

# THE EMERGING JBIG2 STANDARD

Paul G. Howard<sup>1</sup>, Faouzi Kossentini<sup>2</sup>,  
Bo Martins<sup>3</sup>, Søren Forchhammer<sup>4</sup>, William J. Rucklidge<sup>5</sup>,  
Fumitaka Ono<sup>6</sup>

## ABSTRACT

The Joint Bi-level Image Experts Group (JBIG), an international study group affiliated with ISO/IEC and ITU - T, is in the process of drafting a new standard for lossy and lossless compression of bi-level images. The new standard, informally referred to as JBIG2, will support model-based coding for text and halftones to permit compression ratios up to three times those of existing standards for lossless compression. JBIG2 will also permit lossy preprocessing without specifying how it is to be done. In this case compression ratios up to eight times those of existing standards may be obtained with imperceptible loss of quality. It is expected that JBIG2 will become an International Standard by 2000.

## 1 Introduction

JBIG2 is an emerging ISO/IEC International Standard for lossy and lossless bi-level image compression. It is being drafted by the Joint Bi-level Image Experts Group (JBIG), a “Collaborative Team” that reports both to ISO/IEC JTC 1 / SC 29 / WG 1<sup>7</sup> and to ITU - T SG 8<sup>8</sup>. As the result of a process that ended in 1993, JBIG as a “Collaborative Interchange” produced

---

<sup>1</sup>JBIG2 Editor, AT&T Labs - Research, Red Bank, NJ, [pgh@research.att.com](mailto:pgh@research.att.com)

<sup>2</sup>University of British Columbia, Vancouver, BC, [faouzi@ece.ubc.ca](mailto:faouzi@ece.ubc.ca)

<sup>3</sup>Technical University of Denmark, Lyngby, Denmark, [bm@tele.dtu.dk](mailto:bm@tele.dtu.dk)

<sup>4</sup>Technical University of Denmark, Lyngby, Denmark, [sf@tele.dtu.dk](mailto:sf@tele.dtu.dk)

<sup>5</sup>JBIG2 Editor, Xerox Corp., Palo Alto, CA, [rucklidge@parc.xerox.com](mailto:rucklidge@parc.xerox.com)

<sup>6</sup>JBIG Rapporteur, Mitsubishi Electric Corp., Kanagawa, Japan, [ono@isl.melco.co.jp](mailto:ono@isl.melco.co.jp)

<sup>7</sup>ISO is the International Organization for Standardization. IEC is the International Electrotechnical Commission. JTC 1 is the Joint ISO/IEC Technical Committee on information technology. SC 29 is the subcommittee responsible for coding of audio, picture, multimedia and hypermedia information. WG 1 is the working group that deals with coding of still pictures; it includes both JBIG and JPEG, the Joint Photographic Experts Group.

<sup>8</sup>ITU-T is the Telecommunication Standardization Sector of the International Telecommunication Union. SG 8 is the study group that deals with characteristics of telematic systems.

a bi-level image coding standard formally designated ITU - T Recommendation T.82 | International Standard ISO/IEC 11544, and informally known as JBIG or JBIG1. The authors of this paper are all active members of JBIG, although among us only Dr. Ono was involved in JBIG1.

The JBIG2 standard will define a compression method for bi-level images, that is, images consisting of a single rectangular bit plane, with each pixel taking on one of just two possible colors. Compression of this type of image is addressed by existing facsimile standards [1], in particular by ITU - T Recommendations T.4, T.6, T.82 (JBIG1) and T.85<sup>9</sup>. Besides the obvious facsimile application, JBIG2 will be useful for document storage and archiving, images on the World Wide Web, wireless data transmission, print spooling, and teleconferencing.

JBIG2 will be the first international standard that provides for lossy compression of bi-level images; the existing standards are strictly lossless. Indeed, lossy compression is one important reason that JBIG2 is being drafted. The design goal for JBIG2 is to enable lossless compression performance better than that of the existing standards, and to enable lossy compression at much higher compression ratios than the lossless ratios of the existing standards, with almost no degradation of quality. In addition, JBIG2 will allow both *quality-progressive* coding through refinement stages, with the progression going from lower to higher (or lossless) quality, and *content-progressive* coding, successively adding different types of image data (for example, first text, then halftones). In fact, a typical JBIG2 encoder decomposes the input bi-level image into several regions or image segments (usually based on content), and each of the image segments is separately coded using a different coding method. Such content-based decomposition is very desirable in interactive multimedia applications.

The applications for which JBIG2 will be useful have widely differing requirements. For example, low-end facsimile requires high coding speed and low complexity even at the cost of some loss of compression, while wireless transmission needs maximum compression to make fullest use of its narrow channel. In recognition of the variety of application needs, JBIG2 will not have a baseline implementation. Instead, it will provide a toolkit of alternative standardized mechanisms to be selected and used based on application requirements. Typically there will be two mechanisms for each function, one providing high speed and good compression, the other providing high compression and reasonable speed. In addition, there will be a number of application profiles that specify the recommended or required mechanisms and parameters for specific applications.

As is typical with image compression standards, the JBIG2 standard will explicitly define

---

<sup>9</sup>See the Appendix for a summary of the confusing nomenclature of facsimile standards.

the requirements of a compliant bitstream, and will thus implicitly define decoder behavior. The standard will not explicitly define a standard encoder, but instead will be flexible enough to allow sophisticated encoder design. In fact, encoder design will be a major differentiator among competing JBIG2 implementations.

## 2 Technical description

Although the JBIG2 standard is not yet final, many of its technical specifications have become clear. It will have a control structure that allows efficient encoding of multipage documents in sequential or random-access mode, or embedded in another file format. It will utilize pattern matching techniques to allow good compression of text and some types of halftone images, and it will allow refinement of lossy images either to less lossy images or to lossless images. The introduction of loss will be an encoder issue, outside the scope of the standard. In this section we present details of the standard as we see it now; there will likely be changes before it becomes an International Standard.

### 2.1 Headers and control

A JBIG2 file describes a document that consists of one or more bi-level page images. It may be created and read in sequential or random-access mode. In sequential mode, used for streaming applications like facsimile, it is expected that the decoder will interpret all pages in order. In random-access mode, used for applications like document archiving, it will be possible to access and interpret only the pages desired, in any order.

JBIG2 will have a control structure that facilitates efficient multiple-page processing by allowing the decoder, while decoding a page, to make use of information gathered from other pages. JBIG2 files consist of two types of *data segments*. Image data segments are those that describe how the page should appear. Dictionary data segments describe the elements that make up the page, such as the bitmaps of text characters. There are also several types of control data segments, containing information such as page descriptors, page striping, Huffman tables, and so on. The dependencies between different segments (usually between image segments and dictionary segments) are expressed succinctly in segment headers, one associated with each segment. A segment's header also indicates the segment's type, the page, if any, to which it belongs, and the length of the data part of the segment. In sequential mode, each segment header appears just before its associated data segment. In random-access mode, the segment headers are gathered at the beginning of the file, allowing the decoder to

construct the full dependency graph during initialization.

