CHAPTER 14

DV Compression

Introduction

The DV compression scheme is the product of cooperation among a large number of manufacturers, but with four companies performing most of the development and owning most of the intellectual property. It began, I am told, as an exercise to produce a compression scheme for use in high-definition consumer camcorders, using a data rate of 50 Mbits/s. It soon became apparent that the work presented an opportunity for a new generation of standard-definition camcorders working at 25 Mbits/s and offering the "digital" cachet. The 25 Mbits/s rate was very suitable, because recording at this rate can be implemented quite easily with robust, inexpensive technology.

Even then, DV exceeded its expectations; it is sometimes described as the technology that went from engineers to users before the marketing departments realized what was happening! Not only is it successful in the consumer market, but the technology and its derivatives have been adopted in the profession television broadcast market—in fact, video recorders based on DV it is claimed that have been adopted more rapidly than any other format. Two versions of DV are in use professionally. 25 Mbits/s systems, very similar to the consumer DV products, are used for acquisition, particularly in news environments. A 50 Mbits/s version is used in studios and for postproduction.

Although DV is based on the DCT transform, many aspects are quite different from the MPEG approach. Before looking at the details, it is important to review the design criteria, remembering that the original intention was a consumer product. We have seen that motion estimation is a very expensive process most suitable for an asymmetric environment (few encoders, many decoders). For a consumer product, it was determined that there was no sufficiently economical system of temporal compression, so DV had to be an intraframe system. This approach allows simple editing, also a substantial advantage.

When one is recording on a videotape, it is most helpful if the bit stream to be recorded has a constant number of bits per frame. This was also a design criterion for DV.

MPEG can of course work with I-frames only. However, the MPEG model controls bit rate by a feedback system. It is possible to approximate constant-bit rate compression with MPEG (as specified for the D-10 format mentioned in the previous chapter), but stuffing is generally necessary to achieve an exact fixed bit rate.

MPEG enthusiasts would argue that fixed bit rate is, in fact, undesirable. As we discussed earlier, the human psychovisual system is more sensitive to changes in perceived quality than it is to absolute quality. Constant bit rate with images of varying complexity necessarily implies variable quality of the compressed image. DV proponents, on the other hand, show research results that supposedly demonstrate that DV techniques give a lower minimum quality in a sequence of frames with varying complexity. The arguments will continue!

Finally, the fact that DV was intended for consumer equipment meant cost effectiveness was a prime design goal. It was required that compression be performed on a single chip, and the hope was that the same chip could be used for both compression and decompression. These goals were achieved.

Basic Concepts of DV Compression

DV uses the DCT transform; it also performs quantization of the coefficients, as in JPEG and MPEG. However, DV is specifically designed to generate a fixed number of bits from each frame. It uses some very sophisticated techniques to ensure that bit allocation is optimal for every frame.

The process is described in detail below, but we start with a brief summary. The first step, if necessary, is to reduce the number of color difference samples. For 25 Mbits/s consumer products there are different approaches for 525/60 and 625/50 video. For 525/60 systems the video (normally 4:2:2) is first reduced to 4:1:1. This is an appropriate choice when the video will eventually be coded to NTSC. For 625/50 systems it was decided that DV video should be compatible with the MPEG-2 video transmitted using the Digital Video Broadcasting (DVB) system—adopted in Europe for satellite, cable, and terrestrial transmission—so 4:2:0 coding was selected. For the professional version of DV compression described below, 4:1:1 is used for both 525/60 and 625/50 at 25 Mbits/s; 4:2:2 is used for both standards for 50 Mbits/s compression.

The next step is to break the image up into smaller pieces, each of which is allocated a fixed number of bytes. DV uses macroblocks, much as JPEG and MPEG do. Five macroblocks make a *segment*, and each segment must be compressed to the same number of bytes. The five macroblocks in one segment are taken from different parts of

the frame, in an attempt to ensure that no segment gets more than its share of complexity.

Each 8×8 block in the segment is DCT transformed and analyzed for complexity and, depending on the complexity, is allocated to one of four banks of quantization tables. Each bank is optimized for compression of blocks in a particular energy range. There are effectively 16 tables in each bank, and each of these corresponds to a level of quantization scaling. The system then quantizes the coefficients of all blocks in the segment, using a table from the correct bank for each block. This is performed for each of the 16 possible quantization-scaling factors, and the system selects the scaling factor closest to, but not exceeding, 385 bytes of quantized coefficients. Thus, each segment is compressed to an equal number of bytes, but within that segment each block is quantized using a table appropriate to the energy of that block. Complex blocks receive a greater share, and simple blocks a lesser share, of the bits available. There is a third mechanism that helps distribute the bits optimally; this is discussed in the more detailed description that follows.

Detailed Description

25 Mbits/s Compression

DV compression for consumer products is specified by IEC document 61834. The DV-based compression used for professional products described in this section is specified by SMPTE Standard 314M, and figures used in this section are reproduced from this standard, by kind permission of the Vice President, Engineering, of SMPTE. These two documents are the definitive descriptions of DV compression and recommended reading for anyone working with DV video.

Prior to compression to the 25 Mbits/s professional standard, the signal must be reduced from the (normal) 4:2:2 coding to 4:1:1 coding. The DV standard does not specify any filtering, but merely discards pixels from the 4:2:2 input, as shown in Figure 14-1 and Figure 14-2. Depending on the source of the images, appropriate prefiltering may be necessary to prevent artifacts being generated by this decimation process.

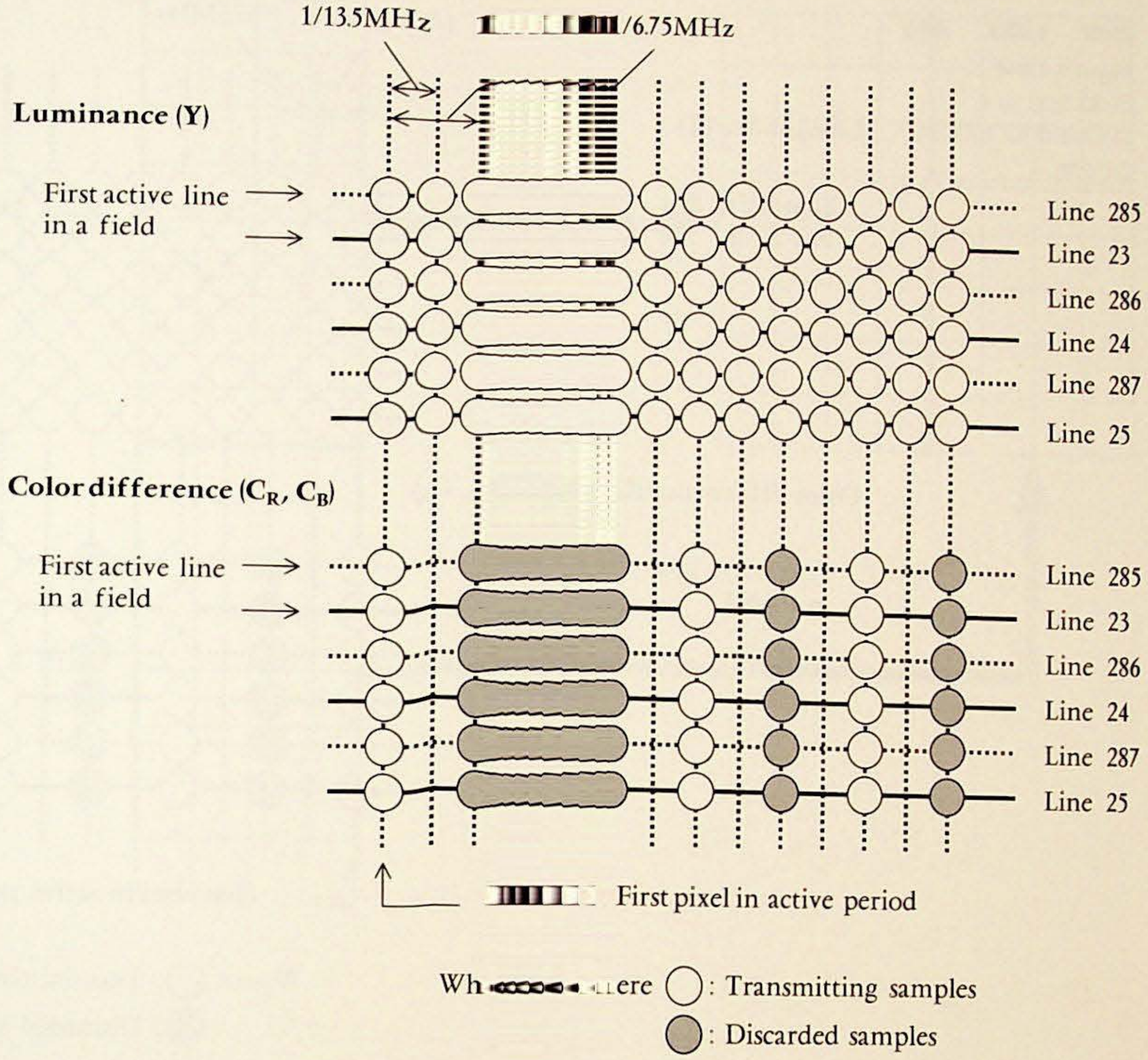


Figure 14-1
Reduction to 4:1:1 sampling for 525/60 systems.

The image is then divided into blocks and macroblocks. In the consumer 625/50 standard with 4:2:0 coding, this process is straightforward, and the same as MPEG. 4:1:1 coding, however, presents a slight problem, because there are only 180 samples of each color difference signal on a line, and 180 is not divisible by 8; we are left with a "half block" at the end of the lines. This is resolved by creating a special "end" macroblock, as shown in Figure 14-3.

The macroblocks are grouped into *superblocks*, as shown in Figures 14-4 and 14-5. Then the *segments* are formed by taking five macroblocks for each segment. The segments are assembled in a pseudo-random manner—within a segment the five macroblocks are taken from different positions in five different superblocks, no two in the same row or column of superblocks. At this stage, the segment consists of 30 blocks of 8x8 pixels, six blocks (four luminance blocks plus

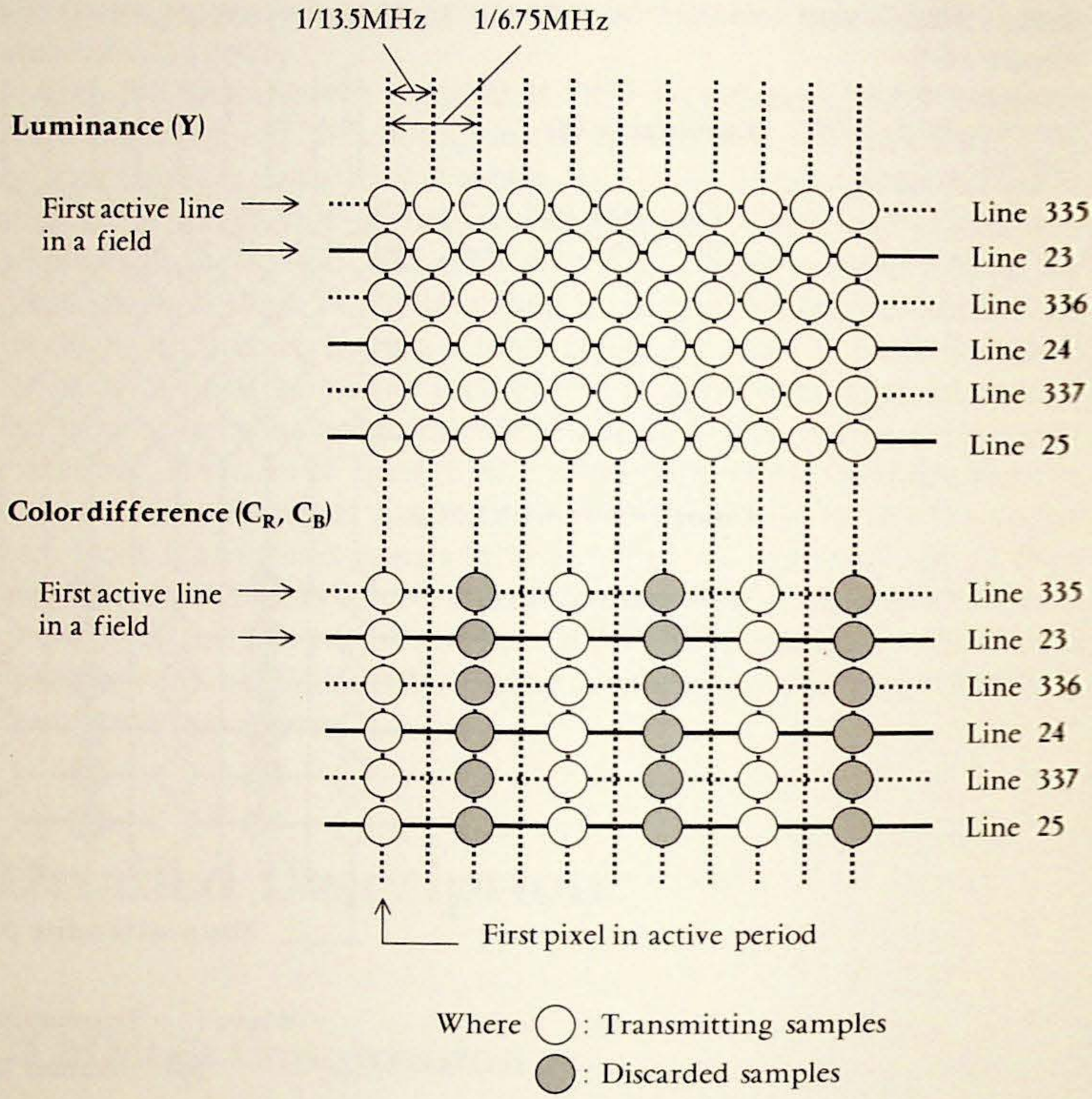
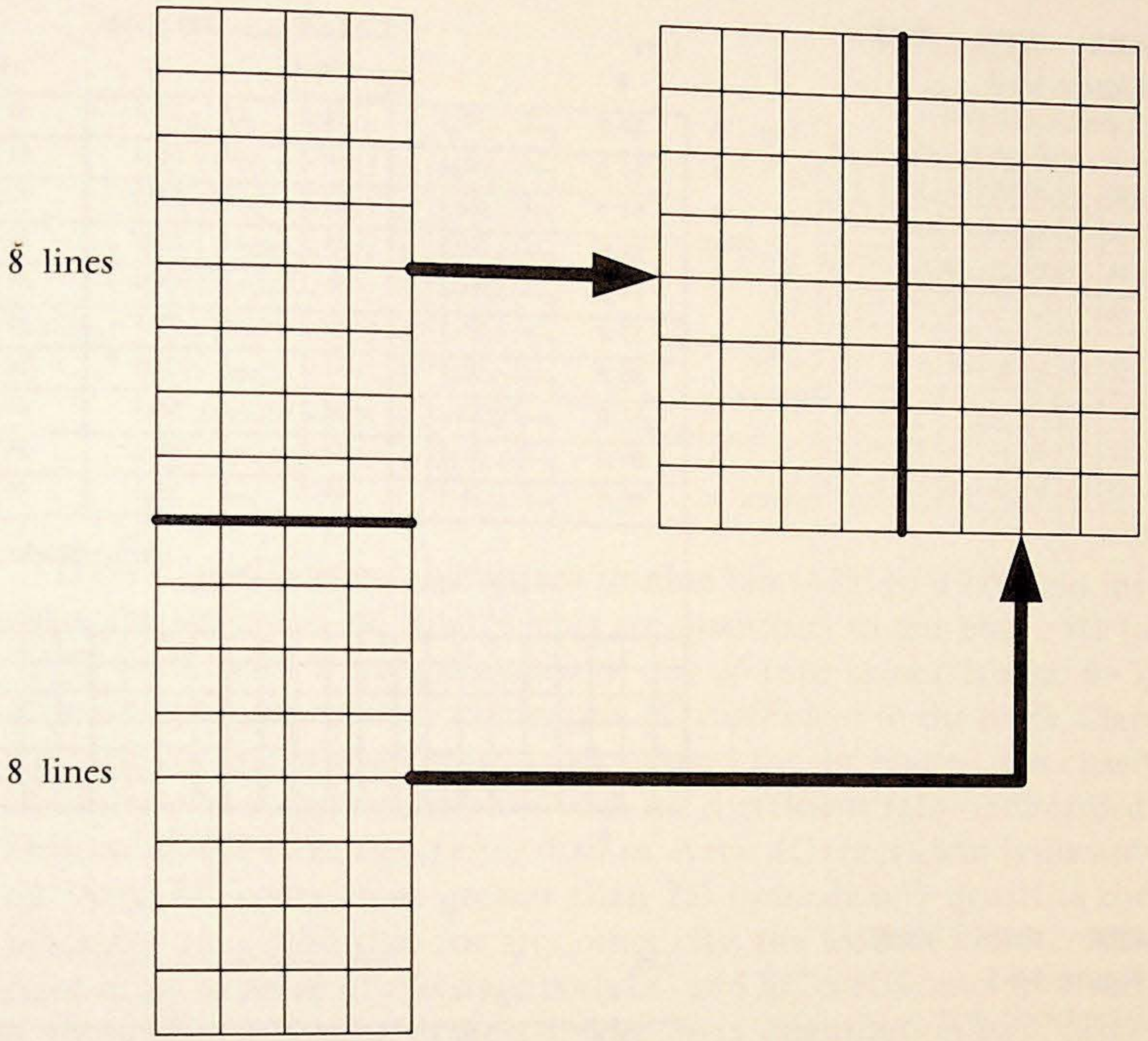


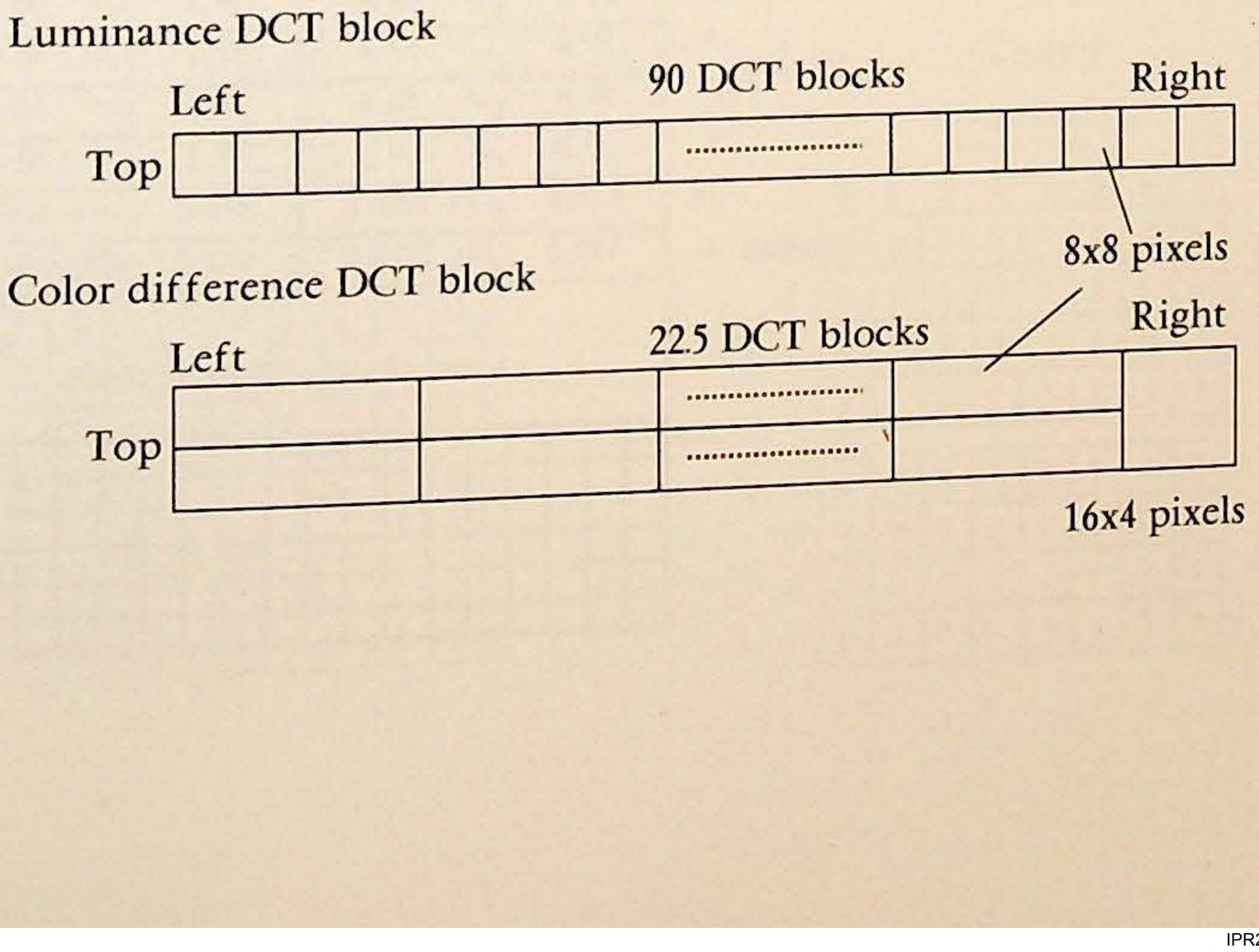
Figure 14-2
Reduction to 4:1:1 sampling for 625/50 systems.

one of each color difference block) in each macroblock. Each pixel value is expressed as an 8-bit word, so the segment comprises 30×8×8 = 1920 bytes of data.

