whole frame loss in video transmission," in *Proceedings of Picture Coding Symposium*, 2007.

[29] L. Fan, S. Ma, and F. Wu, "Overview of AVS video standard," in *Proceedings of IEEE International Conference on Multimedia and Expo*, vol. 1, pp. 423–426, 2004.

[30] M. Flierl and B. Girod, "Multihypothesis motion-compensated prediction with forward-adaptive hypothesis switching," in *Proceedings of International Picture Coding Symposium PCS*, pp. 195–198, April 2001.

[31] M. Flierl and B. Girod, "Multihypothesis motion estimation for video coding," in *Proceedings of IEEE Data Compression Conference*, pp. 341–350, March 2001.

[32] M. Flierl and B. Girod, "Generalized B pictures and the draft H.264/AVC video-compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 587–597, July 2003.

[33] M. Flierl, T. Wiegand, and B. Girod, "A locally optimal design algorithm for block-based multi-hypothesis motion-compensated prediction," in *Proceedings of IEEE Data Compression Conference*, pp. 239–248, April 1998.

[34] M. Flierl, T. Wiegand, and B. Girod, "Rate-constrained multi-hypothesis motion-compensated prediction for video coding," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 150–153, September 2000.

[35] T. Fukuhara, K. Asai, and T. Murakami, "Very low bit-rate video coding with block partitioning and adaptive selection of two time-differential frame memories," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 212–220, February 1997.

[36] S. Fukunaga, T. Nakai, and H. Inoue, "Error-resilient video coding by dynamic replacing of reference pictures," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 3, pp. 1503–1508, November 1996.

[37] B. Girod, "Motion-compensating prediction with fractional-pel accuracy," *IEEE Transactions on Communications*, vol. 41, no. 4, pp. 604–612, April 1993.

[38] B. Girod, "Rate-constrained motion estimation," in *Proceedings of SPIE Visual Communication and Image Processing*, vol. 2308, pp. 1026–1034, 1994.

[39] B. Girod, “Efficiency analysis of multi-hypothesis motion-compensated prediction for video coding,” *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 173–183, February 2000.

[40] M. Gothe and J. Vaisey, “Improving motion compensation using multiple temporal frames,” in *Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, vol. 1, pp. 157–160, May 1993.

[41] D. Hepper, “Efficiency analysis and application of uncovered background prediction in a low bit rate image coder,” *IEEE Transactions on Communications*, vol. 38, no. 9, pp. 1578–1584, September 1990.

[42] D. Hepper and H. Li, “Analysis of uncovered background prediction for image sequence coding,” in *Proceedings of Picture Coding Symposium*, pp. 192–193, 1987.

[43] J. R. Hidalgo and P. Salembier, “Long term selection of reference frame sub-blocks using MPEG-7 indexing metadata,” in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 669–672, April 2007.

[44] M. Horowitz, Demonstration of H.263++ Annex U Performance. ITU-T SG16/Q15, VCEG, Document Q15-J-11, May 2000.

[45] S. Hsu and P. Anandan, “Hierarchical representations for mosaic-based video compression,” in *Proceedings of Picture Coding Symposium*, pp. 395–400, March 1996.

[46] Y. Huang, Z. Liu, S. Goto, and T. Ikenaga, “Adaptive edge detection pre-process multiple reference frames motion estimation in H.264/AVC,” in *Proceedings of International Conference on Communications, Circuits and Systems*, pp. 787–791, July 2007.

[47] Y.-W. Huang, B.-Y. Hsieh, T.-C. Wang, S.-Y. Chien, S.-Y. Ma, C.-F. Shen, and L.-G. Chen, “Analysis and reduction of reference frames for motion estimation in MPEG-4 AVC/JVT/H.264,” in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 145–148, April 2003.

[48] Information technology — Generic coding of moving pictures and associated audio information: Video. ITU-T Recommendation H.262, 2000.

[49] M. Irani, S. Hsu, and P. Anandan, “Mosaic-based video compression,” in *Proceedings of SPIE Digital Video Compression: Algorithms and Technologies*, vol. 2419, February 1995.

[50] ISO/IEC 10918-1:1992 Information technology — Digital Compression and Coding of Continuous-Tone Still Images — Requirements and Guidelines. ISO/IEC JTC1/SC29/WG10, 1992.

[51] ISO/IEC 11172-1:1993 Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s. ISO/IEC JTC1/SC29/WG11, 1993.

[52] ISO/IEC 13818-2:2000 Information technology — Generic coding of moving pictures and associated audio information: Video. ISO/IEC JTC1/SC29/WG11, 1994.

