CHAPTER 20

# Facsimile Compression

KHALID SAYOOD

## OVERVIEW

In this chapter we describe the compression schemes used for facsimile compression. We also briefly describe how these compression schemes have been incorporated into various international standards.

## 20.1 A BRIEF HISTORY

One of the earliest applications of lossless compression was to the compression of facsimile, perhaps because facsimile transmission is one of the oldest forms of digital transmission still in existence. The first facsimile machines were patented soon after Samuel Morse's patenting of the telegraph, in 1836. In 1843 A. Bain patented the *recording telegraph*, followed in 1850 by F. C. Blakewell, who patented the *copying telegraph*. The first commercial facsimile system was set up in France in 1865 using a system, called the pantelegraph, developed by G. Caselli.

It was almost a century before the precursor of the modern fax system made its arrival in 1968 with the development of the Group 1 facsimile standard. The Group 1 standard formalized in Recommendation T.2 of the CCITT, which is now ITU-T, coded the output of a photo-cell using two tones. In the CCITT recommendation a white pixel was represented by a tone at 1300 Hz and a black pixel was represented by a tone at 2300 Hz (in North America a white pixel was represented by a tone at 1500 Hz and a black pixel by a tone at 2300 or 2400 Hz). This standard allowed the transmission of an 8.5 in. by 11 in. page in 6 min. In 1976 the Group 1 fax standard was replaced by the Group 2 fax standard, which used vestigial sideband modulation to transmit a standard page in 3 min.

Compression came into the picture with the Group 3 and Group 4 standards which were published as ITU-T Recommendation T.4 and T.6. These standards permitted the transmission

THE SLEREXE COMPANY LIMITED

SAPORS LANE - BOOLE - DORSET - BH25 8ER

TELEPHONE BOOLE (945 13) 51617 - TELEX 123456

Our Ref. 350/PJC/EAC 18th January, 1972.

Dr. P.N. Cundall,
Mining Surveys Ltd.,
Holroyd Road,
Reading,
Berks.

Dear Pete,

Permit me to introduce you to the facility of facsimile transmission.

In facsimile a photocell is caused to perform a raster scan over the subject copy. The variations of print density on the document cause the photocell to generate an analogous electrical video signal. This signal is used to modulate a carrier, which is transmitted to a remote destination over a radio or cable communications link.

At the remote terminal, demodulation reconstructs the video signal, which is used to modulate the density of print produced by a printing device. This device is scanning in a raster scan synchronised with that at the transmitting terminal. As a result, a facsimile copy of the subject document is produced.

Probably you have uses for this facility in your organisation.

Yours sincerely,

Phil.

P.J. CROSS
Group Leader - Facsimile Research

Registered in England: No. 2566
Registered Office: 60 Victoria [illegible], Ilford, Essex.

**FIGURE 20.1**
A test document image.

time for a page to come down to 1 min (assuming a 4800-bps modem, and excluding the time for handshaking). A "page" in this context refers to a white page with black pixels making up about 4% of the page as in the CCITT test chart 1, also known as the *slerexe page* shown in Fig. 20.1. The page is sampled at 1728 samples per line, with one page consisting of 2376 scan lines. Thus a standard page can be viewed as a bilevel image of size 2376 × 1728. If we were to try and transmit such a page without compression over a 4800-bps line, it would take a little more than 14 min. It would take almost an hour to send a four-page document. Clearly there is a need for compression.

The compression schemes used in Group 3 and Group 4 have also found an application as part of other *de facto* standards such as the Tagged Image File Format (TIFF), which in turn has given rise to a new Internet fax standard known as TIFF-FX. The increase in processing power available to devices has led to more efficient bilevel compression schemes. These are used in the JBIG and JBIG2 standards that were published as ITU-T Recommendation T.82 and T.88. Finally, with the ever-increasing popularity of color scanners and printers a new standard for color facsimile based on an approach called Mixed Raster Compression has been published by ITU-T in Recommendation T.44.

In this chapter we first describe the compression algorithms that are part of the various standards. We then briefly describe how these techniques have been incorporated in the international standards.

## 20.2 THE COMPRESSION ALGORITHMS

In this section we describe the various compression algorithms used for facsimile compression. These compression algorithms take advantage of the particular structure present in documents and are used in various international standards.