Each block is DCT transformed. DV provides for both frame and field DCT coding, but unlike MPEG-2, the decision is made for each block, and affects only that block. The two modes are known as *8-8-DCT*, used for blocks where there is little content variation between the two fields, and *2-4-8-DCT*, used where the content variation is significant (usually as a result of motion in that area of the image). In DV a 2-4-8-DCT is coded as two 8×4 blocks; the top block is the sum of adjacent rows of pixels, the bottom block the difference between adjacent rows. This is illustrated in Figure 14-6.

Figure 14-3
Arrangement of DCT blocks for 4:1:1 encoding.





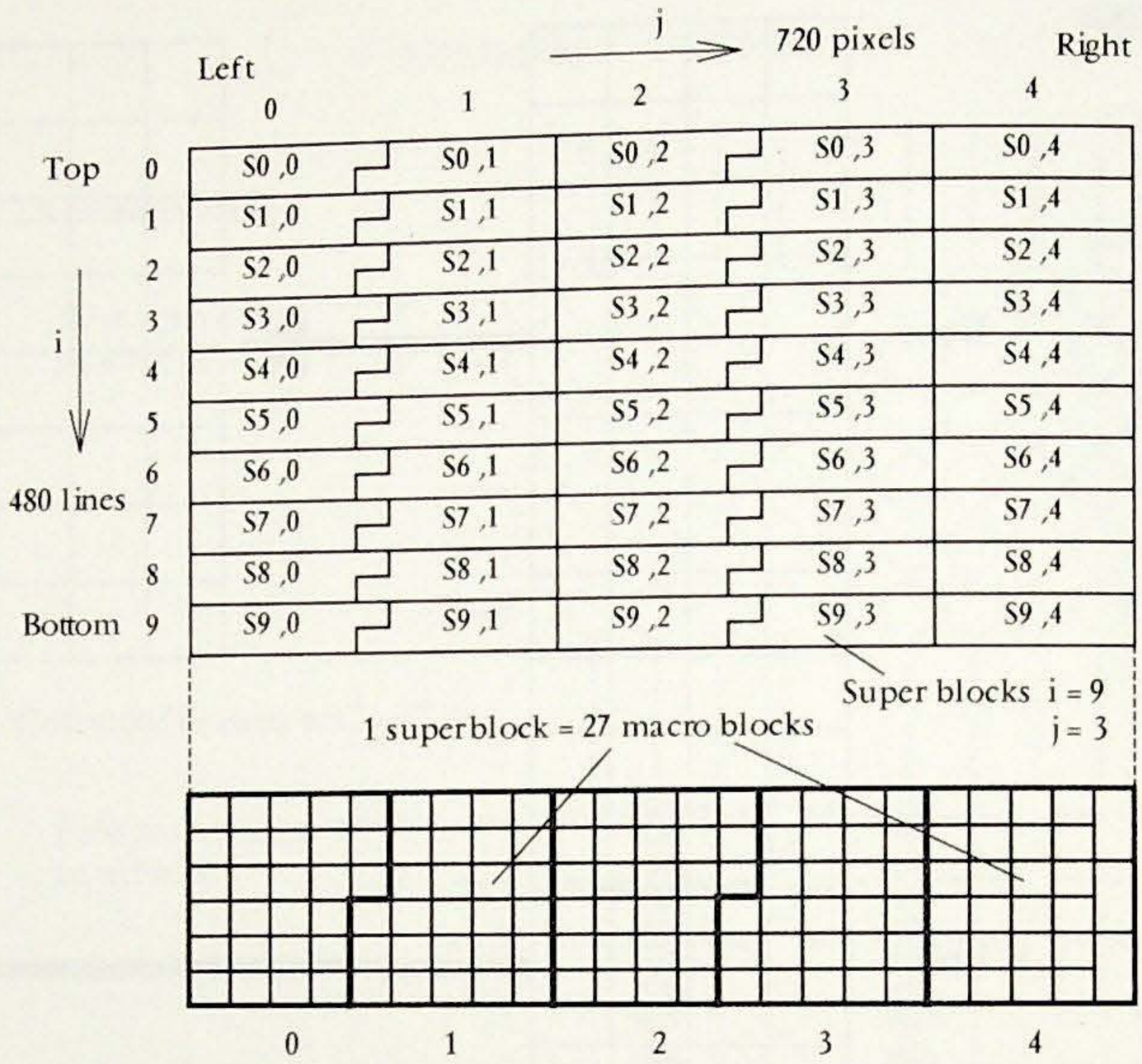


Figure 14-4
Superblocks and macroblocks in one television frame for 525/60 system with 4:1:1 compression.

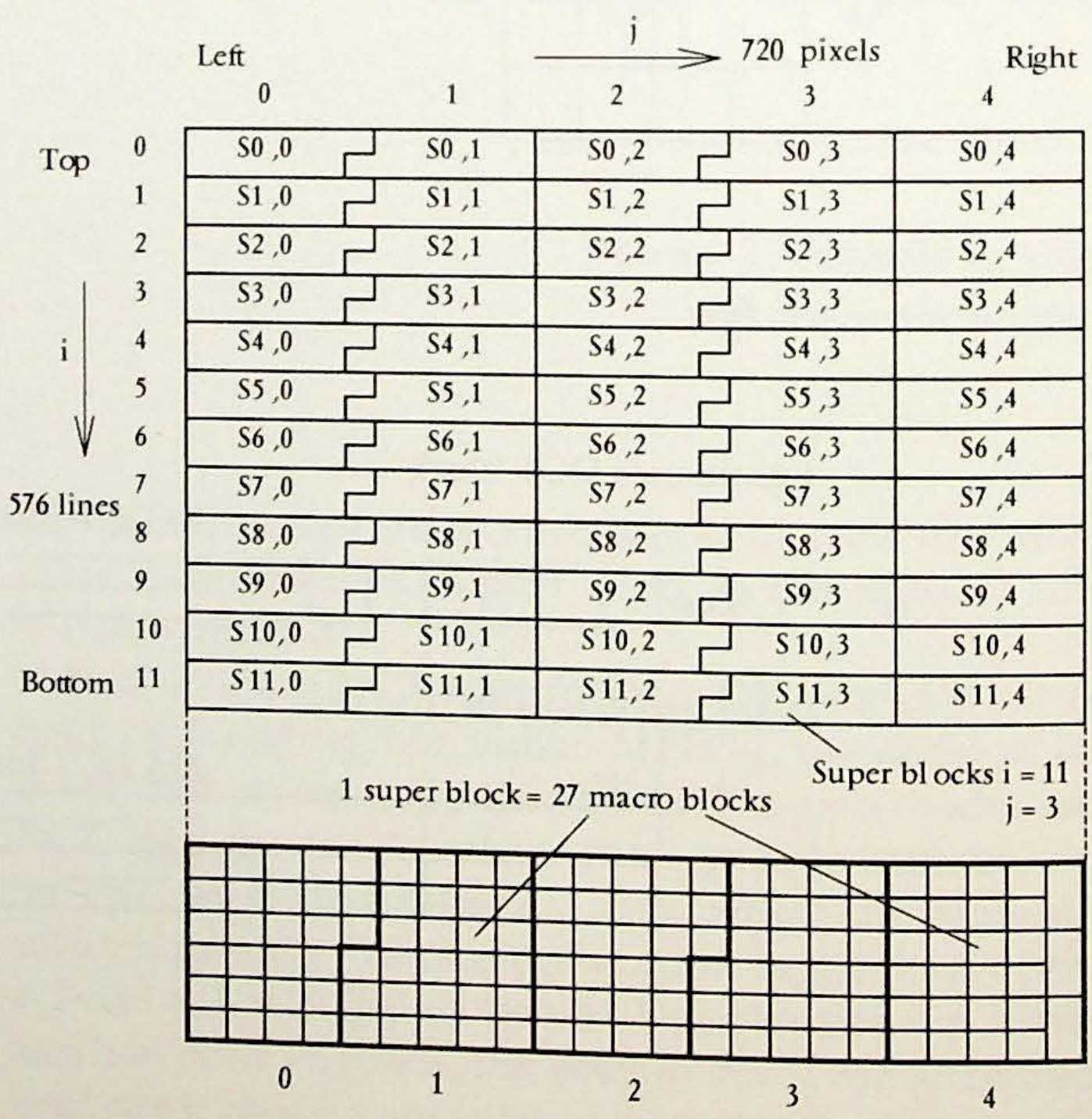


Figure 14-5
Superblocks and macroblocks in one television frame for 625/50 system with 4:1:1 compression.

Figure 14-6
The two DCT modes of DV, and the scanning order of the resulting DCT coefficients.

8-8-DCT

horizontal →

Vertical ↓	0	1	2	3	4	5	6	7
0	1	2	6	7	15	16	28	29
1	3	5	8	14	17	27	30	43
2	4	9	13	18	26	31	42	44
3	10	12	19	25	32	41	45	54
4	11	20	24	33	40	46	53	55
5	21	23	34	39	47	52	56	61
6	22	35	38	48	51	57	60	62
7	36	37	49	50	58	59	63	64

2-4-8-DCT

horizontal →

(sum) / Vertical ↓	0	1	2	3	4	5	6	7
0	1	3	7	19	21	35	37	51
1	5	9	17	23	33	39	49	53
2	11	15	25	31	41	47	55	61
3	13	27	29	43	45	57	59	63
(difference) 4	2	4	8	20	22	36	38	52
5	6	10	18	24	34	40	50	54
6	12	16	26	32	42	48	56	62
7	14	28	30	44	46	58	60	64

The DC coefficient is quantized to nine bits (−255 to +255), and initially the weighted AC coefficients are quantized to ten bits (−511 to +511). Each block is then allocated to one of four classes, labeled 0–3, depending on the value of the largest AC coefficient in the block. Class 0 is used for the lowest energy blocks, class 3 for the highest. For class-3 blocks, the least significant bit of each AC coefficient value is discarded. Otherwise, the most significant digit of every AC coefficient is discarded. (Any AC coefficient greater than 255 immediately qualifies the block for class 3, so that for any other class the MSB of every coefficient must be zero.) At this stage both DC and AC coefficients are 9 bits.

Each 9-bit AC value is then divided by a quantization step, determined by the class number and an area number, dependent on the position of the coefficient in the transformed block. The area numbers are illustrated in Figure 14-7, and the derivation of the quantization step is shown in Figure 14-8.

Figure 14-7
Area numbers.

8-8-DCT

horizontal →

Vertical ↓	0	1	2	3	4	5	6	7
0	DC	0	0	1	1	1	2	2
1	0	0	1	1	1	2	2	2
2	0	1	1	1	2	2	2	3
3	1	1	1	2	2	2	3	3
4	1	1	2	2	2	3	3	3
5	1	2	2	2	3	3	3	3
6	2	2	2	3	3	3	3	3
7	2	2	3	3	3	3	3	3

2-4-8-DCT

horizontal →

(sum) / Vertical ↓	0	1	2	3	4	5	6	7
0	DC	0	1	1	1	2	2	3
1	0	1	1	2	2	2	3	3
2	1	1	2	2	2	3	3	3
3	1	2	2	2	3	3	3	3
(difference) 4	0	0	1	1	2	2	2	3
5	0	1	1	2	2	2	3	3
6	1	1	2	2	2	3	3	3
7	1	2	2	3	3	3	3	3

Figure 14-8
Derivation of quantization steps.

	Class number				Area number			
	0	1	2	3	0	1	2	3
Quantization number (QNO)	15				1	1	1	1
	14				1	1	1	1
	13				1	1	1	1
	12	15			1	1	1	1
	11	14			1	1	1	1
	10	13		15	1	1	1	1
	9	12	15	14	1	1	1	1
	8	11	14	13	1	1	1	2
	7	10	13	12	1	1	2	2
	6	9	12	11	1	1	2	2
	5	8	11	10	1	2	2	4
	4	7	10	9	1	2	2	4
	3	6	9	8	2	2	4	4
	2	5	8	7	2	2	4	4
	1	4	7	6	2	4	4	8
	0	3	6	5	2	4	4	8
		2	5	4	4	4	8	8
		1	4	3	4	4	8	8
		0	3	2	4	8	8	16
			2	1	4	8	8	16
			1	0	8	8	16	16
			0		8	8	16	16

This table needs a little explanation. The *quantization number* is the compression control, and varies between 0 and 15. Zero represents the highest compression (coarser quantization of the coefficients, so fewer bits in the encoded output) and 15 the lowest compression. The system seeks the highest quantization number that can be used, consistent with the final output of the compression system being not more than 385 bytes for the complete segment. Each block in the segment has already been assigned to one of the four classes. Now the system can test each quantization level. Each AC coefficient in each block is divided by the appropriate quantization step. For example, for quantization number 7, in a class-2 block, coefficients in area 0 would be divided by 2, coefficients in areas 1 and 2 would be divided by 4, and those in area 3 by 8. For the same quantization number, coefficients in a class-1 block would be divided by 1, 2, 2, and 4 respectively.

It may appear that the table entries for class 3 are "out of place." This is because the AC coefficients of class-3 blocks have already been divided by 2 as a result of discarding the least significant bit.

The quantized coefficients are read out in the scanning order shown in Figure 14-6. The combination of run-length zeros and the following coefficient value is then variable-length encoded in a manner similar to JPEG and MPEG, but with code tables optimized for

DV. As in JPEG and MPEG a 4-bit *end-of-block* (EOB) code is inserted when there are no more non-zero coefficients. Note that the variable-length—encoding step must be inside the loop that measures the number of bytes required for each quantization number. In other words, for each quantization number, the coefficients must be quantized, scanned, and variable-length—encoded before the length of the encoded segment is known.

When the correct quantization number has been found (the highest number that results in not more than 385 bytes for the segment), the resulting data is arranged as shown in Figure 14-9, which shows the data for one macroblock. After a 4-byte header that identifies the macroblock, 77 bytes are provided for macroblock data (5×77 = 385). The first byte carries 4 bits for the quantization number (0 to 15) and four bits to report error status and error concealment action in a videotape machine. Then comes the first luminance block DC coefficient (9 bits) plus one bit to indicate field or frame DCT, and two bits to indicate the class of the block (0 to 3). Then there is a space of 12½ bytes for AC coefficients. The other five blocks of the macroblock follow in similar manner, except that only 8½ bytes are provided for the AC coefficients of the color-difference blocks.

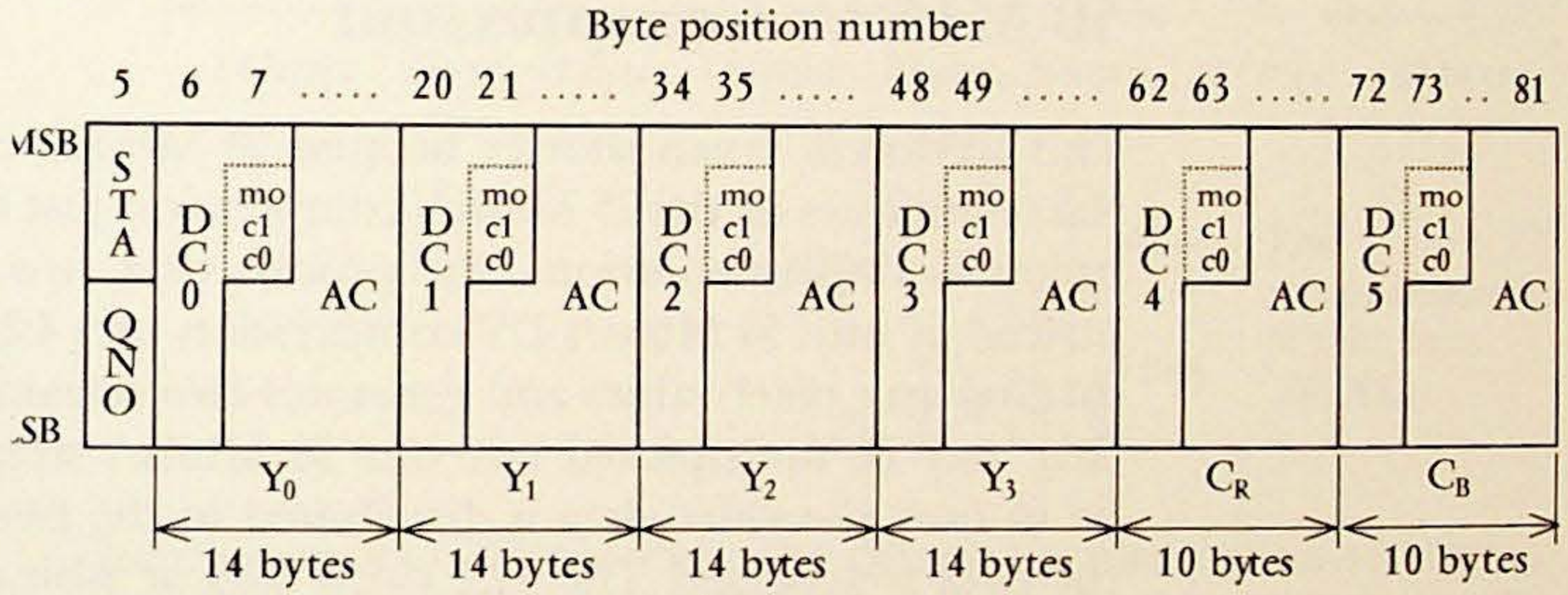


Figure 14-9
Arrangement of a compressed macroblock with 4:1:1 compression.

NOTES

STA:	Error status
QNO:	Quantization number
DC:	DC component
AC:	AC component
EOB:	End of block (0110)
mo:	DCT mode
c0, c1:	Class number

Of course, not all blocks will conveniently generate exactly 10½ or 8½ bytes of variable-length—coded AC coefficients. In some blocks the

EOB will occur before the end of the allocated space; for others there will be too much data for the space provided. DV employs a complex three-pass algorithm to take maximum advantage of the space available. The first pass places the data as described; the second and third passes attempt to place all the remaining data in the "empty" spaces after EOB in the blocks with little data.

So, for 4:1:1 compression of 525/60 there are 1,350 macroblocks per frame, or 270 segments per frame; approximately 30 frames/second. Each segment has a 4-byte header plus 385 bytes of data, so that the data rate is:

$$270 \times 389 \times 30 \times 8 = 25{,}207{,}200 \text{ bits/second}$$

For 625/50 systems with 576 active lines there are, 1,620 macroblocks or 324 segments per frame, and 25 frames per second, so the data rate is:

$$324 \times 389 \times 25 \times 8 = 25{,}207{,}200 \text{ bits/second}$$

Note that the 4:1:1 input data (at 8-bit precision) represents just under 125 Mbits/s, so that this mode of DV provides about 5:1 compression.

50 Mbits/s Compression

This section is much shorter because 50 Mbits/s compression uses all the techniques of the 25 Mbits/s compression just described. In fact, in practical implementations, the process uses two standard DV chips instead of one! 50 Mbits/s DV compression uses 4:2:2 video without discarding any pixel values and generates two streams of 25 Mbits/s data that may be multiplexed into one 50 Mbits/s stream. The differences lie in how the video data is distributed to the two channels and how the data is arranged. The 4:2:2 video data at 8-bit precision represents about 166 Mbits/s, so this mode of DV provides about 3.3:1 compression and is virtually transparent.

The macroblock for 4:2:2 DV compression has only four DCT blocks, two luminance and one each color difference, as shown in Figure 14-10. Again five macroblocks are grouped into one segment, and half the segments are sent to each channel.

With only four blocks per macroblock (and a compression factor of 3.3:1), quantization is less aggressive, and more space is needed for AC coefficients. This results in a revised layout for the compressed macroblock, as shown in Figure 14-11. As with the 25 Mbits/s system, a

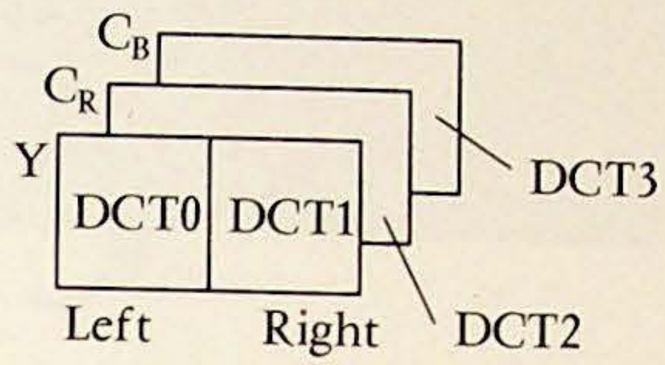


Figure 14-10
Macroblock and DCT blocks for 4:2:2 compression.

NOTES - DCTl: DCT block order
l = 0, 1, 2, 3

three-pass algorithm distributes excess coefficient data into otherwise unused spaces in the data structure.