The data segments defined by JBIG2 will also be able to be embedded in other file formats, such as SPIFF<sup>10</sup>, TIFF<sup>11</sup>, and MRC<sup>12</sup>. When used in this fashion, the JBIG2 segment headers and data segments will be treated as data within the wrapper file format.

One important additional aspect of JBIG2 is its ability to represent multiple pages in a single file. This captures the structure of most documents, and can also be used to improve compression. A character that appears on the first page of a document is likely to appear on other pages; JBIG2 exploits this by allowing dictionaries to be referred to by multiple pages. Thus, the incremental cost of coding additional pages is reduced because of the dictionaries generated for previous pages; this can increase the compression by a factor of two over compressing each page independently.

## 2.2 Cleanup and refinement coding

We anticipate that one of the functions of a JBIG2 encoder will be to segment a page into different classes of image data, in particular textual and halftone data. Some data, such as line art data, may not be identified with one of the standard classes. Such data will be coded by a *cleanup coder*, essentially a basic bitmap coder like JBIG1 or one of the other ITU-T fax coders.

We also provide for the transformation of a lossy character or page image into a less lossy or possibly lossless one. This will be done using *refinement coding*: the image or character will be re-encoded using a two-plane bitmap coder, making use of previously coded information in both the current image and the previously coded lossy image [2]. Such coding may be used more than once to successively refine a character or page image. Refinement coding back to the original lossless image is called *residue coding*. As in JBIG1, it may be possible to obtain faster processing and improved compression by applying *typical prediction* during refinement coding [3].

Textual data will be coded by pattern matching and substitution, possibly with an additional refinement step. Halftone data will be coded by pattern matching and substitution, possibly with refinement, with the patterns corresponding to grayscale values; alternatively, it may be possible to combine halftone data with other data in the cleanup coder, although this may reduce efficiency.

---

<sup>10</sup>SPIFF, the Still Picture Interchange File Format, formally known as ITU-T Recommendation T.84 | International Standard ISO/IEC 10918-4, is the ISO/IEC JTC 1 / SC 29 / WG 1 standardized file format.

<sup>11</sup>TIFF, the Tagged Image File Format, is a trademark of Aldus Corporation.

<sup>12</sup>MRC, a standard for color images with “mixed raster content”, is ITU-T Recommendation T.44.

## 2.3 Model-based coding

It is widely known that the best compression results from using a model of the data that closely matches the structure of the data itself [4]. The earlier facsimile standards use simple models of the structure of bi-level images [1]. ITU-T Recommendations T.4 and T.6 treat each image scan line as a sequence of runs of black and white pixels. Since the strokes in individual characters in a text image are usually several pixels wide and separated by a number of pixels of white space, the run-length model used in MH, MR, and MMR coding<sup>13</sup> provides reasonable compression for images consisting mostly of text. By further taking advantage of the strong correlation between adjacent lines, the two-dimensional run-length model used in MR and MMR coding provides fairly good compression for text images.

The JBIG1 standard treats each pixel as being predicted by some nearby neighbors in positions defined by a fixed template, possibly including a single pixel (the adaptive pixel) whose position is variable. This model captures some of the structure of individual characters, and the adaptive pixel significantly improves performance on periodic halftone images by providing a simple model of the halftone structure. Both the run-length model and the predictive context-based model are very general, and do not directly make use of the textual or halftone nature of the image; however, the price of the generality is to limit the amount of compression possible for specific classes of images.

In JBIG2, we take advantage of our knowledge that typical bi-level images consist mainly of textual and halftone data, and we allow the use of models designed specifically for those data types. JBIG2 is font independent and makes no a priori assumptions about particular character sets (latin, kanji, etc.) or about particular halftone types (periodic dither, error diffusion, etc.). JBIG2 derives representative bitmaps for all patterns within each page, and is thus more general and more robust than OCR or compression methods that utilize font dictionaries.

## 2.4 Pattern matching for text image data

For textual images, we use character based pattern matching techniques [5]. We note that on a typical page of text there are many repeated characters. Therefore, instead of coding all the pixels of each occurrence of each character, we code the bitmap of one representative instance of the character and put it into a “dictionary.” Several names have been used to refer to such bitmaps, including *symbol*, *mark* and *pixel block*. Henceforth, we will refer to the bitmap as a

---

<sup>13</sup>MH, MR, and MMR coding are described in the Appendix.

pixel block.

In this section, we briefly present two encoding methods, *pattern matching and substitution* (PM&S) and *soft pattern matching* (SPM). These methods differ substantially in how they encode pixel blocks. The flexibility of JBIG2 allows the same bitstream format to be used to represent the output of an encoder using either method. JBIG2’s capabilities also allow encoders that are hybrids of these two methods, or that use other encoding methods entirely.

### 2.4.1 Pattern matching and substitution

In a scanned image, two instances of the same character do not match pixel for pixel, but they are certainly close enough that a human observer can see that they are the same. Thus, for each character on the page, we code both a pointer to the corresponding representative bitmap in the dictionary, and the position of the character on the page, usually relative to another previously coded character. If there is no acceptable match, we code the pixel block directly and add it to the dictionary.

Figure 1 shows the block diagram of a typical encoding procedure using PM&S, which involves the following steps: 1) segmentation of the image into pixel blocks, 2) searching for a match in the dictionary, and 3) coding of the associated numerical data if a “good” match is found, or 4) coding of the corresponding bitmap otherwise. These steps are discussed next.

It should be emphasized that a JBIG2 bitstream does not interleave the numerical and bitmap (dictionary) data as Figure 1 implies: what the encoder actually produces is one or more dictionary segments containing the pixel block bitmaps, and one or more image data segments containing the numerical information on where those bitmaps should be drawn to reconstruct the page.

**Segmentation.** The bi-level image is segmented into pixel blocks containing connected black pixels using any standard segmentation technique [6]. Features (e.g., height, width, area, position) for each pixel block are then extracted.

**Dictionary search.** Searching a previously coded pixel block that matches the current pixel block can be done in following steps:

1. Prescreen the potential matching pixel block, skipping it if features such as its width, its height, the area of its bounding box, or the number of black pixels are not close to those of the current pixel block.

2. Compute a match score, and call the potential matching pixel block the *best match* if its score is better than that of any other potential matching pixel block tested so far. A simple example of a match score is the Hamming distance, that is, the count of the number of mismatched pixels between the potential matching pixel block and the current pixel block when they are aligned according to the geometric centers of their bounding boxes [7]. The best match is acceptable if its score is better than a prespecified threshold; the threshold may depend on characteristics of the current pixel block like its size.

