# Speedup Techniques for Text Image Compression with JBIG2

Yan Ye and Pamela Cosman
Electrical and Computer Engineering Department
University of California, San Diego
La Jolla, CA, 92093-0407
Emails: {yye,pcosman}@code.ucsd.edu
Tel:(858)822-1250 FAX:(858)822-3426

## Abstract

*Pattern matching is the most time consuming process in text image compression with JBIG2. In this paper we propose three techniques to speed up the pattern matching process. By limiting the search range for matching symbols in the dictionary, the first technique saves 15% of encoding time with almost no bit rate penalty. By making early decisions about the pattern matching outcome, the second technique saves another 15% of encoding time with no coding loss. The third technique looks at enhanced prescreening using additional symbol features besides symbol size. Using certain topological features, enhanced prescreening can save up to 75% of encoding time with at most 1.6% of bit rate penalty.*

## 1 Introduction

The JBIG2 standard [1, 2, 3] is the new international standard for bi-level image compression. Bi-level images have only one bit-plane, where each pixel takes one of two possible colors. A typical JBIG2 encoder first segments an image into different regions [4] and then uses different coding mechanisms for text and for halftones. In this paper, we are concerned with compressing text images with JBIG2. Text images consist mainly of repeated text characters and possibly some general graphics (e.g., line art). In JBIG2, the coding of text is based on pattern matching techniques [2, 3]. JBIG2 defines two modes for text compression: *pattern matching and substitution* (PM&S) [5] and *soft pattern matching* (SPM) [6].

On a typical page of text, there are many repeated characters. The bitmap of a character instance on the page is called a "symbol." We can extract symbols from the input image using connected component analysis [7]. Rather than coding all the pixels of all the symbols on the page, we code the bitmaps of a representative subset and put them into the *symbol dictionary*. Then, each symbol on the page is coded by giving its position on the page, the index of its best matching symbol in the dictionary, and, in the SPM mode, possibly its actual bitmap which is refinement coded using its matching dictionary symbol [1, 2]. This type of bitmap coding, called refinement coding, is done by context-based arithmetic coding using a context drawn from both the best match bitmap from the dictionary, and the already coded part of the current bitmap [8, 6]. General graphic data not identified as text is encoded at the end using a basic bitmap coder such as specified by JBIG1 [9] or T.6 [10].

A JBIG2 coding system for text images consists of several components: symbol extraction, pattern matching, arithmetic/Huffman integer/bitmap coding, and so on. To speed up arithmetic bitmap coding, JBIG2 allows typical prediction (TP) as specified in JBIG1 [9] and typical prediction for residue (TPR) as proposed in [11]. In this paper, we focus on reducing the encoding time spent on pattern matching. In our work we use the Hamming distance based matching criterion. We measure the percentage of different pixels between two symbols. For SPM-based JBIG2, even using our simple matching criterion, the time spent on pattern matching accounts for as much as 90% of the total encoding time. In this paper we propose three categories of speedup techniques that can significantly reduce the amount of pattern matching time while causing only a very small loss in coding efficiency.

This paper is organized as follows. In Section 2 we propose the three speedup techniques for pattern matching. In Section 3 we show experimental results on coding time saved and bit rate penalty incurred from using these speedup techniques. We conclude our paper in Section 4.

## 2 Speedup techniques for JBIG2 encoding

### 2.1 Limited dictionary symbol search

We proposed the modified-class (MC) dictionary design for the SPM-based JBIG2 in [13]. Experiments showed that the MC dictionary achieves competitive coding performance with relatively low complexity. The design of an MC dictionary consists of two steps. At the first step, we point each symbol to its closest match among all other symbols; we only draw a pointer between a symbol and its best match if the mismatch between them is below the preset threshold. This way the entire symbol set is segmented into small connected graphs, each of which is called a class. We then choose one representative for each class as the symbol with the lowest average mismatch within the class. All class representatives go into the dictionary. The second design step decides the reference relationships among all dictionary symbols, i.e., class representatives. This is done by calculating the matching graphs for all dictionary symbols and forming minimum spanning trees (MSTs) out of these graphs [12, 13].

Suppose a symbol $S$ belongs to a certain class $C$, whose representative is symbol $R$, which, after the MST construction procedure, lands in MST $T$. Therefore we know that symbol $S$ and symbol $R$ are similar, and that symbol $R$ is similar (to different degrees) to the symbols in tree $T$. Therefore, when symbol $S$ searches for its best match in the dictionary, we only search among all the symbols that belong to MST $T$. To do this, for each symbol on the page, we maintain a tree-ID value that specifies the MST in which this symbol's representative belongs. Hence, in the previous example, symbols $S$ and $R$ and all other symbols in the MST $T$ will have the same tree-ID. When the current symbol is matched with the dictionary, it only searches among those dictionary symbols that have the same tree-ID. This significantly reduces the number of dictionary symbols with which the current symbol is matched. Whether this limited search algorithm will suffer significant bit rate penalty depends on how many symbols and their best dictionary matches actually belong to the same MST. Section 3 shows that this limited search algorithm can save encoding time at almost no coding loss.