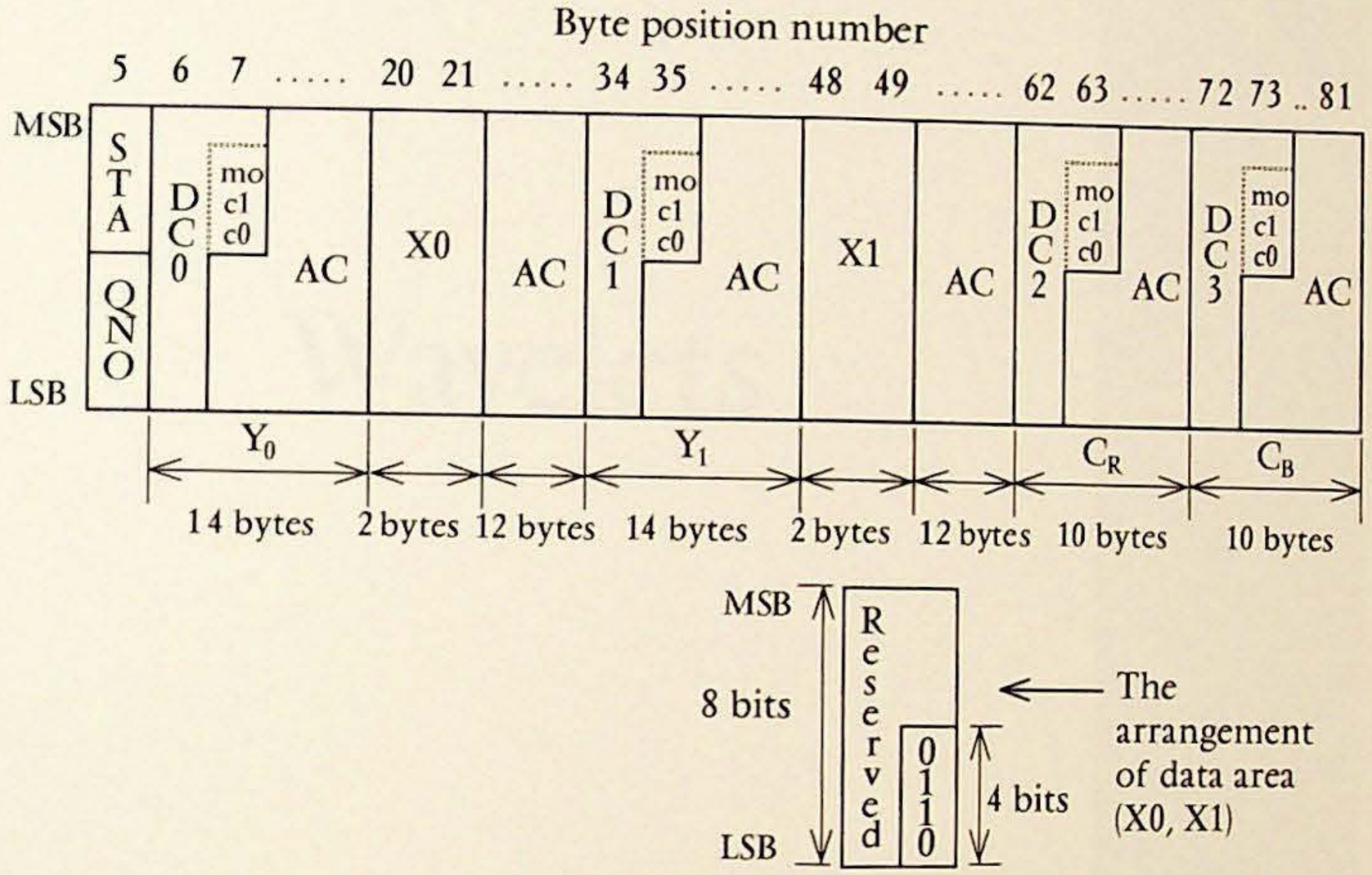


Figure 14-11
Arrangement of a compressed macroblock with 4:2:2 compression.

There are more differences in the detailed syntax of the bitstream, but that is the essence of 4:2:2 50 Mbits/s compression in DV. All the other processes described under the previous section apply.

CHAPTER 15

Wavelets

Introduction

Wavelet technology offers the most viable alternative to DCT for image compression. The techniques were investigated by workers in many different fields, and existed under various names until the connection was spotted by a mathematician in the mid 1980s. Although the favorite child of theorists for many years, until recently practical image compression systems have failed to meet expectations.

The theory of wavelets is quite complex, but the principles involved are not. Like Fourier transforms and DCT, wavelets transform information into a representation using frequency-dependent coefficients, but wavelets differ in that useful positional information is retained. Before we look at wavelets, it will be useful to review some of the qualitative aspects of Fourier transforms.

More about Fourier Transforms

Let's look at the issue of frequency and location information for a moment because it's really important. If we have a series of samples of a signal, say in the horizontal direction, we can look at any sample and know immediately the amplitude information for that particular location in the horizontal direction. What do we know about the horizontal spatial frequency? Nothing! We need the information from many samples to gain much information about the frequency content of the signal, and from all of the samples to get all of the information about frequency content.

Now let's perform a Fourier transform. This gives us a number of values representing the frequency components of the signal. We can get the information pertaining to any given frequency by examining the single coefficient that represents that frequency. However, we have no information about where any amplitude event (such as a step) may occur. To find this we need to assess the contributions of all the frequency components—in fact, perform a inverse Fourier transform.

In general terms, in the spatial domain we have excellent location information but little frequency information; in the frequency domain we have excellent frequency information but little location information. Put like this it sounds very obvious, but it is a fundamental issue that is important to understand. One of the great things

about wavelets is that they can give both frequency and location information.

It will help in understanding the operation of wavelets to examine in a little more detail how the Fourier transform works. The general form of a Fourier series for a periodic function is:

$$A_0 + A_1\sin(2\pi.\, f + \varphi_1) + A_2\sin(2\pi.2\, f + \varphi_2) + A_3\sin(2\pi.3\, f + \varphi_3) + \text{---}$$

where A_0 is the DC component, f is the frequency of periodicity; *2f, 3f,* etc., are the multiples of the first frequency, and φ_1, φ_2, etc. are the phase angles of each of the component frequencies. Remember, as discussed in Chapter 5, Fourier transforms require two coefficients for each frequency; instead of using phase angles directly, we could have used one sine and one cosine term for each frequency, each with an appropriate coefficient. Let's examine how we arrive at these coefficients. Let's build a simple periodic waveform by adding a few Fourier terms:

$$f(x) = 3\sin(2\pi.x) + 5\sin(2\pi.3x) + 4\sin(2\pi.5x)$$

The waveform is shown in Figure 15-1. In this case I have made all the phase angles zero, so we need only worry about one coefficient for each frequency. The DC component is also zero.

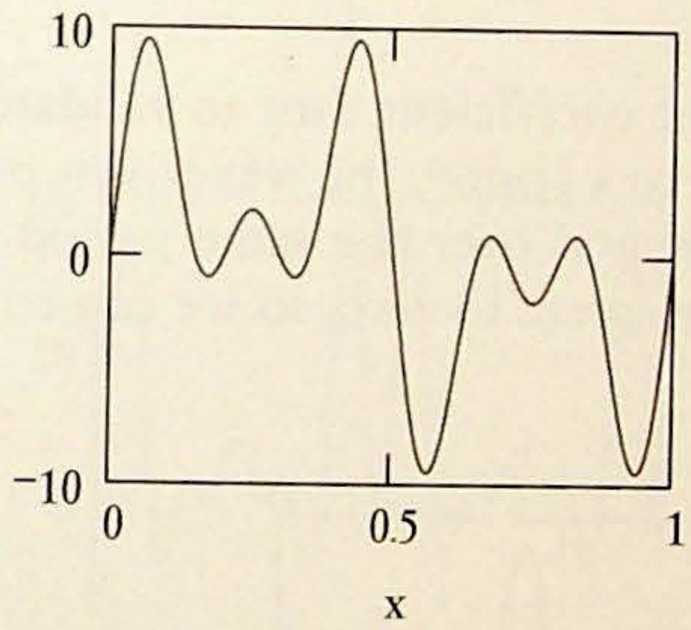


Figure 15-1
A periodic wafveform made from three sine components.

A Fourier coefficient may be calculated by multiplying the waveform by the corresponding Fourier component frequency and integrating the result over the period of the input waveform. (The DC coefficient is obtained by integrating the original waveform itself over the same period.) Normally we would have to use both sine and cosine components to get both coefficients, but in this case sine waves

alone will be sufficient. So, if we multiply our sample waveform by the fundamental frequency (the frequency of periodicity), we have:

$$g(x) = 2.\sin(2\pi.x).[3\sin(2\pi.x) + 5\sin(2\pi.3x) + 4\sin(2\pi.5x)]$$

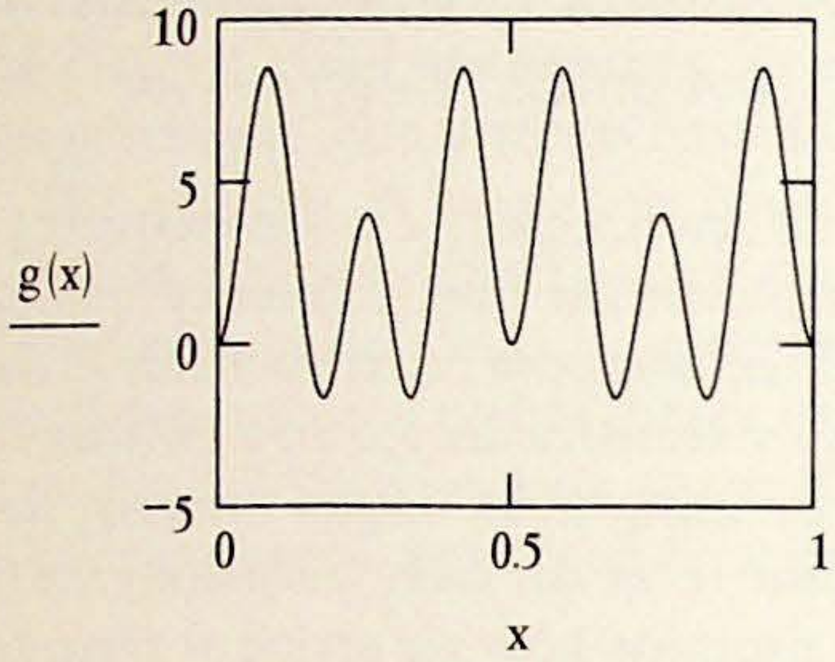


Figure 15-2
The waveform multiplied by a sine wave at the frequency of periodicity. The result is asymmetric, so yields a non-zero coefficient.

This product is shown in Figure 15-2. (There is probably a good mathematical reason for the multiplier "2" but I put it there to make the numbers come out right!) If we integrate this waveform over the full period:

$$\int_0^1 g(x)dx = 3$$

we have our first coefficient. Just to validate the simplification, if we change the sine to a cosine, the waveform product is shown in Figure 15-3, and the integral over the same period is zero. In fact, all cosine products will integrate to zero, so we can continue to look just at sine products.

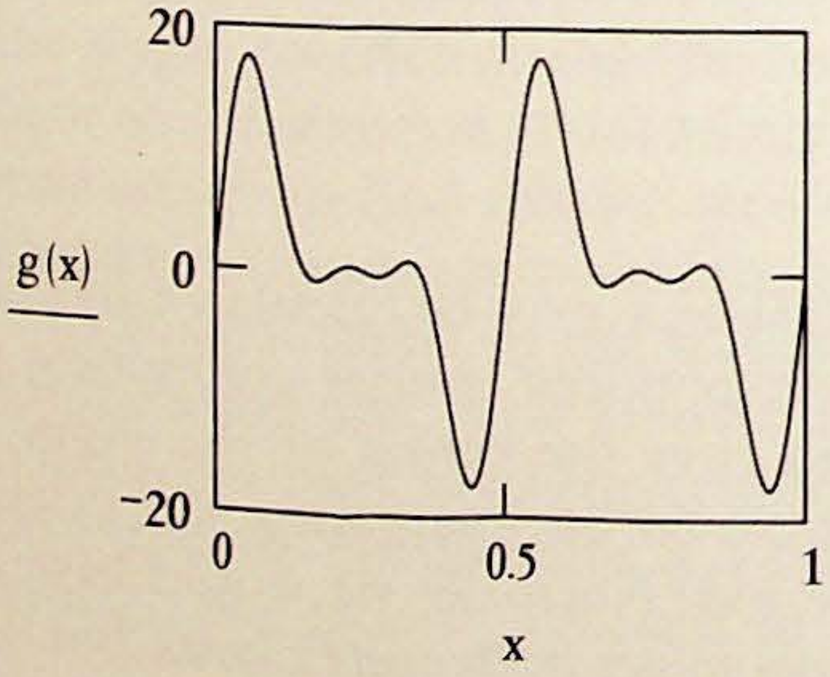


Figure 15-3
In this special case, multiplying by a cosine wave at the same frequency gives a symmetric product that integrates to zero.

The behavior I am trying to demonstrate can be seen if we compare the results using the fourth and fifth harmonics of the fundamental. We expect to see a non-zero coefficient for the fifth harmonic, because that was part of our original waveform definition. There was no fourth harmonic in the original definition, so we expect to see a zero coefficient here. The result of the two multiplications is shown in Figures 15-4 and 15-5. In Figure 15-4 the original waveform is multiplied by the fourth harmonic. Because this frequency does not exist in the original waveform, the product is symmetrical and integrates to zero. In Figure 15-5 the fifth harmonic is used as a multiplier. Because this frequency was present in the original, we get a sine-squared component in the product. This component is always positive, so the product is asymmetrical and integrates to a nonzero result. In fact, the integral evaluates to 4, the coefficient value we would expect.

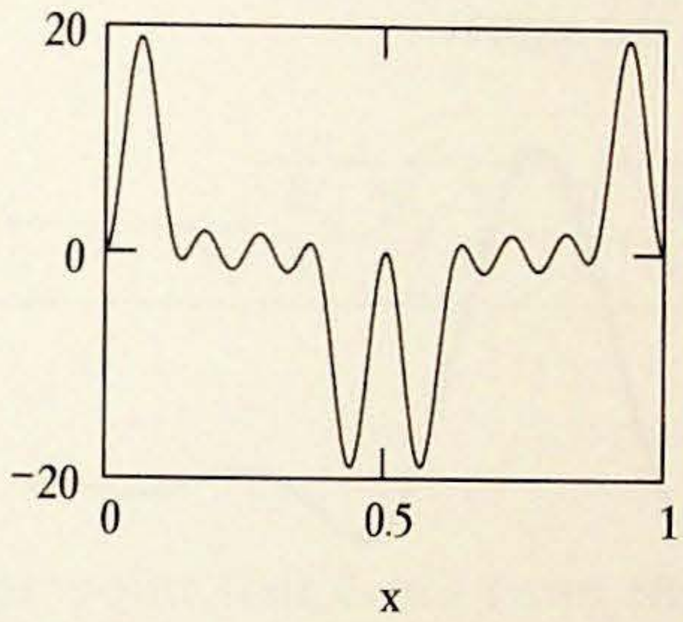


Figure 15-4
The fourth harmonic is not present in the waveform, so the product is symmetric, and we get a zero coefficient.

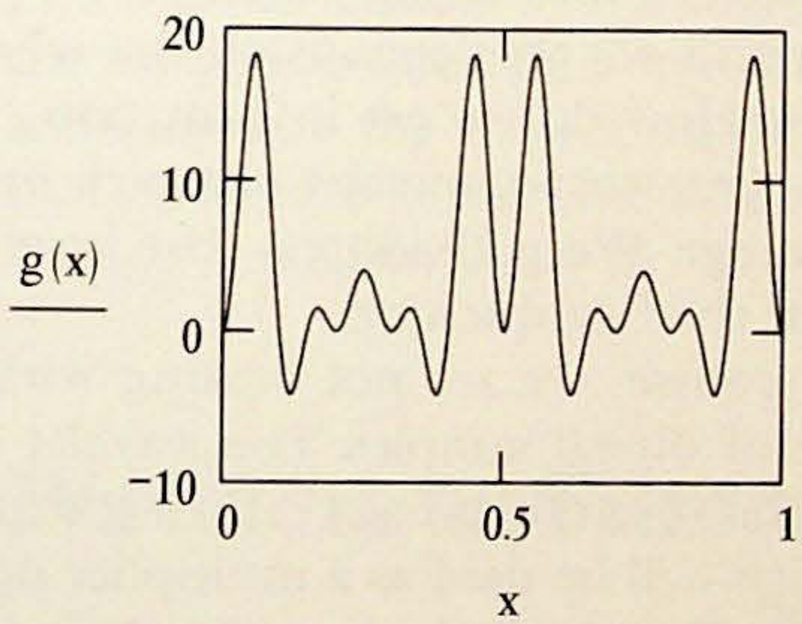


Figure 15-5
The fifth harmonic is present, so the sine-squared products are asymmetric; the integral and the resulting coefficient are non-zero.

The behavior this demonstrates is that multiplying by a sine wave of given frequency "picks out" that frequency in the original waveform by creating a sine-squared product that does not integrate to zero. Any frequency not present in the original does not do this and

(if it is a harmonic of the original's frequency of periodicity) creates a symmetrical product that integrates to zero.

Wavelets Concept

There are many ways of looking at wavelets, but one that fits with the concept we have just been discussing is to view wavelets as a burst of energy with a dominant frequency. A wavelet is shown in Figure 15-6. It is fairly obvious that if we were to multiply a signal by this waveform, the product would contain asymmetric information where the original signal had frequency content similar to that of the wavelet. Just as with the Fourier transform, integrating the product would give us a nonzero coefficient. Multiplication by the wavelet "picks out" detail from the signal.

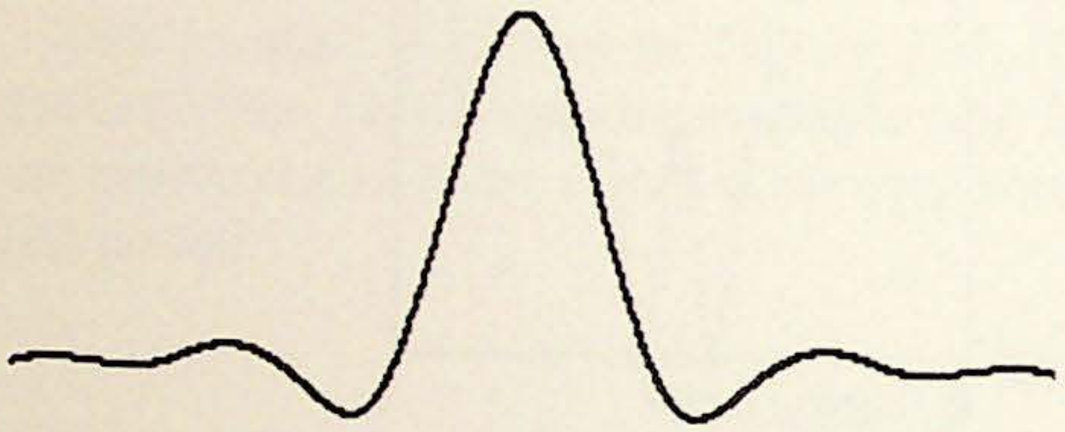


Figure 15-6
A wavelet is a burst of energy.

That is obvious and interesting behavior, but two questions arise. Obviously, the result we get depends on the where on the signal we place the wavelet. How do we get information on the whole signal? Also, it would appear that a wavelet can pick out the detail in only one frequency range. We will address that issue a little later; meanwhile, let's look at the first question.