**Coding of numerical data.** If an acceptable match is found, the associated numerical data (dictionary index, position) are either bit-wise or Huffman-based encoded. Details can be found in [7] or [8].

**Coding of bitmap.** If there is no acceptable match, the bitmap of the current pixel block is encoded using MMR or JBIG1 based techniques.

This method of pattern matching and substitution (PM&S) allows high lossy compression levels. However, use of PM&S results in infrequent but inevitable substitution errors. For cases where such errors are unacceptable but where extra coded bits and extra coding time are acceptable, JBIG2 allows either residue coding (yielding a lossless compression level slightly higher than that of MMR and JBIG1), or a technique called *soft pattern matching* (SPM) [2, 9].

#### 2.4.2 Soft pattern matching

This method differs from pattern matching and substitution in that, in addition to a pointer to the dictionary and position information as in PM&S, we include refinement data that can be used to recreate the original character on the page, yielding lossless compression. Although lossy compression can still be obtained using preprocessing techniques (discussed in the next section), substitution errors are very unlikely.

This refinement data consists of the pixels of the current desired character, coded making use of pixels from both the current character and the matching character. The current character is highly correlated with the matching character since that is the basis for the declaration of a match, so that prediction of the current pixel is now more accurate. The SPM method, illustrated in Figure 2, is similar to the lossy PM&S method discussed earlier. The only difference (shown in *italics* in the figure) is that lossy direct substitution of the matched character is replaced by a lossless encoding that uses the matched character in the coding

context. The refined pixel block may be identical to the original pixel block but the encoder has the freedom to refine merely to a less lossy pixel block. This procedure is lossy SPM. Unlike PM&S, lossy SPM does not need a very safe and intelligent matching procedure to avoid substitution errors (though, as in any lossy technique, errors are possible).

Like the PM&S method, the first part of the SPM method consists of the following steps: 1) segmentation of the image into pixel blocks, 2) searching for a match in the dictionary, and 3) coding of the associated numerical data if a “good” match is found, or 4) coding of the corresponding bitmap otherwise. These steps are similar to the PM&S ones except that the numerical data is encoded differently, using arithmetic coding.

The second part of the SPM method consists of losslessly encoding the bitmap of the current pixel block as follows: we align the geometric center of the current pixel block with the center of the matching pixel block. We then encode all the pixels within the bounding box of the current pixel block in raster scan order, using an arithmetic coder as in JBIG1, but with a different template. Each pixel’s template consists of a combination of some pixels from the causal region of the current pixel block (pixels already seen and coded) and some more pixels from the matching pixel block in the neighborhood of the pixel in the matching pixel block that corresponds to the current pixel<sup>14</sup>. The template shown in Figure 3 is used in the SPM coder discussed in [9].

The SPM method has a distinct advantage over the PM&S method. In the latter method, a matching error can lead directly to a character substitution error. A PM&S method can neither guarantee that there will be no mismatches nor detect them when they occur. In the SPM method, the matching pixel block is used only in the template, to improve the accuracy of our prediction of each pixel’s color. Even using a totally mismatched pixel block in the template leads only to reduced compression efficiency, not to any errors in the final reconstructed image. If the pixel block is well-matched, we take full advantage of our knowledge of it.

To summarize, most of the numerical data can be coded using either multi-alphabet arithmetic coding or Huffman coding. Moreover, the basic coding of bitmaps may be done in two ways: either using a pixel-by-pixel context-based model with arithmetic coding, as in JBIG1, or using an MMR coder, as in T.6.

---

<sup>14</sup>We do not have to worry about causality in the matching pixel block since the matching pixel block is already entirely known by the decoder.



## 2.5 Halftones

Two methods for compressing halftone images have been proposed for inclusion in JBIG2. The first is similar to the context-based arithmetic coding treatment used in JBIG1, although the new standard will allow the context template to include as many as 16 template pixels, as many as 4 of which may be adaptive [10, 11]. An example of a (16, 4) template is shown in Figure 4. The larger templates are intended to exploit specific types of redundancies that exist in halftone images, usually yielding a significant improvement in compression efficiency.

The second method involves descreening the halftone image (converting it back to grayscale) and transmitting the grayscale values. In this method, the bi-level image may be divided into pixel blocks of  $m_b$  rows and  $n_b$  columns. If necessary the bi-level image can be zero-padded at the right side and at the bottom. For a bi-level image with  $m$  rows and  $n$  columns, we can obtain a grayscale image of dimensions  $m_g \times n_g$  where  $m_g = \lfloor (m + (m_b - 1)) / m_b \rfloor$  and  $n_g = \lfloor (n + (n_b - 1)) / n_b \rfloor$ . The grayscale value may be the sum of the binary pixels values in the corresponding  $m_b \times n_b$  block. The grayscale image will be Gray-coded and the bitplanes will be coded using context-based arithmetic coding, as in JBIG1. The grayscale values are then used as indices of fixed-size bitmap patterns in a halftone bitmap dictionary, so the decoder can render the image simply by making the indexed dictionary bitmap patterns about each other [12]. To provide for better quality of halftones with an angled period, the bitmap patterns corresponding to the transmitted grayscale values may also be placed along an angled grid. The rendered halftone is defined by the rule for combining overlapping patterns. The idea of using indices to represent grayscale values of  $m_b \times n_b$  pixel blocks is similar to that of the PM&S and SPM methods, and it can often yield good compression results.

## 2.6 Lossy preprocessing and postprocessing

Although JBIG2 provides the opportunity for lossy compression, the permissible kinds of loss are not specified in the standard. The standard specifies how the decoder must interpret a compliant bitstream; in effect, the decoder is guaranteed to be lossless with respect to the desired image as encoded by the encoder, but not necessarily with respect to the original image. The original image may be modified by the encoder during a preprocessing phase to increase coding efficiency. Of course, the use of direct substitution without refinement, after pattern matching, introduces loss as well. Thus, the PM&S methods are inherently lossy.

Most preprocessing techniques will lower the code length of the image without affecting the general appearance of the image (possible even improving the appearance). In general, loss

may be perceived as flipping pixels. In the PM&S methods, pixel flipping has conceptionally occurred in those positions in the image where an original pixel block and its match does not have the same color. Some possible preprocessing techniques are described next.

**Quantization of offsets.** We can obtain some improvement in compression efficiency by quantizing the offsets. For English text on a portrait-oriented page, character positions can be safely quantized to about 0.015 inch in the horizontal dimension and to about 0.01 inch in the vertical dimension; any more quantization causes noticeable distortion, but does not seriously affect legibility. Unfortunately, the increase in compression efficiency is small, on the order of one percent, and the restored images do not look as good, so this is usually not a useful procedure.

