[Previous] [Next]

CCITT (Huffman) Encoding

Many facsimile and document imaging file formats support a form of lossless data compression often described as CCITT encoding. The CCITT (International Telegraph and Telephone Consultative Committee) is a standards organization that has developed a series of communications protocols for the facsimile transmission of black-and-white images over telephone lines and data networks. These protocols are known officially as the CCITT T.4 and T.6 standards but are more commonly referred to as CCITT Group 3 and Group 4 compression, respectively.

Sometimes CCITT encoding is referred to, not entirely accurately, as Huffman encoding. Huffman encoding is a simple compression algorithm introduced by David Huffman in 1952. CCITT 1-dimensional encoding, described in a subsection below, is a specific type of Huffman encoding. The other types of CCITT encodings are not, however, implementations of the Huffman scheme.

Group 3 and Group 4 encodings are compression algorithms that are specifically designed for encoding 1-bit image data. Many document and FAX file formats support Group 3 compression, and several, including TIFF, also support Group 4.

Group 3 encoding was designed specifically for bi-level, black-and-white image data telecommunications. All modern FAX machines and FAX modems support Group 3 facsimile transmissions. Group 3 encoding and decoding is fast, maintains a good compression ratio for a wide variety of document data, and contains information that aids a Group 3 decoder in detecting and correcting errors without special hardware.

Group 4 is a more efficient form of bi-level compression that has almost entirely replaced the use of Group 3 in many conventional document image storage systems. (An exception is facsimile document storage systems where original Group 3 images are required to be stored in an unaltered state.)

Group 4 encoded data is approximately half the size of 1-dimensional Group 3-encoded data. Although Group 4 is fairly difficult to implement efficiently, it encodes at least as fast as Group 3 and in some implementations decodes even faster. Also, Group 4 was designed for use on data networks, so it does not contain the synchronization codes used for error detection that Group 3 does, making it a poor choice for an image transfer protocol.

Group 4 is sometimes confused with the IBM MMR (Modified Modified READ) compression method. In fact, Group 4 and MMR are almost exactly the same algorithm and achieve almost identical compression results. IBM released MMR in 1979 with the introduction of its Scanmaster product before Group 4 was standardized. MMR became IBM's own document compression standard and is still used in many IBM imaging systems today.

Document-imaging systems that store large amounts of facsimile data have adopted these CCITT compression schemes to conserve disk space. CCITT-encoded data can be decompressed quickly for printing or viewing (assuming that enough memory and CPU resources are available). The same data can also be transmitted using modem or facsimile protocol technology without needing to be encoded first.

The CCITT algorithms are non-adaptive. That is, they do not adjust the encoding algorithm to encode each bitmap with optimal efficiency. They use a fixed table of code values that were selected according to a reference set of documents containing both text and graphics. The reference set of documents were considered to be representative of documents that would be transmitted by facsimile.

Group 3 normally achieves a compression ratio of 5:1 to 8:1 on a standard 200-dpi (204x196 dpi), A4-sized document. Group 4 results are roughly twice as efficient as Group 3, achieving compression ratios upwards of 15:1 with the same document. Claims that the CCITT algorithms are capable of far better compression on standard business documents are exaggerated--largely by hardware vendors.

Because the CCITT algorithms have been optimized for type and handwritten documents, it stands to reason that images radically different in composition will not compress very well. This is all too true. Bi-level bitmaps that contain a high frequency of short runs, as typically found in digitally halftoned continuous-tone images, do not compress as well using the CCITT algorithms. Such images will usually result in a compression ratio of 3:1 or even lower, and many will actually compress to a size larger than the original.

The CCITT actually defines three algorithms for the encoding of bi-level image data:

- Group 3 One-Dimensional (G31D)
- Group 3 Two-Dimensional (G32D)
- Group 4 Two-Dimensional (G42D)

G31D is the simplest of the algorithms and the easiest to implement. For this reason, it is discussed in its entirety in the first subsection below. G32D and G42D are much more complex in their design and operation and are described only in general terms below.

The Group 3 and Group 4 algorithms are standards and therefore produce the same compression results for everybody. If you have heard any claims made to the contrary, it is for one of these reasons:

- Non-CCITT test images are being used as benchmarks.
- Proprietary modifications have been made to the algorithm.
- Pre- or post-processing is being applied to the encoded image data.
- You have been listening to a misinformed salesperson.