In reality, of course, we are not dealing with analog signals but with a sequence of digital samples. The wavelet does not really look like Figure 15-6, but more like Figure 15-7—a sequence of sample values, each of which will be used as a multiplier operating on a sample of the waveform. The particular wavelet shown is a sequence of 13 values, 6 either side of the origin. In applying a wavelet transform, this sequence is "walked across" the sample sequence (waveform) being transformed. This process is known as *convolution*, and is shown in Figure 15-8. For simplicity, this figure shows a short wavelet with just five

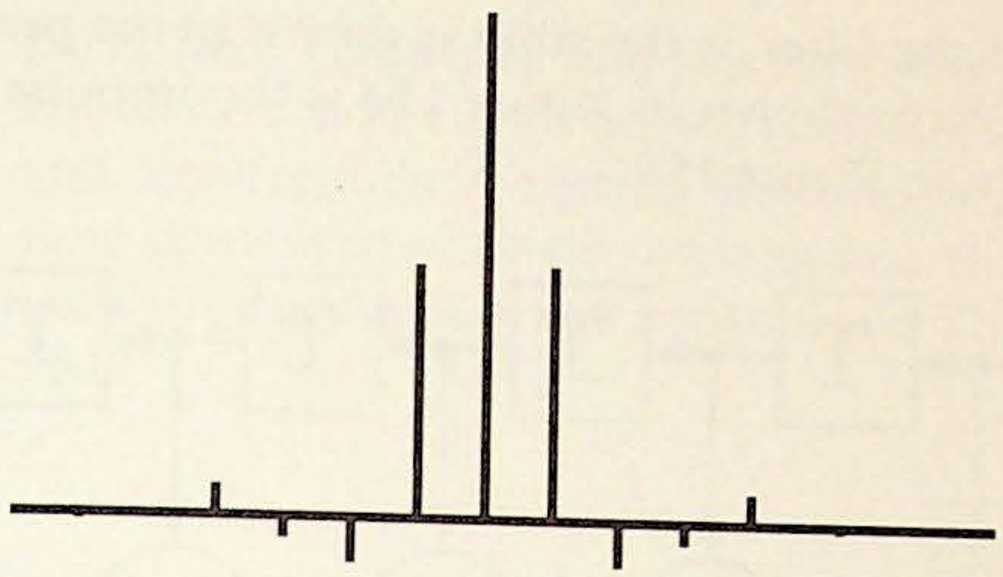


Figure 15-7
The wavelet of Figure 15-6 represented as a sequence of samples.

values, two either side of the origin. The wavelet is moved along, one sample at a time, and the convolution product evaluated for each position. For example, the wavelet is shown operating on sample S_7, and the output of the convolution is a new value S_7', where:

$$S_7' = W_{-2}.S_5 + W_{-1}.S_6 + W_0.S_7 + W_1.S_8 + W_2.S_9$$

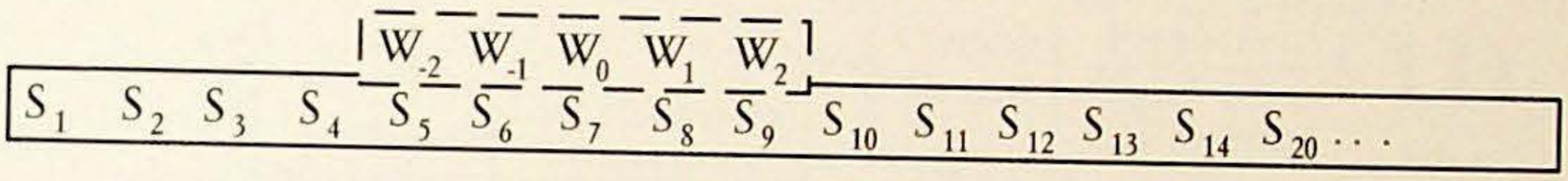


Figure 15-8
Convolving a five-sample wavelet, W, with the samples of a signal, S.

One point that arises from this operation is that to convolve the wavelet over the whole sample set, such as an image, additional samples must be created at the edges. Typically an artificial extension of the image is created by a "reflecting" the edge samples far enough to accommodate the wavelet. In this example, only two additional samples are needed at each end of the waveform. At the left-hand side we create $S_{-1} = S_0$ and $S_{-2} = S_1$.

Wavelets as Filters

Some readers will recognize that the process just described is exactly the same as that shown in the Figure 15-9, which is a classical representation of digital filter. In this diagram, the samples of S are input sequentially to the left-hand side and pass through four delays, T, equal to the sample interval. The output, S_7', when sample S_7 is at the

center of the filter, is the same as shown in the previous equation. The wavelet shape shown in Figure 15-6 is the impulse response of the filter shown in Figure 15-9.

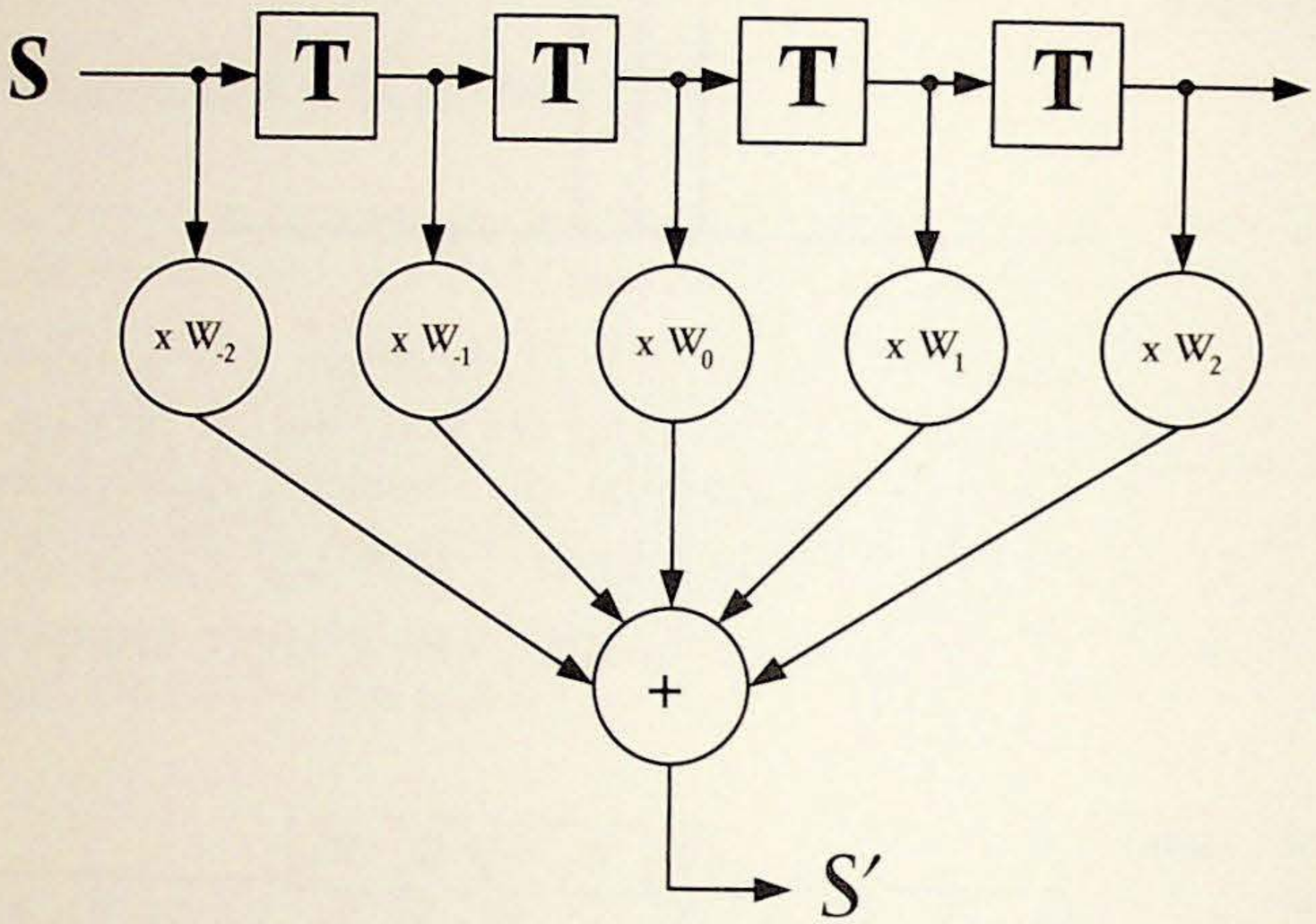


Figure 15-9
The five-sample wavelet shown as a five-tap filter.

The idea of moving a wavelet over the image and picking out detail shows us how wavelets can give both frequency and location information. The concept of wavelets as filters gives us a better idea of how to use them for image compression. It also introduces another point—a wavelet is actually two complementary functions, or two complementary filters. One is known as the wavelet, the other as the *scaling function*. Although there are a great number of complexities in the design and implementation, the concept of a *fractal compressor* is quite simple, and in some ways quite familiar.

The image is filtered by convolving it with the wavelet, which behaves like a high-pass filter and extracts the high-frequency detail of the image. The image is also convolved with the complementary scaling function that removes the high frequencies. So now we have a set of wavelet coefficients representing the fine detail of the image, and an image from which the fine detail has been removed. It is possible to construct two-dimensional wavelets, but the transform is separable, so generally the two-dimensional image is handled by applying the wavelet and scaling functions in both the horizontal and vertical directions. The process is very similar to the *quadrature mirror filter*, described in more detail in Chapter 17. The image is split into two

parts, high frequency and low frequency, operating, say, in the horizontal direction first. The two resulting subimages contain both high- and low-frequency vertical information. Each of the sub images is now convolved with the wavelet and the scaling function vertically, each producing two new separations. The process is shown in Figure 15-10.

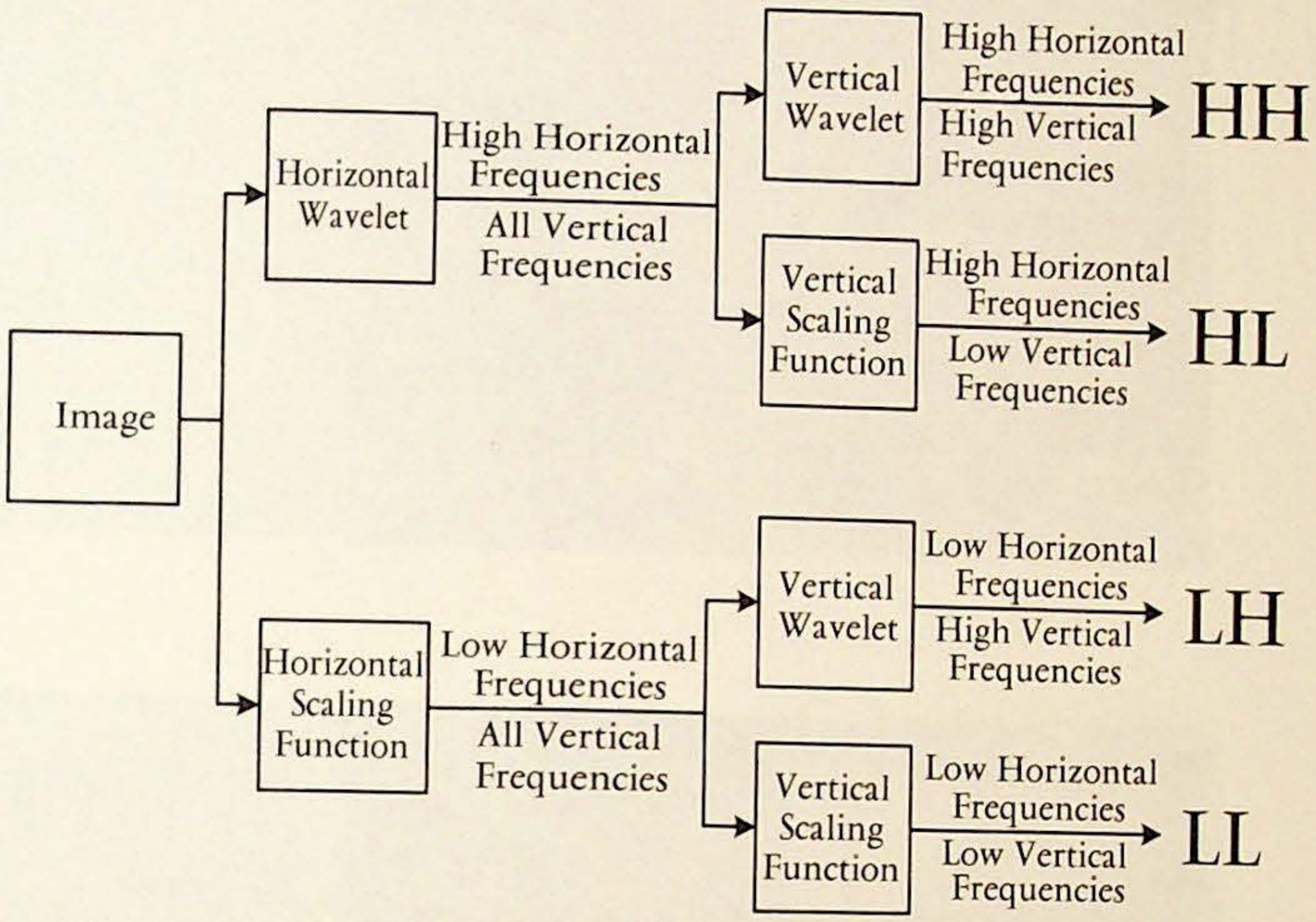


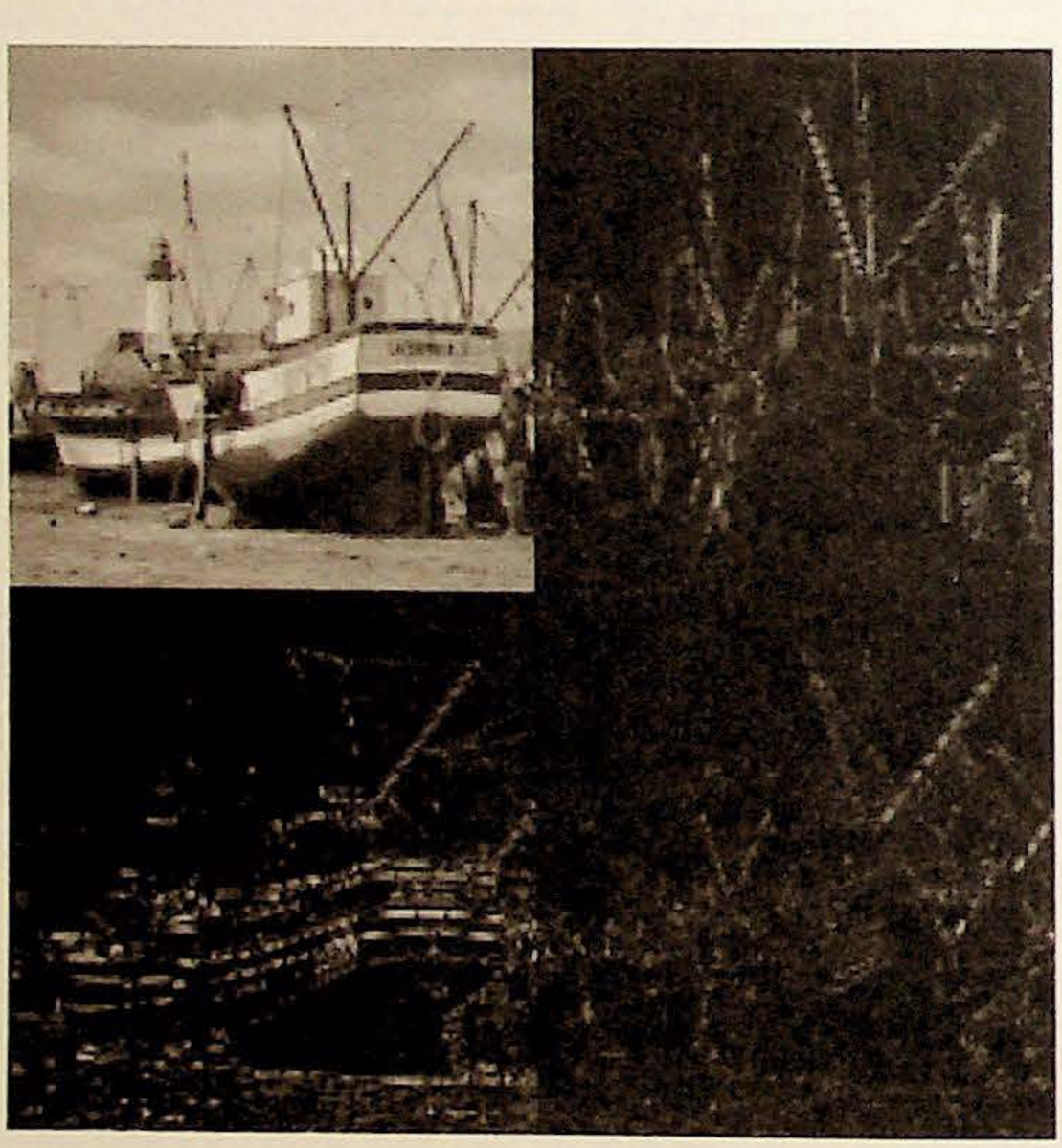
Figure 15-10
Division of the image into four subimages.

After this process, we have four subimages. One contains high horizontal and high vertical frequencies (HH), one has high horizontal and low vertical frequencies (HL); a third has low horizontal and high vertical frequencies (LH). Finally, the fourth has only low frequencies, horizontal and vertical (LL). Because of the bandwidth restrictions, the sampling density may be halved, horizontally and vertically, as explained in Chapter 17. Each subimage has, therefore, just one quarter of the samples or pixels of the original. This is shown in Figure 15-11(b).

It should be noted that the illustrations in Figure 15-11 exaggerate the amplitude of the wavelet coefficients. Each subimage is scaled to display the largest coefficient as white, to make visable as many coefficients as possible. This gives a better idea of the structure of the subimages, but a misleading impression of the energy of the high-frequency coefficients.

(a) The original image, as used elsewhere in this book.

This image is 256 x 256 pixels.

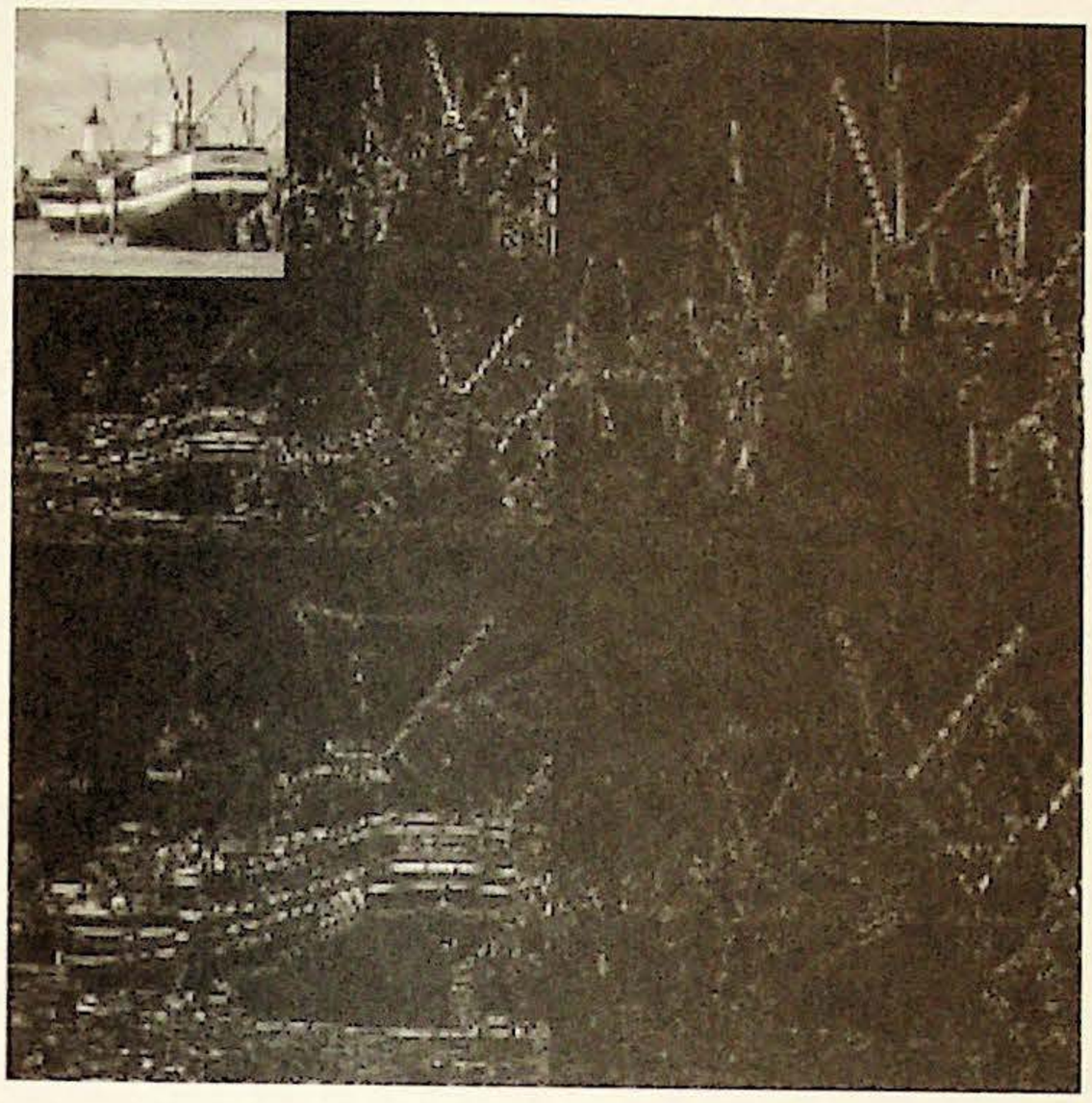


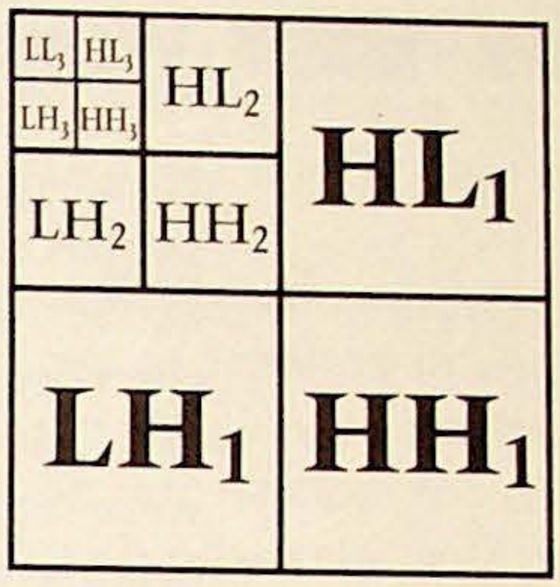
LL	HL
LH	HH

(b) A single pass of the transform, showing the three sets of coefficients, and the residual filtered image.