**Noise removal and smoothing.** For images consisting mainly of text at a resolution commensurate with the character sizes, loss can be introduced while still maintaining a near-zero probability of substitution errors [13]. For example, eliminating very small pixel blocks that represent noise on the page improves compression efficiency. We can also achieve improvement by smoothing each pixel block (following rules designed to prevent substitution errors) before compressing it and thus before entering it into the list of potential matching pixel blocks. One simple smoothing that can be done is to remove single protruding pixels (white or black) along edges within pixel blocks. Smoothing has the effect of standardizing local edge shapes, which improves prediction accuracy and increases the number of matching pixels between those of the current pixel block and those of the potential matching pixel blocks. The increase in compression ratio is typically about ten percent.

**Bit flipping** Loss may also be introduced by flipping bi-level pixels [9, 14, 10, 15]. This may be done in a preprocessing step at the encoder, and does not affect the complexity of the decoder. The principle is simple, but it must be done in a controlled manner. Some control is needed both to ensure that the code length actually decreases and to avoid artifacts. Flipping a pixel not only affects the code length of the pixel itself but also the code length of all the pixels for which it appears in the template. As a second order effect, it also slightly changes pixel-color statistics for the rest of the image. Artifacts such as avalanche effects may appear as a result of flipping pixels. In [14], pixels are flipped on-line such that the best tradeoff between rate and weighted distortion is achieved. In [10, 15], a method is presented, where a greedy rate-distortion based algorithm is used to control pixel flipping. In the original algorithm, statistics are first collected, then the effect of flipping candidate pixels on the total

code length is calculated based on the collected statistics, and finally pixels are flipped such that the highest gain in rate-distortion tradeoff is achieved. Less complex variations of the original algorithm are also presented in [10, 15]. For large and flexible templates such as those used in JBIG2, the encoder is usually able to capture the image structure, and the artifacts encountered are often quite small. For smaller templates or with simpler flipping techniques, flipping avalanches may be avoided by applying a control mechanism based on error-diffusion [10, 15]. Pixel flipping offers a continuous trade-off of rate and distortion in the near-lossless area up to some maximum, image-dependent, distortion. Although the above algorithms can be applied to all bi-level material, they provide the highest improvement for halftones.

Finally, we note that postprocessing, also not specified in the JBIG2 standard, can also be used to improve the quality of reconstructed bi-level images. Postprocessing is especially beneficial in the case of halftone images, where more visually pleasing rendered images can be obtained, for example by tuning the reconstructed image to a particular output device.

### 3 Experimental Results

To illustrate the potential advantages of the emerging JBIG2 standard, we present simulation results in which a typical JBIG2 compliant coder is compared against three coders achieving some of the highest compression performance levels reported in the literature: the standard bi-level image coder (JBIG1) and the multi-level image coders, SPHIT[16] and TCQ[17]. The latter coder is selected as the baseline for JPEG-2000, the next JPEG generation. In our simulation experiments, the bi-level image **s06a** (also known as **ccitt2** in the JPEG-2000 test set) is used as the test image. The image **s06a**, shown in Figure 5, contains both text and halftone material. Such an image allows us to demonstrate, clearly, the effectiveness of the text and halftone coding methods.

Table 1 shows the lossless compression ratios achieved by JBIG1, SPM, JBIG1-like halftone, SPHIT and TCQ. Notice that, because it is optimized for text, SPM does not perform well in comparison to JBIG1. Moreover, the JBIG1-like halftone coder achieves a 1.3 : 1 advantage due to its use of more template pixels and more adaptive pixels. As expected, both the SPHIT and TCQ coders do not perform well in comparison to JBIG1. The especially poor performance of SPHIT, which employs wavelet filters, indicates that wavelet filter banks should not be used to encode bi-level images. This fact was already discovered during the development of the TCQ coder, so the TCQ coder disables the filtering process when bi-level image data is detected.

When loss is allowed, large gains in compression efficiency can be achieved by JBIG2 coders. As shown in Table 2, 15% and 53% savings in compression bits in comparison to JBIG1 are obtained for the SPM and halftone coders (respectively), while the reconstructed **s06a** image is indistinguishable from the original image. Since they can achieve precise bit rate control, the SPHIT and TCQ coders are used to lossy encode the image **s06a** at the same bit rate as the one achieved by the lossy SPM coder. The subjective quality of the resulting SPHIT and TCQ reconstructed images is, however, very poor.

Since the image **s06a** contains both text and different types of halftone material, better compression performance is expected if it is properly segmented, and each of the image segments is separately coded using the appropriate coding method. Indeed, we have segmented the image **s06a** into 4 image segments (halftone1, halftone2, halftone3, text), and applied several combinations of JBIG2 component coders (i.e., PM&S, SPM, halftone), yielding the lossless coders JBIG2-I and JBIG2-II, and the lossy coders JBIG2-III, JBIG2-IV and JBIG2-V. Compression performance results for these coders are shown in Table 3.

JBIG2-I consists of applying different versions of the JBIG1 like context-based halftone coder to each of the halftone segments and SPM to the text segment. Clearly, JBIG2-I achieves the highest lossless compression ratio of 11.4 : 1. JBIG2-II is that same coder as JBIG2-I, except that the halftone coder is also used to encode the text segment. Notice that segmentation yields some lossless compression gains (approximately 7%) even if the same component coder is applied to the segments.

In the lossy coders JBIG2-III, JBIG2-IV and JBIG2-V, a lossy halftone coder is applied to each of the halftone segments. The corresponding compression file sizes are shown in Table 3, and the corresponding reconstructed images are shown in Figures 6, 7, and 8. Clearly, the reconstructed images are almost indistinguishable from the original ones, but a saving between 30% and 50% in compression bits is achieved as compared to JBIG1. The PM&S, SPM and halftone coders are applied to the text segment in JBIG2-III, JBIG2-IV and JBIG2-V (respectively), with corresponding reconstructed text images shown in Figure 9. Notice that, again, the quality of the reconstructed images is very good, while requiring slightly higher than half the number of bits required by JBIG1. It should be noted that, although current encoders do not do this, JBIG2 allows the possibility of segmenting halftone2 (Figure 7) into the underlying halftone and the overlying white text. Using this segmentation, further gain in compression efficiency can be achieved.

To summarize, we conclude that 1) the quality of reconstructed text images is very good and compression efficiency is high if PM&S or SPM based coding is used, 2) the quality

of reconstructed halftone images is also very good and compression efficiency is also high if halftone coding is used, 3) the quality of reconstructed text/halftone images is also very good and compression efficiency is high-to-moderate if halftone coding is used, 4) the quality of text/halftone images is excellent and compression efficiency is very high if a good segmentation algorithm and the PM&S/SPM/halftone coding methods are appropriately used.

