5

Advanced Video Coding (AVC)/ H.264 Standard

5.1 Digital Video Compression Standards

It has long been recognized that to promote interoperability and achieve economics of scale there is a strong need to have international standards. Two organizations – ITU-T and ISO/IEC – have been most involved in setting those standards. In ITU-T, the Video Coding Experts Group (VCEG) under Study Group 16 has focused on developing video coding standards. In ISO/IEC the Moving Picture Experts Group (MPEG), formally known as Working Group 11 (WG-11) under SC-29 has concentrated on developing international coding standards for compressing digital video and audio. In ITU-T the main focal applications have been Video Conferencing and Video Telephony and in MPEG the main focal applications have been Digital Television transmission and storage. However, more recently, those lines have been blurring. ITU-T and ISO/IEC jointly developed two video coding standards: 1) MPEG-2/H.262 [1]; and 2) MPEG-4 Part 10/H.264 [2] also known as Advanced Video Coding (AVC). As the latter standard is known as both MPEG-4 AVC and H.264, in this text it is referred to as AVC/H.264 or sometimes simply as AVC. This standard is expected to play a critical role in the distribution of digital video over 4G networks. Full detailed description of this standard will require a complete book dedicated only to this subject. Due to limited space, only a high level description of this standard is provided to whet the appetite and give reader a good basic understanding of the coding tools used in the standard that provide higher coding efficiency over previous standards.

The movement towards creating international standards started in the mid 1980s with ITU-T approving H.120 standard in 1988. After that, as shown in Figure 5.1, several international standards have been developed and are maintained by ITU-T and ISO/IEC. This figure provides only a rough approximate order of the development time frame of these standards. The time line and time periods shown in the figure correspond approximately to the time when the bulk of the technical work was done. Amendments, extensions and corrections to these standards continued long after the periods shown.

4G Wireless Video Communications Haohong Wang, Lisimachos P. Kondi, Ajay Luthra and Song Ci
 ISBN: 978-0-470-77307-9

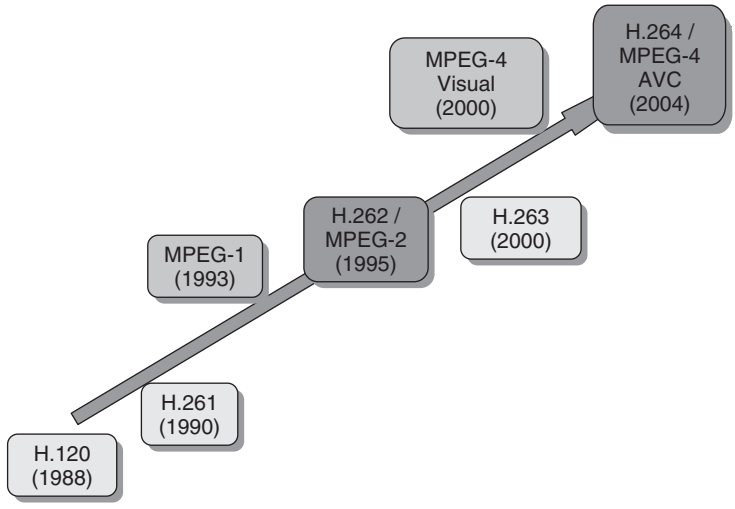

Figure 5.1 International digital video coding standards

In the 1980s and 1990s it was generally believed that one would need different basic codec structures and algorithms depending upon the compressed bit rate. In video conferencing it was believed that one would need a separate standard for compression for bit rates that were a multiple of 64 (m $\times$ 64) up to 384 kbps and for bit rates higher than 384 kbps. It was not the case. Similarly, it was also widely believed that one would need separate compression standards for compressing Standard Definition Television (SDTV) and High Definition Television (HDTV). MPEG-2 was in the beginning targeted towards SDTV and the name MPEG-3 was reserved for compressing HDTV. As the understanding of compression algorithms matured, it was discovered that there was no such need and that the basic algorithm developed for SDTV was also equally appropriate for HDTV. The development of MEPG-3 was dropped and the MPEG-2 specification was extended to cover the resolutions and bit rates corresponding to HDTV.

The MPEG-2 standard was developed jointly by ISO/IEC's WG-11 under SC29 and ITU-T and is also known as the ITU-T H.262 standard. After the completion of this standard, WG-11's and ITU-T's video coding groups moved in two separate directions. WG-11 started to work on MPEG-4 Part 2 [3], which was focused not only on improving coding efficiency over MPEG-2 but also on the interactive applications. This was the first standard that developed object oriented compression methodology. However, there have been few applications developed for the object oriented representation of video. Therefore, the most used profiles for MPEG-4 Part 2 so far are Simple Profile (SP) and Advanced Simple Profile (ASP). SP was focused on lowering the complexity of the compression algorithm over MPEG-2 and ASP was focused on increasing coding efficiency over MPEG-2. ASP was successful in improving coding efficiency by a factor of about 1.4 over MPEG-2. ITU-T's Video Coding Experts Group (VCEG) focused on developing more efficient video coding standards, known as H.263 and H.263++, for video phone and conferencing applications [4]. After these standards were developed, VCEG began

to work towards improving compression efficiency under the informal project name of H.26L. The resulting H.26L codec showed promising coding efficiency and performed very well in the Call for Proposal (CfP) sent out by MPEG for the next generation codec beyond MPEG-4 Part 2 and MPEG-2. As a result, it was agreed by WG-11 and SG-16 that MPEG and VCEG should join forces with the aim of developing jointly the next generation of advanced video standards. The Joint Video Team (JVT), consisting of experts from MPEG and VCEG, was formed in December 2001 to develop the next generation of video coding standards. As H.26L performed well in MPEG's testing of various algorithms, it was decided to keep the basic syntax of H.26L and to extend it so as to expand the focus from video phone and video conferencing applications to also include digital TV and digital video creation in studio applications and to add new coding tools in order to improve its performance for those applications. One of the prime goals was to improve the coding efficiency by a factor of 2 over MPEG-2.

Figure 5.2 provides some examples of the coding efficiency comparisons of MPEG-2, MPEG-4 ASP and MPEG-4 AVC/H.264 for standard MPEG test sequences 'Mobile & Calendar' and 'Bus'. In Figure 5.2, CIF stands for the Common Intermediate Format video with a picture (frame) resolution of 352×288 pixels, while HHR represents the Half-Horizontal Resolution video with a picture (frame) resolution of 352×480 pixels. It is important to note that the results will vary from codec to codec and sequence to sequence.

In this figure the bit rates are such that they provide 32 dB Peak Signal to Noise (PSNR) and are normalized for the MPEG-2 bit rate to be 100% for each sequence. As shown in equation (5.1), PSNR is obtained by taking the ratio of the peak signal power to the compression noise power, where the compression noise power is obtained by subtracting the decompressed video from the original video and obtaining the average power. PSNR

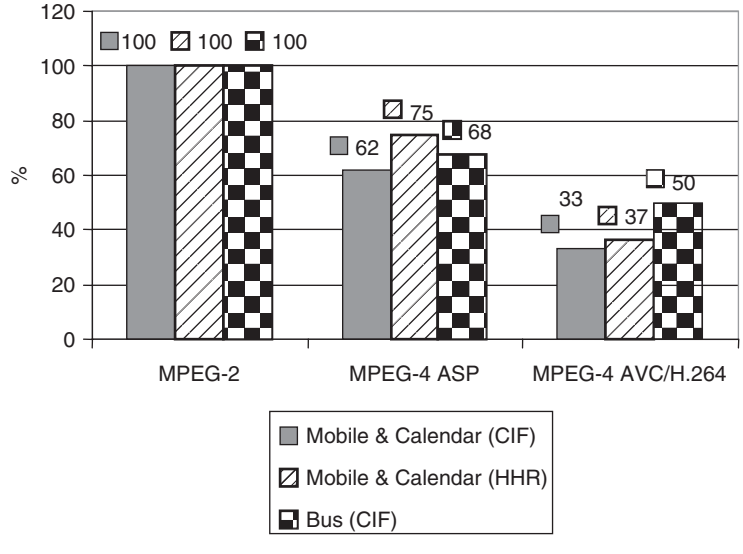

Figure 5.2 Percentage bit rate required for the same PSNR (~32 dB)

in dB is given by:

$$PSNR = 10\log_{10}\left[\frac{255^2}{\frac{1}{Total\ pixels}\sum_{pixels}(original\ pixel\ value - decoded\ pixel\ value)^2}\right] \tag{5.1}$$

Over the last 15 years, the coding efficiency of the coding algorithms and standards has improved by a factor of about 4. Interestingly, the basic core structure has remained the same – motion compensated hybrid transform-DPCM. The transform has also remained the same – Discrete Cosine Transform (DCT) or DCT-like integer transform. Several rounds of testing have been done at the international level and this hybrid coding structure with DCT or DCT-like transform always came out ahead in terms of coding performance. On the algorithm side, the primary reason for the improvement in coding efficiency has been the ability to perform more accurate predictions of the pixels from their spatially and temporally neighboring pixels. This is due to a significant increase in the complexity of the spatial and temporal prediction algorithms over the years. As implementation and integration technology progressed, more and more complex algorithms for performing predictions could be added to the standards. This has resulted in lower prediction error in the pixels being compressed thus requiring a lesser number of bits and therefore providing better coding efficiency. In addition, these new algorithms also made AVC/H.264 the first standard to bridge the gap between video conferencing and digital television applications. Until the development of this standard, there were two categories of standards. One, such as MPEG-2, that was more efficient and used for entertainment video (TV) and another, such as H.263 or H.263++, was more efficient and used for video conferencing and telephony. AVC/H.264 was the first standard that was equally applicable and highly efficient for both categories and the entire range of bit rates – from 32 kbps to more than 250 Mbps. That is five orders of magnitude of bit rate range!

5.2 AVC/H.264 Coding Algorithm

As is also the case with the previous standards, the AVC/H.264 standard standardizes decoder and bit stream syntax. This specifies indirectly the coding tools which can be used by an encoder to generate a bitstream that can be decoded by a compliant decoder. Figure 5.3 shows basic coding structure of the corresponding encoder.