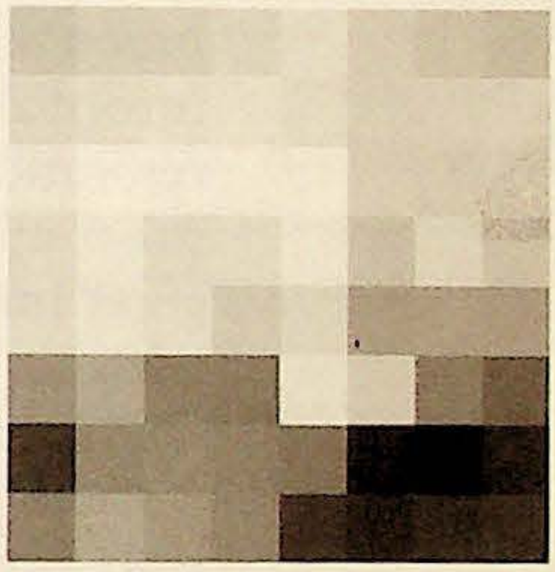
The amplitude of the coefficients is exaggerated by rescaling to make them visible.

Figure 15-11 Wavelet transforms of "Boats." (a) The original image, as used elsewhere in this book, at 256 × 256 pixels. (b) A single pass of the transform, showing the three sets of coefficients, and the residual filtered image. The amplitude of the coefficients is exaggerated by rescaling to make them visible.





(c) This example shows three iterations and the resulting ten subbands.



(d) After five iterations the residual image is reduced to just 64 pixels, shown above.

Figure 15-11 (c) This example shows three iterations and the resulting ten subbands. (d) After five iterations the residual image is reduced to just 64 pixels, shown above.

Wavelet Compression

Now we can start to put a system together. We left a question unanswered earlier—how do we deal with the fact that a wavelet appears to extract only one frequency range? In the previous exercise, we had the three subimages that contained high frequencies (HH, LH, and HL), plus the low-frequency subimage, LL. All of these subimages were reduced to half-sampling density. Now, suppose we convolve the original wavelet with the subimage LL. The wavelet is unchanged, but the image has been subsampled. A frequency that had, say, 8 samples per period in the original image now has 4 samples per period. Thus, the same wavelet captures frequencies at half of those captured in the first pass. Similarly, the unchanged scaling function now eliminates high frequencies down to half the cut-off frequency of the first pass.

Now we can see how the wavelet elegantly spans the void between spatial and frequency representations. Spatial representation gives excellent location information and no frequency information. Frequency representation gives excellent frequency information and no location information. On the first pass a wavelet gives coarse frequency information (the top half of the bandwidth) and fine spatial resolution (half the original sampling density). Each successive pass gives information about a smaller band of frequencies (finer frequency resolution), but with coarser location information, because of the subsampling that occurs on each pass. Eventually we would have good information about the lowest frequencies in the image, but with no positional information because we would have only one sample left!

Clearly we have an iterative process here. On each pass the image is split into four subimages, and on the subsequent pass the same process is applied to the quarter-size LL subimage, generating four sub-subimages! This process is easy to see in Figure 15-11(c), which shows three iterations. Figure 15-11(d) shows five iterations, and LL_5 is now reduced to eight pixels in each direction (from the original 256). We could, in theory, continue until LL had only one pixel (eight iterations with this image), but there comes a time when the overhead of identifying the different subimages exceeds the benefit of additional passes. A typical software compression application would use six passes on this image, leaving just 16 values from LL_6 to be transmitted.

This may seem like an enormous computational task. Certainly the convolution process is quite expensive, but we do not get the full

impact on every pass because the image is smaller each time. If the first pass uses *N* processor cycles, the next uses *N/4* cycles, the next *N/16*, and so on. For the six iterations discussed above, only about *4N/3* cycles are used in total.

Wavelet-based compression is similar to DCT-based compression in yet another way. Nothing we have done yet as resulted in compression! However, the wavelet coefficients of an image have similar characteristics to the DCT coefficients. Many are zero or close to zero, and the remainder may be coarsely quantized with relatively little perceptual impact. We saw that the human psychovisual system permitted coarse quantization particularly of the high-frequency coefficients. The first pass of wavelet compression yields the largest number of coefficients (because the image is at its largest), and these are all high-frequency coefficients.

As with DCT-based compression, the next step is to organize the remaining coefficients to maximize the benefit of lossless compression techniques. In the discussion of JPEG we saw that the zigzag scanning "collected" the most significant coefficients early in the scan, and maximized the runs of zeros. These subimages resulting from the wavelet transform require more complex operations. One such technique is called *zerotree coding* and uses *significance maps*. This algorithm assumes that if a coefficient is insignificant in a low-resolution subimage, then the corresponding coefficients in the higher resolution subimages will also be insignificant. Further details are beyond the scope of this book, but the bibliography contains a wealth of references for those who wish to delve deeper.

So, wavelet-based compression and DCT-based compression are similar in many ways. Figure 15-12 shows a DCT-based compression system; Figure 15-13 shows a wavelet-based system. The only fundamental difference is the transform and, as we have seen, both DCT and wavelets are closely linked to Fourier transforms.

Wavelet compression does require more computational power than DCT-based compression. But, according to its proponents, it offers a number of substantial advantages.

- Wavelet coding is inherently scalable. The transform process, shown in Figure 15-10, is repeated for as many iterations as required. The decoder performs the inverse process, *but may stop at anytime* if the full resolution of the original is not required. A bitstream may carry the full resolution, but a decoder may use a subset of the bitstream, depending on its capabilities and the resolution of the display being used.

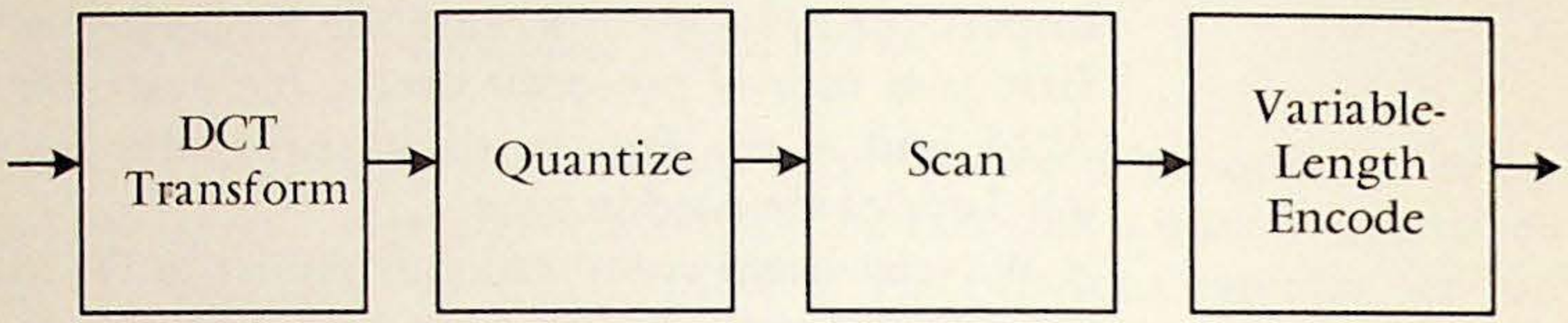


Figure 15-12
DCT-based compression.

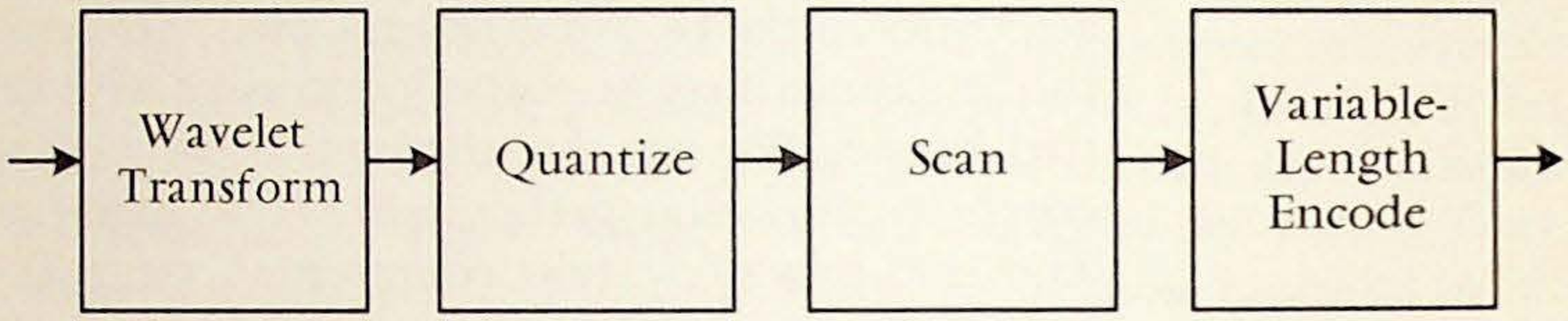


Figure 15-13
Wavelet-based compression.

- Wavelet compression is claimed to be more efficient at low bit rates. Probably the most significant element here is the fact that wavelet-generated artifacts are generally less objectionable than DCT-generated artifacts. When DCT is used at low bit rates, the excessive quantization tends to result in level differences between adjacent DCT blocks. These straight-line correlated errors are precisely the artifacts to which the human psychovisual system is most sensitive. Excessive quantization of wavelet coefficients leads to "smearing" of detail, but there is nothing in the system that will generate straight lines.

It is difficult to show a useful comparison between the two systems on the printed page, but Figure 15-14 serves to illustrate the qualitative differences. It shows the image "Boats" (originally 256 x 256 pixels, 64 kBytes) compressed by JPEG and by a wavelet-based compression system. In each case the compression ratio is about 17: 1, for a file size of just under 4 kBytes. The CD-ROM include some tools that allow experimenters to make their own comparisons.

At the time of writing, wavelet compression has made very little impact compared to DCT-based compression. There are number of reasons for this. Wavelets have been less successful than DCT-based systems in achieving good efficiency at the near-transparent compression ratios. Also, once DCT was adopted by MPEG, most development effort went into producing integrated circuits for MPEG— that is, DCT. Until recently, little specialist silicon was available for wavelet compression. However, this is changing, and now that wavelet compression has been adopted in MPEG-4 (for static textures) and in JPEG2000, wavelet implementations are likely to become much more common.

Figure 15-14
Compression of "Boats" at 17:1 by JPEG (above) and wavelets (below).

CHAPTER 16

JPEG2000

Introduction

We discussed JPEG back in Chapter 7, and indeed both historically and as a source of the technology used in other systems, that is its logical place. The various iterations of MPEG, the development of DV, and the evolution of wavelet compression all followed JPEG. But the JPEG committee has not been idle. The JPEG2000 standard is now at the stage of Final Committee Draft and is expected to become an International Standard during 2001. JPEG2000, like MPEG, is a very rich standard, and this short chapter can offer only a brief summary and references to additional information.

Limitations of the Original JPEG System

The original JPEG was and is a powerful system. It provided the groundwork for the dominant compression systems in use today. When used in accordance with its design criteria it provides excellent results. Nevertheless, as we have seen in the course of this book, it does have its limitations. JPEG2000 set out to address all weaknesses of JPEG, so a good starting point is to summarize these weaknesses.

- JPEG is designed for continuous-tone images and performs badly on imagery with different characteristics.
- Specifically, JPEG is unsuitable for much computer-generated imagery, binary (black/white) images such as text, and compound documents with different types of content.
- JPEG has many modes, of which a large proportion are application specific. There is no universal decoder architecture.
- At low bit rates, JPEG artifacts, such as blocking, are very visible and restrict the usefulness of the standard.
- JPEG is very susceptible to transmission errors; typically even a small error rate results in substantial image degradation.
- JPEG has no convenient mechanism for handling very large images.
- JPEG is inflexible by today's standards. It cannot support lossy and lossless compression in the same bitstream, and there is no ability to encode the important parts of an image to a higher quality than the rest.

Goals of JPEG2000

All these factors were taken into account in specifying the requirements for the new JPEG2000 standard. These include:

- Backward compatibility with JPEG; a JPEG2000 decoder should be capable of decoding a JPEG bitstream.
- Superior low bit-rate performance; initial tests show 20 to 50 percent reduction in bit rate for comparable quality, depending on image content.
- Ability to process large and/or high-precision images. JPEG2000 permits image height and width up to about 4 billion pixels. There can be up to 255 components in each image (e.g., *Y, U, V,* alpha, etc.) and each component may have any depth from 1 to 32 bits.
- Continuous-tone and bilevel compression.
- Lossless and lossy compression. JPEG2000 provides wavelets like those discussed in the previous chapter, plus wavelets with integer coefficients to permit lossless transforms. It also provides a special form of color difference coding using integer coefficients to move between *RGB* and *YUV,* permitting perfect reversibility.
- Progressive transmission, providing the gradual buildup of an image by resolution (spatial scalability) and by pixel accuracy (signal-to-noise scalability). Also, the ability to extract a lower-resolution image from the bitstream by a less-powerful decoder.
- Random access to the bitstream (start decoding at any point).
- Robustness in the presence of bit errors.
- Region of interest (ROI) coding and decoding, permitting higher quality for the most important parts of the image. Regions can be defined in a manner similar to shape coding in MPEG-4.
- Content-based description and image security, including encryption capability, indelible copyright information, and the like.
- Provision of an *alpha* or transparency channel.

JPEG 2000 is based on wavelet compression and offers a wide range of lossy and lossless filter combinations. At the time of writing most available software is experimental, but some excellent results may be demonstrated. Figure 16-1 shows a comparison between the conventional JPEG implementation of a popular graphics program, and an experimental JPEG2000 package. The compression is about 60:1 in each case (the original file is 64kB, the JPEG files is 1,361 bytes, and the JPEG2000 file is 1,305 bytes.

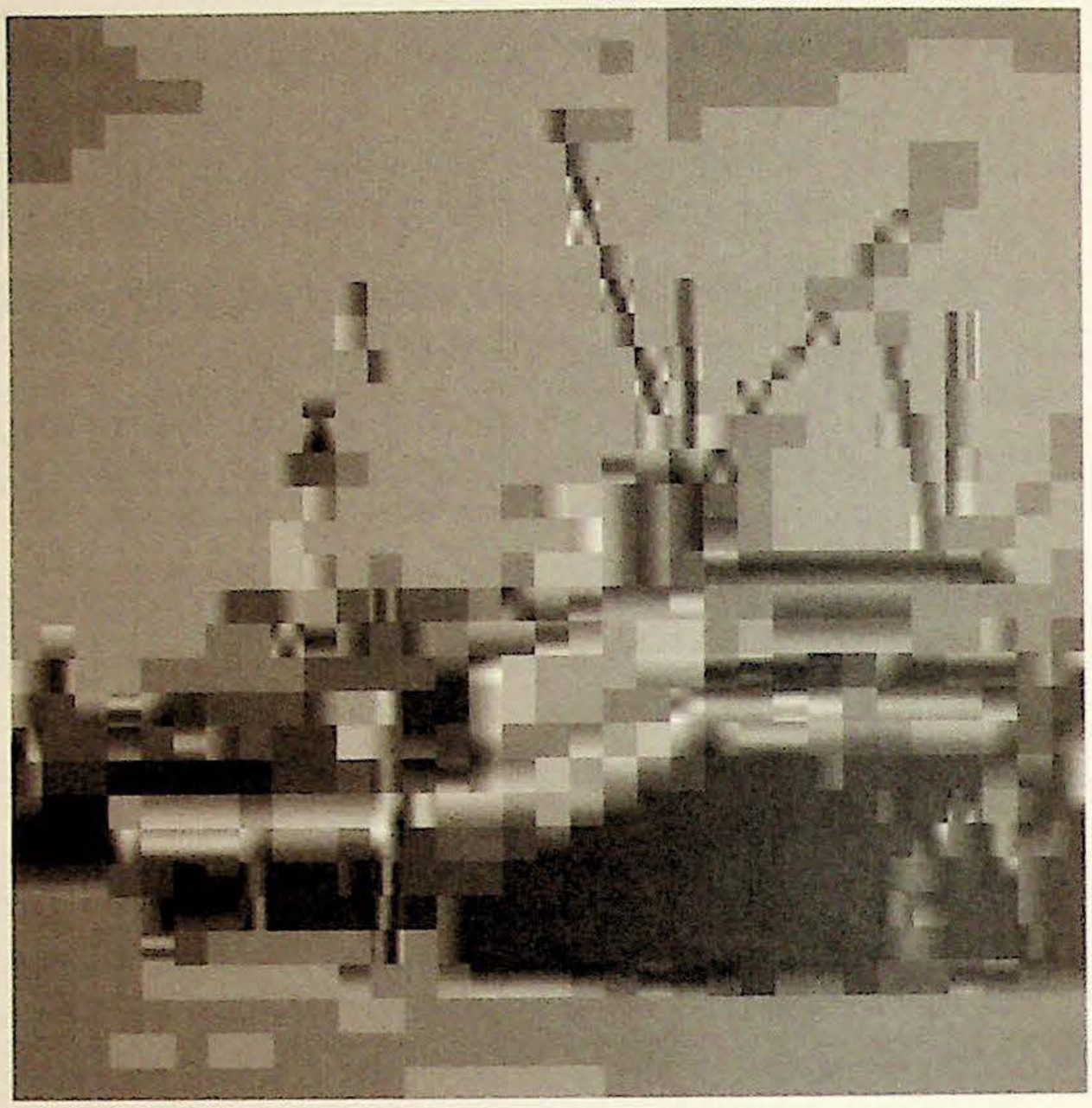
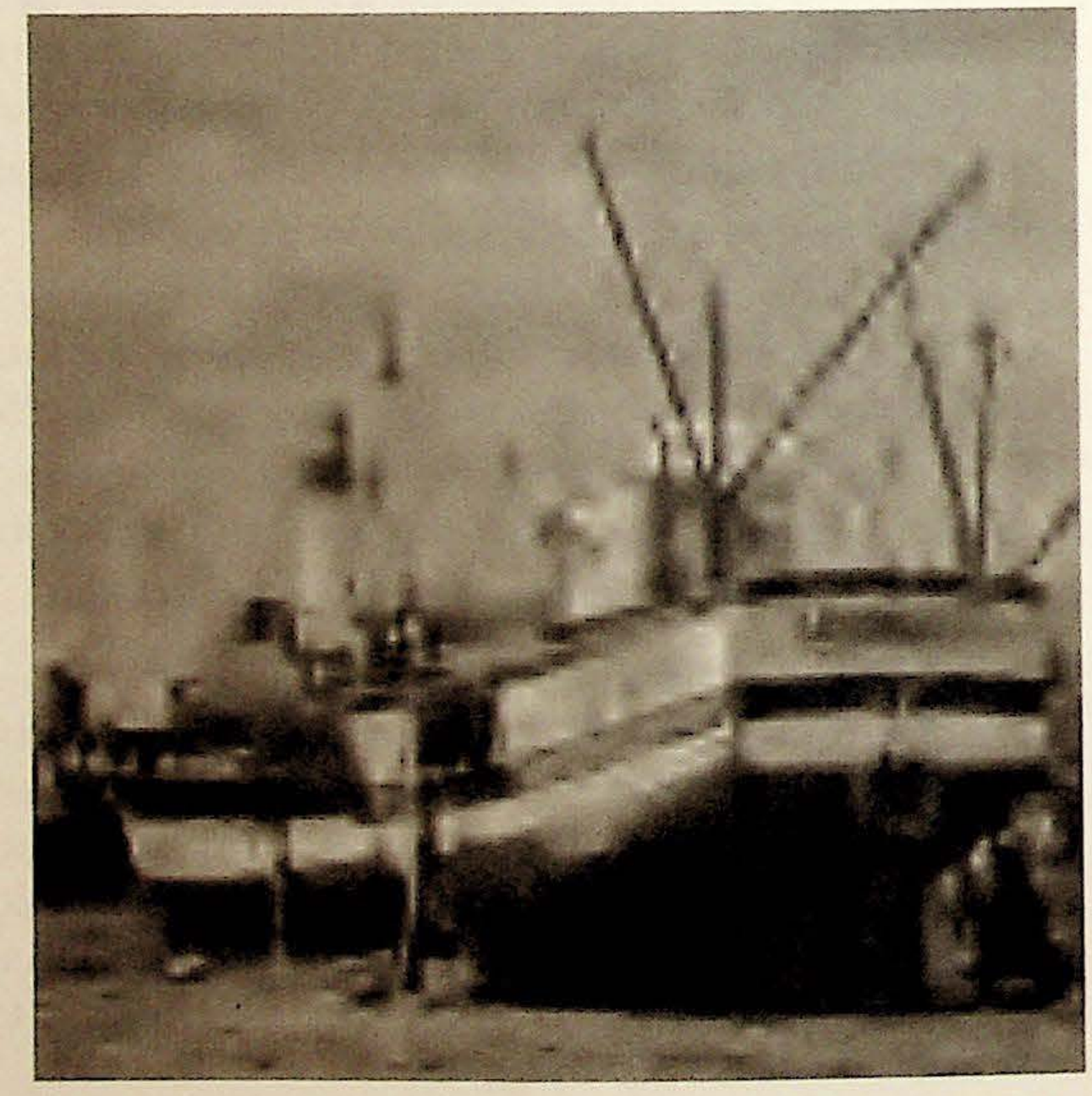


Figure 16-1
JPEG (above) and JPEG2000 (below) compression of "Boats" by about 60:1.