## 4 Prospects

When adopted, the JBIG2 standard will facilitate low bit rate transmission, storage and interactive use of bi-level images while maintaining high reproduction quality. Compared with MR coding, the method commonly used in fax machines, JBIG2 compression applied after preprocessing a 200 dots-per-inch textual image (introducing mild loss) can provide up to 3 to 5 times the compression (and thus requires only roughly 20 to 35 percent as many bits); for normal type sizes (down to about 8 point) there is virtually no visible degradation of quality. Alternatively, we can apply JBIG2 compression to lossily-preprocessed higher resolution images (600 dpi); the resulting compressed bitstream (for text images) uses fewer bits than MR coding at  $100 \times 200$  dpi (a typical usage of Group 3 fax) and gives considerably higher quality. Moreover, as illustrated in the previous section, JBIG2 outperforms significantly JBIG1 and the wavelet/subband SPHIT and TCQ coders in both the lossless and lossy cases, especially when a good segmentation algorithm is properly used.

At the time of this writing, the JBIG study group is preparing a formal Working Draft for JBIG2. After the required sequence of drafts, ballots, comment periods, and dispositions of comments, JBIG2 is expected to become International Standard ISO/IEC 14492 by 2000.

## 5 Acknowledgments

The authors would like to acknowledge the contributions of Jaehan In (University of British Columbia, BC, Canada) and those of the JBIG members, in particular Ronald B. Arps (IBM Almaden Research Center, San Jose, CA), Corneliu Constantinescu (IBM Almaden Research Center, San Jose, CA), Dave Fernandes (DSI Datotech Systems, Inc., Vancouver, BC, Canada), Hyung Hwa Ko (Kwangwoon University, Seoul, Korea), Istvan Sebestyen (Siemens AG, Munich, Germany), Stephen Swift (Image Power, Inc., Vancouver, BC, Canada), and Stephen Urban (Delta Information Systems, Inc., Horsham, PA)

## Appendix: Summary of existing facsimile standards

This appendix is included in an attempt to dispel some of the confusion surrounding facsimile standards.

Transmission of facsimile data is standardized by a number of ITU - T Recommendations and by one ISO/IEC International Standard. ITU - T is the Telecommunication Standardization Sector of the International Telecommunication Union<sup>15</sup>. The ITU ([www.itu.int](http://www.itu.int)) is an international organization within which governments and the private sector coordinate global telecom networks and services. ITU - T issues “Recommendations.” ISO ([www.iso.ch](http://www.iso.ch)) is the International Organization for Standardization, a non-governmental worldwide federation of national standards bodies. Its mission is to promote the development of standardization to facilitate the international exchange of goods and services, and to develop cooperation in the spheres of intellectual, scientific, technological and economic activity. ISO collaborates closely with the International Electrotechnical Commission (IEC, [www.iec.ch](http://www.iec.ch)) on matters of electrotechnical standardization. ISO issues “International Standards.”

ITU - T Recommendation T.4 defines the MH and MR methods for facsimile coding. MH (Modified Huffman) is a one-dimensional encoding of runs of black and white pixels, with the run lengths coded using a two-level Huffman code; the first level, used only if a run is 64 pixels or longer, codes the number of multiples of 64 pixels in the run, and the second level codes the number of remaining pixels after accounting for the multiples of 64 pixels. MR (Modified READ<sup>16</sup>) is a two-dimensional method in which the position of each changing element on the present line is coded relative to either a corresponding changing element on the previous line or to the preceding changing element on the present line. In MR coding, every  $K$ th line ( $K = 1, 2$ , or  $4$ ) is coded using one-dimensional MH coding. There are fill characters and an end-of-line indicator at the end of each line, so the effect of transmission errors can be limited.

ITU - T Recommendation T.6 defines the MMR method for facsimile coding. MMR (Modified Modified READ) is similar to MR, except that there are no fill characters, no end-of-line indication, and no one-dimensional coding. (In effect,  $K = \infty$  and the line above the top image line is considered to be all white.) Since MMR is intended to be used in error-free circumstances, it can be used in Group 4 or in Group 3 Error Correction Mode (ECM), in which an ARQ error-correction facility is used in an attempt to eliminate transmission errors.

ITU - T Recommendation T.82 is identical to International Standard ISO/IEC 11544. It is informally known as the JBIG standard; to avoid confusion with the new JBIG2 standard

---

<sup>15</sup>The ITU was formerly known as CCITT.

<sup>16</sup>READ stands for Relative Element Address Designate.

we recommend calling it JBIG1. T.85 is the application profile for facsimile using T.82.

T.82 defines two methods for bi-level compression, progressive and non-progressive. Progressive coding, which presumes real time coding and soft copy display, is not adopted in the current facsimile standards. In non-progressive coding, the image is coded in raster-scan order. Each pixel is coded using arithmetic coding, with the probability of each of the two colors being conditioned on a context of ten nearby pixels. Two different templates are permitted for the context, one containing pixels from the current scan line and the previous scan line, the other containing pixels from the current scan line and the two previous scan lines. In each template, one of the ten pixels, called the adaptive (AT) pixel, may be moved from its default position.

Group 3 and Group 4 are ITU - T facsimile recommendations that define coding methods as well as other aspects of facsimile transmission, such as communication protocols. The coding methods in Group 3 facsimile are defined by T.4 T.6, and T.82; MH, MR, MMR and JBIG1 coding may be used. The communication protocols for Group 3 are defined in T.30. The coding methods in Group 4 facsimile are defined by T.6 and T.82. There are about 10 million installed Group 3 fax machines today. Group 4 is used to a limited extent in Japan and Europe, and as standardization of facsimile has evolved over the years most of the functionality of Group 4 has been incorporated into Group 3.

Recently ITU - T defined color facsimile services for both Group 3 and Group 4. Three categories are recommended. One uses JPEG coding, defined by ITU - T recommendation T.81 | International Standard ISO/IEC 10918; it is good for the transmission of natural color images. The second uses JBIG1 coding, and is good for limited color images or palletized images. The third category is Mixed Raster Content (MRC), defined by ITU - T Recommendation T.44. It is good for a mixture of these two categories.