Group 3 One-Dimensional (G31D)

Group 3 One-Dimensional encoding (G31D) is a variation of the Huffman keyed compression scheme. A bi-level image is composed of a series of black-and-white 1-bit pixel runs of various lengths (1 = black and 0 = white). A Group 3 encoder determines the length of a pixel run in a scan line and outputs a variable-length binary code word representing the length and color of the run. Because the code word output is shorter than the input, pixel data compression is achieved.

The run-length code words are taken from a predefined table of values representing runs of black or white pixels. This table is part of the T.4 specification and is used to encode and decode all Group 3 data.

The size of the code words were originally determined by the CCITT, based statistically on the average frequency of black-and-white runs occurring in typical type and handwritten documents. The documents included line art and were written in several different languages. Run lengths that occur more frequently are assigned smaller code words while run lengths that occur less frequently are assigned larger code words.

In printed and handwritten documents, short runs occur more frequently than long runs. Two- to 4-pixel black runs are the most frequent in occurrence. The maximum size of a run length is bounded by the maximum width of a Group 3 scan line.

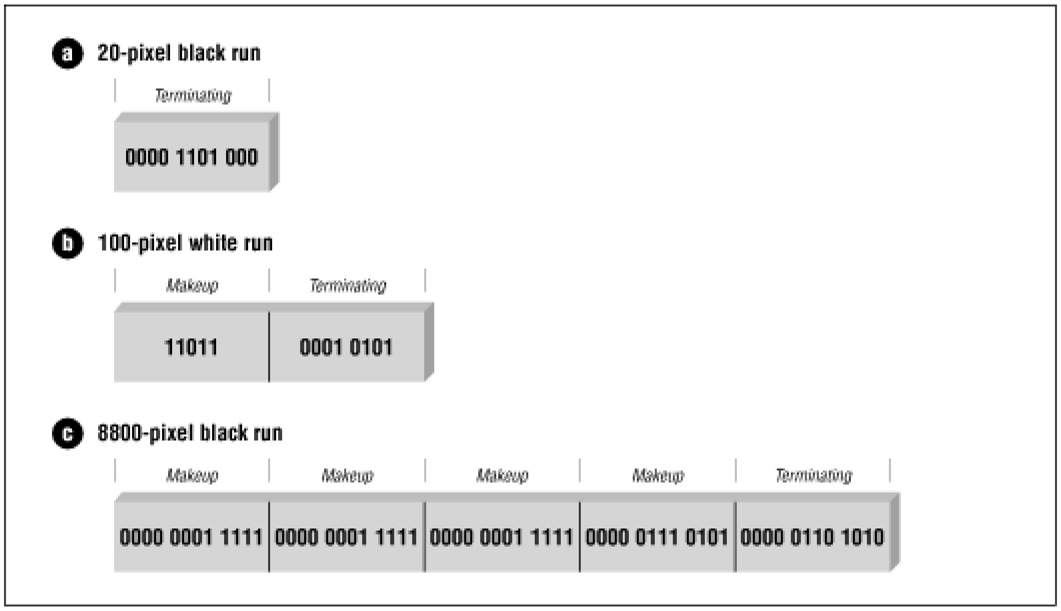
Run lengths are represented by two types of code words: *makeup* and *terminating*. An encoded pixel run is made up of zero or more makeup code words and a terminating code word. Terminating code words represent shorter runs, and makeup codes represent longer runs. There are separate terminating and makeup code words for both black and white runs.

Pixel runs with a length of 0 to 63 are encoded using a single terminating code. Runs of 64 to 2623 pixels are encoded by a single makeup code and a terminating code. Run lengths greater than 2623 pixels are encoded using one or more makeup codes and a terminating code. The run length is the sum of the length values represented by each code word.