Digital video is a three – two spatial and one temporal – dimensional signal. Therefore, the prediction of a certain pixel value can be done based on the neighboring pixels in the same picture (spatial prediction) or the in other pictures in the past or the future (temporal prediction). Once the predicted value (at point D) is obtained, the difference of the true value and the predicted value is calculated. This difference is also called the prediction error. A DCT-like (sometimes also called Integer DCT) transform of the difference signal is taken and the transform coefficients are quantized. In some coding modes, described later, a scaling matrix can also be applied to each coefficient in order to shape the coefficient values so that higher frequency coefficients are quantized more than the lower ones. The quantized coefficients are then scanned out as a one dimensional signal and passed through two possible lossless entropy coding algorithms – Context

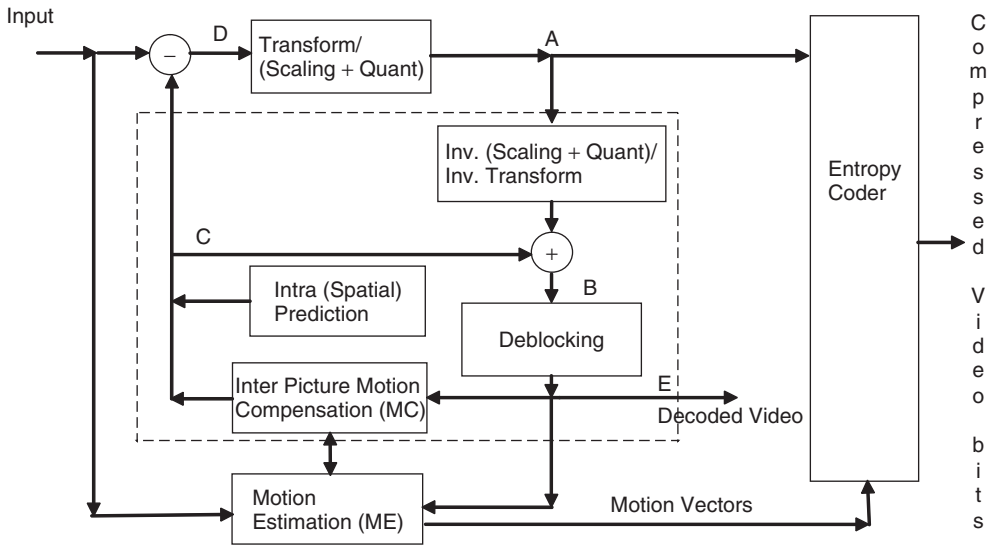

Figure 5.3 AVC encoder structure

Adaptive Variable Length Coder (CAVLC) or Context Adaptive Binary Arithmetic Coder (CABAC).

In this figure, one more layer of temporal prediction – Motion Estimation (ME) and Motion Compensation (MC) – and intra (Spatial) prediction are shown. As described in sections 3.4.1, 3.4.2 and 3.4.3, the area within the dotted box corresponds to the local decoder loop within the encoder and the signal at point E is the same as the output of a decoder. Motion is estimated between current blocks and the *decoded* blocks in previously compressed pictures. As explained in section 3.4.1, this avoids drifting between an encoder and a decoder. In many implementations ME is done in two stages. In the first stage motion is estimated as a pre-encoding step (which can be done ahead of the rest of the encoding process) by using original pictures and MVs are then refined to the final values by using the decoded reference pictures to obtain the prediction error values at point D in Figure 5.3. There is an additional functional block called Deblocking in that loop. At a high level, the functional blocks of the encoder are as shown in Figure 5.4 and are described in more detail below.

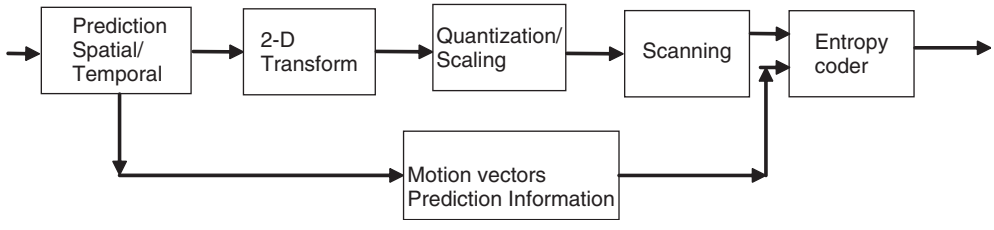

Figure 5.4 Functional blocks of an AVC encoder

5.2.1 Temporal Prediction

Temporal prediction consists of two steps – 1) Motion Estimation (ME); and 2) Motion Compensation (MC). Motion Estimation is the single most computationally intensive step in an encoder. As the name suggests, in this step an estimate of the motion of the pixels in a picture is obtained. This allows one to estimate where the pixels belonging to a certain object in a picture were in the past and will be in the future. The Motion Compensation step uses the motion vectors obtained in the ME step and provides the estimate of the current pixel value. If the estimate is exact, the difference between the pixel values in other pictures and that in the current picture will be zero. However, there are also some changes that happen between the pixel value in the current picture and other pictures. Those could be due to the rotation of an object, occlusion, change in the intensity and many other such reasons. Therefore, the difference between the predicted value and the true value of a pixel is not always zero. The difference is then transformed and quantized. The decoder also needs to know the motion vectors used to create the difference. The encoder compresses the motion vectors (MVs) used for prediction and sends them along with the coefficient values.

5.2.1.1 Motion Estimation

It is not only very hard and costly to estimate the motion of every pixel, but it also takes a large number of bits to send the motion vectors (MVs) to the decoder. Therefore, motion is estimated for a group of pixels together. The group of pixels was selected to be 16 × 16 (16 pixels horizontally and 16 pixels vertically) in earlier standards such as H.261 and MPEG-2. This group of pixels is called the macroblock (MB). In MPEG-2 for interlaced video, the size of the macroblock was 16 × 8 for the macroblocks that were coded in the field mode. The size of the MB was reached as a compromise between implementation complexity and compression efficiency. In the 1990s it was very costly to implement ME and MC blocks smaller than 16 × 16 in an encoder and a decoder. When AVC/H.264 was developed the Very Large Scale Integration (VLSI) technology had progressed significantly and it was reasonable to implement motion estimation for an area smaller than 16 × 16. In this standard, motion can be estimated for groups of pixels of sizes 16 × 16, 16 × 8, 8 × 16, 8 × 8, 8 × 4, 4 × 8 and 4 × 4, as shown in Figure 5.5. A 16 × 16 MB can be further partitioned into two 16 × 8, or two 8 × 16, or four 8 × 8 size MBs. An 8 × 8 MB can be further partitioned into two 8 × 4, or two 4 × 8, or four 4 × 4 sub-macroblocks. In MPEG-2 the term macroblock was used to refer to the group of pixels (16 × 16 or 16 × 8 for field MB) to which MV was assigned for motion prediction. This size was different than the size of the group of pixels for which transform (8 × 8) was taken and that group was referred as a 'block'. In AVC/H.264 that line of separation

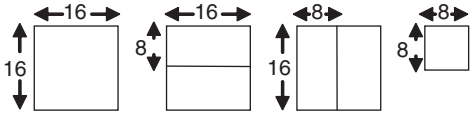

Figure 5.5(a) MB sizes

Figure 5.5(b) Sub-macroblock partitions of 8 × 8 MB

between the MV block and transform size block has blurred. Therefore, the terms MB, sub-MB, 'block' and 'partition' are used interchangeably in this text unless there is a need to be explicit. The context of the terms makes it clear whether a block corresponding to MV is referred to or a block corresponding to transform is referred to.

An encoder is given the flexibility to select the size that is the most appropriate for a given region of a picture and the cost of ME. As shown in Figure 5.6, in the areas closer to the moving edges, encoders can break a MB in smaller sizes.

In this way a better motion estimation can be obtained around moving edges which will result in a smaller prediction error. Note that breaking an MB into smaller parts increases

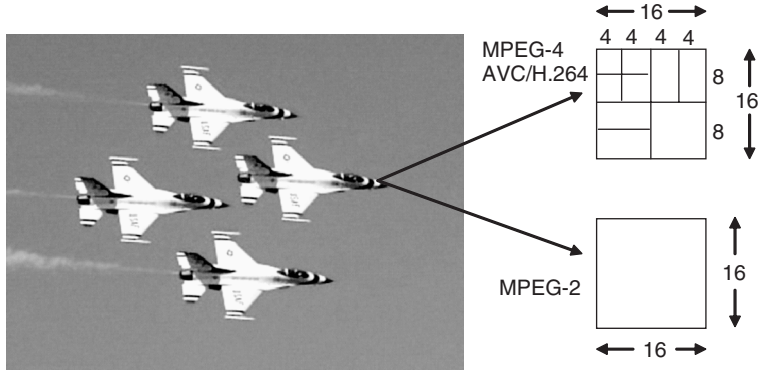

Figure 5.6(a) Picture to be compressed

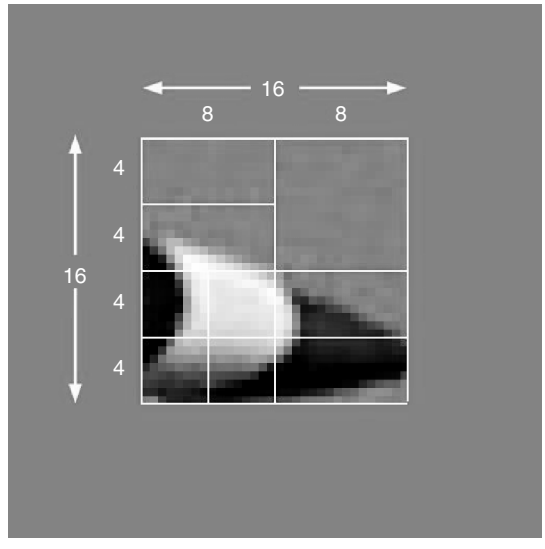

Figure 5.6(b) Zoomed portion at the nose of a plane in Figure 5.6(a) showing need for variable block size for the motion estimation

the number of bits required to send the motion information to the decoder as there will be more motion vectors associated with a greater number of sub-macroblocks. Therefore, an encoder needs to make smart decisions and do a trade-off between the higher cost (number of bits) of sending more motion vectors versus the lower cost (number of bits) associated with sending smaller prediction errors.

As with earlier standards, the AVC/H.264 standard does not describe how motion is estimated for a given macroblock/sub-macroblock. It is left up to the encoders to develop their own algorithms. There are many different algorithms developed in order to estimate the motion [5, 6]. They provide different implementation cost versus coding efficiency trade offs. They can be classified into two main categories:

1. Time domain.
2. Frequency domain.

The majority of motion estimation algorithms fall within the time domain category. In the time domain block matching based ME approaches are common where the current block is matched against the blocks in past or future pictures. The match with the smallest difference – typically the Sum of Absolute Difference (SAD) – is taken to be the location of that sub-macroblock in that picture and is used as reference for prediction. The full search method, where every sub-macroblock in the reference picture is compared against the current sub-macroblock in the current picture, provides the minimum SAD value. However, it sometimes becomes very computationally intensive especially for an HD picture. Therefore, many fast motion estimation techniques have been developed. They provide various cost-performance compromises and are used depending upon the implementation architecture.