## References

- [1] R. Hunter and A. H. Robinson, “International digital facsimile coding standards,” in *Proceedings of IEEE*, vol. 68, pp. 854–867, July 1980.
- [2] K. Mohiuddin, J. J. Rissanen, and R. Arps, “Lossless binary image compression based on pattern matching,” in *Proceedings of International Conference on Computers, Systems, and Signal Processing*, (Bangalore, India), pp. 447–451, 1984.
- [3] C. Constantinescu and R. Arps, “Fast residue coding for lossless textual image compression,” in *DCC’97*, (Snowbird, Utah, USA), pp. 397–406, 97.
- [4] K. Sayood, *Introduction to Data Compression*. San Francisco, CA: Morgan Kaufmann Publishers, 1996.
- [5] R. N. Ascher and G. Nagy, “A means for achieving a high degree of compaction on scan-digitized printed text,” *IEEE Transactions on Computers*, vol. C-23, pp. 1174–1179, Nov 1974.
- [6] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*. Reading, MA: Addison-Wesley Publishing Company, 1974.
- [7] W. K. Pratt, P. J. Capitant, W. H. Chen, E. R. Hamilton, and R. H. Walls, “Combined symbol matching facsimile data compression system,” *Proceedings of the IEEE*, vol. 68, pp. 786–796, July 1980.
- [8] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*. New York: Van Nostrand Reinhold, 1994.
- [9] P. G. Howard, “Text image compression using soft pattern matching,” *Computer Journal* 40:2-3, 1997.
- [10] B. Martins and S. Forchhammer, “Lossless/lossy compression of bi-level images,” in *Proceedings of IS&T/SPIE Symposium on Electronic Imaging: Science and Technology, SPIE-3018*, pp. 38–49, 1997.
- [11] B. Martins and S. Forchhammer, “Tree coding of bi-level images,” *IEEE Trans. Image Processing*, (to appear), April 1998.
- [12] S. Forchhammer and K. S. Jensen, “Data compression of scanned halftones images,” *IEEE Trans. Commun.*, vol. 42, pp. 1881–1893, Feb-Apr 1994.



- [13] P. G. Howard, “Lossless and lossy compression of text images by soft pattern matching,” in *Proc. of Data Compression Conference*, pp. 210–219, 1996.
- [14] F. Kossentini and R. Ward, “An analysis-compression technique for black and white documents,” in *IEEE Southwest Symposium on Image Analysis and Interpretation*, (San Antonio, TX, USA), pp. 141–144, Apr. 1996.
- [15] B. Martins and S. Forchhammer, “Lossless, near-lossless, and refinement coding of bi-level images.” submitted to *IEEE Trans. Image Processing*.
- [16] A. Said and W. A. Pearlman, “A new fast and efficient image codec based on set partitioning in hierarchical trees,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243–250, June 1996.
- [17] A. Bilgin, P. Sementilli, and M. Marcellin, “Progressive image coding using trellis coded quantization,” *Submitted to IEEE Transactions on Image Processing*, Nov. 1997.

## List of Tables

1	Compression results for different lossless coders. . . . .	20
2	Compression results for different lossy coders. . . . .	20
3	Compression results for different JBIG2 coders. . . . .	20

## List of Figures

1	Block diagram of a typical pattern matching and substitution algorithm. . . .	21
2	Block diagram of a typical soft pattern matching algorithm. . . . .	22
3	Template for refinement coding of the current pixel block. The numbered pixels form the context for coding the pixel marked “P”. (a) Pixels taken from the causal part of the current pixel block. (b) Pixels taken from the matching pixel block. The pixel numbered “7” corresponds to pixel “P” when the pixel blocks are aligned according to their centers. . . . .	23
4	3-line template for lossless and lossy coding. The pixels marked “a” are default positions for the 4 adaptive template pixels. . . . .	23
5	Original image <b>s06a</b> (CCITT2) . . . . .	24
6	(a) Original image “halftone1” and (b) the reconstructed image using a lossy halftone coder. . . . .	25
7	(a) Original image “halftone2” and (b) the reconstructed image using a lossy halftone coder. . . . .	25
8	(a) Original image “halftone3” and (b) the reconstructed image using a lossy halftone coder. . . . .	26
9	Original image “text” and the reconstructed images using three lossy coders (only a fragment of each image is shown): (a) original image “text”, (b) the reconstructed image using a lossy SPM coder, (c) the reconstructed image using a lossy halftone coder, and (d) the reconstructed image using a PM&S coder. .	27

	Original Size	JBIG1	SPM (lossless)	Halftone (lossless)	SPHIT (lossless)	TCQ (lossless)
<b>s06a (ccitt2)</b>	1671168	217252	294125	165378	815529	274071
Compression ratio		7.7:1	5.7:1	10.1:1	2:1	6.1:1

Table 1: Compression results for different lossless coders.

	Original Size	SPM (lossy)	Halftone (lossy)	SPHIT (lossy)	TCQ (lossy)
<b>s06a (ccitt2)</b>	1671168	185499	101691	185499	185499
Compression ratio		9:1	16.4:1	9:1	9:1

Table 2: Compression results for different lossy coders.

Segmented images	Original size	JBIG2-I (Lossless)	JBIG2-II (Lossless)	JBIG2-III (Lossy)	JBIG2-IV (Lossy)	JBIG2-V (Lossy)
halftone1	100313	13372 (HT)	13372 (HT)	8934 (LHT)	8934 (LHT)	8934 (LHT)
halftone2	404253	51264 (HT)	51264 (HT)	30317 (LHT)	30317 (LHT)	30317 (LHT)
halftone3	224656	30161 (HT)	30161 (HT)	14654 (LHT)	14654 (LHT)	14654 (LHT)
text	1671168	51891 (SPM)	59234 (HT)	27562 (PM&S)	30636 (LSPM)	44906 (LHT)
Total		146688	154031	81467	84541	98811
Compression ratio		11.4:1	10.8:1	20.5:1	19.8:1	16.9:1

Table 3: Compression results for different JBIG2 coders.