Here are some examples of several different encoded runs:

- A run of 20 black pixels would be represented by the terminating code for a black run length of 20. This reduces a 20-bit run to the size of an 11-bit code word, a compression ratio of nearly 2:1. This is illustrated in Figure 9-7, a.
- A white run of 100 pixels would be encoded using the makeup code for a white run length of 64 pixels followed by the terminating code for a white run length of 36 pixels (64 + 36 = 100). This encoding reduces 100 bits to 13 bits, or a compression ratio of over 7:1. This is illustrated in Figure 9-7, b.
- A run of 8800 black pixels would be encoded as three makeup codes of 2560 black pixels (7680 pixels), a makeup code of 1088 black pixels, followed by the terminating code for 32 black pixels (2560 + 2560 + 2560 + 1088 + 32 = 8800). In this case, we will have encoded 8800 run-length bits into five code words with a total length of 61 bits, for an approximate compression ratio of 144:1. This is illustrated in Figure 9-7, c.

Figure 9-7: CCITT Group 3 encoding



The use of run lengths encoded with multiple makeup codes has become a de facto extension to Group 3, because such encoders are necessary for images with higher resolutions. And while most Group 3 decoders do support this extension, do not expect them to do so in all cases.

Decoding Group 3 data requires methods different from most other compression schemes. Because each code word varies in length, the encoded data stream must be read one bit at a time until a code word is recognized. This can be a slow and tedious process at best. To make this job easier, a state table can be used to process the encoded data one byte at a time. This is the quickest and most efficient way to implement a CCITT decoder.

All scan lines are encoded to begin with a white run-length code word (most document image scan lines begin with white run lengths). If an actual scan line begins with a black run, a zero-length white run-length code word will be prepended to the scan line.

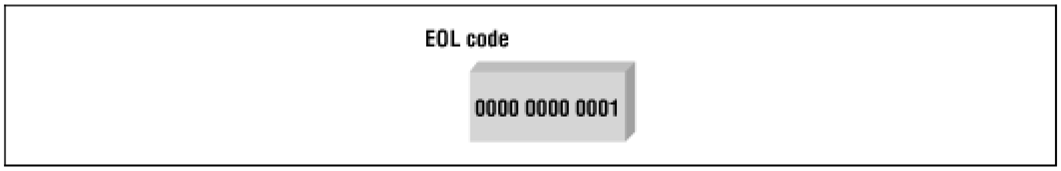
A decoder keeps track of the color of the run it is decoding. Comparing the current bit pattern to values in the opposite color bit table is wasteful. That is, if a black run is being decoded, there is no reason to check the table for white run-length codes.

Several special code words are also defined in a Group 3-encoded data stream. These codes are used to provide synchronization in the event that a phone transmission experiences a burst of noise. By recognizing this special code, a CCITT decoder may identify transmission errors and attempt to apply a recovery algorithm that approximates the lost data.

The EOL code is a 12-bit code word that begins each line in a Group 3 transmission. This unique code word is used to detect the start/end of a scan line during the image transmission. If a burst of noise temporarily corrupts the signal, a Group 3 decoder throws away the unrecognized data it receives until it encounteres an EOL code. The decoder would then start receiving the transmission as normal again, assuming that the data following the EOL is the beginning of the next scan line. The decoder might also replace the bad line with a predefined set of data, such as a white scan line.

A decoder also uses EOL codes for several purposes. It uses them to keep track of the width of a decoded scan line. (An incorrect scan-line width may be an error, or it may be an indication to pad with white pixels to the EOL.) In addition, it uses EOL codes to keep track of the number of scan lines in an image, in order to detect a short image. If it finds one, it pads the remaining length with scan lines of all white pixels. A Group 3 EOL code is illustrated in Figure 9-8.

Figure 9-8: CCITT Group 3 encoding (EOL code)

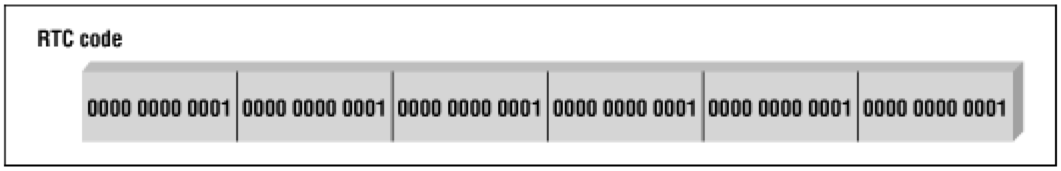


Most FAX machines transmit data of an "unlimited length," in which case the decoder cannot detect how long the image is supposed to be. Also, it is faster not to transmit the all-white space at the end of a page, and many FAX machines stop when they detect that the rest of a page is all white; they expect the receiver to do white padding to the end of the negotiated page size.