### 20.2.1 Modified Huffman

Looking at any document we can clearly see that black and white pixels are not randomly scattered through the page. They occur in groups. If we are looking at a single scan line, we will see a string of white pixels followed by a string of black pixels and so on. Therefore, it makes sense to encode the number of pixels in each string, or run, rather than the pixels themselves. This is called *run-length coding*. In order to obtain compression the run-lengths that occur more often should be encoded with shorter codewords than the run-lengths that occur less often. A coding scheme that does this is Huffman coding (see Chapter 3). However, the number of values that the run-lengths can take on is very large. You could have run-lengths all the way up to 1728. Creating a Huffman codebook of this size is clearly cumbersome. Furthermore decoding a Huffman code with 1728 entries in the codebook would impose an unrealistic computational load. The Modified Huffman (MH) algorithm represents the run-length $r$ with two numbers, $m$ and $t$, where

$$r = 64m + t.$$

Notice that $t$ can take on values from 0 to 63, and $m$ can take on values from 1 to 27. The MH algorithm uses separate variable-length codes to encode the values of $m$ and $t$. The values of $m$ are encoded using the *make-up* codes shown in Table 20.1 and Table 20.2. The values of $t$ are encoded using the *terminating* codes shown in Table 20.3 and Table 20.4. Note that there are different codes for runs of white pixels and runs of black pixels. Table 20.5 lists the extended make-up codes for runs of black and white pixels.

The one-dimensional coding consists of codewords corresponding to alternate runs of white and black. We assume that the first run will be a run of white pixels. If this assumption is violated, then the code for a run-length of 0 white pixels is transmitted.

### 20.2.2 Modified READ

A facsimile is a two-dimensional object with a very high correlation between pixels on neighboring lines. The modified READ algorithm uses this fact to provide a substantial improvement in compression. The main feature of documents used in the two-dimensional algorithm is that the

**Table 20.1 Make-up White Codes**

| Code | Length | Run | Code | Length | Run |
|---|---|---|---|---|---|
| 11011 | 5 | 64 | 011010100 | 9 | 960 |
| 10010 | 5 | 128 | 011010101 | 9 | 1024 |
| 010111 | 6 | 192 | 011010110 | 9 | 1088 |
| 0110111 | 7 | 256 | 011010111 | 9 | 1152 |
| 00110110 | 8 | 320 | 011011000 | 9 | 1216 |
| 00110111 | 8 | 384 | 011011001 | 9 | 1280 |
| 01100100 | 8 | 448 | 011011010 | 9 | 1344 |
| 01100101 | 8 | 512 | 011011011 | 9 | 1408 |
| 01101000 | 8 | 576 | 010011000 | 9 | 1472 |
| 01100111 | 8 | 640 | 010011001 | 9 | 1536 |
| 011001100 | 9 | 704 | 010011010 | 9 | 1600 |
| 011001101 | 9 | 768 | 011000 | 6 | 1664 |
| 011010010 | 9 | 832 | 010011011 | 9 | 1728 |
| 011010011 | 9 | 896 | | | |

**Table 20.2 Make-up Black Codes**

| Code | Length | Run | Code | Length | Run |
|---|---|---|---|---|---|
| 0000001111 | 10 | 64 | 0000001110011 | 13 | 960 |
| 000011001000 | 12 | 128 | 0000001110100 | 13 | 1024 |
| 000011001001 | 12 | 192 | 0000001110101 | 13 | 1088 |
| 000001011011 | 12 | 256 | 0000001110110 | 13 | 1152 |
| 000000110011 | 12 | 320 | 0000001110111 | 13 | 1216 |
| 000000110100 | 12 | 384 | 0000001010010 | 13 | 1280 |
| 000000110101 | 12 | 448 | 0000001010011 | 13 | 1344 |
| 0000001101100 | 13 | 512 | 0000001010100 | 13 | 1408 |
| 0000001101101 | 13 | 576 | 0000001010101 | 13 | 1472 |
| 0000001001010 | 13 | 640 | 0000001011010 | 13 | 1536 |
| 0000001001011 | 13 | 704 | 0000001011011 | 13 | 1600 |
| 0000001001100 | 13 | 768 | 0000001100100 | 13 | 1664 |
| 0000001001101 | 13 | 832 | 0000001100101 | 13 | 1728 |
| 0000001110010 | 13 | 896 | | | |

transition from a white pixel to a black pixel, or from a black pixel to a white pixel, in any given line will occur at almost the same location as a similar transition in the previous line. The algorithm is a modification of a Japanese proposal to CCITT called the Relative Element Address Designate of READ algorithm [1]. Hence it is referred to as the modified READ or MR algorithm.