A Rate-Distortion (RD) optimization based approach for ME is also used by many encoders. In this approach the cost of representing motion vectors and the prediction error are modeled as the following Lagrangian cost function:

$$C = D + \lambda \times R \tag{5.2}$$

where D is the measure of the distortion in the picture and R is the total number of bits for representing motion information. Typically, D is based on the sum of the absolute difference of the luminance components corresponding to various possible motion vectors. In this approach a motion vector is selected that minimizes the cost function C. Unfortunately, as there is no close form value available for λ, its value is decided experimentally. This type of RD optimization requires significant computational power that may not be available in some implementations. Thus, encoders need to make implementation specific compromises on RD optimization or use other motion estimation methods. For example, many encoders do not use sub-8×8 sizes while encoding HD or SD size pictures as they are more useful at SIF/CIF/QVGA or smaller picture sizes and some encoders may choose to use only 16×16 and 8×8 partitions.

5.2.1.2 P and B MBs

A motion predicted MB can be of one of the two types – Predicted (P) or Bi-predicted (B). In a P MB there is only one MV associated with that MB or its partitions. However,

unlike MPEG-2, that motion vector may be derived from reference pictures that can either be in the past or in the future in a captured video sequence relative to the current picture being compressed. In B MB, up to two motion vectors may be used by the MB or sub-MBs. Unlike MPEG-2, both the reference pictures corresponding to those motion vectors can either be in the past or future or one in the past and one in the future relative to the current picture being compressed.

5.2.1.3 Multiple References

In AVC/H.264, more than one picture can be used as references in order to estimate the motion of various different parts of a picture, as shown in Figure 5.7. Therefore, if motion is such that there is an occlusion and parts of an object in the current picture are hidden in the picture that is in the immediate past or future, one can look for a better match in other pictures that are further away and may contain a better match. This is also useful in the situation where the motion is periodic and a better match for the current macroblock is not necessarily in the immediate neighboring picture. Many times, a video consists of pictures with sudden changes in brightness due to flashes. Those flashes interfere with motion estimation. In that case, a better estimate of motion can be obtained by skipping over the flashes and using pictures further away in time. Therefore, in AVC, the use of multiple reference frames allows for an encoder to optimize and adapt the ME process to the content being compressed.

5.2.1.4 Motion Estimation Accuracy

A picture is a sampled image in which the motion is continuous. Therefore, a better estimate of motion can be obtained by interpolating between the pixels, especially for the sharp moving edges. Due to implementation costs, the interpolation in MPEG-2 was

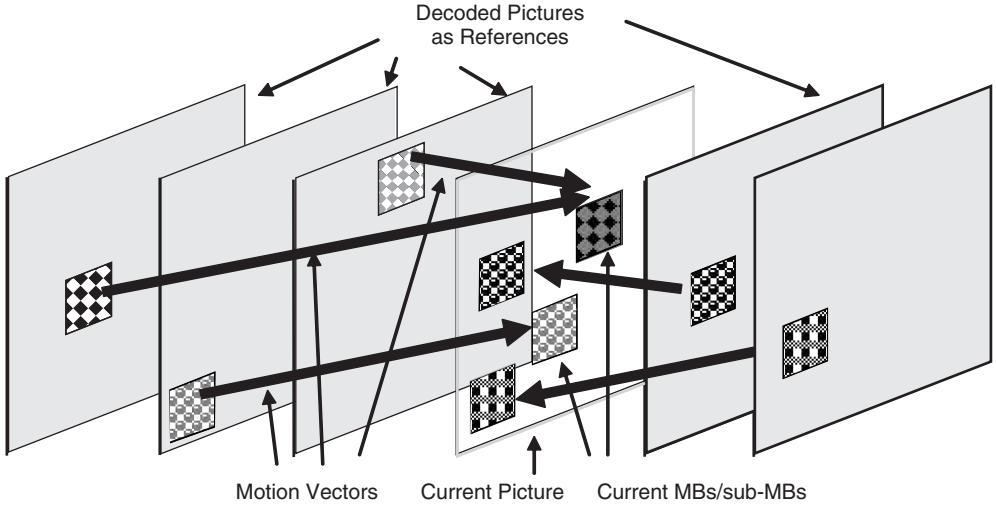

Figure 5.7 Multiple reference pictures

limited to a factor of 2 and motion was estimated with a $\frac{1}{2}$ pixel accuracy. In AVC/H.264 (and also in MPEG-4 Part 2) the interpolation was expanded to be up to a factor of 4 and the motion can be estimated with 1/4th pixel accuracy. For the luma samples, the 1/4th pixel interpolation is done in two steps. In the first step, pixel values at half way between the pixels in the captured picture are calculated by using a 6-tap filter (1, −5, 20, 20, −5, 1)/32. The pixel values at 1/4th the pixel locations are then calculated by using linear interpolation. For chroma samples, the prediction values are obtained by bilinear interpolation. For 4:2:0, since the sampling grid of chroma has half the resolution of the luma sampling grid, the displacements used for chroma have a 1/8th sample position accuracy [12].

5.2.1.5 Weighted Prediction

Unlike prior standards, the reference pixels can also be weighted by a weighting factor before obtaining the prediction error. This can improve coding efficiency significantly for scenes containing fades, and can be used flexibly for other purposes as well. In a fade, the intensity of the video picture dims to black or comes out of black to a regular intensity. Owing to changes in intensity between pictures, a proper weighting in order to compensate for the variation in the intensity provides a better estimate of the current pixel values and hence results in smaller prediction error. The standard does not specify an algorithm for finding the weighting and it is left up to the encoders to implement any desired algorithm. The standard only specifies the syntax corresponding to the weighting values and how to send them to a decoder, if used.

5.2.1.6 Frame and Field MV

A video sequence may consist of a progressively or interlaced scanned picture. In progressively scanned pictures the lines in a video frame are captured contiguously. Interlaced scanned frame consists of two fields. In interlaced scanned pictures the alternate lines are captured at one given time (see Figure 3.1). In a 525 line standard the spacing between the two fields is 1/60th of second. As two fields are captured at different time, parts of the scene that are in motion appear at different locations in a frame, even though they belong to the same object. AVC/H.264 gives the encoder the option (there are Profile related restrictions that are discussed below) of breaking a MB in a frame into two fields and estimating the motion separately for each field. This is beneficial in obtaining motion more accurately for moving objects, especially for the pixels around the edges or when the motion is non-linear and the displacement of the pixels is not the same in the top and bottom field.

AVC/H.264 allows for two degrees of freedom with regard to frame or field based motion estimation. One option encoders may chose is to break the entire frame into two fields and treat each field as an independent picture for compression. This decision can be made independently for each picture. This process is called Picture Adaptive Frame or Field (PicAFF or PAFF) based compression. Another choice which encoders have is to compress the frames in the frame mode and make the frame and field decision adaptively and independently for a portion of a frame. The smallest size of the frame for which a frame or field decision can be made is 16×32, which is 2 MB high. This process is called MacroBlock Adaptive Frame Field (MBAFF) compression. Notice that, unlike the

MPEG-2 standard, in MBAFF the field/frame decision is made at MB-pair level and not at each MB level. The reason for selecting (vertical) MB-pair as the smallest quanta for that decision was that once a 16×32 size area is divided into two fields, it provides two macroblocks, each of 16×16 size and now all the coding tools, e.g. 16×16 Spatial Prediction, 16×16 or 16×8 or 8×16 ME, etc., used in the standard for a frame MB can now also be used for a field MB. However, note that the same numbers of lines in a field span twice as large a space as the same numbers of lines in a frame. This not only impacts upon the decision to choose the frame or field MB mode but also other mode decisions made while compressing a MB. Ideally, it would provide the best compression by compressing an MB in all three PicAFF, MBAFF and frame only modes and picking the one that provides the best compression. However, that becomes a very computationally intensive and expensive process. Therefore, the decision of which of these three modes to use is made fairly early in the compression process. Those decisions are largely based on the motion in the frame. Frames with large motion (e.g. frames with panning) are generally compressed using PicAFF. The MBAFF compression tool can provide better coding efficiency for frames with mixed, moving and static, zones. In PicAFF, as each field is compressed as independent pictures, the MBs in the second field of a frame can use MBs in the first field as references. However, in MBAFF the decoding order is as shown in Figure 5.8 and the current MB can not use the MBs in the same frame that occurs in the future because they are not available (i.e. compressed and decompressed) when the current MB is being compressed.

Although the decoding process for the interlaced scanned video is not significantly harder or more complex than a progressively scanned video, compressing an interlaced scanned video at the encoding end is significantly more complex. That complexity gives rise to relatively less coding efficiency in a typical encoder available today for an interlaced scanned video than that for a progressively scanned video.

5.2.1.7 MV Compression

The motion vectors used to estimate the motion and prediction of the current MB need to be conveyed to the decoder so that it can also use the same motion

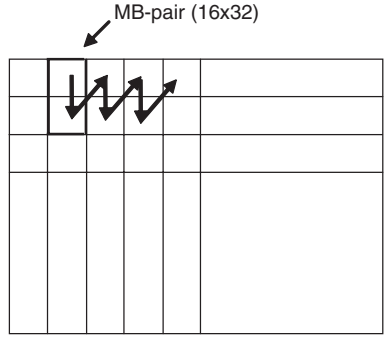

Figure 5.8 Decoding order for MB AFF frame

vectors to calculate the pixel values from the prediction errors and motion vector information. Therefore, the motion vector information is also compressed before sending it to a decoder. Techniques used to compress MV can be divided into two categories.

In the first category, motion vector corresponding to a sub-MB is compressed by a DPCM-like process. A reference MV is computed based on the neighboring blocks. As shown in Figure 5.9(a), for 8×16 partitions the motion vector of neighboring block to the left or up in the diagonal right position is used as reference. For 16×8 partitions the motion vector of neighboring blocks up or to the left of the partition is used as reference.

For other partitions the median MV of the upper, upper right and left neighboring blocks is taken as reference. An example is shown in Figure 5.9(b) where neighboring blocks N_1, N_2 and N_3 are the neighbors used. The details of exactly which neighbors to use for various cases (such as MBAFF, non-availability of neighbors, etc.) are provided in the standard. The difference of the MV corresponding to the current MB and the reference MV is compressed using lossless variable length coding.

