JBIG2 - THE ULTIMATE BI-LEVEL IMAGE CODING STANDARD

Fumitaka Ono (Tokyo Institute of Polytechnics, JAPAN)
William Rucklidge (Intelligent Markets, USA)
Ronald Arps (IBM Almaden Research Center, USA)
Cornel Constantinescu (IBM Almaden Research Center, USA)

ABSTRACT

JBIG2 is a new standard for lossy and lossless compression of bi-level images. It exploits model-based coding for text and halftones; as well as nearby neighbor based coding for generic bi-level images. It can achieve compression ratios of up to three times those of existing standards for lossless textual image compression. JBIG2 also allows loss to be introduced while maintaining "visually lossless" quality for textual images, for which it can yield compression ratios of up to eight times those of existing standards. It produces even greater compression ratios when exploiting similarities between multiple pages and efficiently codes and integrates both scanned and generated bi-level images. JBIG2 was developed for applications having widely differing requirements such as facsimile, document storage and archiving, Web image coding, wireless data transmission, print spooling, and even teleconferencing. Accordingly, JBIG2 is structured as a "toolkit" of alternate capabilities to be selected based on the requirements of such applications

1. INTRODUCTION

The Joint Bi-level Image Experts Group (JBIG), an international study group affiliated with ISO/IEC JTC 1/SC29/WG1 and ITU-T SG8, has developed a new standard, informally referred to as JBIG2, for lossy, lossless, and lossy-to-lossless compression of bilevel (two-color) images. Compression of this type of image data is also addressed by existing facsimile standards, such as ITU-T Recommendations T.4 (MH, MR), T.6 (MMR, commonly called "Group 4"), and T.82|ISO/IEC 11544 (called JBIG or JBIG1).

JBIG1, which was the first coding standard produced by the JBIG committee in 1993, provided for lossless and progressive (lossy-to-lossless) coding. Though it also has a capability for lossy coding, the lossy images produced by JBIG1 have significantly lower quality than their original images because the number of pixels in the lossy image cannot exceed one quarter of those in the original image.

In contrast, JBIG2 was explicitly prepared for lossy, lossless, and lossy-to-lossless image compression. Its design goals were to attain lossless compression performance better than that of the existing standards, and to include lossy compression with almost no visible degradation of quality and much higher compression ratios than the lossless ratios obtained with existing standards.

JBIG2 achieved these goals and added a capability for content-progressive coding, (successively adding different types of image data, for example, first text and then halftones). Such content-based decomposition is very desirable especially in interactive multimedia applications. JBIG2 completed formal standardization this year as ISO/IEC Standard 14492 and ITU-T Recommendation T.88.

2. TECHNICAL FEATURES OF JBIG2

2.1. Overview

A typical JBIG2 encoder decomposes the input bi-level image into several regions and encodes each of the regions separately using a different coding method. Figure 1 shows the block diagram of a JBIG2 decoder. In this figure, thick lines and arrows show the procedures that decode data, and how these procedures invoke one another. Thin lines and arrows show the data storage elements used by the decoder, and how decoded data flows through the system. Not shown on the figure is the control decoding procedure, which controls the flow of encoded data.

JBIG2 utilizes model based coding to take advantage of our knowledge that typical bi-level images contain significant amounts of textual and halftone image data, and that in textual and halftone data, the same basic shapes appear repeatedly. In the model, no assumptions about particular character sets (Latin, Kanji, etc.) or about particular halftone types (periodic