To see how the algorithm operates we use an example of two lines of a facsimile image shown in Fig. 20.2. Assume that the contents of the top line is known to both encoder and decoder. On the bottom line the last pixel known to both encoder and decoder is the pixel marked $a_0$. We have labeled some of the transition pixels in the figure. The pixel labeled $b_1$ is the first pixel on the top line which is to the right of the pixel labeled $a_0$ and is of a different color. The pixel labeled $b_2$ is the first pixel to the right of the pixel labeled $b_1$ which is of a different color than pixel $b_1$. The pixel labeled $a_1$ is the first pixel of a different color from the pixel labeled $a_0$ on the line being encoded and to the right of pixel $a_0$. The pixel labeled $a_2$ is the first pixel to the right of pixel $a_1$

**Table 20.3 Terminating White Codes**

| Code | Length | Run | Code | Length | Run | Code | Length | Run |
|---|---|---|---|---|---|---|---|---|
| 00110101 | 8 | 0 | 0010111 | 7 | 21 | 00101011 | 8 | 42 |
| 000111 | 6 | 1 | 0000011 | 7 | 22 | 00101100 | 8 | 43 |
| 0111 | 4 | 2 | 0000100 | 7 | 23 | 00101101 | 8 | 44 |
| 1000 | 4 | 3 | 0101000 | 7 | 24 | 00000100 | 8 | 45 |
| 1011 | 4 | 4 | 0101011 | 7 | 25 | 00000101 | 8 | 46 |
| 1100 | 4 | 5 | 0010011 | 7 | 26 | 00001010 | 8 | 47 |
| 1110 | 4 | 6 | 0100100 | 7 | 27 | 00001011 | 8 | 48 |
| 1111 | 4 | 7 | 0011000 | 7 | 28 | 01010010 | 8 | 49 |
| 10011 | 5 | 8 | 00000010 | 8 | 29 | 01010011 | 8 | 50 |
| 10100 | 5 | 9 | 00000011 | 8 | 30 | 01010100 | 8 | 51 |
| 00111 | 5 | 10 | 00011010 | 8 | 31 | 01010101 | 8 | 52 |
| 01000 | 5 | 11 | 00011011 | 8 | 32 | 00100100 | 8 | 53 |
| 001000 | 6 | 12 | 00010010 | 8 | 33 | 00100101 | 8 | 54 |
| 000011 | 6 | 13 | 00010011 | 8 | 34 | 01011000 | 8 | 55 |
| 110100 | 6 | 14 | 00010100 | 8 | 35 | 01011001 | 8 | 56 |
| 110101 | 6 | 15 | 00010101 | 8 | 36 | 01011010 | 8 | 57 |
| 101010 | 6 | 16 | 00010110 | 8 | 37 | 01011011 | 8 | 58 |
| 101011 | 6 | 17 | 00010111 | 8 | 38 | 01001010 | 8 | 59 |
| 0100111 | 7 | 18 | 00101000 | 8 | 39 | 01001011 | 8 | 60 |
| 0001100 | 7 | 19 | 00101001 | 8 | 40 | 00110010 | 8 | 61 |
| 0001000 | 7 | 20 | 00101010 | 8 | 41 | 00110011 | 8 | 62 |
| | | | | | | 00110100 | 8 | 63 |