In the second category, the motion vector of a MB is not sent explicitly to a decoder and is derived by the decoder based on the motion vectors of neighboring pixels in temporal and spatial dimensions. An encoder can use one of the two possible modes: Direct and Skip modes. There are two types of Direct modes: Temporal and Spatial. In Temporal Direct mode the MV is derived by scaling the MV of the co-located pixels, as shown in Figure 5.9(c).

In Spatial Direct mode the MV of the current block is derived by examining the motion vectors of a collocated MB without the scaling process, and using the motion vectors of neighboring blocks for generating prediction motion vectors.

Direct modes are very helpful when there is high correlation among the MVs of the blocks within a region. This allows for a significant reduction in bit rate as the MVs of the current blocks are not sent explicitly in the bit streams.

Like Direct mode, there is another mode where the MVs are not sent explicitly and are derived by the decoders based on the MVs of the neighboring blocks. It is called the Skip mode. It exploits the fact that sometimes not only the motion is highly correlated but also the prediction is so good that there are no prediction errors after quantization. In this mode neither the MV are sent explicitly nor the coefficient values.

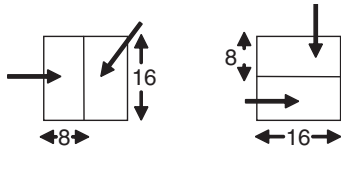

Figure 5.9(a) Prediction directions for motion vectors for 8×16 and 16×8 partitions

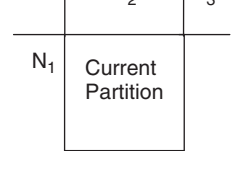

Figure 5.9(b) MV prediction based on median MV of neighbors' MVs

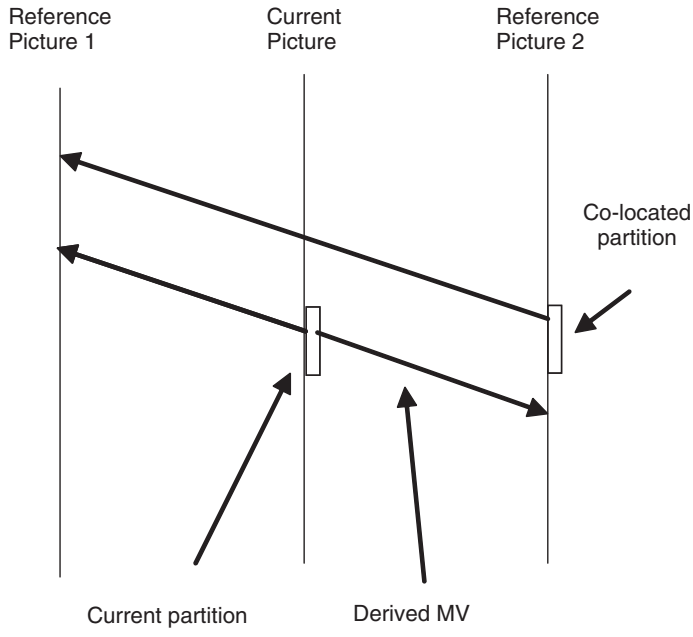

Figure 5.9(c) MV prediction in direct mode

5.2.2 Spatial Prediction

AVC.H.264 allows for extensive spatial prediction modes. These modes are used when temporal prediction is not used. MB with no temporal prediction is also called Intra (I) MB. The spatial prediction of I MB is called intra prediction. In AVC/H.264 such prediction is done in pixel-domain. By comparison, in MPEG-2 only simple intra predictions of the DC coefficients are performed.

An encoder may use one of the three possible spatial prediction modes: 16×16 luma (for chroma, corresponding chroma block size is used), 8×8 luma and 4×4 luma (there are profile related restrictions, discussed below, on when these modes can be used). In 16×16 mode, the pixel values of a 16×16 MB can be predicted in one of four different modes: Vertical, Horizontal, DC and Plane. In Horizontal prediction, the pixels in an MB are predicted from the pixels in the left side, P(−1,j) of the MB, as shown in Figure 5.10.

The −1 value in the notation P(−1,j) denotes the column of pixels left of the 16×16 macroblock and j, from 0 to 15, is the pixel number along the y-axis. In Vertical prediction, the pixels in an MB are predicted from the pixels on the top of the MB, pixels P(i, −1), shown in Figure 5.10, where −1 signifies the row above the top row of the 16×16 macroblock and i, ranging from 0 to 15, is the pixel number along the x-axis. In the DC prediction mode, the pixels are predicted from the DC (average) values of the neighboring pixels. In the Plane mode it is assumed that intensity of the pixels increases or decreases linearly in the MB. The prediction mode that gives the least prediction error is chosen.

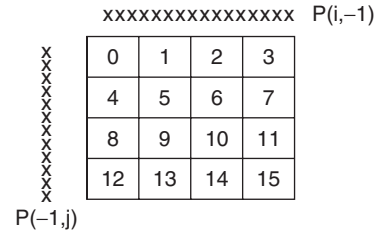

Figure 5.10 Spatial prediction for 16 × 16 mode

In 8 × 8 luma prediction mode, one of the eight different directions, as shown in Figure 5.11, and DC prediction can be used. Including DC, there are nine possible intra prediction modes. The directional modes are as shown in the figure. Mode 0 is the vertical direction, 1 is horizontal and so on. DC prediction is given mode number 2. Similarly, in 4 × 4 luma prediction mode, one of the eight different directions, shown in Figure 5.11, and DC prediction can be used for predicting pixels of a 4 × 4 block. The spatial prediction mode, selected for the prediction, is further compressed and sent in the bit stream.

Chroma intra prediction operates using full macroblock prediction. Because of differences in the size of the chroma arrays for the macroblock in different chroma formats (i.e., 8 × 8 chroma in 4:2:0 MBs, 8 × 16 chroma in 4:2:2 MBs and 16 × 16 chroma in 4:4:4 MBs), chroma prediction is defined for three possible block sizes. The prediction type for the chroma is selected independently of the prediction type for the luma.

In large flat zones the 16 × 16 mode is more efficient to use. The 4 × 4 mode is used for a region with high detail and fast changing intensity. The 8 × 8 mode is used for regions that fall in between the two extremes. This flexibility allows AVC/H.264 to achieve a very high coding efficiency for the Intra blocks.

5.2.3 The Transform

As shown in Figure 5.3, prediction errors are transformed before quantization. In the initial stage of the development of the standard, only 4 × 4 sized transform was used.

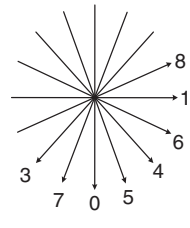

Figure 5.11 Possible spatial (intra) prediction directions

As the focus of the standard moved towards large picture sizes, such as HD and 4k × 2k, and applications such as video creation in studios, an additional flexibility of using 8 × 8 transform was allowed for encoders.

5.2.3.1 4 × 4 Integer DCT and Inverse Integer DCT Transform

As mentioned above, the standard provides the specification of a decoder and not that of an encoder. Therefore, it standardizes inverse transform used in the decoder. To match that transform, an encoder needs to use a matching forward transform. The 4 × 4 transform, corresponding to the inverse transform specified in the standard, is given by:

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

Note that all the coefficients of this transform are integers. This is not only relatively simple to implement but also allows an encoder to implement the transform exactly without causing precision related errors. In prior standards, like H.261 or MPEG-2, true DCT was used. Some coefficients of the DCT are irrational numbers and cannot be implemented without introducing precision errors. It poses two problems: 1) they cannot be implemented without truncation errors; and, more importantly, 2) the decoder part in an encoder (the dotted area in Figure 5.3) will likely not be same as the decoder in a receiver designed by some other company. As encoders and decoders will generally implement those DCT coefficients with differing precision, there will be a mismatch between the two. It can potentially cause some drifting of encoded and decoded pixel values. To avoid that drifting, a small amount of dithering (called mismatch control) was introduced in the MPEG-2 standard. Although this potential drifting does not happen in real signals and is not a problem, it was decided to remove this potential source of error in AVC/H.264. In addition to the simplicity of the design and implementation, this turns out to be another benefit of the integer forward and inverse DCT transform in AVC/H.264.

One way to analyze the forward transform is as follows [7, 8]. A 4 × 4 DCT transform is given by:

$$Y = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix}$$

where, $a = \frac{1}{2}$, $b = \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) = 0.653281482438\ldots$, *and* $c = \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) = 0.382683432365\ldots$ and p_{ij} is $(i, j)^{th}$ pixel value. This can be manipulated into the

following form:

$$Y = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & b \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{bmatrix} \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & b \end{bmatrix}$$

$$= \left[\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{bmatrix} \right] \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix}$$

$$= C \cdot P \cdot C^T \otimes A$$

where, $d = \frac{c}{b} = 0.41421356237\ldots$ and $\otimes$ denotes element by element multiplication for the matrices to the left and the right of the symbol. The matrix A can now be absorbed into the quantization matrix which also performs element by element operation. So, the transform is now given by the C matrix above. However, it still contains d which is an irrational number. Therefore, the value of d was truncated to 0.5. With this change the approximated 'DCT' transform matrix becomes:

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{bmatrix}$$

However, the division by 2 causes loss of accuracy. Therefore, the second and fourth rows are multiplied by 2 and appropriate changes are done down the encoding path. Thus, the final form of the transform becomes:

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

Transform can now not only be implemented accurately but also easily. As it can be implemented without the use of a multiplier and using only additions and shifts, it is sometimes also stated to be a multiplier free transform. Note that owing to the approximation made above, the transform is no longer the true cosine transform. However, it does not hurt the coding efficiency in any significant way.

5.2.3.2 8 × 8 Transform

In a similar style to the 4 × 4 transform, integer 8 × 8 inverse DCT transform is specified in the standard and the corresponding forward 8 × 8 integer DCT transform is

given by:

$$C = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & 12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix}$$

5.2.3.3 Hadamard Transform for DC

When the 16×16 Intra prediction mode is used with the 4×4 transform, the DC coefficients of the sixteen 4×4 luma blocks in the macroblock are further transformed by using the following Hadamard transform:

$$H_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

The DC coefficients of the 4×4 chroma block samples in a macroblock are also further transformed by Hadamard transform. In 4:2:0 video format, there are only four DC coefficients. They are transformed by using the following 2×2 Hadamard transform:

$$H_{2\times2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

For 4:4:4 video format, $H_{4\times4}$ above is used to further transform 4×4 chroma DC coefficients.

5.2.4 Quantization and Scaling

When 4×4 transform is used, coefficients cannot be weighted explicitly before quantization. Coefficients in a macroblock can be quantized using one of the 52 possible values for 8 bit video. For higher bit depth video, the number of possible values increases by 6 for each additional bit. The quantization step is controlled by the quantization parameter. Quantization step sizes are not linearly related to the quantization parameter. The step size doubles for every six increments of the quantization parameter.