0-7803-6297-7/00/$10.00 © 2000 IEEE

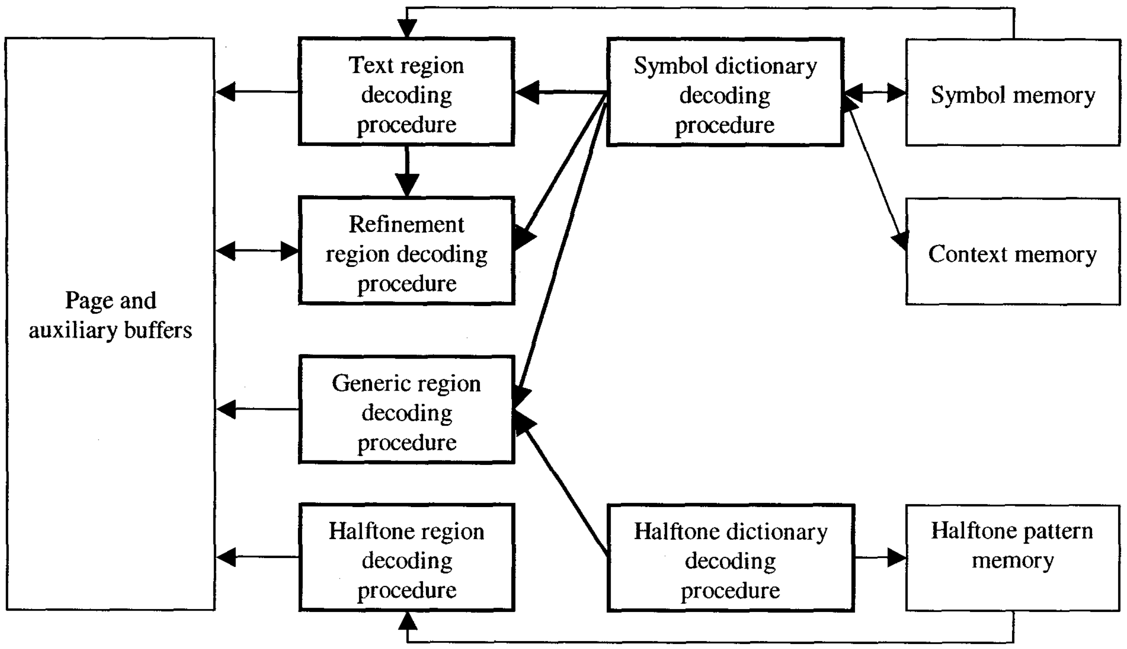
Page and auxiliary buffers
Text region decoding procedure
Refinement region decoding procedure
Generic region decoding procedure
Halftone region decoding procedure
Symbol dictionary decoding procedure
Symbol memory
Context memory
Halftone dictionary decoding procedure
Halftone pattern memory

Figure 1: Components of a JBIG2 Decoder

dither, error diffusion, etc.) are necessary. This provides flexible and robust compression performance over a wide variety of unanticipated data. JBIG2 has several coding modes, which are designed for these typical types of data found in bi-level images. Textual image data is coded by pattern matching and substitution [1], with possibly one or more additional "refinement" steps [2]. Other bi-level image data such as line art is coded using a pure bitmap "generic" coder. Halftone data may also be coded by pattern matching and substitution using pseudo-characters, or with the generic coder.

One of the functions in a maximally efficient system using JBIG2 is to segment a page into text, halftone and generic regions. Such segmentation techniques are not described in the normative part of the standard, because they are preprocessing decisions that don't affect compatibility with the specification. JBIG2 also has a capability that facilitates efficient multiple-page processing by allowing the decoder to make use of information, such as the dictionaries of character shapes, gathered from other pages. For entropy coding, all of the above methods can use either arithmetic coding (the MQ-coder) or Huffman coding (including MMR coding of bitmaps). JBIG2 also facilitates transformation of entire lossy images into lossless ones; by providing an image refinement capability; it also provides a character-by-character refinement capability.

JBIG2 allows lossy compression due to pre-processing of the original image, which may consist of flyspeck noise removal and text filtering to still be visually lossless [3], or even complex halftone image processing algorithms based on rate-distortion theory [4]. JBIG2 also permits the lossy compression that results from pattern matching and substitution in the encoder. The permissible level of such losses is not specified in the standard. The decoder is only guaranteed to be lossless with respect to information provided to the entropy encoders, but not necessarily with respect to the original image. However, if desired, lossy image representations may be followed by refinement data to decrease loss in any amount, including perfect reconstruction of the original image.

2.2. Text coding

In bi-level textual data, two instances of the same characters may not exactly match pixel by pixel in a scanned image. But they are usually close enough that a human observer can tell that they are the same. So instead of coding all the individual bitmaps of each occurrence of the same character, we code the bitmap of one representative instance (or a synthetic instance, usually built by averaging) of that character and put it into a "symbol dictionary." Then for each character instance on the page, we code both an index of the corresponding representative bitmap in the dictionary, and the position of the character instance, usually relative to another previously coded character instance [1]. The bitmaps in the dictionary, the indices and the position information may be coded using either context-based arithmetic coding or Huffman coding.