**Table 20.4 Terminating Black Codes**

| Code | Length | Run | Code | Length | Run | Code | Length | Run |
|---|---|---|---|---|---|---|---|---|
| 0000110111 | 10 | 0 | 00000110111 | 11 | 22 | 000011011011 | 12 | 43 |
| 010 | 3 | 1 | 00000101000 | 11 | 23 | 000001010100 | 12 | 44 |
| 11 | 2 | 2 | 00000010111 | 11 | 24 | 000001010101 | 12 | 45 |
| 10 | 2 | 3 | 00000011000 | 11 | 25 | 000001010110 | 12 | 46 |
| 011 | 3 | 4 | 000011001010 | 12 | 26 | 000001010111 | 12 | 47 |
| 0011 | 4 | 5 | 000011001011 | 12 | 27 | 000001100100 | 12 | 48 |
| 0010 | 4 | 6 | 000011001100 | 12 | 28 | 000001100101 | 12 | 49 |
| 00011 | 5 | 7 | 000011001101 | 12 | 29 | 000001010010 | 12 | 50 |
| 000101 | 6 | 8 | 000001101000 | 12 | 30 | 000001010011 | 12 | 51 |
| 000100 | 6 | 9 | 000001101001 | 12 | 31 | 000000100100 | 12 | 52 |
| 0000100 | 7 | 10 | 000001101010 | 12 | 32 | 000000110111 | 12 | 53 |
| 0000101 | 7 | 11 | 000001101011 | 12 | 33 | 000000111000 | 12 | 54 |
| 0000111 | 7 | 12 | 000011010010 | 12 | 34 | 000000100111 | 12 | 55 |
| 00000100 | 8 | 13 | 000011010011 | 12 | 35 | 000000101000 | 12 | 56 |
| 00000111 | 8 | 14 | 000011010100 | 12 | 36 | 000001011000 | 12 | 57 |
| 000011000 | 9 | 15 | 000011010101 | 12 | 37 | 000001011001 | 12 | 58 |
| 0000010111 | 10 | 16 | 000011010110 | 12 | 38 | 000000101011 | 12 | 59 |
| 0000011000 | 10 | 17 | 000011010111 | 12 | 39 | 000000101100 | 12 | 60 |
| 0000001000 | 10 | 18 | 000001101100 | 12 | 40 | 000001011010 | 12 | 61 |
| 00001100111 | 11 | 19 | 000001101101 | 12 | 41 | 000001100110 | 12 | 62 |
| 00001101000 | 11 | 20 | 000011011010 | 12 | 42 | 000001100111 | 12 | 63 |
| 00001101100 | 11 | 21 | | | | | | |

**Table 20.5 Extended Make-up Codes (Black and White)**

| Code | Length | Run |
|---|---|---|
| 00000001000 | 11 | 1792 |
| 00000001100 | 11 | 1856 |
| 00000001101 | 11 | 1920 |
| 000000010010 | 12 | 1984 |
| 000000010011 | 12 | 2048 |
| 000000010100 | 12 | 2112 |
| 000000010101 | 12 | 2176 |
| 000000010110 | 12 | 2240 |
| 000000010111 | 12 | 2304 |
| 000000011100 | 12 | 2368 |
| 000000011101 | 12 | 2432 |
| 000000011110 | 12 | 2496 |
| 000000011111 | 12 | 2560 |

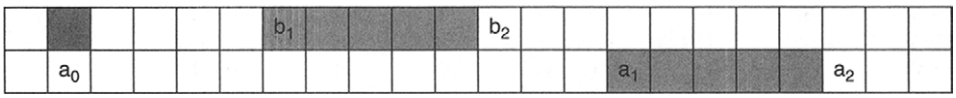


**FIGURE 20.2**
Two lines of a facsimile image.

which is of a different color from pixel $a_1$. Note that the locations of pixel $a_0$, pixel $b_1$, and pixel $b_2$ are known to both encoder and decoder, while the locations of pixel $a_1$ and pixel $a_2$ are known only to the encoder.

The encoder functions in one of three modes based on the relative locations of pixels $a_1$, $b_1$, and $b_2$. If pixel $a_1$ is to the right of pixel $b_2$, the encoder is in *pass mode*. If pixel $a_1$ is not to the right of pixel $b_2$ and within three pixels of pixel $b_1$ the encoder is said to be in *vertical mode*. If neither of these conditions is satisfied, the encoder is said to be in *horizontal mode*. This procedure is summarized in Fig. 20.3.

In the *pass mode* the encoder sends the codeword 0001 to the decoder. This lets the decoder know that all pixels from pixel $a_1$ to just under pixel $b_2$ have the same color as pixel $a_1$. The pixel under pixel $b_2$ becomes the new pixel $a_0$ and both encoder and decoder update the locations for the other transition pixels with reference to the new pixel $a_0$. Encoding continues.

In the *vertical mode* the encoder encodes the location of $a_1$ using the following codes:

| | |
|---|---|
| $a_1$ directly under $b_1$ | 1 |
| $a_1$ to the right of $b_1$ by one pixel | 011 |
| $a_1$ to the right of $b_1$ by two pixels | 000011 |
| $a_1$ to the right of $b_1$ by three pixels | 0000011 |
| $a_1$ to the left of $b_1$ by one pixel | 010 |
| $a_1$ to the left of $b_1$ by two pixels | 000010 |
| $a_1$ to the left of $b_1$ by three pixels | 0000010 |

Once the decoder learns the current location of the pixel $a_1$, that pixel is relabeled $a_0$ and we repeat the procedure.