[53] ISO/IEC 14496-2:1999 Information technology — Coding of audio-visual objects — Part 2: Visual. ISO/IEC JTC1/SC29/WG11, 1999.

[54] B. Jung, B. Jeon, M.-D. Kim, B. Suh, and S.-I. Choi, "Selective Temporal Error Concealment Algorithm for H.264/AVC," in *Proceedings of IEEE International Conference on Image Processing*, October 2004.

[55] C. Kim and C.-C. J. Kuo, "Efficient temporal search range prediction for motion estimation in H.264," in *Proceedings of IEEE 7th Workshop on Multimedia Signal Processing*, pp. 1–4, October 2005.

[56] C.-S. Kim, R.-C. Kim, and S.-U. Lee, "Robust transmission of video sequence using double-vector motion compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 9, pp. 1011–1021, September 2001.

[57] C.-S. Kim, S. Ma, and C.-C. J. Kuo, "Fast H.264 motion estimation with block-size adaptive referencing (BAR)," in *Proceedings of IEEE International Conference on Image Processing*, pp. 57–60, October 2006.

[58] M.-D. Kim, S.-I. Choi, and S.-W. Ra, "Hybrid error concealment method for H.264/AVC," in *Proceedings of the 7th International Conference on Advanced Communication Technology*, pp. 408–411, 2005.

[59] S. H. Kim, Y. K. Kim, K. H. Yang, and S. U. Lee, "Multiple reference frame based scalable video coding for low-delay Internet transmission," in *Proceedings of International Workshop on Very Low Bitrate Video Coding*, 2001.

[60] W.-Y. Kung, C.-S. Kim, and C.-C. J. Kuo, "Analysis of multi-hypothesis motion compensated prediction for robust video transmission," in *Proceedings of IEEE International Symposium on Circuits and Systems*, May 2004.

[61] R. Kutka, "Content-adaptive long-term prediction with reduced memory," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 817–820, September 2003.

[62] W. M. Lam, A. R. Reibman, and B. Liu, "Recovery of lost or erroneously received motion vectors," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 417–420, 1993.

[63] P.-J. Lee and M.-L. Lin, "Fuzzy logic based temporal error concealment for H.264 video," *ETRI Journal*, vol. 28, no. 5, pp. 574–582, October 2006.

[64] Y.-C. Lee, Y. Altunbasak, and R. M. Mersereau, "A temporal error concealment method for MPEG coded video using a multi-frame boundary matching algorithm," in *Proceedings of IEEE International Conference on Image Processing*, vol. 1, pp. 990–993, October 2001.

[65] A. Leontaris and P. C. Cosman, "Video compression for lossy packet networks with mode switching and a dual-frame buffer," *IEEE Transactions on Image Processing*, vol. 13, no. 7, pp. 885–897, July 2004.

[66] A. Leontaris and A. M. Tourapis, "Weighted prediction alternative to adaptive interpolation mechanisms," Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, JVT-AB033, July 2008.

[67] X. Li, E. Q. Li, and Y.-K. Chen, "Fast multi-frame motion estimation algorithm with adaptive search strategies in H.264," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004.

[68] Y. Liang, M. Flierl, and B. Girod, "Low-latency video transmission over lossy packet networks using rate-distortion optimized reference picture selection," in *Proceedings of IEEE International Conference on Image Processing*, vol. 2, pp. 181–184, September 2002.

[69] Y. J. Liang, E. Setton, and B. Girod, "Channel-adaptive video streaming using packet path diversity and rate-distortion optimized reference picture selection," in *Proceedings of IEEE Workshop on Multimedia Signal Processing*, pp. 420–423, December 2002.

[70] Z. Liang, J. Zhou, O. C. Au, and L. Guo, "Content-adaptive temporal search range control based on frame buffer utilization," in *Proceedings of IEEE 8th Workshop on Multimedia Signal Processing*, pp. 399–402, October 2006.

[71] S. Lin, S. Mao, Y. Wang, and S. Panwar, "A reference picture selection scheme for video transmission over ad-hoc networks using multiple paths," in *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 96–99, August 2001.

[72] S. Lin and Y. Wang, "Error resilience property of multihypothesis motion-compensated prediction," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 545–548, September 2002.

[73] Y. Liu, J. Bu, C. Chen, L. Mo, and K. He, "Multiframe error concealment for whole-frame loss in H.264/AVC," in *Proceedings of IEEE International Conference on Image Processing*, vol. 4, pp. 281–284, September 2007.