When Group 3 data is encapsulated in an image file, information regarding the length and width of the image is typically stored in the image file header and is read by the decoder prior to decoding.

Group 3 message transmissions are terminated by a return to control (RTC) code that is appended to the end of every Group 3 data stream and is used to indicate the end of the message transmission. An RTC code word is simply six EOL codes occurring consecutively. The RTC is actually part of the facsimile protocol and not part of the encoded message data. It is used to signal the receiver that it should drop the high-speed message carrier and listen on the low-speed carrier for the post-page command. A Group 3 RTC code is illustrated in Figure 9-9.

Figure 9-9: CCITT Group 3 encoding (RTC code)



A fill (FILL) is not actually a code word but a run of one or more zero bits that occurs between the encoded scan-line data and the EOL code (but never in the encoded scan line itself). Fill bits are used to pad out the length of an encoded scan line to increase the transmission time of the line to a required length. Fill bits may also be used to pad the RTC code word out to end on a byte boundary.

TIFF Compression Type 2

The TIFF compression Type 2 scheme (in which the compression tag value is equal to 2) is a variation of CCITT G31D encoding. The TIFF Type 3 and TIFF Type 4 compression methods follow exactly the CCITT Group 3 and Group 4 specifications, respectively. Type 2 compression, on the other hand, implements Group 3 encoding but does not use EOL or RTC code words. For this reason, TIFF Type 2 compression is also called "Group 3, No EOLs." Also, fill bits are never used except to pad out the last byte in a scan line to the next byte boundary.

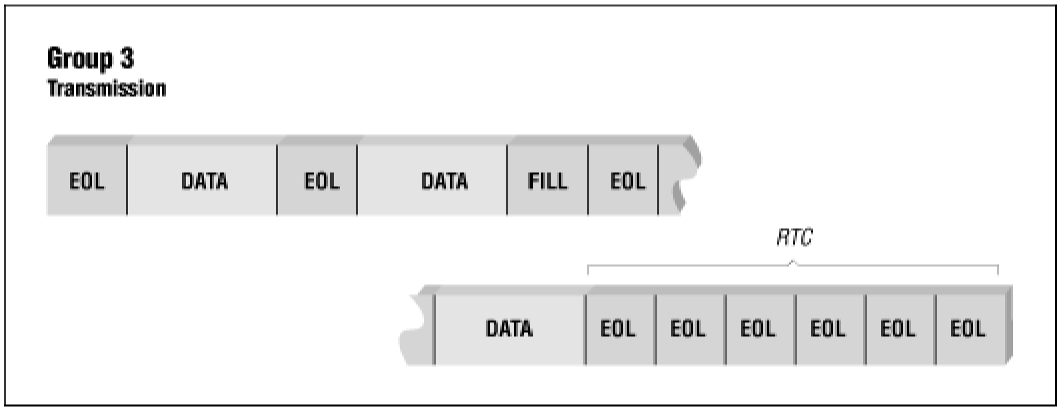
These modifications were incorporated into the TIFF specification because EOL and RTC codes are not needed to read data stored on tape or disk. A typical letter-size image (1728x2200 pixels) would contain 26,484 bits (3310.5 bytes) of EOL and RTC information. When storing Group 3 data to a file, the following are not needed:

- The initial 12-bit EOL
- The 12 EOL bits per scan line

- The 72 RTC bits tacked onto the end of each image

Conventional Group 3 decoders cannot handle these modifications and will either refuse to read the TIFF Type 2-encoded data or will simply return a stream of decoded garbage. However, many decoders have been designed to accept these Group 3 trivial "modifications" and have no problems reading this type of data. Group 3 encoding is illustrated in Figure 9-10.

Figure 9-10: Group 3 CCITT encoding (TIFF Compression Type 2)



TIFF Class F

There is nearly one facsimile file format for every brand of computer FAX hardware and software made. Many compress the facsimile data using RLE (presumably so it will be quicker to display) or store it in its original Group 3 encoding. Perhaps the most widely used FAX file format is the unofficial TIFF Class F format. (See the TIFF article for more information about TIFF Class F.)