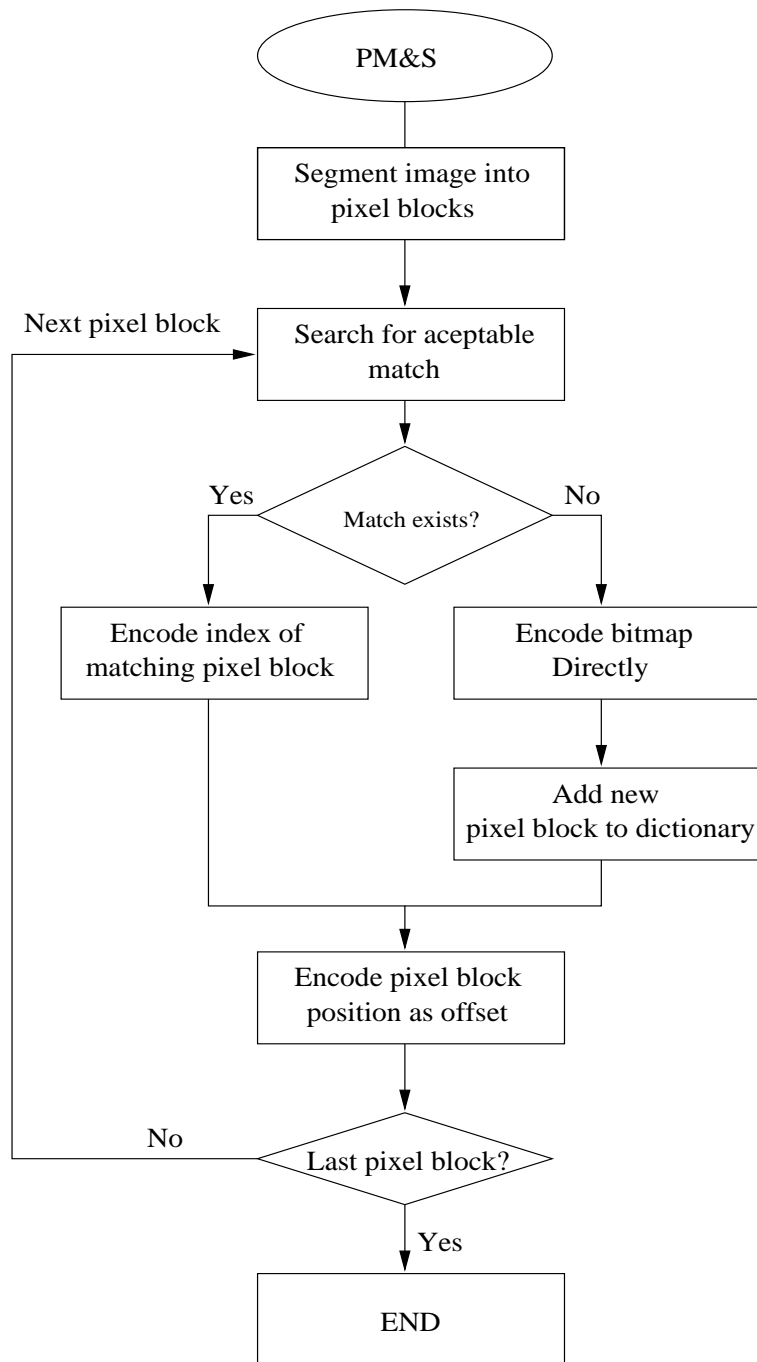


Figure 1: Block diagram of a typical pattern matching and substitution algorithm.

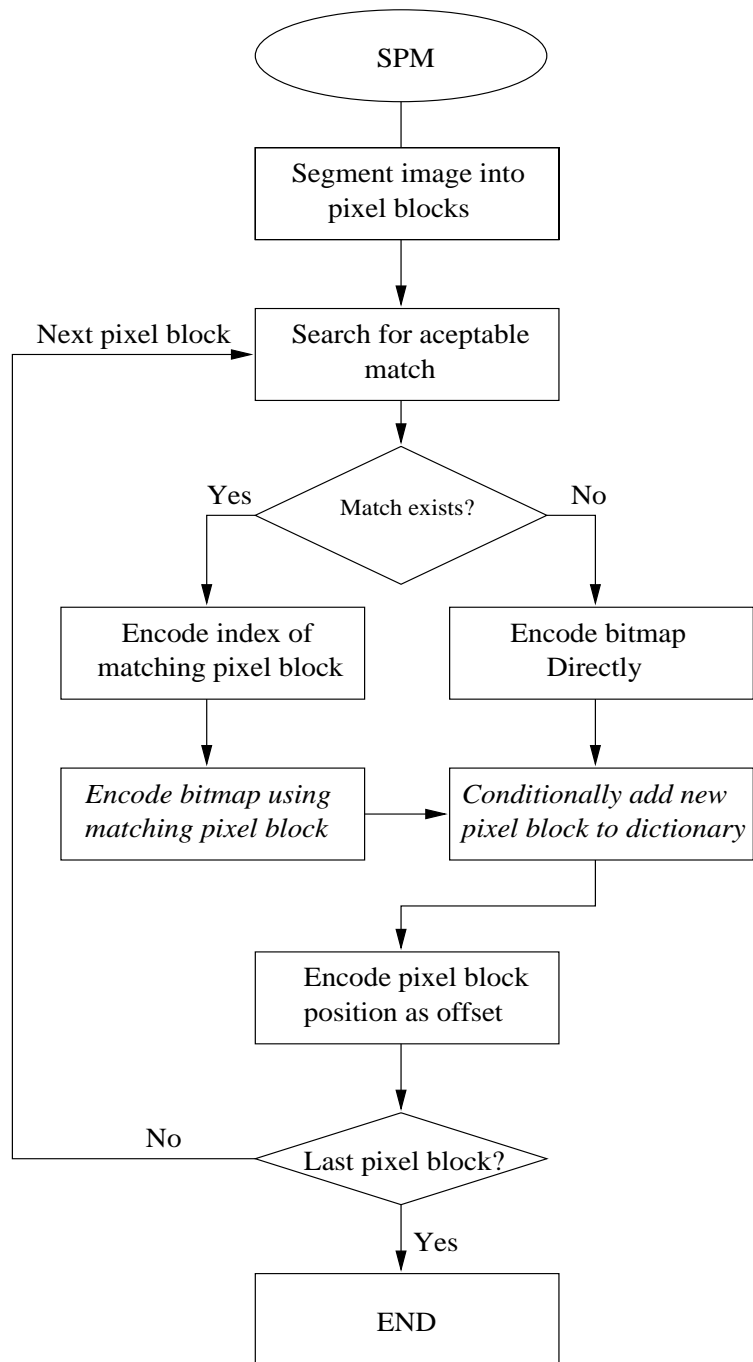


Figure 2: Block diagram of a typical soft pattern matching algorithm.

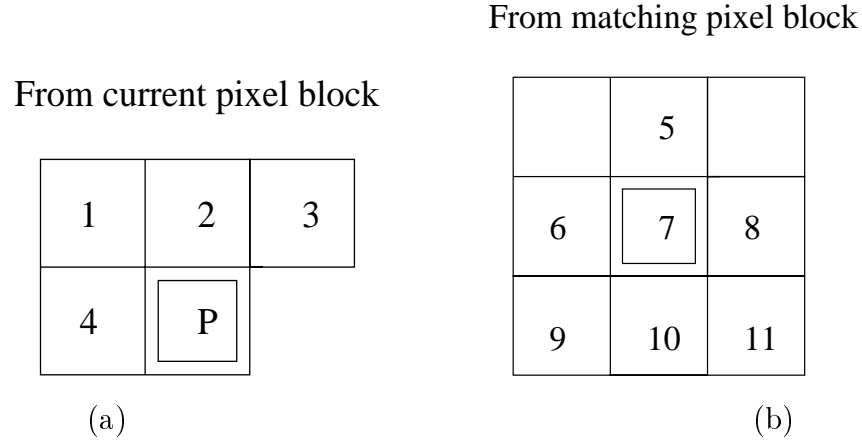


Figure 3: Template for refinement coding of the current pixel block. The numbered pixels form the context for coding the pixel marked “P”. (a) Pixels taken from the causal part of the current pixel block. (b) Pixels taken from the matching pixel block. The pixel numbered “7” corresponds to pixel “P” when the pixel blocks are aligned according to their centers.

				a	X	X	X	a				
			a	X	X	X	X	X	a			
		X	X	X	X	?						