CHAPTER 17

Audio Compression

Introduction

This will be a short chapter, partly because of limited knowledge on my part, but also because audio compression is built on many of the same principles as video compression. Only in one sense is audio compression radically different from video compression—it is much older and has been commercially exploited in the analog domain. It was not called *compression;* it was known as *noise reduction,* but essentially these are the same. A noise reducer modifies the signal such that when reverse processing is applied after transmission or recording, the noise of the transmission or recording channel is attenuated. The well known Dolby A, B, and C systems are actually simple, analog, subband companding systems—terms we encounter below.

Nevertheless, our interest is on compressing digital audio. Depending on the quality requirements, audio may be sampled at a variety of frequencies, usually in the range 8 to 48 kHz, and with sample resolutions from 8 to 24 bits. The lower limits provide telephone-grade audio; the higher can match or exceed human hearing capabilities. Of most interest in the professional world are two standards at the high end. Compact disc audio is sampled at 44.1 kHz, with a word length of 16 bits for each of two channels, a total of just over 1.4 Mb/s. Professional studio audio is usually sampled at 48 kHz with a word length of 20 bits, almost 1 Mb/s per channel. This rate can accommodate audio frequencies to over 20 kHz with a dynamic range of 120 dB.

As with any form of compression, the objective is to reduce the data by eliminating redundancy. There is "absolute" redundancy—data can be removed by an encoder then unambiguously recreated by a decoder. Removal of this redundancy is lossless compression. There is also perceptual redundancy—data can be removed without significant change to the experience of a human observer.

Perceptual redundancy itself falls into two categories. There are phenomena to which the observer is intrinsically insensitive. In video we saw that the higher-frequency transform coefficients could be coarsely quantized without significant impact on the viewer. This is true for all conditions. Another effect, used in bit allocation in video systems, is masking. An artifact that might be visible and objectionable in an image area with little activity (empty sky, for example), becomes invisible or insignificant in a busy area of an image.

All these effects exist in audio, and all are exploited by compression systems, but masking is by far the most significant contributor to data

savings in a single-channel audio system. (Multiple channel systems also benefit from correlation between the various channels.) To understand the mechanisms of masking in the human hearing process we must examine the workings of the inner ear.

Masking in Human Hearing

The human hearing system has remarkable performance. Its useful range covers about 10 octaves and, in the more sensitive regions, it provides a dynamic range of more than 100 dB. The discriminating power of the ear—brain combination is incredible; we can resolve conversation and other intelligence from noise levels and interfering signals to a degree that seems to defy the laws of physics. This performance requires some very sophisticated processing.

One of the key elements of the ear is the basilar membrane of the inner ear. It separates two of the fluid-filled chambers of the cochlea and is in contact with the hairs of the organ of Corti. These hairs drive the actual sensors that send nerve messages to the brain. The basilar membrane varies in width, thickness, and rigidity along its length. This makes it frequency sensitive; different areas vibrate at different frequencies. The basilar is not just a passive membrane; it has an active mechanism that can provide positive feedback to low-amplitude vibrations. The combination of these characteristics leads to interesting behavior. (This explanation is somewhat simplistic. Various authorities have suggested different mechanisms for the behavior of the inner ear, particularly in respect to the active attributes. Some suggest nonlinearity in the cochlea; others attribute the active element to the organ of Corti. Suffice it to say that the behavior is complex, certainly not linear, and that there are active elements within the inner ear. We concentrate on the effects of this complexity.)

Some authorities estimate that there are 24 regions in the basilar membrane; others suggest a higher number, but it is certainly finite. Each region can vibrate over a small range of frequencies, but at only one frequency at a time within that range. When positive feedback is applied, the effect is that of a very high-Q tuned circuit. Each region vibrates at a frequency determined by the strongest stimulus within its range, *and is unaffected by any smaller stimuli.* This means that within each frequency band, only the loudest frequency contributes to the signal received by the brain, and so to what we hear.

This effect is the single most significant contributor to our ability to compress audio.

The effect is called *frequency masking*, and it contributes to our ability to compress in two distinct ways. As mentioned above, only the strongest stimulus within a region is significant. Any frequency in the signal in the same band, but lower in amplitude, need not be coded. Also, noise within the band is irrelevant, provided it is sufficiently below the prime stimulus. This allows encoding of the prime stimulus with a relatively small number of bits; the resulting quantization noise is masked just like any other noise. This effect is quite remarkable and, thanks to Dolby Laboratories, there is a demonstration on the CD-ROM that is well worth trying.

The positive feedback of the basilar contributes to another effect, again similar to that of a high-Q tuned circuit. The vibration responds slowly to changes in the amplitude of the stimulus, and this results in *temporal masking*—we fail to hear sounds a short time before, and a longer time after, a strong stimulus.

Simple Audio Compression Schemes

Many of today's consumer devices use some form of audio compression. Sometimes this is used to increase storage capacity. One example is found in 8-mm video recorders; compression is employed to reduce the tape area needed for audio. Audio workstations become much more cost effective if the capacity of the hard drives can be increased by using compression, provided the necessary high quality can be maintained.

In some applications, the compression is used for more than one purpose. The mini compact disc achieves its recording density by means of compression, but data reduction is also used to improve replay under adverse conditions. Data is read from the mini CD at the same speed as from a regular CD—about twice as fast as necessary for normal replay. The data enters a buffer, and after the buffer delay (typically a few seconds) it is decompressed and presented to the listener. If the listener is an energetic jogger and jars the player enough to move the read head, the player can continue to play from the buffer while the head is repositioned, and the additional data speed allows the buffer to be filled again, ready for the next "jog."

Simple schemes such as these generally use some form of *companding*. The number of bits required by each word is reduced by dynamically altering the level or gain according to the instantaneous amplitude, or the amplitude range of a group of samples.

Video 8, for example, uses a simple four-step gain control based on instantaneous signal amplitude. Low amplitudes are passed at unity gain; higher amplitudes are gain-reduced by factors of 2, 4, or 8. The result looks to a video engineer like a gamma curve (see Figure 17-1). The higher levels are coarsely quantized, but the increased quantization noise is masked by the high signal level. (In the case of Video 8, this mild reduction in data rate is supplemented by an analog compressor.)

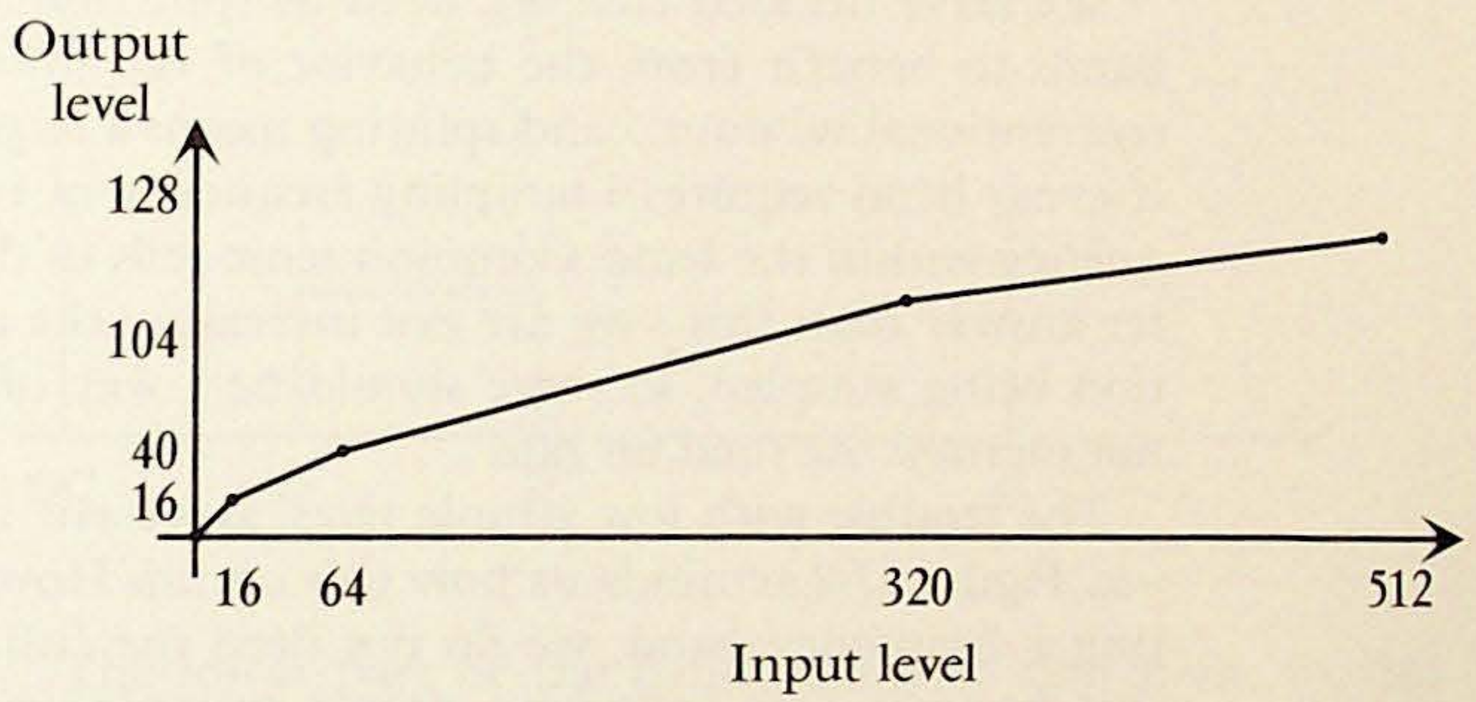


Figure 17-1
Transfer function for Video 8 digital compression.

More sophisticated companding systems usually divide the input data stream into frames, typically between 1 and 32 ms long. In a 16-bit system the values for each sample can range from −32,768 to +32,767. However, it is rare for the sound in one frame to occupy the full range of sample values. If it does not, each sample can be expressed by a shorter word, combined with a gain or scaling factor applied to the complete frame. In other words, the levels are expressed in floating-point form, but with the restriction that the mantissa must have the same value throughout a frame. The NICAM systems, as used in stereo television, use this technique to reduce 14-bit words to 10-bit words, plus a 3-bit mantissa for each frame of 32 words.

There are two real problems with such systems. A frame with a single high sample value can cause low-level audio to be coarsely quantized. This is not usually audible if the frame length is short, typically not more than 1 millisecond. The greater problem is that they do not provide the large compression ratios we would like to see for many applications.

Quadrature Mirror Filters

To make efficient use of the masking effects of the human hearing system, it is necessary to split the audio band into regions as small as or smaller than the regions of the basilar membrane's response. Most modern audio compression schemes are based upon some system that divides the audio band in this way. We will examine subband systems, where the division is performed with filters, and transform systems, which use techniques more like those we have seen in video systems. Before we explore subband systems, we must learn about an important trick, the *quadrature mirror filter* (QMF).

We have decided that we need to split the audio spectrum into bands to benefit from the behavior of the inner ear. However, by conventional wisdom, band splitting means a large increase in bit rate if every band requires a sampling frequency of twice the highest frequency within the band. Common sense tells us there should be a better answer than this—we are not increasing the amount of information being sampled, so there should be a way of sampling that does not increase the total bit rate.

The trouble with low sample rates, as we saw in Chapter 2, is aliasing. Figure 17-2 reminds us how this occurs. However, if we are splitting a frequency band, we do not need the full spectrum shown in this drawing. Let's consider a simple example where we want to split the band into two. For ease of nomenclature we'll call the full spectrum 0 to $2f_a$, which we want to split into two bands, 0 to f_a and f_a to $2f_a$. To sample the whole spectrum, we need a sampling frequency of twice $2f_a$, or $4f_a$ (Figure 17-3). Now let's split the band as shown in Figure 17-4. We now have two half-band signals, each sampled at the $4f_a$ rate, so that we have twice as much sampled data as we started with.

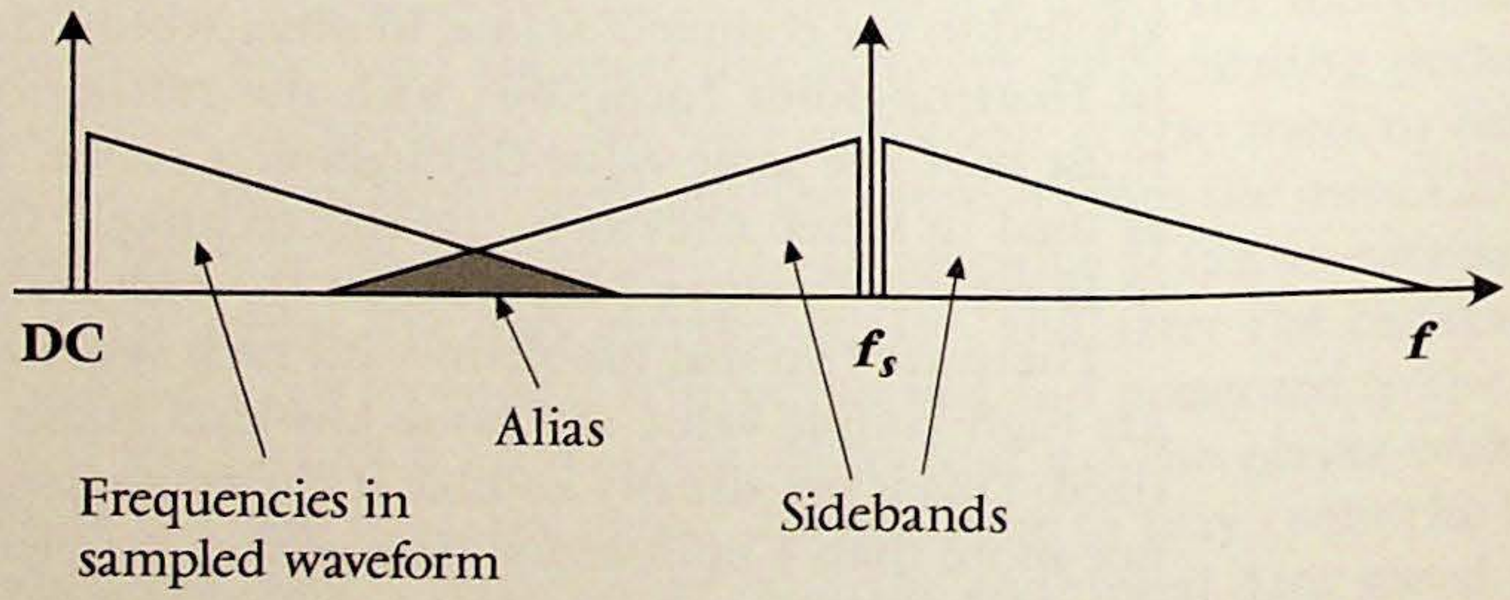


Figure 17-2
Frequencies above half the sampling frequency cause baseband and sideband overlap, resulting in aliasing.

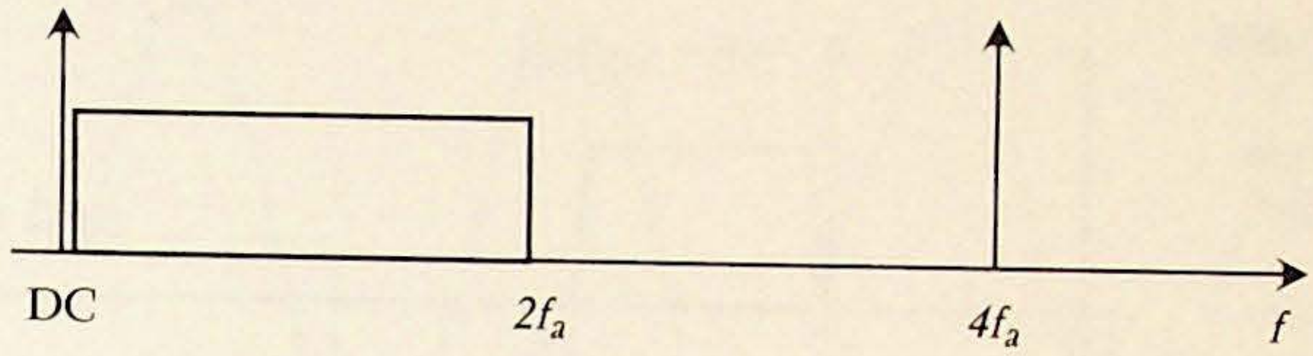


Figure 17-3
Input range sampled at twice highest frequency.

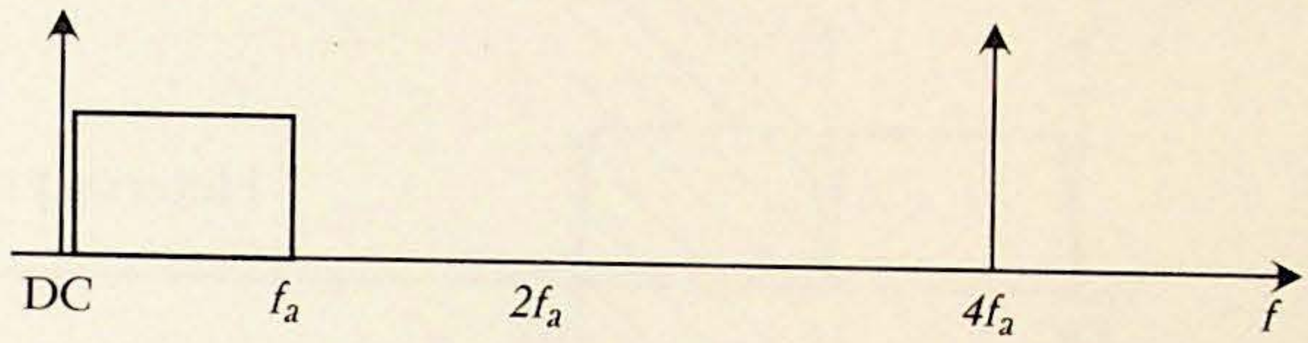
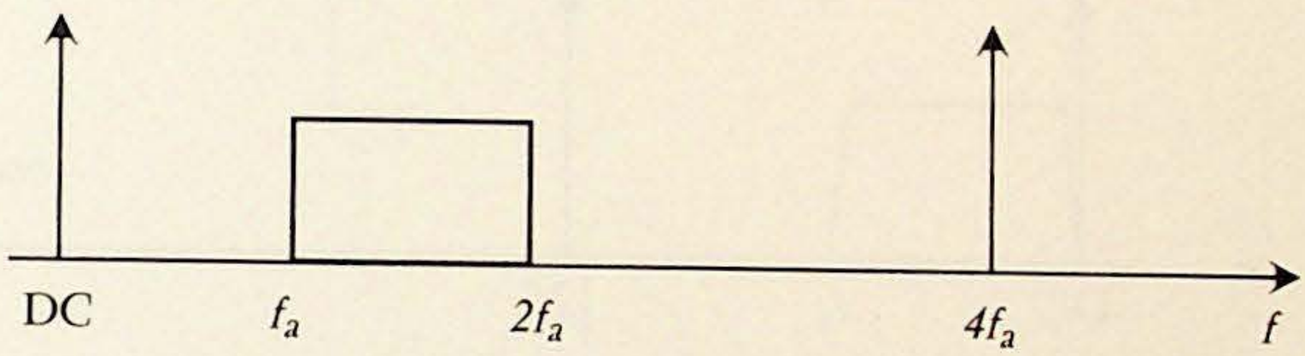


Figure 17-4
Input range bisected.



The lower part of the band is easy to deal with; we can discard every other sample, reducing the sampling frequency to $2f_a$ without risk of aliasing. What can we do about the other half-band? Thinking back to analog techniques, we could heterodyne the upper half-band down to the same frequency range as the lower band by mixing with a suitable frequency and filtering the result. We could then sample the resultant at $2f_a$, and after reconstruction heterodyne back to the original frequency range (Figure 17-5).

In fact there is a much simpler mechanism. Again, we'll repeat a drawing from Chapter 2, showing how the signal and its alias are indistinguishable because they have identical samples (Figure 17-6). But if we eliminate the lower half of the band, *only* the "alias" gets sampled, so there is nothing to confuse with it! All we need to do is use a bandpass reconstruction filter *that will generate only frequencies in the upper half-band,* and we have an unambiguous situation again. So, in fact, we can again discard alternate samples and the upper half-band also has an effective sampling frequency of $2f_a$ (Figure 17-7). We have succeeded in splitting the band while maintaining the same total number of samples.