The method of "pattern matching and substitution" allows excellent compression, but substitution errors are possible, especially at low resolutions. For cases where such errors are unacceptable JBIG2 supports a technique called soft pattern matching [3], which combines judicious preprocessing with efficient lossless restoration of all matched characters. In soft pattern matching, in addition to an index of the dictionary and position information, refinement data is also sent to restore the current character instance as was produced after careful preprocessing. This refinement data is the coded data of the current character instance, using both the causal pixels of current character instance and the entire bitmap of the matched character in the dictionary. Since the current character instance is highly correlated with the matched character (after all, that is the basis for the declaration of a match) the resulting predictions and compression factors are excellent for this visually lossless result..

Coding bitmaps for the characters in a dictionary is done either by a nearby neighbor based Markov model using arithmetic coding, similar to JBIG1, or by MMR coding as specified in ITU-T T.6. The main differences between the JBIG2 arithmetic bitmap coder and that of JBIG1 are that that the arithmetic coder is slightly different (MQ rather than QM), and that the number of pixels in the template can be larger (10, 13 or 16 pixels rather than 10 pixels). The character refinement coding required by soft pattern matching must use arithmetic coding.

Since most characters appear on every page of a document, JBIG2 allows symbol dictionaries to be used by multiple pages.

2.3 Halftone coding

In JBIG2, halftones can be compressed in two ways. The first uses the arithmetic form of generic coding, exploiting adaptively positioned template pixels to capture correlations between adjacent halftone dots [4]. The MMR form of generic coding can also be used but it does not exploit such correlations and its compression will not be as good. The second method involves descreening the halftone image (converting it back to grayscale) and encoding the grayscale values. The grayscale values are used as indices into a halftone pattern dictionary, containing fixed-size tiny images representing the indicated grayscale values in halftone form. The decoder can then render a halftone image without needing explicit position or pattern size information, by simply drawing indexed dictionary images in accordance with a sequence of grayscale values at locations specified by the halftone grid.

For either method of coding halftoned images it may make sense to post-process the image in order to make it more visually acceptable, perhaps by tuning it for a particular output device. Such post-processing is outside of the scope of the standard.

2.4 Generic and refinement coding for image regions

The same selection of Huffman and arithmetic methods used for coding dictionary bitmaps or patterns are also used for coding the generic parts of a bi-level image that do not utilize the textual or halftone coding models. Similarly, JBIG2 provides for refinement of a lossy image region using the same coding method as used in character refinement. The lossy image is refined using a two-plane bitmap coder, making use of information from pixels in the lossy image already sent and causal pixels from the target (usually lossless) image region to be reconstructed. Again, only arithmetic coding is used for the region refinement coding method.

2.5 Arithmetic entropy coding

JBIG2 uses the MQ coder for its arithmetic coder. The MQ coder is an adaptive binary arithmetic coder, which is characterized by multiplier-free approximation and renormalization-driven update of probability estimator, and bit-stuffing introduced by the Q-coder [5], enhanced by the conditional exchange derived from MELCODE [6], and the state transition table known as JPEG-FA [7].

At any given point in the general arithmetic coding process, the string of symbols which have been observed so far, is mapped to a unique sub-interval, [c, c+a] represented by

$$c=C\cdot 2^{-M-N} \quad \text{and} \quad a=A\cdot 2^{-M-N}$$

where C and A are integers and N is the total number of normalization shifts which have been employed to ensure that $2^{M-1} <= A < 2^{M}$, where M is the width of the A register.

Upon completion, the compressed bit-stream is C, but C can be terminated to less than or equal to N+2 bits, depending on encoder and decoder termination rules.