Even with the latest release of TIFF, revision 6.0, Class F has never been officially included in the standard, despite the wishes of the TIFF Advisory Council. The reason for this is that Aldus feels that supporting applications that require facsimile data storage and retrieval is outside of the scope of TIFF. (TIFF was designed primarily with scanners and desktop publishing in mind.) This is too bad, considering that one of TIFF's main goals is to aid in promoting image data interchangeability between hardware platforms and software applications.

Group 3 Two-Dimensional (G32D)

Group 3 One-Dimensional (G31D) encoding, which we've discussed above, encodes each scan line independent of the other scan lines. Only one run length at a time is considered during the encoding and decoding process. The data occurring before and after each run length is not important to the encoding step; only the data occurring in the present run is needed.

With Group 3 Two-Dimensional (G32D) encoding, on the other hand, the way a scan line is encoded may depend on the immediately preceding scan-line data. Many images have a high degree of vertical coherence (redundancy). By describing the differences between two scan lines, rather than describing the scan line contents, 2D encoding achieves better compression.

The first pixel of each run length is called a *changing element*. Each changing element marks a color transition within a scan line (the point where a run of one color ends and a run of the next color begins).

The position of each changing element in a scan line is described as being a certain number of pixels from a changing element in the current, coding line (horizontal coding is performed) or in the preceding, reference line (vertical coding is performed). The output codes used to describe the actual positional information are called Relative Element Address Designate (READ) codes.

Shorter code words are used to describe the color transitions that are less than four pixels away from each other on the code line or the reference line. Longer code words are used to describe color transitions lying a greater distance from the current changing element.

2D encoding is more efficient than 1-dimensional because the usual data that is compressed (typed or handwritten documents) contains a high amount of 2D coherence.

Because a G32D-encoded scan line is dependent on the correctness of the preceding scan line, an error, such as a burst of line noise, can affect multiple, 2-dimensionally encoded scan lines. If a transmission error corrupts a segment of encoded scan line data, that line cannot be decoded. But, worse still, all scan lines occurring after it also decode improperly.

To minimize the damage created by noise, G32D uses a variable called a K factor and 2-dimensionally encodes K-1 lines following a 1-dimensionally encoded line. If corruption of the data transmission occurs, only K-1 scan lines of data will be lost. The decoder will be able to resync the decoding at the next available EOL code.

The typical value for K is 2 or 4. G32D data that is encoded with a K value of 4 appears as a single block of data. Each block contains three lines of 2D scan-line data followed by a scan line of 1-dimensionally encoded data.

The K variable is not normally used in decoding the G32D data. Instead, the EOL code is modified to indicate the algorithm used to encode the line following it. If a 1 bit is appended to the EOL code, the line following is 1-dimensionally encoded; if a 0 bit is appended, the line following the EOL code is 2-dimensionally encoded. All other transmission code word markers (FILL and RTC) follow the same rule as in G31D encoding. K is only needed in decoding if regeneration of the previous 1-dimensionally encoded scan line is necessary for error recovery.

Group 4 Two-Dimensional (G42D)

Group 4 Two-Dimensional (G42D) encoding was developed from the G32D algorithm as a better 2D compression scheme--so much better, in fact, that Group 4 has almost completely replaced G32D in commercial use.

Group 4 encoding is identical to G32D encoding except for a few modifications. Group 4 is basically the G32D algorithm with no EOL codes and a K variable set to infinity. Group 4 was designed specifically to encode data residing on disk drives and data networks. The built-in transmission error detection/correction found in Group 3 is therefore not needed by Group 4 data.

The first reference line in Group 4 encoding is an imaginary scan line containing all white pixels. In G32D encoding, the first reference line is the first scan line of the image. In Group 4 encoding, the RTC code word is replaced by an end of facsimile block (EOFB) code, which consists of two consecutive Group 3 EOL code words. Like the Group 3 RTC, the EOFB is also part of the transmission protocol and not actually part of the image data. Also, Group 4-encoded image data may be padded out with fill bits after the EOFB to end on a byte boundary.

Group 4 encoding will usually result in an image compressed twice as small as if it were done with G31D encoding. The main tradeoff is that Group 4 encoding is more complex and requires more time to perform. When implemented in hardware, however, the difference in execution speed between the Group 3 and Group 4 algorithms is not significant, which usually makes Group 4 a better choice in most imaging system implementations.