When 8×8 transform is used, the encoder is given the additional flexibility of using its own weighting matrix during the quantization step. The weighting matrix can be adapted from picture to picture.

5.2.5 Scanning

The two dimensional transformed coefficients are scanned out in a particular order so as to produce a one dimensional string of coefficients, which is then input to the variable

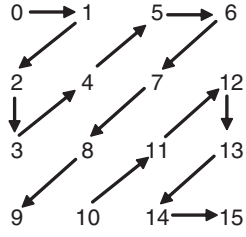

Figure 5.12 Zig-zag scanning for progressive video

length encoder, as shown in Figure 5.3. Two different styles of scanning are used – one for progressively scanned and another for interlaced scanned video where MBs are compressed in the field mode. In progressively scanned video or for frame MBs, a traditional zig-zag scanning order, as shown in Figure 5.12, is used. The zig-zag scanning is used as it provides longer run of zeros, in a typical compressed video, which can be compressed more efficiently by the VLCs used in the standard.

However, for interlaced video, the statistics of the coefficients is different for the field MBs than it is for the MBs in a progressively scanned video. Therefore, for field MBs the scanning order is as shown in Figure 5.13. These scanning orders are tuned more optimally for a field MB.

Similarly, zig-zag and alternate scans for 8×8 transformed blocks are also specified in the standard.

5.2.6 Variable Length Lossless Codecs

The AVC/H.264 standard includes three different types of lossless encoders – Exp-Golomb, CAVLC (Context Adaptive VLC) and CABAC (Context Adaptive Binary Arithmetic Coding). Exp-Golomb codes are used only for those high level syntax elements that are not voluminous in the number of bits, e.g. chroma format, bit depth, number of reference frames, etc. Quantized transform coefficients are coded using either CAVLC or CABAC. As described below, some receivers are required to implement only CAVLC while others are required to implement both. Both of these coding techniques recognize the fact that digital video is a non-stationary signal and there are large

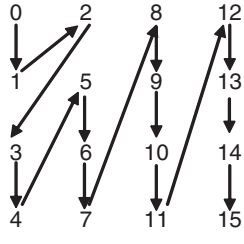

Figure 5.13 Alternate scan for field MBs in interlaced video

variations in the probability of various syntax elements from one sequence to another. They both adopt the philosophy of adapting the coder to the characteristics of the digital video being compressed.

5.2.6.1 Exp-Golomb Code

This code is simple to implement. The bit stream is generated for a given code value, as shown in Table 5.1. In this table x_i is either 0 or 1. These codes consist of prefix part: 1, 01, 001, etc. and the suffix bits are as shown in the table. For n bits in the suffix, there are 2^n codes for a given prefix part.

Various syntax elements are mapped into the code values as specified in the standard. Table 5.2 shows in explicit form the bit stream generated corresponding to those values.

For the syntax elements with signed value, like change in the quantization values, mapping of the signed value to the code value is specified in the standard and is shown in Table 5.3.

Table 5.1 Exp-Golomb code

Bits	Code Values
1	0
0 1 x_0	1–2
0 0 1 x_1 x_0	3–6
0 0 0 1 x_2 x_1 x_0	7–14
0 0 0 0 1 x_3 x_2 x_1 x_0	15–30
0 0 0 0 0 1 x_4 x_3 x_2 x_1 x_0	31–62
...	...

Table 5.2 Exp-Golomb code examples

Bits	Code Values
1	0
0 1 0	1
0 1 1	2
0 0 1 0 0	3
0 0 1 0 1	4
0 0 1 1 0	5
0 0 1 1 1	6
0 0 0 1 0 0 0	7
0 0 0 1 0 0 1	8
0 0 0 1 0 1 0	9
...	...

Table 5.3 Mapping of signed syntax values to code values

Code Values	Signed Syntax Element Value
0	0
1	1
2	−1
3	2
4	−2
5	3
6	−3
...	...

5.2.6.2 CAVLC (Context Adaptive VLC)

The core philosophy of CAVLC is similar to that in the previous standards where the more frequently occurring data elements are coded with shorter length codes and the length of run of zeros is also taken into account while developing the code tables. A major basic change in comparison to the VLCs used in previous standards is that it exploits the fact that in the tail-end there are small valued (± 1) non-zero coefficients. In addition, it also adapts the table used for coding based on past information such that one can use more optimal statistics that are adapted to the video. These coefficients are coded and conveyed by sending: total non-zero coefficients; trailing ones, sign of trailing ones, total zeros, the number of consecutive zero valued coefficients, run of zeros and non-zero coefficient values. To give a high level view let us consider an example. Let the coefficients after zig-zag scan be 10,0,0,6,1,0,0,−1,0,0,0,0,0,0,0,0. In this case, there are four non-zero coefficients, two trailing ones,four zeros between 10 and −1, the run of zeros between −1 and 1 is 2 and the run of zeros between 10 and 6 is also 2.

The statistics of the coefficient values have a narrower distribution in the end than in the beginning. Therefore, the coefficients are coded in reverse order. In the example above, 6 is the first coded coefficient value. A default coding table is used to code the first coefficient. The table used for the next coefficient is then selected based on the context as adapted by previously coded levels in the reverse scan. To adapt to a wide variety of input statistics several structured VLC tables are used.

5.2.6.3 CABAC

AVC/H.264 also allows the use of an Arithmetic Coder. Arithmetic coding has been used extensively for image and data compression [9, 10]. It is used for the first time here in a video coding standard. The Arithmetic Coder has an increased complexity over the traditional one dimensional run-length Huffman style encoder but it is capable of providing higher coding efficiency, especially for symbols that have fractional bit entropy. There are three basic steps for CABAC – 1) binarization; 2) context modeling and 3) binary arithmetic coding [11]. As the syntax elements, such as coding modes, coefficients, motion vectors, MB and Sub-MB types, etc., are not binary, the first step for CABAC is

Table 5.4 Example of binarization

Slice Type	Sub-MB Type	Bin string
P Slice	P 8 × 8	1
	P 8 × 4	0 0
	P 4 × 8	0 1 1
	P 4 × 4	0 1 0

to map those non-binary values into unique binary code words called 'bin strings'. As an example, the mapping of sub-MB types in a P slice is shown in Table 5.4.

The next step is context modeling. In this step, the probability models for various symbols are developed. As the probability model makes a significant impact on the coding efficiency of the coder, it is very important that a right model is developed and is updated during encoding and that the probability statistics of various symbols are adapted to the past video information. This adaptation allows the encoder to tailor the arithmetic encoding process to the video being compressed rather than having non-optimal fixed statistics based encoding.

The final stage of CABAC is a Binary Arithmetic Coder. Arithmetic encoding is based on recursive interval sub-division depending upon the probability of the symbols. The step of converting syntax elements to binary values allows one to simplify the implementation of the arithmetic coding engine. Low complexity binary arithmetic coding engines have been used in image compression algorithms and standards such as JPEG. In AVC, a multiplication free binary arithmetic coding scheme was developed in order to achieve a reasonable trade-off between complexity and coding efficiency.

CABAC is reported to achieve about 10 to 15% better coding efficiency in the range of video quality (PSNR) in a digital TV application [11–13]. This improvement can be lower or higher depending upon the video content.

5.2.7 Deblocking Filter

When a video is highly compressed, the compression artifacts start to become visible in the decoded pictures. Three of the most commonly seen artifacts are blocking, mosquito and ringing noises. In blocking noise, the blocks corresponding to the motion estimation and transformation start to become visible. In mosquito noise, small size noise starts to appear around moving edges. This is primarily because the motion of the region covered by a macroblock consists of pixels with mixed motion resulting in large prediction error and also the presence of edges produces higher frequency components that become quantized. In ringing noise, the sharp edges or lines are accompanied by their faint replicas. Although owing to flexible MB size in AVC/H.264 these noises are reduced significantly, they do start to appear as the compression factor increases. To minimize the impact of these noises, deblocking filters have been used. There are two schools of thought – deblocking as a post processing step or in-loop deblocking. In the first, deblocking filters are applied after the pictures are decoded. As in MPEG-2 and MPEG-4 Part 2, the design of those filters is not standardized and it is left up to the decoder manufacturers to decide how to design and implement these and whether to apply deblocking filters or not. This approach

has the benefit of keeping the cost of decoder implementation low and leaves some room for innovation at the decoding and displaying end. A disadvantage in this approach is that it is hard to control the quality of the displayed video from the encoding/sending side. In addition, the pictures that are used as references contain these noises and they accumulate as more and more pictures use past pictures as references. Furthermore, these noises are a function of content as well as quantization. It becomes harder to implement a deblocking filter that takes those factors into account in the post processing step. To avoid those disadvantages, AVC/H.264 specifies an in-loop deblocking filter, as shown in Figure 5.3. As the name suggests, the deblocking is now done within the decoding loop.

The in-loop deblocking filter in AVC reduces the blockiness introduced in a picture. The filtered pictures are used to predict the motion for other pictures. The deblocking filter is a content and quantization adaptive filter that adjusts its strength depending upon the MB mode (Intra or Inter), the quantization parameter, motion vector, frame or field coding decision and the pixel values.

The deblocking filtering is designed so that the real edges in a scene are not smoothed out and the blocking artifacts are reduced. Several filters are used and their lengths and strengths are adjusted depending upon the coding mode (like Intra or Inter prediction, frame or field mode), size of the motion vector, neighboring pixels and quantization parameters. When the quantization size is small, the effect of the filter is reduced, and when the quantization size is very small, the filter is shut off. For the frame intra predicted blocks the filters are strongest and are four pixels long. In the field mode the vertical filters are only two pixels long as the field lines are twice as far apart as the frame lines. This reduces blurring due to filtering. All compliant decoders are required to implement the deblocking filter as specified in the standard.

An encoder is also allowed the option of not using the deblocking filter and signaling to a decoder in the syntax whether a deblocking filter is used or not. This allows encoders to choose between compression artifacts or deblocking artifacts. For very high bit rate applications, where the blocking noise is not significant, it may not be desirable to add blurring distortion due to deblocking filter. Encoders are also allowed the option of reducing the level of filtering by not using default parameters and optimizing them to a given application.

5.2.8 Hierarchy in the Coded Video

The basic coding structure of AVC/H.264 is similar to that of earlier standards and is commonly referred to as a motion-compensated-transform coding structure. The coding of video is performed picture by picture. Each picture to be coded is first partitioned into a number of slices (it is also possible to have one slice per picture). As in earlier standards, a slice consists of a sequence of integer numbers of coded MBs or MB pairs. To maximize coding efficiency, only one slice per picture may be used. In an error prone environment, multiple slices per picture are used so that the impact of an error in the bit stream is confined only to a small portion of a picture corresponding to that slice.