[74] N. Mukawa and H. Kuroda, "Uncovered background prediction in interframe coding," *IEEE Transactions on Communications*, vol. 33, no. 11, pp. 1227–1231, November 1985.

[75] H. G. Musmann, P. Pirsch, and H.-J. Grallert, "Advances in picture coding," *Proceedings of IEEE*, vol. 73, no. 4, pp. 523–548, April 1985.

[76] S. Nogaki and M. Ohta, "An overlapped block motion compensation for high quality motion picture coding," in *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 184–187, May 1992.

[77] M. T. Orchard and G. J. Sullivan, "Overlapped block motion compensation: An estimation-theoretic approach," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 693–699, September 1994.

[78] I. Park and D. W. Capson, "Dynamic reference frame selection for improved motion estimation time in H.264/AVC," in *Proceedings of IEEE Southwest Symposium on Image Analysis and Interpretation*, pp. 97–100, March 2008.

[79] Y. O. Park, C.-S. Kim, and S.-U. Lee, "Multi-hypothesis error concealment algorithm for H.26L video," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 465–468, September 2003.

[80] A. Puri, R. Aravind, B. G. Haskell, and R. Leonardi, "Video coding with motion-compensated interpolation for CD-ROM applications," *Signal Processing: Image Communication*, vol. 2, no. 2, pp. 127–144, 1990.

[81] F. H. P. Spaan, R. L. Lagendijk, and J. Biemond, "Error robust video coding based on H.263," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 472–476, October 1998.

[82] Standard for Television: VC-1 Compressed Video Bitstream Format and Decoding Process. SMPTE 421M, 2006.

[83] Y. Su and M.-T. Sun, "Fast multiple reference frame motion estimation for H.264," in *Proceedings of IEEE International Conference on Multimedia and Expo*, vol. 1, pp. 695–698, October 2004.

[84] G. Sullivan, T. Wiegand, D. Marpe, and A. Luthra, Text of ISO/IEC 14496-10 Advanced Video Coding 3rd Edition. ISO/IEC JTC 1/SC 29/WG11 N6540, July 2004.

[85] G. J. Sullivan, "Multi-hypothesis motion compensation for low bit-rate video coding," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 437–440, April 1993.

[86] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, pp. 74–90, November 1998.

[87] C. N. Taylor and S. Dey, "ORBit: An adaptive data shaping technique for robust wireless video clip communication," in *Proceedings of 37th Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1567–1571, November 2003.

[88] C.-W. Ting, W.-H. Lam, and L.-M. Po, "Fast block-matching motion estimation by recent-biased search for multiple reference frames," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 1445–1448, October 2004.

[89] C.-W. Ting, L.-M. Po, and C.-H. Cheung, "Center-biased frame selection algorithms for fast multi-frame motion estimation in H.264," in *Proceedings of IEEE International Conference on Neural Networks and Signal Processing*, pp. 1262–1265, December 2003.

[90] M. Tiwari and P. Cosman, "Dual frame video coding with pulsed quality and a lookahead window," in *Proceedings of IEEE Data Compression Conference*, pp. 372–381, March 2006.

[91] M. Tiwari and P. C. Cosman, "Selection of long-term reference frames in dual-frame video coding using simulated annealing," *IEEE Signal Processing Letters*, vol. 15, pp. 249–252, 2008.

[92] Y. Tomita, T. Kimura, and T. Ichikawa, "Error resilient modified inter-frame coding system for limited reference picture memories," in *Proceedings of Picture Coding Symposium*, pp. 743–748, September 1997.

[93] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," in *Proceedings of SPIE Visual Communication and Image Processing*, January 2002.

[94] A. M. Tourapis, O. C. Au, and M. L. Liou, "Highly efficient predictive zonal algorithms for fast block-matching motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 10, pp. 934–947, October 2002.

[95] A. M. Tourapis, H. Y. Cheong, O. C. Au, and M. L. Liou, "N-Dimensional zonal algorithms. The future of block based motion estimation?," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 506–509, October 2001.

[96] A. M. Tourapis, F. Wu, and S. Li, "Enabling VCR functionalities in streaming media (intelligent interactive streaming I2 stream)," in *Proceedings of IEEE International Conference on Information Technology: Research and Education*, pp. 1–5, August 2003.

[97] H.-Y. C. Tourapis and A. M. Tourapis, "Fast motion estimation within the H.264 codec," in *Proceedings of IEEE International Conference on Multimedia and Expo*, vol. 3, pp. 517–520, July 2003.