Tips for Designing CCITT Encoders and Decoders

Here are some general guidelines to follow if you are using the CCITT encoding method to encode or decode.

- Ignore bits occurring after the RTC or EOFB markers. These markers indicate the end of the image data, and all bits occurring after them can be considered filler.

- You must know the number of pixels in a scan line before decoding. Any row that decodes to fewer or greater pixels than expected is normally considered corrupt, and further decoding of the image block (2D encoding only) should not be attempted. Some encoding schemes will produce short scan lines if the line contains all white pixels. The decoder is expected to realize this and pad out the entire line as a single white run.

- Be aware that decoded scan-line widths will not always be a multiple of eight, and decoders should not expect byte-boundary padding to always occur. Robust decoders should be able to read non-byte-aligned data.

If a decoder encounters an RTC or EOFB marker before the expected number of scan lines has been decoded, assume that the remaining scan lines are all white. If the expected number of scan lines has been decoded, but an RTC or EOFB has not been encountered, stop decoding. A decoder should then produce a warning that an unusual condition has been detected.

- Note that a well-designed CCITT decoder should be able to handle the typical color-sex and bit-sex problems associated with 1-bit data, as described below:

  The CCITT defines a pixel value of 0 as white and a pixel value of 1 as black. Many bitmaps, however, may be stored using the opposite pixel color values and the decoder would interpret a "negative" of the image in this case (a color-sex problem). Different machine architectures also store the bits within a byte in different ways. Some store the most significant bit first, and some store the least significant bit first. If bitmap data is read in the opposite format from the one in which it was stored, the image will appear fragmented and disrupted (a bit-sex problem). To prevent these problems, always design a CCITT decoder that is capable of reading data using either of the color-sex and bit-sex schemes to suit the requirements of the user.

For Further Information About CCITT

The original specifications for the CCITT Group 3 and Group 4 algorithms are in CCITT (1985) Volume VII, Fascicle VII.3, Recommendations T.4 and T.6:

> "Standardization of Group 3 Facsimile Apparatus for Document Transmission," *Recommendation T.4, Volume VII, Fascicle VII.3, Terminal Equipment and Protocols for Telematic Services*, The International Telegraph and Telephone Consultative Committee (CCITT), Geneva, Switzerland, 1985, pp. 16-31.

> "Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus," *Recommendation T.6, Volume VII, Fascicle VII.3, Terminal Equipment and Protocols for Telematic Services*, The International Telegraph and Telephone Consultative Committee (CCITT), Geneva, Switzerland, 1985, pp. 40-48.

The latest specification is CCITT (1992) Volume VII, Fascicle VII.3, Recommendations T.0 through T.63.

Both the CCITT and ANSI documents may be obtained from the following source:

American National Standards Institute, Inc.
Attn: Sales
1430 Broadway
New York, NY 10018 USA
Voice: 212-642-4900

See also the following references. (For information on getting RFCs [Requests for Comments], send email to *rfc-info@isi.edu* with a subject line of "getting rfcs" and a body of "help: ways _to_get_rfcs".)

> Hunter, R. and A.H. Robinson, "International Digital Facsimile Coding Standards," *Proceedings of the IEEE*, vol. 68, no. 7, July 1980, pp. 854-867.

> *RFC 804--CCITT Draft Recommendation T.4 (Standardization of Group 3 Facsimile Apparatus for Document Transmission)*.

> *RFC 1314--A File Format for the Exchange of Images in the Internet*.

> *FAX: Digital Facsimile Technology & Applications*, McConnell, Artech House, Norwood, MA; second edition, 1992.

> Marking, Michael P., "Decoding Group 3 Images," *The C Users Journal*, June 1990, pp. 45-54.

Information on MMR encoding may be found in the following references:

> "Binary-image-manipulation Algorithms in the Image View Facility," *IBM Journal of Research and Development*, vol. 31, no. 1, January 1987.

Information on Huffman encoding may be found in the following references:

Huffman, David, "A Method for the Construction of Minimum Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 9, 1952, pp. 1098-1101.

Kruger, Anton, "Huffman Data Compression," *C Gazette*, vol. 5, no. 4, 1991, pp.71-77.

[Previous] [Next]

Feedback Printer friendly

Terms of Service | Privacy Policy | Contact Info