The hierarchy of video data organization is as follows:

Video sequence { picture { slices [MBs (sub-MBs (blocks (pixels)))] }}.

In AVC/H.264, slices are coded individually and are the coding units, while pictures are specified as the access units and each access unit consists of coded slices with associated data.

5.2.8.1 Basic Picture Types (I, P, B, B_R)

There are four different basic picture types that can be used by encoders: I, P, B and B_R (the standard does not formally use these names but they are most commonly used in the applications using the standard). I-pictures (Intra pictures) contain intra coded slices and consist of macroblocks that do not use any temporal references. As only spatial prediction is allowed, these are good places in a bitstream to perform random access, channel change, etc. They also stop the propagation of errors made in the decoding of pictures in the past. AVC also defines a special type of Intra picture called the Instantaneous Decoder Refresh (IDR) picture. As described in Chapter 10, it is a regular Intra picture with the constraint that pictures appearing after it in the bistream cannot use the pictures appearing before it as references. IDR pictures may be used as random access points as the decoding process does not depend upon the history before the IDR.

The P-pictures (Predicted pictures) consist of MBs or sub-MBs that can use up to one motion vector for prediction. Various parts of the P-pictures may use different pictures as references to estimate the motion. Unlike in MPEG-2, those reference pictures can be either in the past or in the future of the current frame in the video stream. P-pictures can also contain intra MBs. The B-pictures (Bi-predicted pictures) consist of MBs or sub-MBs that use up to two motion vectors for prediction. B pictures can contain MBs with one or zero (intra) motion vectors. Various parts of B-pictures also may use different pictures as references. Unlike in MPEG-2, both of the reference pictures, corresponding to the two MVs, can either be in the past or future or one can be in the past and the other in the future of the current picture in the video stream. B_R pictures are the B pictures that are used as references for temporal prediction by other pictures in the past or in the future. Note that in MPEG-2 one cannot use B-pictures as references.

5.2.8.2 SP and SI Pictures

SP and SI pictures are special types of pictures that are used for switching the bitstream from one rate to another. In some internet streaming environments with no Quality of Service there is no guaranteed bandwidth. Therefore, available channel capacity varies significantly and there is a need to change the compressed bit rate with time. To facilitate switching the bit rate quickly, SP and SI pictures can be used [14]. Consider a video compressed at two different bit rates, as shown in Figure 5.14. Let us assume that switching from Stream 1 to Stream 2 is desired. As shown in Figure 5.14, there are several switching points, like S_1 and S_2, created in the streams. In a normal bitstream, after switching, the pictures appearing in the future do not have the correct past references as the pictures received in the past belong to different bit rate. This causes distortion in the decoded video. To avoid this problem, S_1 and S_2 are created so that one can send another picture S_{12} which will allow one to create S_2 exactly but by using past pictures of Stream 1. Therefore, the decoding of Stream 2 picture after this point can occur without any

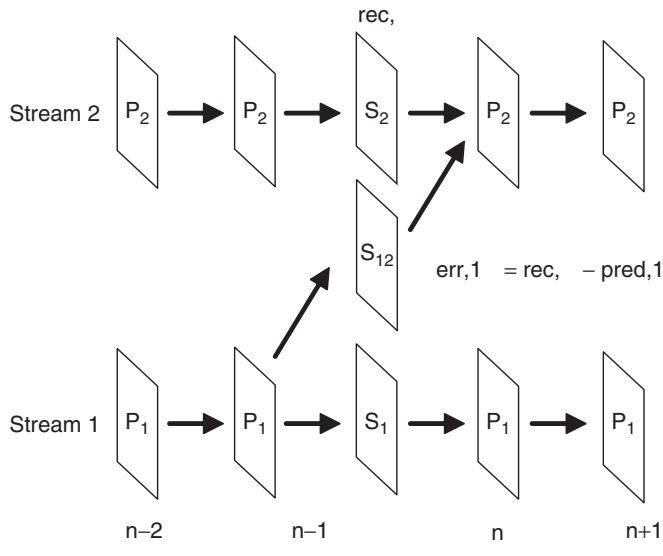

Figure 5.14 SP and SI pictures

distortion. If S_{12} is created by using temporal predictions it is called the SP picture. If only the spatial prediction is used then it is called the SI picture.

There are several issues related to SP and SI pictures. These pictures work well only when the switching occurs between the streams corresponding to the same video content. In addition, they are more complex and cause reduction in coding efficiency. Therefore, they are not widely used.

5.2.9 Buffers

The number of bits per picture varies significantly depending upon the content type and the picture type. Therefore, the compressed pictures are produced at an irregular and unpredictable rate. However, a picture typically needs to be decoded and displayed in a fixed picture time. Therefore, there is a mismatch between the number of bits that arrive at a decoder and the number of bits consumed by the decoder in one picture time. Therefore, the incoming bits are stored in a buffer called the Coded Picture Buffer (CPB) which holds the bits corresponding to the pictures yet to be decoded. To avoid the overflow and underflow of bits at the input of a decoder, it is important to ensure that an encoder knows how much input buffer a decoder has, to hold the incoming compressed bitstreams. To provide consistency and interoperability and not require a two way communication between an encoder and a decoder to convey that information, which is not practical in many applications, the standard specifies the minimum CPB buffer each decoder must have. It is an encoder's responsibility to generate the bitstream so that the CPB does not overflow and underflow.

In order to make sure that the encoders do not generate a bitstream that is not compliant with AVC/H.264, a hypothetical reference decoder (HRD) model is provided. This model

contains input CPB, an instantaneous decoding process and an output decoded picture buffer (DPB). It also includes the timing models – rate and time when the bytes arrive at the input of those buffers and when the bytes are removed from those buffers. An encoder must create the bitstreams so that the CPB and DPB buffers of HRD do not overflow or underflow.

There are two types of conformance that can be claimed by a decoder – output order conformance and output timing conformance. To check the conformance of a decoder, test bitstreams conforming to the claimed Profile and Level are delivered by a hypothetical stream scheduler to both HRD and the decoder under test. For an output order conformant decoder, the values of all decoded pixels and the order of the output pictures must be the same as those of HRD. For an output timing conformant decoder, in addition, the output timing of the pictures must also be the same as that of HRD.

5.2.10 Encapsulation/Packetization

To be able to adapt the coded bit stream easily for diverse applications such as broadcasting over Cable, Satellite and Terrestrial networks, streaming over IP networks, video telephony or conferencing over wireless or ISDN channels, the video syntax is divided into two layers – the Video Coding Layer (VCL), and non-VCL layer. The VCL consists of bits associated with compressed video. Non-VCL information consists of sequence and picture parameter sets, filler data, Supplemental Enhancement Information (SEI), display parameters, picture timing, etc. VCL and non-VCL bits are encapsulated into NAL Units (NALU). Originally NAL stood for Network Abstraction Layer but as the standard progressed, its purpose was modified so that only the acronym NAL was kept in the standard. The format of a NALU is as shown in Figure 5.15.

The first byte of each NALU is a header byte and the rest is the data. The first bit of the header is a 0 bit. The next two bits indicate whether the content of NALU consists of a sequence or picture parameter set or a slice of a reference picture. The next five bits indicate the NALU type corresponding to the type of data being carried in that NALU. There are 32 types of NALUs allowed. These are classified in two categories: VCL NAL Units and non-VCL NAL Units. NALU types 1 through 5 are VCL NALUs and contain data corresponding to the VCL. NALUs with NALU type indicator value higher than 5 are non-VCL NALUs and carry information like SEI, Sequence and Picture Parameter set, Access Unit Delimiter etc. For example, NALU type 7 carries the Sequence Parameter Set and type 8 carries the Picture Parameter Set. Non-VCL NALU may or may not be present in bitstream and may be sent separately by any means of external communication.

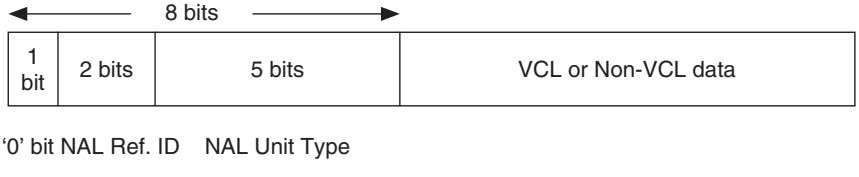

Figure 5.15 NAL unit format

5.2.11 Profiles

The AVC/H.264 standard was developed for a variety of applications ranging from video phone to entertainment TV to mobile TV to video conferencing to content creation in a studio. Those applications have different requirements. Requiring all of the decoders to implement all of the tools would make them unnecessarily complex and costly. Therefore, the standard divides the coding tools into different categories, called Profiles, based on various collections of applications. Each profile contains a sub-set of all of the coding tools specified in the standard. A decoder compliant with a certain profile must implement *all* of the tools specified in that profile. An encoder generating bitstream to be decoded by a decoder compliant to a certain profile is not allowed to use any tools that are not specified in that profile. However, it may choose to use a sub-set of tools specified in that profile.

Figures 5.16, 5.17 and 5.18 depict the various profiles specified in the standard. For consumer applications today, 4:2:0 format is used. Four profiles defined with those applications in mind are:

- Baseline
- Extended
- Main
- High.

At the time this text was written, JVT was in the process of creating fifth profile called Constrained Baseline Profile.