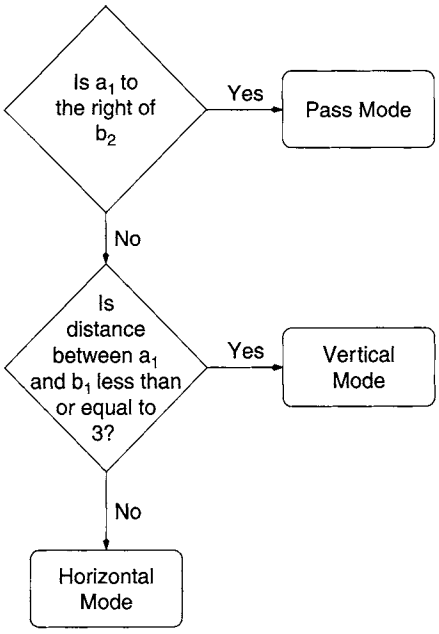


**FIGURE 20.3**
Selection of coding mode.

In the horizontal mode the encoder sends the code 001. The next two run-lengths, that is, the distance between the pixels labeled $a_0$ and $a_1$ and that between the pixels labeled $a_1$ and $a_2$, are transmitted using modified Huffman codes.

Note that all the codewords mentioned above form a prefix code; no codeword is a prefix of any other codeword.

### 20.2.3 Context-Based Arithmetic Coding

Arithmetic coding was developed in its current form in the mid-1970s [2, 3] and has rapidly gained in popularity. The details of the arithmetic coding technique are presented in Chapter 5. Arithmetic coding is especially useful for the encoding of facsimile documents because of the structure of the data. The technique is particularly effective when the probabilities of the symbols being encoded are highly skewed. The probabilities of black and white pixels in facsimiles are highly skewed if we take into account the neighborhood, or the context, of the pixel being encoded. In context-based arithmetic coding the probabilities used to encode a particular symbol depend on the context in which that symbol occurs. A table of frequency counts is kept for the different context values. Consider the pixel marked $X$ in Fig. 20.4. The bold outline delineates the context for the pixel $X$. As the pixels can take on only two values, the context, which contains 10 pixels, can take on 1024 values. In a context-based scheme one might have a separate table of frequency of occurrence for each context value.

Taking the contexts into account provides a much more accurate estimate of the probability of a given pixels, which in turn results in higher compression. The size and shape of the context are important factors in determining the effectiveness of this approach.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | X | ? | ? | ? | ? | ? | ? | ? |
| ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |

**FIGURE 20.4**
Context for coding pixel $X$.

### 20.2.4 Run-Length Color Encoding

This is a simple extension of standard run-length encoding and is formalized in ITU-T Recommendation T.45. The bitstream is divided into *header* and *data*. The header specifies the number of color components, the number of bytes used to represent each component value, and the number of different color values that are encoded.

The data consist of alternating run-lengths and values. The run-length is encoded using 1 or 3 bytes. If the run-length is between 1 and 255, a single byte is sufficient. The all-zero byte is used to signal the fact that the run-length is greater than 255. The next 2 bytes contain the value of the run. The color value is encoded using $n \times k$ bytes, where $n$ is the number of components and $k$ is the number of bytes used to represent the value of each component.

## 20.3 THE STANDARDS

The compression algorithms described in the previous section have been incorporated in a number of different international standards. These standards have evolved with the development of technology and the evolution of documents. In this section we briefly describe these standards.

### 20.3.1 ITU-T Group 3 (T.4)

The Group 3 standard contains two types of coding schemes; the one-dimensional coding scheme known as the Modified Huffman scheme and the two-dimensional scheme known as the Modified Modified READ scheme.

In the one dimensional scheme the modified Huffman codewords for each line of the facsimile is terminated by an end-of-line (EOL) codeword. This is a unique codeword consisting of 11 zeros followed by a one. The use of this symbol allows the decoder to synchronize after an error burst.