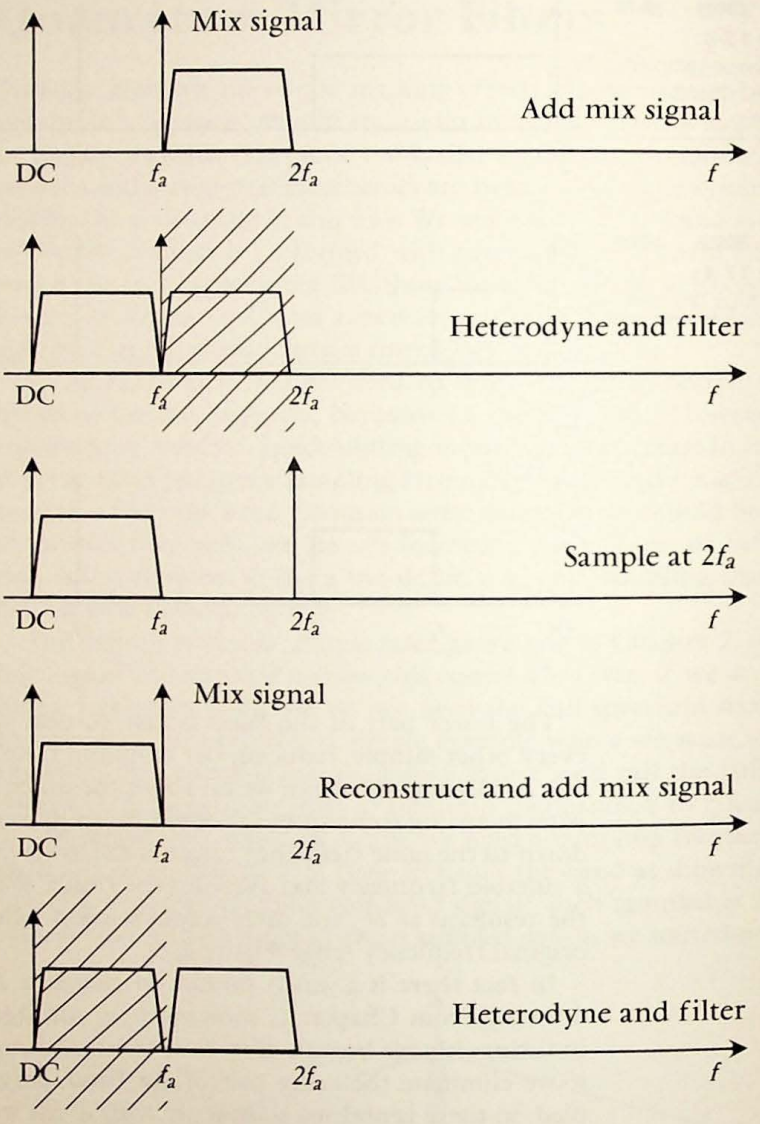


Figure 17.5
Analog approach to sampling the upper half band at $2f_a$ instead of at $4f_a$ (for illustration only).

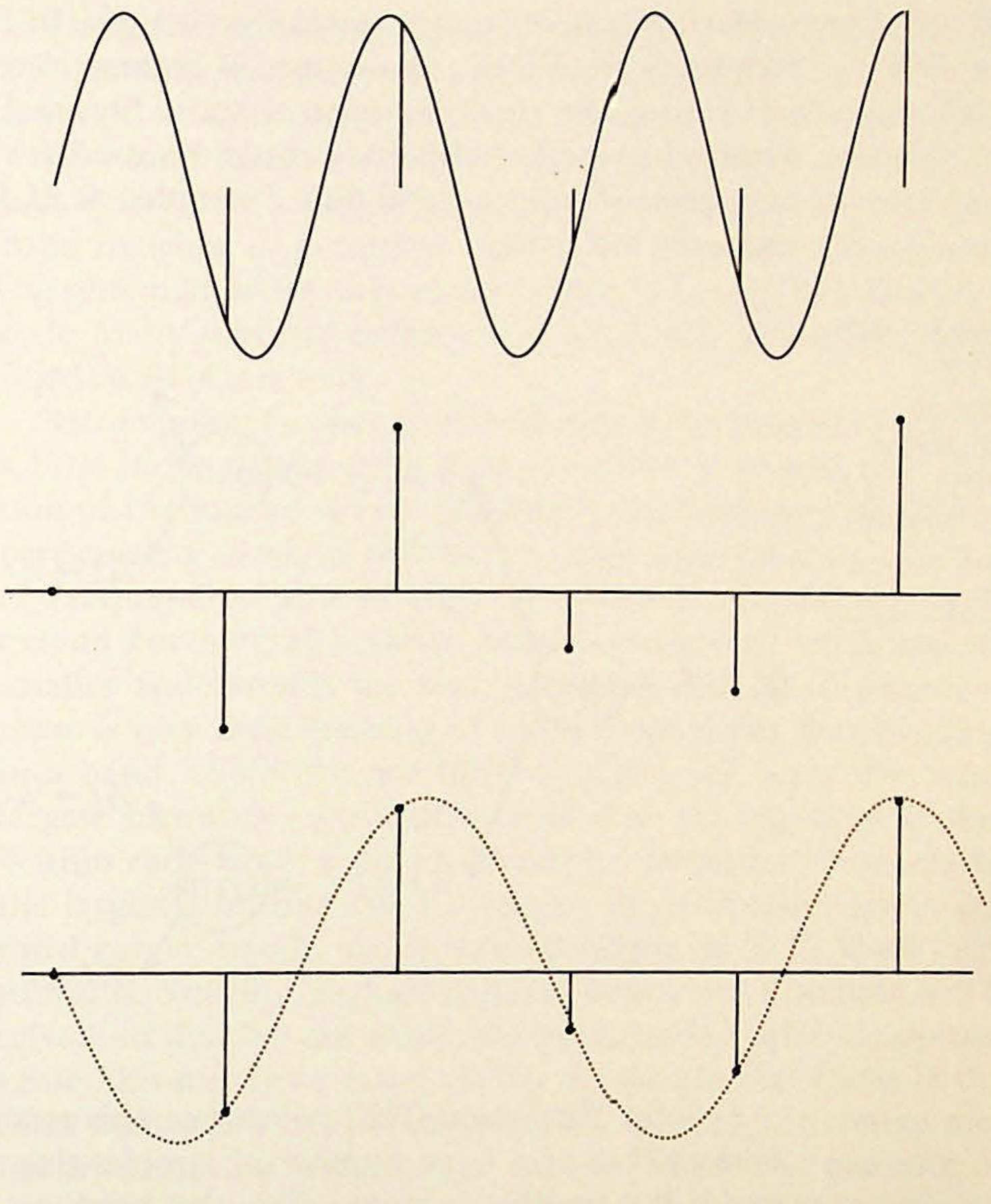


Figure 17-6
The same set of samples can represent either signal. Normally the lower frequency is in the valid (Nyquist sampled) band, but if this band is not present the samples unambiguously represent the higher frequency.

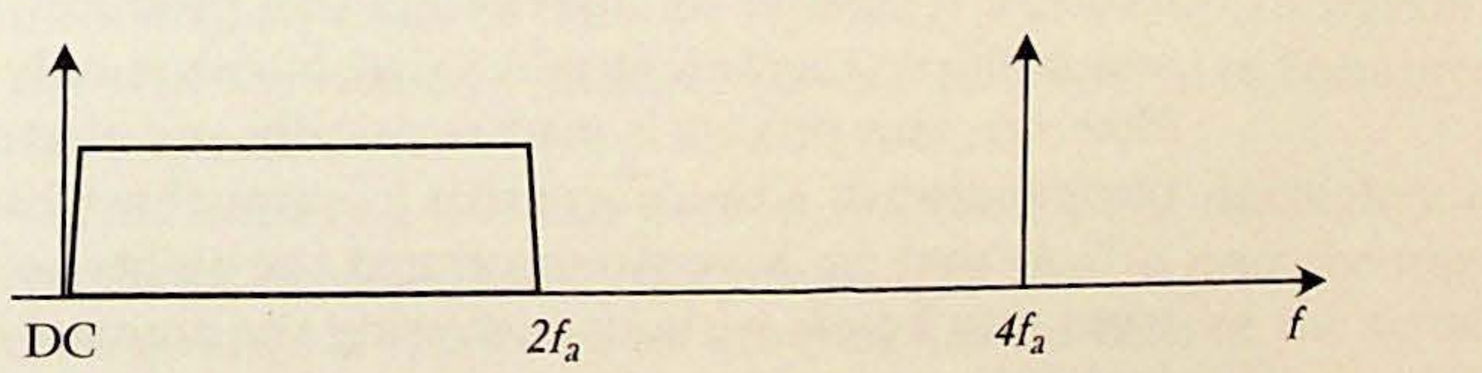
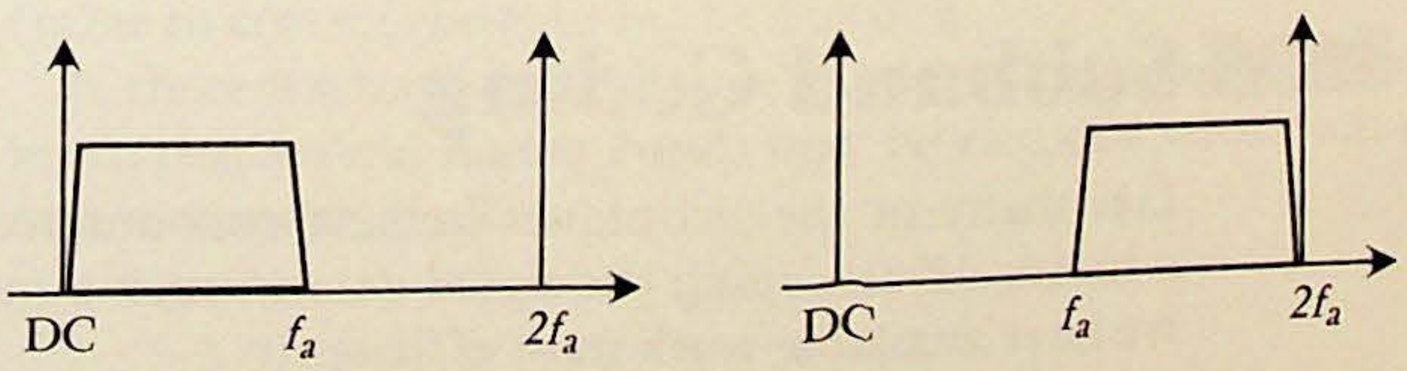


Figure 17-7
The band is split without increasing the total data.



Having done this once, we can do it again. In fact, we can split the frequency band into any number of binary-related subbands without increasing the total amount of data. Figure 17-8 shows a nested arrangement of QMFs that could be used for this purpose. This arrangement splits a band 0 to *f*, sampled at 2*f*, into eight subbands, each sampled at *f/4*.

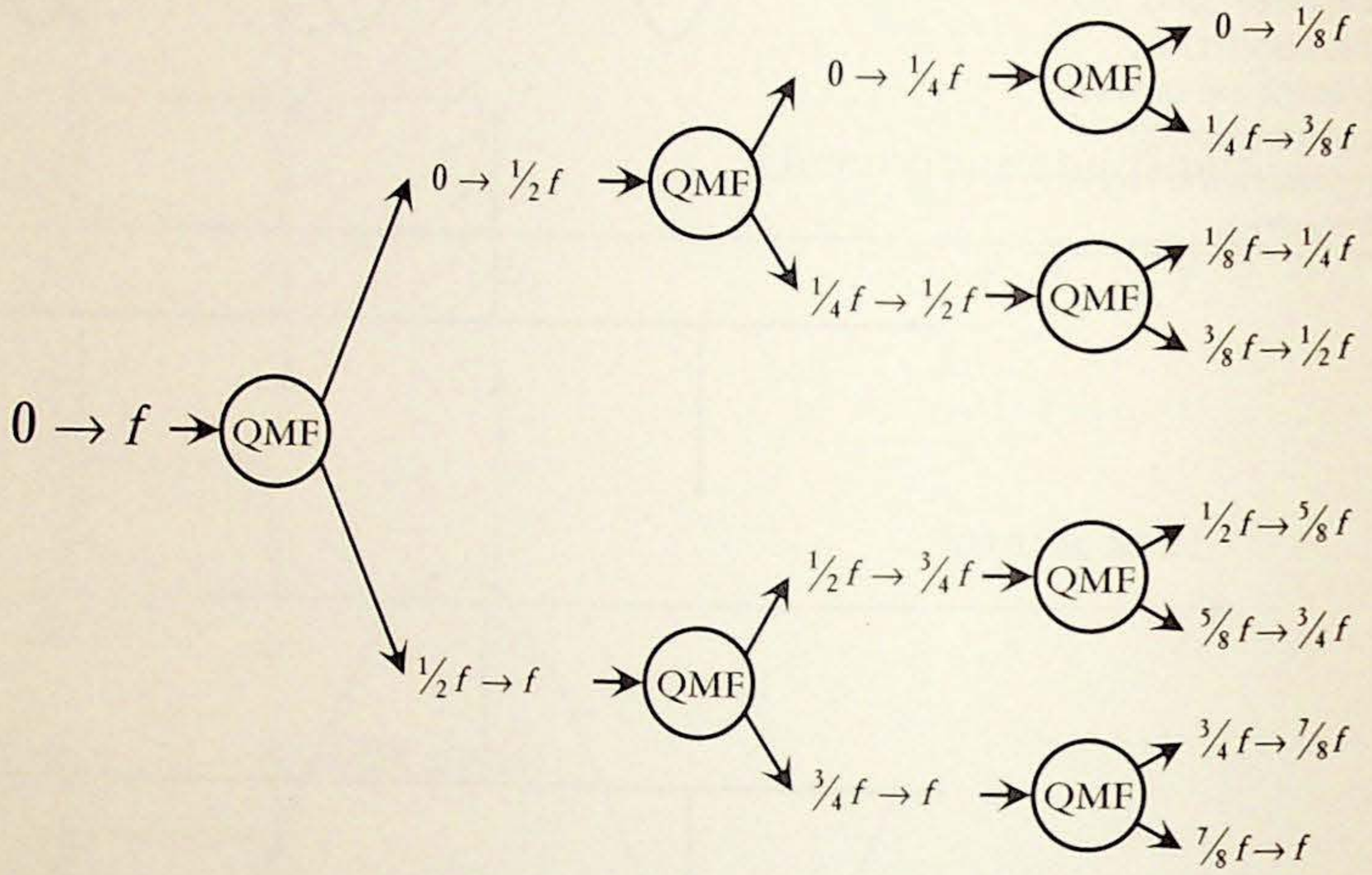


Figure 17-8
Using a tree of quadrature mirror filters (QMF) to split a frequency band into eight equal subbands.

As John Watkinson (1995) points out, this arrangement calls for the computation of a large number of samples that are subsequently discarded. It is possible to build a filter that produces, say, 32 subbands and uses multiple coefficient tables that calculate only the needed samples. However, this process is mathematically and electrically so close to the computation of a transform that it is arguable what it should be called.

Now that we have demonstrated the ability to split the audio spectrum into bands without increasing the amount of data, we can now look at how to reduce the data in each band.

Subband Coding

Like many of the techniques we have encountered, subband coding is a means of arranging the signal data in such a manner that the tools we have available work most efficiently.

The principal tool for audio compression is companding. In simple companding systems we rely on the total signal lying within a limited range. If we split the audio into subbands it is much more likely that, within each band, we will encounter a limited range—in fact some bands may have no signal at all. Furthermore, we have seen that if the band matches, or is smaller than, a critical band of the human hearing system, masking may mean that there is very little that we need to code. Many subband coders use 32 subbands, each approximately one-third of an octave wide.

Before going further in this direction, we need to look at masking a little more closely. Like most functions associated with the operation of the human senses, masking is nonlinear and asymmetrical. In particular, a stimulus will have a substantial masking effect at higher frequencies, and at high stimulus amplitudes this effect can extend for several octaves. At lower stimulus levels the effect is smaller and extends for less than an octave. At all stimulus levels, there is very little masking of lower frequencies. When coding within a band, therefore, we must consider the worst case where the largest stimulus within that band is at the top edge of the band. Within each band, a threshold can be determined from the level of the largest stimulus. For reasonably high stimulus levels, and one-third octave bands, the threshold might be 20 to 30 dB below the stimulus. Nothing, including noise, below this threshold will be perceived, so that we can allow the quantization noise to approach this value; this means we can code the samples in that frame in that subband with 4 or 5 bits per sample! This is enough to convey the essential information within that narrow band. The presence of that information ensures that we will not hear the quantization noise, nor will we miss any smaller signals that might have been present in the original.

This is really a two-step process. On the basis of the highest level in the frame, a scale factor that moves this level to near the top of the coding range is applied to all samples. On the basis of the actual level, the masking threshold is determined, and the modified samples are truncated to this number of bits. The scale factor is sent with the frame to correct the gain in the decoder.

If there is a large signal in one band, the masking effect of this may mean that several higher bands may be encoded with even fewer bits, or not encoded at all.

Bit Allocation

All the above assumes we will always operate on the edge of audible errors. Rapid change of signal amplitude within a frame, not to mention the variability of human ears, means that a practical system should strive for some headroom above the theoretical threshold in each subband. Compromises may have to be made, because a practical system may be required to deliver a constant bit rate, irrespective of the input signal. The compression system needs a controller that allocates bits to the different subbands, based upon the needs of each band, the overall availability of bits, and its psychoacoustic model of human hearing. This controller may operate by analyzing the data in the subbands, or it may use a Fourier (or similar) transform of the input signal that can give a more detailed analysis of the input.

Whatever decisions are made in allocating bits must also be known at the decoder. There are three possible strategies.

In a *forward adaptive* system, the coder performs all calculations, and the allocation is encoded and sent to the decoder. This strategy has an advantage in that the encoder algorithms may be updated without affecting decoder operation, but some part of the bandwidth must be used to send the bit allocations to the decoder.

Backward adaptive systems perform the same calculations in the encoder and decoder. The decoder will always reach the same conclusions as the encoder, so that no allocation data needs to be sent. However, the decoder cost is significantly higher, and the encoder cannot be changed.

A compromise is found in *forward/backward adaptive* systems. Complex calculations are performed at the encoder, and a very small transmission bandwidth is used to send a few key parameters to the decoder, which then need only perform simple calculations. As in the backward adaptive approach, the encoder cannot be significantly changed, but some parameters can be changed to adjust behavior.

Transform Coding

In studying subband coding, we see that a number of factors point to the advantage of narrow subbands. (Remember, we have shown that we can split the input signal into any number of bands without increasing the data.)

- A typical audio spectrum is made up of many discrete frequencies. With wide subbands, most or all subbands will contain one or more components, and so have to be encoded. With narrower bands, more subbands will fall within gaps in the spectrum and contain no components. These bands do not have to be encoded at all.
- The number of bits needed to encode a subband depends on the degree of masking. We must always consider the worst case, where the stimulus is at the top edge of the band, because of the steep slope of the masking curve below the stimulus frequency. For this reason, narrow bands mask much higher noise levels and can be encoded with fewer bits.

Band-splitting filters are an efficient means of obtaining a reasonable number of subbands, such as the 32 used by many subband encoders. Beyond this, however, use of filters becomes very cumbersome, and a frequency transform is used to analyze the signal. Very large numbers of subbands may be obtained by this method. A 256-subband transform filter has approximately the same complexity as a conventional filter for 32 subbands.

There is a drawback to transform filters. Filters with high-frequency resolution have poor temporal resolution, and this affects the coding of frames that include transients. If the full coding gain is taken, the resulting quantization noise is present throughout the frame. A transient in the frame will probably mask this noise after the transient, but there is very little "negative" temporal masking; thus, the noise may be heard at the beginning of the frame.

To counter this effect, some systems use transient detection and switch to a lower number of bands for frames that include a transient. A correctly designed transform filter bank may be switched quite easily without any significant increase in complexity.

Example Compression Systems

Audio Compression in MPEG

MPEG has standardized on compression schemes based on the MUSICAM system developed by Philips. Three coding layers are defined for MPEG-1, but Layer II is by far the most used.

Layer II supports audio sampled at 32 kHz, 44.1 kHz, and 48 kHz. It uses 24-ms frames (with 48-kHz sampling) and codes 32 equal subbands. For each subband a 6-bit scale factor is used, giving a range of 120 dB. The scale factor can operate over the complete frame, but can be changed on 8-ms boundaries if required.

This system uses forward adaptive bit allocation and fractional bit quantization. It can code mono or stereo at bit rates of 32 to 384 kb/s. Layer II has been adopted for several direct broadcast satellite (DBS) systems and by the European Digital Video Broadcasting (DVB) system. It is also the standard for multimedia applications such as CD-ROMs.

MPEG-2 added some flexibility to MPEG-1 audio. Half-sample rates were added (16 kHz, 22.05 kHz, and 24 kHz) and provision was made for a multichannel extension. This approach provides backward compatibility with MPEG-1. If multichannel sound is required, a stereo mix must be derived and transmitted as Layer II audio. The additional information is sent in an enhancement stream. Multichannel decoders accept both streams, and remultiplex for the required number of channels. Simple decoders receive only the stereo bit stream and ignore the enhancement layer. Total bit rates of up to 1 Mbit/s are permitted for highest-quality multichannel.

The most well-known audio compression system is, of course, MP3—the designation for MPEG Audio Layer III. This uses more sophisticated algorithms to achieve greater compression efficiency, and is claimed to offer CD quality at about 64 kbits/s per channel. The availability of many thousands of MP3 compressed files on the Internet has done more to change the way we think about entertainment content than any other single development. It is far from certain how the copyright issues will be solved, and who will win the battle for control of digital entertainment distribution, but it is certain that the new techniques are here to stay.