In the multiplier-free approximation, the interval width after LPS (Less Probable Symbol) is approximated to $A_L = b\cdot p$ instead of $a\cdot p$, where b is about $0.71\cdot 2^{M}$, and p is the estimated probability. The interval width after MPS (More Probable Symbol) is approximated to $A_M = A - b\cdot p$

Document	Pages	MMR	JBIG1	Lossless JBIG2	Lossy JBIG2
F04_300	1	95,879	77,642	73,422	14,234
Technical report	23	1,260,357	926,229	842,918	184,470
Book	512	45,719,356	34,674,283	2,633,977	N/A

Table 1: JBIG2 Performance

The adoption of conditional exchange improves the approximation of the interval length, and the adoption of JPEG-FA reduces the learning time.

Carry propagation is a well-known common problem of arithmetic coders and in the MQ coder; it is solved by a bit-stuffing process.

2.6. Profiles

Since JBIG2 is a tool-kit for various applications, it is expected that different applications will use different subsets, or profiles, of JBIG2. The standard currently provides seven such profiles. Two of these profiles, designed for low-memory applications, are intended to be basic starting points, suitable for many applications.

3. PERFORMANCE

This section shows some typical numbers using JBIG2 coding, on three sample documents. All file sizes are given in bytes.

The first document is one of the sample pages used to test MMR coding. It is scanned at a resolution of 300dpi. Lossy JBIG2 clearly outperforms MMR and JBIG1, while lossless JBIG2 is somewhat better than JBIG1. The second document is a 23-page scanned technical report, scanned at 600dpi. For this document, lossless JBIG2 is significantly smaller than JBIG1, while lossy JBIG2 is five times smaller than JBIG1. The third document is a 512-page book whose page images were produced by a PostScript interpreter, and thus contain no scanner noise. For such generated documents, pattern matching and substitution offers lossless coding without any refinement coding being necessary. Also, for such a long document, the savings due to sharing symbol dictionary between pages are enormous. Lossless JBIG2 thus outperforms MMR and JBIG1 by a large factor.

4. CONCLUSION

The JBIG2 specification was approved this year as an International Standard in ISO/IEC and a formal Recommendation in ITU-T. Its pattern matching and substitution foundation is considered to be the most efficient coding method for textual image documents in the current art. For generic coding, the optimum number of contexts in template-based coding can be matched, in accordance with the number of pixels to be coded in a page. Compared to JBIG1 we can now better adaptively compress very high resolution documents, by using up to 16 nearby neighbor pixels in the compression models. The ability of JBIG2 to exploit the repetition of shapes across different pages offers unprecedented compression.

The toolkit design of JBIG2 allows for the most appropriate coding method to be selected for different regions of each page in a document. Also, differing applications can choose different parts of the JBIG2 toolkit to either (1) achieve the best possible compression using the arithmetic coding options, or (2) achieve extremely high decoding speeds (over 1 gigapixel per second in software) using the Huffman and MMR coding options. We believe that JBIG2 can now provide the best compression performance or fastest decoding speeds for all applications involving scanned or generated bi-level images.

5. REFERENCES

1] R.N. Ascher & G. Nagy, "A Means for Achieving a High Degree of Compaction on Scan-Digitized Printed Text," IEEE Trans. on Computers, C-23, pp.1174-1179, Nov. 1974.

[2] K. Mohiuddin, J.J. Rissanen, & R. Arps, "Lossless Binary Image Compression Based on Pattern Matching", Proceedings of International Conference on Computers, Systems, and Signal Processing, Bangalore, India, pp.447-4513, 1984.

[3] P.G. Howard, "Text Image Compression Using Soft Pattern Matching", Computer Journal, 40:2, 1997.

[4] B. Martins & S. Forchhammer, "Lossless/Lossy Compression of Bi-level Images", Proceedings of IS&T/SPIE Symposium on Electronic Imaging: Science and Technology, SPIE,3018, pp.38-49, 1997.

[5] W. Pennebaker, J. Mitchell, G. Langdon, and R. Arps "An overview of the basic principles of the Q-coder- adaptive binary arithmetic coder," IBM J. Res. Develop., Vol.36, pp.717-726, Nov. 1988

[6] F. Ono et al, "Bi-level Image Coding with MELCODE-Comparison of Block Type code and Arithmetic Type Codes" 7.6.1-6, Globecom 89, Nov. 1989

[7] W.Pennebaker, and J. Mitchell, "JPEG: still image data compression standard," Van Nostrand Reinhold, NY, 1992.