[98] Y.-C. Tsai and C.-W. Lin, "H.264 Error resilience coding based on multihypothesis motion compensated prediction," in *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 952–955, July 2005.

[99] W. Tu and E. Steinbach, "Proxy-based reference picture selection for real-time video transmission over mobile networks," in *Proceedings of IEEE International Conference on Multimedia and Expo*, July 2005.

[100] S. C. Vadapalli, H. Shetiya, and S. Sethuraman, "Efficient alternative to intra refresh using reliable reference frames," in *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 124–127, July 2007.

[101] N. Vasconcelos and A. Lippman, "Library-based image coding," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. v, pp. 489–492, April 1994.

[102] A. Vetro, P. Pandit, A. Smolic, and Y.-K. Wang, Joint draft 7.0 on multiview video coding. Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, JVT-AA209, April 2008.

[103] Video codec for audiovisual services at p x 64 kbit/s. ITU-T Recommendation H.261, 1993.

[104] Video coding for low bit rate communication. ITU-T Recommendation H.263, 2005.

[105] Y.-K. Wang, M. M. Hannuksela, and M. Gabbouj, "Error resilient video coding using unequally protected key pictures," in *Proceedings of International Workshop on Very Low Bitrate Video Coding*, pp. 290–297, September 2003.

[106] H. Watanabe and S. Singhal, "Windowed motion compensation," in *Proceedings of SPIE Visual Communication and Image Processing VCIP*, vol. 1605, pp. 582–589, November 1991.

[107] T. Wiegand, Proposed draft on Annex U for enhanced reference picture selection. ITU-T SG16/Q15, VCEG, Document Q15-H-30, August 1999.

[108] T. Wiegand, N. Färber, K. Stuhlmüller, and B. Girod, "Error-resilient video transmission using long-term memory motion compensated prediction," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 1050–1062, June 2000.

[109] T. Wiegand and B. Girod, *Multi-frame Motion-compensated Prediction for Video Transmission*. Norwell, MA: Kluwer Academic Publishers, September 2001.

[110] T. Wiegand, B. Lincoln, and B. Girod, "Fast search for long-term memory motion-compensated prediction," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 619–622, October 1998.

[111] T. Wiegand, E. Steinbach, and B. Girod, "Affine multipicture motion-compensated prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 2, pp. 197–209, February 2005.

[112] T. Wiegand, E. Steinbach, A. Stensrud, and B. Girod, "Multiple reference picture coding using polynomial motion models," in *Proceedings of SPIE Visual Communication and Image Processing VCIP*, vol. 3309, no. 2, pp. 134–145, January 1998.

[113] T. Wiegand, X. Zhang, and B. Girod, "Long-term memory motion-compensated prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 1, pp. 70–84, February 1999.

[114] T. Wiegand, X. Zhang, B. Girod, and B. Andrews, Long-term memory motion-compensated prediction. ITU-T SG16/Q15, VCEG, Document Q15-C-11, December 1997.

[115] S. W. Wu and A. Gersho, "Joint estimation of forward and backward motion vectors for interpolative prediction of video," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 684–687, September 1994.

[116] X. Yuan, "Hierarchical uncovered background prediction in a low bit rate video coder," in *Proceedings of Picture Coding Symposium*, p. 12.1, 1993.

[117] K. Zhang and J. Kittler, "A background memory update scheme for H.263 video codec," in *Proceedings of European Signal Processing Conference*, vol. 4, pp. 2101–2104, September 1998.

[118] K. Zhang and J. Kittler, "Using background memory for efficient video coding," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 944–947, October 1998.

[119] K. Zhang and J. Kittler, Proposed amendments for Annex U on enhanced reference picture selection. ITU-T SG16/Q15, VCEG, Document Q15-H-09, August 1999.

[120] R. Zhang, S. L. Regunathan, and K. Rose, "Video coding with optimal inter/intra-mode switching for packet loss resilience," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 966–976, June 2000.

[121] J. Zheng and L.-P. Chau, "Error-resilient coding of H.264 based on periodic macroblock," *IEEE Transactions on Broadcasting*, vol. 52, no. 2, pp. 223–229, June 2006.

[122] X. Zhou and C.-C. J. Kuo, "Robust streaming of offline coded H.264/AVC video via alternative macroblock coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 4, pp. 425–438, April 2008.

[123] X.-J. Zhu and S.-A. Zhu, "Fast mode decision and reduction of reference frames for H.264 encoder," in *Proceedings of International Conference on Control and Automation*, vol. 2, pp. 1040–1043, June 2005.