Audio Compression for ATSC

During the process leading to the U.S. ATSC standard, it was decided to use MPEG-2 (MP@HL) for video coding. However, no agreement could be reached on the audio compression system to be used.

One proponent offered a system where six channels were individually compressed, but the main contenders were Philips with an enhanced MUSICAM system and Dolby with its AC-3 system. Both systems offered "5.1" channels. This means five full-bandwidth channels for

front left, front center, front right, left surround, and right surround. An additional channel has only narrow bandwidth and is intended to drive subwoofers, chair-shakers, and other very low-frequency devices. It is also known as the *low-frequency effects* (LFE) channel.

In a bitterly fought contest, expert listeners compared the two systems under carefully controlled studio conditions. AC-3 was selected, on the basis of its slightly superior performance on certain vibraphone passages. Philips contested the decision on the grounds that the MUSICAM system had not been performing correctly at the tests, but the decision remained. In fairness it should be said that when either system is working optimally, expert listeners cannot distinguish the results from the original recording, although Dolby claims this performance level at a lower bit rate. AC-3 supports bit rates from 32 to 640 kbits/s; the ATSC tests were performed at the intended transmission rate of 384 kbits/s (the maximum permissible rate has now been increased to 448 kbits/s, the same maximum as specified for DVDs). This represents a compression ratio of about 10:1, making the quality a remarkable achievement.

AC-3 is a transform system, switching between 256 subbands for normal operation and 128 subbands for transients. It supports the usual sampling rates of 32 kHz, 44.1 kHz, and 48 kHz, and uses 32-ms framing at 48 kHz. It uses a 4.5-bit scale factor over a range of 144 dB and forward/backward adaptive bit allocation.

Dolby incorporated a number of interesting features in AC-3, to be known as Dolby Digital in the commercial world. Although the system can encode up to 5.1 channels, the decoder can produce whatever downmix is appropriate for the receiver capabilities. Decoders can downmix to mono or to stereo or, of course, support full 5.1-channel output. In addition, each Dolby Digital (DD) decoder incorporates a Dolby Surround Pro Logic (DSPL) decoder to support the most common surround-sound installations. In fact, Dolby Digital includes a specific pass-through mode for DSPL encoded audio. The two DSPL signals are carried on the left and right front channels of the 5.1 system and passed directly to the DSPL decoder.

It is important to recognize the existence of this mode. Many videotapes, particularly film-tape transfers, have DSPL audio. DSPL is an analog compression scheme that carries center and surround information coded into the left and right channels; it is a four- to two- to four-channel system. It might be assumed that a DSPL signal could be decoded to four channels and fed to a DD system, perhaps feeding the surround signal to both left and right rear channels. This would

work if there were five loudspeakers at the receiver, but there is a hidden trap.

DSPL is an analog system where the matrixing can never be perfect, so the system is designed with this in mind. Specifically, the system is designed to avoid sensitivity to phase variations in high frequencies in the surround channel. Also, DSPL is not a true surround-sound system in that it does not attempt to place sounds accurately in a 360° stage. It provides accurate location in front of the listener, plus a general (and very effective) ambiance from behind. As a part of this process the DSPL decoder low-pass—filters and delays the derived ambiance signal. Provided the four signals are then conveyed transparently to four loudspeakers, all will be well.

However, if the DSPL signal is decoded and fed to a DD system and decoded at a receiver with only (two-channel) stereo capability, the DD decoder downmixes for the stereo output. If the signals it mixes include the filtered and delayed surround from a DSPL decoder, the downmix will be quite wrong and objectionable.

To avoid this problem, DD provides the DSPL mode described above. The two DSPL-encoded channels are fed to the left and right front channels of the DD system, and the "message" that this is DSPL is passed with the bit stream to the decoder, which passes the two channels directly to the incorporated DSPL decoder. No downmix is required for stereo, and the two DSPL channels can be combined for mono.

Dolby Digital also addresses the perennial problem of loudness variation between programs and between channels. The system uses dialog level to normalize receiver volume; the bit stream carries an indication of the dialog level and the decoder gain is varied to keep this essentially constant.

Finally, DD allows for adjustable dynamic range at the decoder. Wide dynamic range is perfect for home theater environments, but quite unsuitable for a portable television in a noisy kitchen. The decoder can reduce dynamic range as determined by the receiver manufacturer—or according to the preferences of the listener if user controls are provided—with separate control of compression above and below the nominal dialog level.

Some issues surrounding the use of DD for digital television have not yet been resolved. One is the metadata (descriptive data) that should accompany the audio to the encoder. The encoder must know if its input is DSPL, so as to invoke the pass-through mode. It also needs to know the nominal dialog level and the dynamic range setting in order to convey these to the receiver. This presents a practical issue

in a television plant where no obvious mechanism exists for routing, recording, and distributing the metadata with the audio.

Initially most potential users assumed that DD would be used for network distribution as well as final transmission to the home. However, fades, voiceovers, and the like. cannot be performed on the coded signal, and such operations in a local TV station require decoding and re-encoding. As with any compression system, this results in quality loss. It has been suggested that DD coded at a higher rate than that used for transmission could be used for distribution to provide the necessary quality headroom. However, it is possible that a different coding system, Dolby-E, may be preferable for distribution; this is discussed briefly in Chapter 19 under Mezzanine Compression Systems. As with many aspects of digital television, implementation is very slow, and new decisions are being made all the time.

CHAPTER 18

Streaming Media

Introduction

Streaming is a technique used to deliver multimedia content, often including video and audio, usually compressed. It is used on networks, including private intranets and the Internet. Streaming brings together characteristics of two fundamentally different delivery systems.

In the world of digital studio installations, both video and audio, connections between pieces of equipment are usually made by dedicated circuits. These circuits carry the bitstream, usually uncompressed, in real time with no significant transit delay. They are deterministic and isochronous—bits go in to the circuit at a fixed rate and bits come out at the same fixed rate. For every bit that goes in, one comes out. The data is received by hardware elements designed specifically for the communication circuit being used, and every usable element, be it an audio sample or a video pixel, is processed as it arrives.

In this scenario there is really no concept of a beginning or end; the bitstream may last for a second, or for many years. The transmitting device sends the information it has; the receiving device accepts the information as it receives it. Even when there is no real information to send—if the video is black or the audio silent—as long as the equipment is powered, it will faithfully send the codes for black or silence, and they will be diligently received.

In the world of computers, the usual method of transferring information is a file transfer. One application creates a file and causes it to be stored on some medium. Another application may access the file from its original storage location, or some separate mechanism may cause the file to be transferred to a different storage location before the second application accesses it. The transfer mechanism may move a physical medium like a floppy disk ("sneakernet"), but today it is more likely that the file is transferred over some form of network. The file may be transferred quickly or slowly, perhaps in many parts, and the parts may not arrive in the same order as they were sent. It is the job of the transfer mechanism to ensure that when the job is finished, the received file is the same as the original.

Whatever the mechanism, the process is essentially serial. The file is created, then moved, then accessed. If there is no information to send, no file is created. No access to a file is possible until the transfer is complete.

This is the mechanism used, for example, when downloading an MP3 file from a music distributor. The file is transferred over the Internet from the server to the user's hard drive. Only when the transfer is complete can the file be accessed by an MP3 player.

The first system described above is really a very specialized form of streaming, but the term today is generally used to refer to transmission of content over a network. What makes streamed data special is that the content may be accessed and used without waiting for the completion of a file transfer. An example of streaming is the sample audio clips offered by on-line vendors of CDs. Many offer the ability to hear a few seconds of one or more tracks before making a purchase decision.

Streaming in this environment is not the same as the isochronous studio interface. We must assume a connection that supports a rate at least close to the data rate of the clip being played. If the clip is encoded at 384 kbits/s and we attempt to stream over a 20 kbits/s link, the best we could offer is one second of audio followed by a 20-second wait, then one more second of audio. In this case, waiting for a complete file transfer and playing the clip from the hard disk is the only practical solution.

But even if the average link speed is sufficient to support approximately real-time transfer, the transfer rate will rarely be constant. Delivery via the Internet generally involves many links and many routers. Transmission is usually "bursty" and packets may not be delivered in the order that they were transmitted. Clearly buffering is needed to permit an acceptable presentation. An intelligent player application will adjust the buffer time according to the characteristics of the incoming data; if the buffer empties (resulting in a break in presentation) the buffer size will be increased to make future interruptions less likely.

Applications for Streaming Media

The usefulness of streaming media is totally dependent on the availability of adequate bandwidth. The comments that follow are based on the environment as seen late in the year 2000. I make some pessimistic predictions below, and I hope I shall be proved wrong, as most commentators are! Time will tell.

In corporate intranets this may not be an issue: 100 Mbits/s Ethernet is starting to replace 10 Mbits/s, star network topology is rapidly replacing loops, and switches are replacing hubs. All of these factors help to increase the bandwidth available to a user and to reduce bottlenecks in the network. Streaming video over corporate networks is viable today, and the situation is continually improving.

Away from this environment the situation is much less encouraging. Audio streaming is useful on the Internet. The standard home connection is a 56 kbits/s modem, often achieving about 40 kbits/s. Modern techniques of audio compression can offer acceptable speech quality at 10 kbits/s or less and reasonable music quality from about 20 kbits/s. The standard connection, and the Internet itself, offer sufficient overhead for these rates to be supported quite consistently.

As mentioned above, e-commerce vendors can offer music samples that genuinely improve the shopping experience. Congressional hearings and shareholder meetings are now routinely "broadcast" over the Internet.

Video is a very different story and has, to date, achieved little more than novelty value over the Internet. High-quality, full-screen, full—refresh-rate video still requires megabits per second. Improvements in compression technology will help, but there is no way we will be able to deliver serious video in a few hundred kilobits per second. While we are confined to connections with even lower bandwidths, Internet streaming video will likely remain postage-stamp sized, and/or have very a low refresh rate.

There are useful applications for streaming video. As mentioned above, corporate intranets can provide a suitable infrastructure. Aside from this, video conferencing and distance learning are attractive applications that can deliver real benefits even today, but the value is greatest where the video is a supplement to other content (audio, faxed documents, prepublished hard copy material, electronic slide presentations, etc.). Generally video conferencing and distance learning use leased network connections or dial-up ISDN, offering at least 128 kbits/s. Even at this, quality is low, update is slow, and either the video is delayed with respect to the audio, or both are delayed, making conversation difficult.

We hear talk of the "bandwidth explosion." Will the situation improve soon? Personally I think not, even though there is a great deal of unused fiber on long-distance routes. High-speed home connections are available in the form of DSL and cable modems. These certainly offer an improved experience, but mainly because there are so few of them. There is little evidence that the switching infrastructure supporting these connections is growing at anything like the necessary pace. More and more Web sites are offering content that requires more bandwidth. I see an explosion of bandwidth demand and creeping infrastructure bandwidth growth.

The spread of multicasting technology will certainly help with applications such as "live" broadcast. Today, almost all Internet stream-

ing relies on unicast operation—the server must deliver a separate bitstream to each individual user, addressed appropriately, even if thousands of users are watching a live event. When more multicast technology is available, servers will be able to deliver a single bitstream to which multiple users may subscribe.

Standards for Streaming Media

The first standard to achieve widespread adoption was H.261, an ITU Standard approved in 1990. It is intended for use on ISDN circuits and uses multiples of 64 kbits/s. It supports the common image format (CIF) at 352 × 276 pixels and the quarter-size QCIF at 176 × 144 pixels. It uses a combination of motion compensation, DCT, and VLC, and can code at rates up to 30 frames/s. Unlike MPEG, H.261 and other streaming media standards permit frame dropping as a means to control bit rate. It is very common in video conferencing systems to see the image freeze when some rapid motion, such as a pan, occurs.

H.263 followed in 1996. Initially this standard was intended for very low bit rates only (up to 64 kbits/s), but this restriction was removed and the standard is likely to replace H.261. It employs improved coding algorithms and supports QCIF and SQCIF (approximately half-QCIF). As an option, implementations can support higher resolutions up to 16 CIF at 1408 × 1152 pixels.

More recently, the H.323 series of standards provides higher quality over corporate intranets and to institutions with second-generation Internet connections. Operation is typically in the hundreds of kilobits/s range. The higher quality offered by these standards and bandwidths is particularly attractive to academic and medical conferencing.

In the world of personal computers there is intense rivalry among three companies, Apple, RealNetworks, and Microsoft. The original format, and still very popular, is *Quicktime* by Apple. This format supports a wide range of bit rates and compression schemes—nowadays almost anything can be made into a Quicktime movie. On the CD included with this book you will find commercial stock footage at full resolution, and low bit rate preview versions of the same clips; both are supplied as Quicktime movies. There is a free version of the Quicktime player, but Apple does not seem to offer any free version of the encoder.

RealNetworks range of products including *RealAudio* and *RealVideo* have probably shown the fastest growth in recent years as many web sites have added audio and video elements. As mentioned previously, we now expect to hear audio samples from CDs before we buy them online. The RealNetworks solutions have proven very attractive at the low bit rates available on today's Internet. RealNetworks offer a free basic player, and a trial version of the encoder is also available for download without charge. More sophisticated players, and fully functional encoders, are available for a price.

The most recent major contender is Microsoft Corporation. It has mounted a major effort to redefine streaming media with the *Advanced Streaming Format* (ASF), offering end-to-end solutions. Microsoft offers a complete range of tools including players, encoders, servers, and supplementary tools at its *Windows Media* site. Microsoft's new Media Player, available at this site and included with the latest operating systems, supports a wide range of coding formats (except Real), and the latest encoder supports MPEG-4 (Version 1 at the time of writing). The CD includes sample Windows Media files, at audio/video bit rates ranging from 28 kbits/s to 2Mbits/s. Microsoft's determination to become a substantial player in the field of streaming media may be deduced from the fact that (at the time of writing) *all* of these tools are available without charge.

Some of the tools mentioned above will be found on the CD-ROM included with this book; others may be obtained from the respective websites. In all cases, it would be wise to check the websites for the latest versions of these products.

Microsoft Corporation has mounted a major effort to redefine streaming media with *Advanced Streaming Format* (ASF) offering end-to-end solutions. Microsoft offers a full range of free tools, including encoders, players, and servers, plus a wealth of information at its Web site. Microsoft's new Media Player, available at this site, and included with the latest operating systems, supports a vast range of coding formats (except Real), and the latest encoder supports MPEG-4 (Version 1 at the time of writing).

Some of these tools are on the CD-ROM included with this book, but check out the Web sites for latest versions.

CHAPTER 19

Closing Thoughts

Introduction

This chapter is the dumping ground for all the bits and pieces that would not fit elsewhere. For example, I take some time here to look at the issue of concatenated compression systems—a subject of great importance since compression is used in acquisition, in editing, and in delivery. There is a discussion of mezzanine, or "in between," compression systems that may offer a partial solution to some of these issues.

The use of compression systems for television distribution presents many new challenges. Common functions, such as the insertion of commercials at local stations or at cable head ends, may depend on the ability to switch compressed bit streams. We examine some of the difficulties inherent in this process.

Finally, we take a brief look at some future directions—those that are fairly close, and some that perhaps require "suspension of disbelief."

Fractal Compression

Fractal compression requires enormous computing power for encoding, but a relatively simple decoder. Fractal compression takes a block of pixels, and attempts to find another block that can be transformed into the current block. Rotation, reflection, scaling, gain, offset, and the like are all allowed. If large parts of the image can be reduced to a recursive relationship, these parts of the image may be represented by a relatively small number of parameters of a *contractive mapping*.

Some workers claim spectacular results with this method, but as yet it has found little general acceptance. Perhaps the most significant claim of fractal compression is that it produces a file that may be rendered back at any resolution, lower or higher than the original. Of course, no additional detail is created by enlarging the image, but the fractal approach ensures that typical artifacts of enlargement, such as pixel blocking, do not appear.

The CD-ROM with this book includes two fractal-based applications for compressing static images. I encourage you to experiment with these and compare the results with those obtained from DCT- and wavelet-based applications (also on the CD). The fractal programs are not commercial products and do not have the same degree of "polish" as the for-sale applications, but are suitable for experiment. It is interesting to note that the only fractal-based commercial program I

know of has been withdrawn (though the shareware version is still available at some Internet locations).

Statistical Multiplexing

In Chapter 9 we saw that temporal compression necessarily results in variable bit rate at the output of the compression encoder. The efficiency gained by predictive encoding in P- and B-frames means that they will be smaller than I-frames; conversely, for an I-frame to be a good reference for generating P- and B-frames, it must be as error free as possible—in other words, as big as possible. Chapter 9 described the type of rate control mechanism and the buffers necessary to interface an MPEG encoder to a fixed-rate transmission circuit.

The above discussion addresses only a part of the problem, in that it assumes constant complexity of the video being encoded. In practice, video varies a great deal in complexity. A "talking-head" shot is easier to encode than a scene with moving objects. A static graphic is easier still; at the other extreme, a zooming shot of complex motion in a basketball game, with a crowd behind, is about as difficult as video can get. It is fairly obvious that there will be both short- and long-term variations. A sport program is intrinsically more "bit hungry" than a talk show, but within each program there will be some parts that are much more complex than others. Unfortunately, even the short-term variations are likely to have time scales of tens of seconds or minutes, so increased buffering is not a practical solution.

The problem is a real one; at a fixed bit rate greater complexity means lower quality, lesser complexity means higher quality. Unfortunately, the human psychovisual system is much more sensitive to variations in quality than it is to absolute quality, and such variations are likely to be visible and annoying.

In a system with a single program stream and fixed bit rate there is little to be done, except to try to design the system so that the lowest quality received (most-complex video) is sufficiently good that improvements in quality are not readily apparent. If this is achieved, there is a choice—the quality can be allowed to vary, or the quality can be held down to the acceptable level and the surplus bits used for some other purpose or just "stuffed."

Some applications allow variation in bit rate. A movie on DVD (previously digital video disc or digital versatile disc, now just DVD) uses

an average bit rate of about 5 Mbits/s, but the DVD system is capable of a peak bit rate of some 10 Mbits/s. This allows the compression device (or the human compressionist) considerable leeway in the allocation of bits to scenes of different complexity.

Another solution is available when a fixed-rate MPEG transport bit stream carries a number of program streams (as described in Chapter 10), as used by direct broadcast satellite systems. Statistical multiplexing is a technique based on the assumption that not all of the program streams will carry complex video at the same time. Within the fixed-rate bit stream, at any instant more bits can be allocated to the program streams carrying complex video, and fewer to those carrying simple video, stills, and other data. Three approaches are available to control the distribution of bits:

- The bit allocation of a program may be determined by its type; a football game will need more bits than a sitcom, and a movie coded at 24 frames per second will probably need even less bits than the sitcom. In a simple system this determination can be used to divide the bit allocations between the various program streams. In a more complex system with dynamic control of bit allocation, program type may be used for the initial estimate of bit rate requirements for each program stream.
- Using conventional rate control techniques, information from the rate controller of each compression encoder can be considered by a statistical multiplex controller. This controller will provide a modifier control back to each rate controller so that encoders operating at a low quantization scale factor will be instructed to quantize more aggressively, thereby freeing up additional bits for encoders that are being forced to use a high scale factor. In this way, the statistical multiplex controller can balance out the bit allocations so that all program streams are using a similar scale factor and providing a similar level of quality.
- The most sophisticated controllers may use *look-ahead* techniques to measure the complexity of upcoming frames (as may be done by the encoder itself). This information may be used to further "tune" the rate controllers. For example, if several streams have complex material arriving shortly, it may be appropriate to increase all quantization scale factors to reduce buffer fullness and provide for a peak in bit requirements.

The above techniques work best if the transport stream is carrying a considerable number of program streams, and if there is a good mix

of program types. Obviously, statistical multiplexing is complex and expensive; however, the payoffs are considerable. Use of this technique not only averages out the impairment level of each program stream, but for a given quality level it allows more program streams within a channel of given capacity. In fact, the gain is even higher in the commercial world; if the quality is not varying appreciably, a lower quality can be tolerated. (Expressed simply, the human psychovisual system, and viewers' psychology, will accept a given quality level if it is not shown how much better it could be!) These factors make it possible to substantially increase the number of program streams of acceptable quality in a given channel. In commercial terms, each satellite transponder can carry more programs. Given that each transponder in place represents an investment of tens of millions of dollars, the advantage is compelling.

Concatenated Compression Systems

I have discussed at length the sensitivity of compression systems to artifacts and noise on the signal to be encoded. These can cause substantial losses in efficiency. A compression system adds artifacts and quantization noise to an image sequence, so what happens when we feed one compression system with the output of another? This is far from an idle question. We are seeing increasing use of compression systems for acquisition and in studio operations, and we are about to see the large-scale introduction of heavily compressed digital transmission systems. There would seem to be ample reason for concern.

Experimental results suggest that compression systems are very tolerant of their own artifacts. In most of the compression systems in use today, a quality loss is apparent on the first pass, but if the signal is not changed in any way, subsequent passes through an identical compression system add almost no loss.

In one sense we can see that this is intuitively reasonable. A compression system discards information, particularly information that should have little effect on the perceived image. A second pass through an identical system attempts to throw away the same information, but it is no longer there. The second system should be able to code the remaining information just as happily as the first.

Jim Wilkinson of Sony first explained this to me as a theoretical model. Most of the elements in the compression encode and decode