The two-dimensional coding is substantially more effective in terms of compression than the one-dimensional scheme. However, it is also vulnerable to error propagation. An error in one line will get propagated to the next line and so on. In order to prevent runaway error the Group 3 standard requires that every $K$ lines one-dimensional coding be used where $K$ is either 2 or 4.

The standard also includes provisions for optional error limiting and error correction modes. The error limiting mode is used only in the one-dimensional coding scheme. In this mode a line of 1728 pixels is divided into 12 groups of 144 pixels each. A 12-bit header is then constructed with 1 bit for each group of 144 pixels. If all pixels in a group are white, the corresponding bit in the header has a value of 0. If not, it has a value of 1. The all-white groups do not need to be encoded as the header contains all information necessary to reconstruct them. The other groups are encoded using modified Huffman codes.

In the error correction mode the bitstream is broken up into packets with each packet containing an error-detecting code. If a packet is detected to be in error, a request for retransmission is sent to the encoder. The request is not honored until the entire page has been transmitted. Each packet can be retransmitted a maximum of four times.

The end of document transmission is indicated by sending six consecutive EOLs.

### 20.3.2 Group 4 (T.6)

The Group 4 algorithms were standardized in ITU-T Recommendation T.6. Essentially, it takes advantage of the much cleaner digital lines to remove the restriction of using one-dimensional coding from the modified READ algorithm. As the two-dimensional algorithm is substantially more effective than the one-dimensional algorithm, this can dramatically increase the amount of compression. To reflect the fact that this algorithm is a (slightly) modified version of modified READ, it is called the *Modified Modified READ* algorithm or MMR.

### 20.3.3 JBIG and JBIG2 (T.82 and T.88)

In recent years two new standards have been developed by the Joint Bi-Level Experts Group, which is a joint committee of ITU-T, ISO, and IEC. These standards provide much higher compression as well as progressive transmission. The JBIG2 standard even allows for lossy compression of facsimile documents. The compression approach used in the JBIG standard is based on arithmetic coding. The standard describes a progressive transmission approach in which the context for encoding of a pixels can be made up of pixels from the neighbors of the pixel in the current layer or a lower resolution layer.

The more recent JBIG2 standard reflects the evolving nature of documents that increasingly contain images. The document to be encoded is segmented into *halftone regions, symbol regions*, and *generic regions*. Halftone regions are parts of the document containing halftone images, symbol regions are parts of the document containing text symbols, and the generic regions are regions that do not fit into either of the first two categories. Each region is encoded using techniques appropriate to the type of structures present in the data corresponding to the region.

Details on both the JBIG and the JBIG2 standards can be found in Chapter 17.

### 20.3.4 MRC—T.44

As society develops and evolves so do documents. In recent years, documents contain more and more multicolored text and images. When such documents are converted to bilevel images, there can be considerable loss of resolution. This level of quality can be unacceptable to consumers. To deal with the changing nature of the document and consumer expectations the ITU-T has accepted the Mixed Raster Content (MRC) approach as a new standard for facsimile transmission.

The MRC approach recognizes the fact that no compression scheme works well with both continuous tone images and text. The JPEG standard and the new JPEG2000 standard provide high compression with excellent quality for natural images. However, if the text portion of the document is compressed to the same level, there is a loss of crispness in text. The JBIG and JBIG2 standards provide excellent compression for text but they do not provide a very high level of compression for continuous tone images. Recognizing this fact the MRC strategy is to decompose the facsimile image into three layers (there is an option to increase the number of layers). The *background layer* consists of the portions of the document that consist of continuous tone images. The foreground layer consists of portions of the image containing text or line art.