Figure 4: 3-line template for lossless and lossy coding. The pixels marked “a” are default positions for the 4 adaptive template pixels.



Figure 5: Original image s06a (CCITT2)



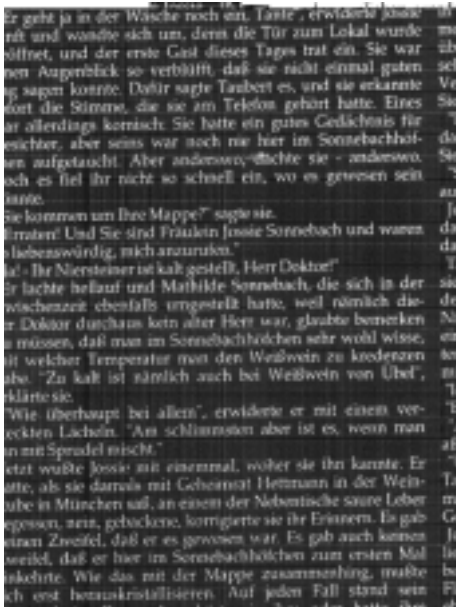


(a)

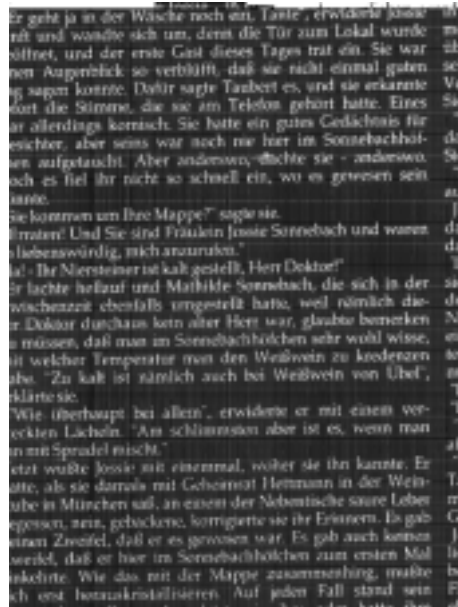


(b)

Figure 6: (a) Original image “halftone1” and (b) the reconstructed image using a lossy halftone coder.



(a)



(b)

Figure 7: (a) Original image “halftone2” and (b) the reconstructed image using a lossy halftone coder.



(a)



(b)

Figure 8: (a) Original image “halftone3” and (b) the reconstructed image using a lossy halftone coder.

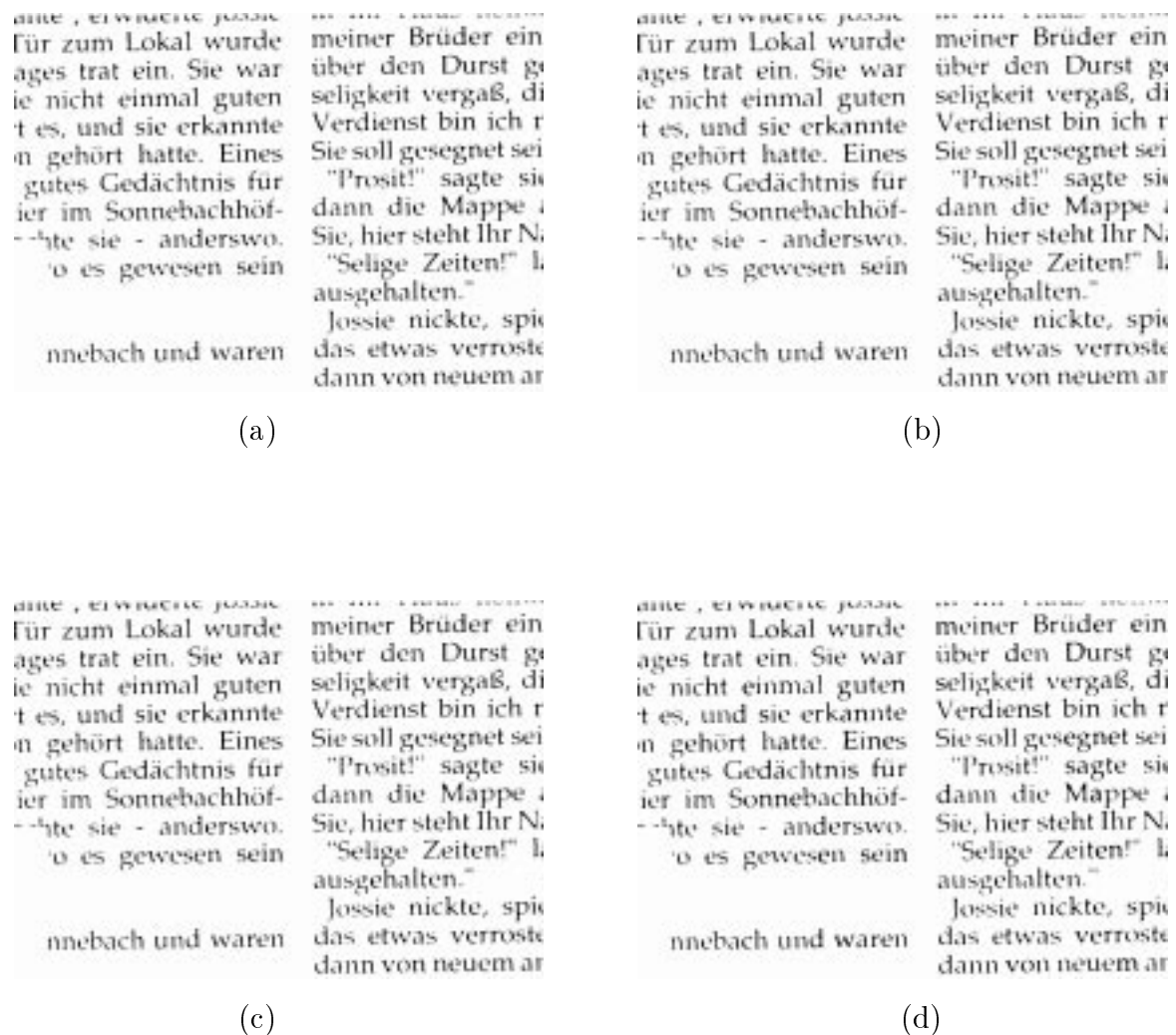


Figure 9: Original image "text" and the reconstructed images using three lossy coders (only a fragment of each image is shown): (a) original image "text", (b) the reconstructed image using a lossy SPM coder, (c) the reconstructed image using a lossy halftone coder, and (d) the reconstructed image using a PM&S coder.