5.2.11.1 Baseline Profile

This was designed with mobile, cell phones and video conferencing applications in mind. In these applications the power consumption plays a very important role. Therefore, the complexity of encoders and decoders is desired to be low. B or B_R pictures and CABAC

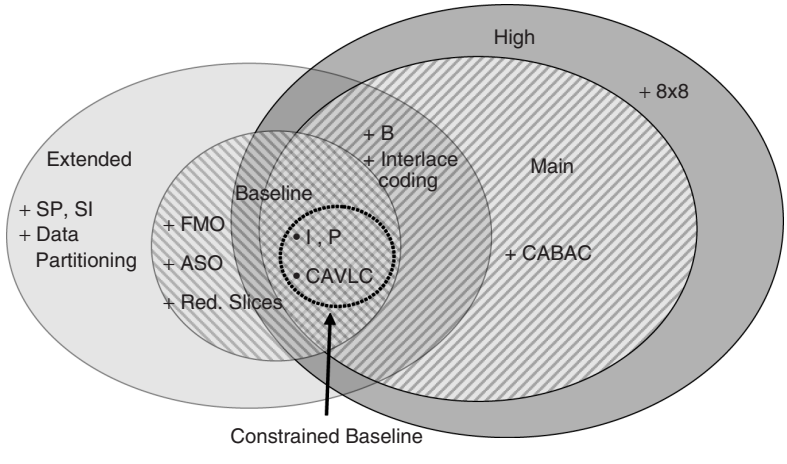

Figure 5.16 Profiles for 4:2:0 format

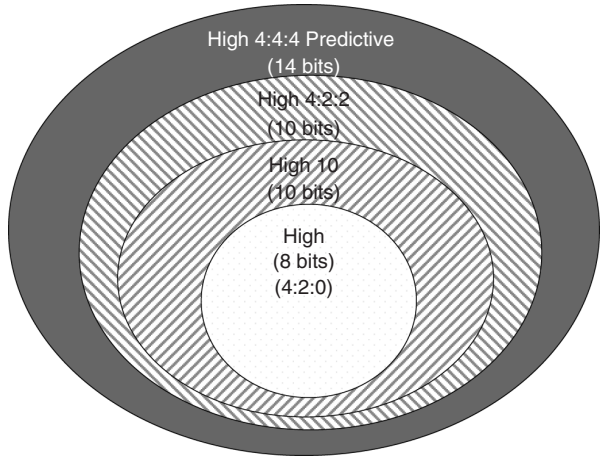

Figure 5.17 Hierarchy of high profiles

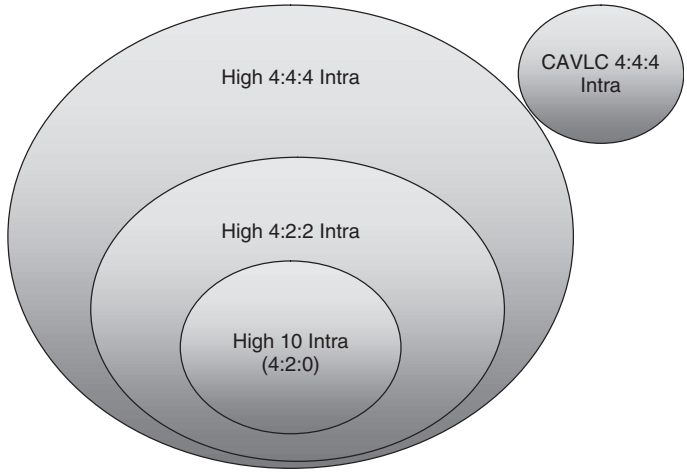

Figure 5.18 Intra-only profiles

coding tools are not allowed. Also, as the pictures in those applications are expected to be CIF or QVGA sizes, interlaced coding tools are also not included. Note that in this standard the motion in P pictures can be estimated based on the references in the future. To use the references that are in the future, an encoder has to wait for those references to arrive before starting the encoding. Therefore, P pictures do not necessarily imply that the bitstream is generated with low delay. Similarly, both the motion vectors in B pictures may use the pictures in the past as references. Therefore, the presence of B pictures in a bitsteam does not necessarily imply longer delays. In addition to the low power consumption, many of these applications operate in an environment that has a significant bit error rate. To allow for robust communication in that environment, three

error resilience tools are also added – Flexible Macroblock Ordering (FMO), Arbitrary Slice Order (ASO) and Redundant Slices. In FMO, the macroblocks are not sent in the order in which they are scanned and displayed. Their order is selected by an encoder depending upon burst error length. As the MBs are not in display order, errors within them are scattered over a wider picture area and become relatively less noticeable. In ASO, the slices can be sent in an arbitrary order and in Redundant Slices, as the name suggests, the same slices can be sent more than once so that the receiver can pick the ones with less error. Note that Baseline profile is not a sub-set of Main and High profiles as these profiles do not support the error resilience tools mentioned above. Therefore, a Baseline profile compliant bitstream cannot be decoded by Main and High profile decoders unless the error resilience tools are not used.

Constrained Baseline Profile

In many environments (for example, in Portable Media Players), the bit error rates are not high and there is a strong desire to simplify the encoder and decoder implementations as well as to be compatible with Main and High profiles so that devices like Set Top Boxes and TVs, that implement those popularly used profiles in Digital Television, can also decode the bit stream. In order to achieve this goal, the AVC standard allows for the setting of a flag, called 'constraint_set1_flag', in the bitstream syntax to 0 or 1. When this flag is set to 1, it indicates that those error resilience tools are not used. Currently, there is no specific name assigned to this setting. However, at the time this text was written, JVT was in the process of formally adopting the name Constrained Baseline Profile corresponding to this setting.

5.2.11.2 Extended Profile

This profile is a superset of the Baseline profile and extends it to streaming video applications, as perceived at the time the standard was designed. In addition to including all of the tools allowed in the Baseline profile, it allows for the usage of SP and SI so that if there is a sudden change in the channel bandwidth the bitstream can be switched to the one using a lower bit rate. In order to increase coding efficiency and allow for the use of higher resolution (SD and HD) pictures, interlaced coding tools and B pictures are allowed. To keep the complexity of a decoder lower, it does not allow CABAC.

5.2.11.3 Main Profile

This profile was designed so as to provide high coding efficiency with Digital TV as one of the main applications. Therefore, it includes B-pictures, CABAC and Interlaced Coding tools. Note that it supports both CABAC and CAVLC. This allows for compatibility with Constrained Baseline profile. As, the error rates after FEC (Forward Error Correction) are not expected to be high (typically, less than 10^{-6}), the error resilience tools (FMO, ASO and RS) are not included in this profile.

5.2.11.4 High Profile

This profile provides the highest coding efficiency and highest possible flexibility with regard to the coding tools that can be used on the encoder side for a 4:2:0 color format

video. It is a super set of the Main profile. In addition to all the tools used in the Main profile, it includes 8×8 transform, 8×8 Intra Prediction and downloadable Quantization/Weighting tables. On average, this profile is reported to provide about 10% higher coding efficiency in comparison to the Main Profile for 720p formats. This also allows one to fine tune the look and visual nature of video, e.g. preservation of film grain or picture by picture control of the distortion in order to provide more pleasing or desirable visual quality.

5.2.11.5 High10 Profile

This profile allows for the use of 10 bit video for 4:2:0 formats. This was developed in order to provide higher quality video for Electronic Cinema, contribution grade video, studios and possibly next generation High Definition DVD applications. As these applications do not typically use interlaced scanning, the 4:2:0 format is considered to be adequate for the lower end of those applications.

5.2.11.6 High 4:2:2 Profile

This profile is a super set of the High10 profile and allows for the use of the 4:2:2 color format for compression. As this format is very commonly used to capture, store and process video, the High 4:2:2 profile allows for compression without requiring the steps of converting video to the 4:2:0 format and then back to the 4:2:2 format. This avoids the artifacts introduced in converting 4:2:2 to 4:2:0 and back to the 4:2:2 format.

5.2.11.7 High 4:4:4 Predictive Profile

This profile is a super set of the High10 and High 4:2:2 profiles and allows for the highest color resolution and picture quality. Chroma has the same resolution as Luma. It allows for up to 14 bits per luma and chroma pixel values.

5.2.11.8 Intra Only Profiles

In studios and other environments where a great deal of video editing is done, it is desirable to have the capability to access each picture independently. To support those and Digital Cinema applications, a set of profiles is provided that allows no temporal prediction and where each picture is compressed in intra modes. As shown in Figure 5.18, these profiles cover the 4:2:0, 4:2:2 and 4:4:4 video chroma sampling formats. As is also shown in the figure, except for the CAVLC 4:4:4 Intra Profile, a decoder compliant with the higher chroma resolution profile is required to be able to decode the bitstream compliant with a profile with lower chroma resolution at the same level.

5.2.12 Levels

It is not practical to require every decoder to be able to decode a bitstream corresponding to all of the possible resolutions ranging from QCIF (176×144) at 15 frames/sec to Digital Cinema ($4k \times 2k$) resolution with 72 frames/sec. Therefore, the standard also

specifies Levels for each Profile. A Level specifies the parameters that impact upon the processing power needed to decode the bitstream in real time and also the maximum memory required to decode a bitstream compliant with that level. Unlike MPEG-2, resolution is not specified in terms of maximum horizontal and vertical sizes. Realizing that one of the more fundamental parameters in the design of a decoder is pixels per picture, MPEG-4 Part 2 specified the Level constraints in terms of the number of pixels in a picture rather than the horizontal and vertical sizes of a picture. The same approach was followed in the AVC. Similarly, instead of frame rates, pixel rates are specified. Currently, the standard specifies 16 Levels. The Level of a decoder is another axis of the compliance plane. The compliance of a decoder is indicated by the Profile and the Level.

5.2.12.1 Maximum Bit Rates, Picture Sizes and Frame Rates

Table 5.5 shows the maximum bit rate (for VCL), the maximum number of 16×16 macroblocks (MBs) in a picture and the maximum MB rates for the levels defined in the standard. When NAL is included then the max bit rate increases by 20 %. The last column also provides the typical picture resolution of the picture and frame rate used in products compliant to that Level. Other picture sizes at different frame rates can also be used as long as the maximum number of MBs per picture is less than the one specified in the third column and the maximum frame rate is such that the pixel rate for the given picture size is less than the maximum MBs/sec specified in the fourth column. In addition to the constraints in Table 5.5, the aspect ratio of a picture was also constrained to be such that the horizontal and the vertical sizes cannot be more than *square root* ($8 \times$ *maximum frame size*). This is done to avoid placing the undue burden on the decoders of adding the capability to configure the memory arbitrarily. Furthermore, to limit the maximum frame rate corresponding to small picture sizes at a given level, the maximum frame rate is further restricted to be $\leq$172 fps.

As Table 5.5 shows, this standard is applied from the low rate of 10s of kbps to 100s of Mbps. Unlike previous standards, this standard provides high coding efficiency over 5 orders of magnitude of bit rate range!

Level 3 is also known as the SD level, Level 4 as the HD level and Level 4.2 is also known as the 1080 60p standard in digital TV applications. Currently Level 3 and Level 4 are widely deployed and various activities are under way to specify the use of Level 4.2 (1080 60p) for digital TV (see Chapter 10).