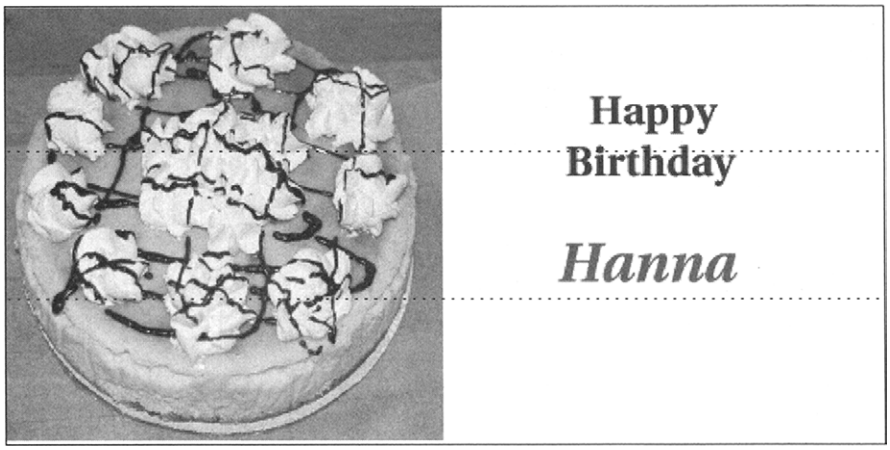


**FIGURE 20.5**
Test image. (See also color insert.)

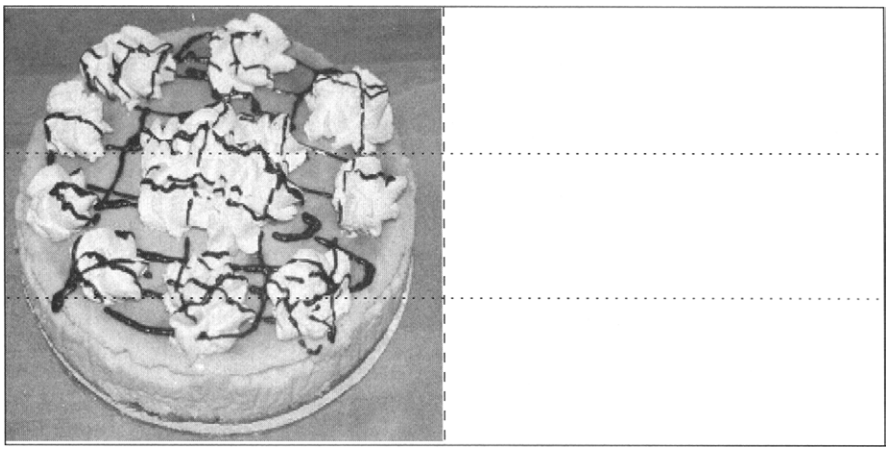

**FIGURE 20.6**
Background for the test image. (See also color insert.)

A bilevel mask layer provides directions for the reconstruction of the image. A black pixel (value of 1) indicates that that particular pixel in the reconstructed image should come from the foreground layer and a white pixel indicates that the pixel in that location in the reconstructed image should come from the background layer.

As an example consider the document shown in Fig. 20.5 (see also color insert). The left half of the picture contains a continuous tone image, while the right half of the image contains multicolored text (we will describe the meaning of the dashed and dotted lines in this and the following figures later in the chapter). The background layer, foreground, and mask layers for this document are shown in Fig. 20.6–20.8 (see also color insert for Figs. 20.6 and 20.7).

Because of memory restrictions in the coder, the document can be partitioned into stripes. As an example we have partitioned the test document shown into three stripes. Notice that not all stripes in Fig. 20.5 contain both foreground and background layers. Furthermore notice that in the

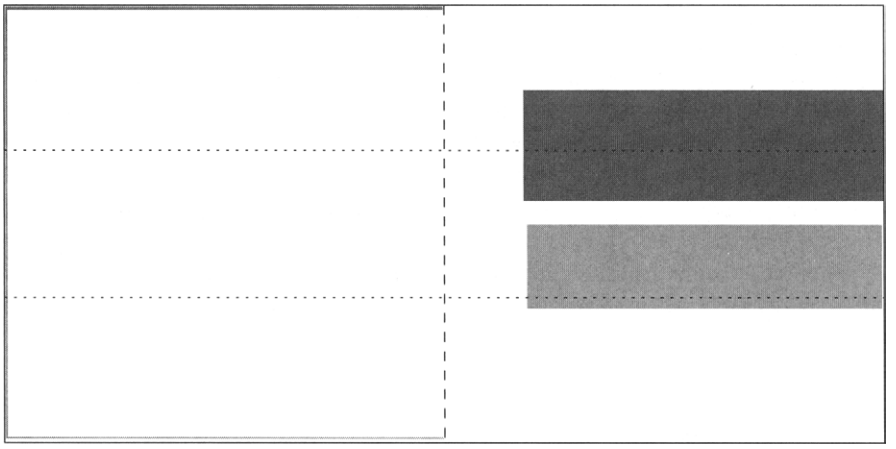

**FIGURE 20.7**
Foreground for the test image. (See also color insert.)

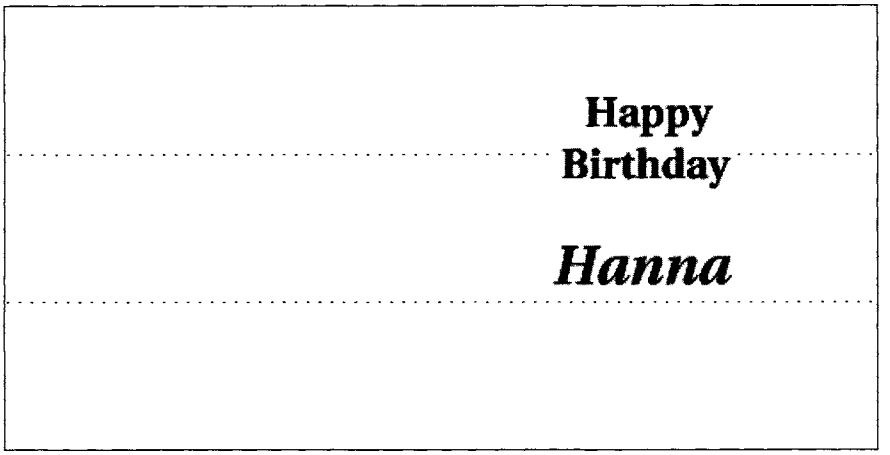


**FIGURE 20.8**
Mask for the test image.

foreground and background not all of the stripe is occupied by picture or text. To take advantage of these the recommendation allows three kinds of stripes: three-layer stripes (3LS), two-layer stripes (2LS), and one-layer stripes (1LS). The three-layer stripes contain mask foreground and background layers. The mask layer is transmitted first followed by the background and then the foreground layer. The two-layer stripes contain the mask layer and either the foreground or the background layer. The layer not included is set to a constant value. This type of stripe is useful when the stripe contains monochrome text or line art and a continuous tone image or no image and multicolored text. In this case also the mask layer is transmitted first followed by the foreground or background layer. The one-layer stripe is useful when the stripe contains only a continuous tone image or monochrome text or line art. This is the case for the lowest stripe in the test image.

Only the mask layer is required to span the entire width of the stripe. The other layers can be specified to cover only part of the width of the strip. In the example shown in the figures, in the background stripe the data to the right of the dashed line need not be encoded.

The mask layer can be coded using Modified Huffman (T.4), Modified READ (T.4), Modified Modified READ (T.6), or JBIG (T.82). The coder used is specified in the set of bytes starting with the 13th byte in the Start of Page marker segment. Currently with only these four coding schemes the coder can be specified by setting one of the bits in a single byte. If the number of encoders gets to be greater than or equal to 7, there will be a need for more than a single byte. The other two layers can be encoded using either JPEG (T.81) or JBIG (T.82). The encoding method used is specified in the byte(s) following the bytes that specify the coder for the mask layer in the Start of Page marker segment.

Unlike previous standards there is an inherent assumption in T.44 and the MRC approach that there will be new developments in compression and in the nature of documents. This flexibility should make it a valuable standard for years to come.

### 20.3.5 Other Standards

There are a number of other facsimile standards including military and NATO (STANAG) standards. However, they are principally variations of the standards described above. With the increased used of the Internet for a variety of communication activities previously restricted to other channels, the Internet Engineering Task Force has been working on a standard for Internet fax. The emerging standard is popularly known as TIFF-FX. This standard allows for inserting data encoded using the facsimile standards described above into the tagged image file format. The TIFF-FX standard can be viewed as a superset of all the standards described in this chapter. Its various profiles support T.4, T.6, T.44, and T.82 as well as JPEG (T.81).

## 20.4 FURTHER READING

- The JBIG and JBIG2 standards are described in Chapter 17.
- Huffman coding and arithmetic coding are described in Chapters 4 and 5, respectively.
- The various ITU standards themselves are quite readable and are the ultimate source for information.

## 20.5 REFERENCES

1. Yasuda, Y., 1980. Overview of digital facsimile coding techniques in Japan. *IEEE Proceedings*, Vol. 68, pp. 830–845, July 1980.
2. Pasco, R., 1976. *Source Coding Algorithms for Fast Data Compression*, Ph.D. thesis, Stanford University.
3. Rissanen, J. J., 1976. Generalized Kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, Vol. 20, pp. 198–203, May 1976.