5.2.12.2 Maximum CPB, DPB and Reference Frames

To limit the maximum memory required in a decoder, the maximum CPB and DPB buffer sizes are also specified. Table 5.6 shows the maximum allowed CPB and DPB sizes at various levels. A decoder is required to have an input buffer memory at least as big as specified in this table. An encoder cannot produce a bit stream so as to require more than the specified buffer size. This standard specifies greater coded picture buffer sizes than previous standards. This was done in order to allow for maximum flexibility for encoders in optimizing video quality and decoding delay and thus achieving the best delay versus video quality trade-off required. By using a larger CPB buffer size an encoder can compress video with larger variation in bits per frame so as to accommodate larger

Table 5.5 Levels and associated bit rate, picture size and frame rate constraints for 4:2:0 format profiles

Level Number	Max. Compressed Bit Rate	Max. MBs per picture	Max. MB Rate (MB/sec)	Typical picture size and Rate
1	64 kbps	99	1485	SQCIF (128 × 96) × 30fps QCIF (176 × 144) × 15fps
1b	128 kbps	99	1485	QCIF (176 × 144) × 15fps
1.1	192 kbps	396	3000	QCIF × 30fps
1.2	384 kbps	396	6000	CIF (352 × 288) × 15fps
1.3	768 kbps	396	11 880	CIF × 30fps
2	2 Mbps	396	11 880	CIF × 30fps
2.1	4 Mbps	792	19 800	525 HHR (352 × 480) × 30fps 625 HHR (352 × 576) × 25fps
2.2	4 Mbps	1620	20 250	525 SD (720 × 480) × 15fps 625 SD (720 × 576) × 12.5fps
3	10 Mbps	1620	40 500	525 SD × 30fps 625 SD × 25fps VGA (640 × 480) × 30fps
3.1	14 Mbps	3600	108 000	720p HD (1280 × 720) × 30fps
3.2	20 Mbps	5120	216 000	720p HD × 60fps
4	20 Mbps	8192	245 760	720p HD × 60fps 1080 HD (1920 × 1088) × 30fps 2k × 1k (2048 × 1024) × 30fps
4.1	50 Mbps	8192	245 760	
4.2	50 Mbps	8704	522 240	1080 HD × 60fps 2k × 1k × 60fps
5	135 Mbps	22 080	589 824	2k × 1k × 72fps
5.1	240 Mbps	36 864	983 040	2k × 1k × 120fps 4k × 2k (4096 × 2048) × 30fps

variations in the complexity of video quality. On the other hand, higher CPB buffer size causes higher decoding delay. The third column in the table provides examples of the maximum delays experienced by bitstreams that use the maximum CPB buffer sizes and are compressed at the maximum bit rates allowed by the levels.

The fourth column provides the maximum DPB size specified in the standard at various levels. This limits the maximum number of reference frames that can be used for those picture sizes. The fifth column shows the maximum number of reference frames corresponding to the DPB size limit specified in the standard. Note that for a given level one can use a smaller picture size and as a result use a greater number of reference frames as

Table 5.6 CPB and DPB sizes for 4:2:0 format profiles

Level Number	Max CPB size	Max delay at max bit rate	Max DPB size (Bytes)	Max number of reference frames for typical pictures sizes
1	175 kbits	2.7 sec	152 064	SQCIF: 8 QCIF: 4
1b	350 kbits	2.7 sec	152 064	QCIF: 4
1.1	500 kbits	2.6 sec	345 600	QCIF: 9
1.2	1 Mbits	2.6 sec	912 384	CIF: 6
1.3	2 Mbits	2.6 sec	912 384	CIF: 6
2	2 Mbits	1 sec	912 384	CIF: 6
2.1	4 Mbits	1 sec	1 824 768	525 HHR: 7 625 HHR: 6
2.2	4 Mbits	1 sec	3 110 400	525 SD: 6 625 SD: 5
3	10 Mbits	1 sec	3 110 400	525 SD: 6 625 SD: 5 VGA: 6
3.1	14 Mbits	1 sec	6 912 000	720p HD: 5
3.2	20 Mbits	1 sec	7 864 320	720p HD: 5
4	25 Mbits	1.25 sec	12 582 912	720p HD: 9 1080 HD: 4 2k × 1k: 4
4.1	62.5 Mbits	1.25 sec	12 582 912	720p HD: 9 1080 HD: 4 2k × 1k: 4
4.2	62.5 Mbits	1.25 sec	13 369 344	1080 HD: 4 2k × 1k: 4
5	135 Mbits	1 sec	42 393 600	2k × 1k: 13
5.1	240 Mbits	1 sec	70 778 880	2k × 1k:16
				4k × 2k: 5

more frames can in that case fit into the maximum allowed DPB size. To avoid the number of reference frames growing out of hand, a maximum limit of 16 frames, irrespective of the picture size, is also specified. Similarly, to cap the frame rate for small picture size for levels that allow for a high pixel rate, a maximum frame rate of 172 frames/sec is also specified.

A decoder compliant with a certain level is also required to be compliant with lower levels.

5.2.13 Parameter Sets

In addition to the compressed bits related to a video sequence, extra information is needed for the smooth operation of a digital video compression system. Furthermore, it is helpful to decoders if some information about some of the parameters used while compressing digital video is provided at a higher level in the syntax. Sequence Parameter Sets (SPS) and Picture Parameter Sets (PPS) provide that information.

5.2.13.1 Sequence Parameter Sets (SPS)

SPS provides information that applies to the entire coded video sequence. It includes information such as Profile and Level that is used to generate the sequence, the bit depth of the luma and chroma, the picture width and height and Video Usability Information (VUI) among other parameters.

Video Usability Information (VUI)

To be able to display a decoded video properly, it is helpful to have information such as color primaries, picture and sample aspect ratios and opto-electronic transfer characteristics, etc. These parameters are conveyed as part of the VUI. This information is sent as part of the Sequence Parameter Sets at the Sequence layer.

5.2.13.2 Picture Parameter Sets (PPS)

PPS provides information that applies to a coded picture. It includes information such as which of the two entropy coders is used, the scaling matrix, whether the default deblocking filter setting is used or not and whether 8×8 transform is used or not, etc.

5.2.14 Supplemental Enhancement Information (SEI)

The normative part of the standard is focused on the bitstream syntax and coding/decoding tools. It does not deal with other aspects of a digital video system that are required for the issues related to display and other information useful in a system. To help in providing that information the SEI messaging syntax is also developed in the standard. A system may or may not use SEI messages. Some key commonly used SEI messages are:

- Buffering period
- Picture timing
- Recovery point
- Pan scan rectangle
- User data

Buffering period SEI specifies parameters such as the delay related to the buffering and the decoding of the bitstream. Picture timing SEI provides information as to whether a frame is an interlace or progressive frame and if it is an interlaced frame then what is the order of fields (top and then bottom or bottom then top) and how should fields be repeated when 24 frames per sec movie material is displayed on TV with a 60 fields

per sec refresh rate. Recovery point SEI provides information about how long it will take to fully recover a decoded picture without any distortion when one enters the stream at a given access point. Pan scan rectangle provides information on what region of a picture with wider aspect ratio should be displayed on a screen with narrower aspect ratio. User data allows a user to send private (not standardized by ISO/IEC and ITU-T committee) data. More detail is provided in Chapter 10.

5.2.15 Subjective Tests

In Figure 5.2, an example of a PSNR based comparison of AVC and MPEG-2 is provided. However, the human visual system perceives video quality differently than PSNR measurement. Therefore, the performance of a digital video processing system must also be evaluated based on visual/subjective tests. Several organizations, including the MPEG committee, have done subjective testing and comparison of AVC with MPEG-2 and other standards. Subjective tests done by the MPEG committee [15, 16] have been published. Those tests showed that the coding efficiency of AVC is about twice the coding efficiency of MPEG-2 also based on subjective tests, or in other words, AVC is able to provide a similar visual quality as MPEG-2 at about half the bit rate.

References

1. ISO/IEC JTC 1 and ITU-T, "Generic coding of moving pictures and associated audio information – Part 2: Video," ISO/IEC 13818-2 (MPEG-2) and ITU-T Rec. H.262, 1994 (and several subsequent amendments and corrigenda).
2. ISO/IEC JTC 1 and ITU-T, "Advanced video coding of generic audiovisual services," ISO/IEC 14496-10 (MPEG-4 Part 10) and ITU-T Rec. H.264, 2005 (and several subsequent amendments and corrigenda).
3. ISO/IEC JTC1, "Coding of audio-visual objects – Part 2: Visual," ISO/IEC 14496-2 (MPEG-4 visual version 1), 1999; Amendment 1 (version 2), 2000; Amendment 4, 2001 (and several subsequent amendments and corrigenda).
4. ITU-T, "Video coding for low bit rate communication," ITU-T Rec. H.263; v1: Nov. 1995, v2: Jan. 1998, v3: Nov. 2000.
5. P. Kuhn, "Algorithms, complexity analysis and VLSI architectures for MPEG-4 motion estimation," Kluwer Academic Publishers, 1999.
6. M. Flierl and B. Girod, "Video coding with superimposed motion-compensated signals – applications to H.264 and beyond," Kluwer Academic Publishers, 2004.
7. H. S. Malvar, A. Hallapuro, M. Karczewicz and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 13, No. 7, pp. 598–603, July 2003.
8. A. Hallapuro, M. Karczewicz and H. Malvar, "Low complexity transform and quantization," JVT-B038, 2nd JVT meeting: Geneva, CH, Jan 29 – Feb 1, 2002.
9. I. Witten, R. Neal, and J. Cleary, "Arithmetic coding for data compression," Comm. of the ACM, Vol. 30, No. 6, pp. 520–540, 1987.
10. W. B. Pennebaker and J. L. Mitchell, "JPEG still image compression standard," Van Nelson Reinhold, 1993.
11. D. Marpe, H. Schwarz and T. Wiegand, "Context-based adaptive binary arithmetic coding in H.264/AVC video compression standard," *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 13, No. 7, pp. 620–636, July 2003.
12. A. Puri, X. Chen, and A. Luthra, "Video coding using the H.264/MPEG-4 AVC compression standard", *Signal Processing: Image Communication*, Vol. 19, pp. 793–849, June 2004.
13. I. Moccagatta and K. Ratakonda "A performance comparison of CABAC and VCL-based entropy coders for SD and HD sequences," JVT-E079r2, 5th JVTMeeting: Geneva, CH, 9-17 October, 2002

14. M. Karczewicz and R. Kurceren, "The SP- and SI-Frames Design for H.264/AVC," *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 13, No. 7, pp. 637–644, July 2003.
15. ISO/IEC JTC 1/SC 29/WG 11 (MPEG), "Report of the formal verification tests on AVC/H.264," MPEG document N6231, Dec., 2003 (publicly available at http://www.chiariglione.org/mpeg/quality_tests.htm).
16. C. Fenimore, V. Baroncini, T. Oelbaum, and T. K. Tan, "Subjective testing methodology in MPEG video verification," SPIE Conference on Applications of Digital Image Processing XXVII, July, 2004.