## 2.2 Early jump-out based on previous best match

When matching one symbol with another, we save the previous lowest mismatch score; the pattern matcher compares on-the-fly the current accumulated mismatch score against the previous lowest one. If the current mismatch is already above the previous lowest, then we terminate the current matching process. Computing the Hamming distance between two symbols is fast because it only requires the exclusive-OR (XOR) operation and incrementing the mismatch score accordingly. On the other hand, comparing the two integer mismatch scores also takes time. Therefore, we do the integer comparison of mismatch scores only once for each row of pixels in the bitmap. At the end of each row, the current accumulated mismatch is checked; if it exceeds the previous lowest, the pattern matching process terminates.

## 2.3 Enhanced prescreening

Before matching a pair of symbols, it is advantageous to prescreen them by certain features. There is no need to apply pattern matching to two symbols that are obviously dissimilar. For example, symbols that differ greatly in size (e.g. a capital "D" and a comma ",") obviously do not match. The original SPM system as proposed in [6] prescreens symbols using size; only symbols with similar sizes (defined as not more than 2 pixels different in either dimension) are given to the pattern matcher which computes their mismatch score. Prescreening is intended to reduce the number of unnecessary pattern matching calls that will not return a match. At the same time, prescreening should not rule out potentially good matches. Otherwise it will incur a high bit rate penalty. Therefore, the ideal prescreening rules out all "unmatchable" symbols and passes on all "matchable" symbols to the more expensive pattern matching subroutine.

Other features can be used in prescreening besides symbol size. One such example is to use symbol area and/or perimeter [7, 14]. However, these two features are not particularly helpful for two reasons: they are correlated with symbol size, and they are usually sensitive to scanning noise and digitization parameters such as contrast [7]. According to our experiments, in the English language, using the Hamming distance based matching criterion, letter pairs that are among the most easily confused include "b" and "h," "c" and "e," and "i" and "l." In this paper we propose two topological features for prescreening: number of holes and number of connected components [16]. Prescreening by these two features can effectively prevent these symbol pairs from being handed over to the pattern matcher (see Figure 1).

Another useful feature for prescreening is introduced in [7]. We call it the quadrant centroid distance. It is calculated as follows. We divide each symbol into four quadrants and calculate the centroid for each quadrant. To prescreen two symbols, we calculate the distance between each pair of corresponding quadrant centroids, sum the four distances and compare the total to a threshold, which is preset to 3 pixels in our implementation. A small total distance means that the two symbols have similar mass distribution in all four quadrants; only such symbol pairs are passed on to pattern matching to be further examined.

## 3 Experimental results

In this section we show experimental results on the three speedup techniques proposed, the limited dictionary search algorithm based on tree-ID (TID), early jump-out (EJO), and enhanced prescreening (PRESCRN). We consider two figures of merit, the encoding time saved and the bit rate penalty incurred.

Our experiments use a set of twelve test images, two

**Table 1. Using the proposed three speedup techniques in SPM JBIG2.**

| | | total time | | match time | | coded size | |
|---|---|---|---|---|---|---|---|
| | | sec | % gain | sec | % gain | bytes | % loss |
| NONE | | 99.44 | - | 90.86 | - | 36,738 | - |
| TID | | 85.82 | 14 | 77.15 | 15 | 36,742 | 0.0 |
| EJO | | 85.12 | 14 | 76.47 | 16 | 36,738 | 0.0 |
| TID+EJO | | 71.18 | 28 | 62.63 | 31 | 36,742 | 0.0 |
| PRESCRN | S+Q | 31.95 | 68 | 23.29 | 74 | 37,128 | 1.1 |
| | S+H+C | 61.85 | 38 | 53.65 | 41 | 36,938 | 0.5 |
| | S+Q+H+C | 28.70 | 71 | 20.09 | 78 | 37,342 | 1.6 |
| ALL | | 26.20 | 74 | 17.62 | 81 | 37,359 | 1.7 |

**Table 2. Using the proposed three speedup techniques in PM&S JBIG2.**

| | | total time | | match time | | coded size | |
|---|---|---|---|---|---|---|---|
| | | sec | % gain | sec | % gain | bytes | % loss |
| NONE | | 24.05 | - | 10.78 | - | 41,404 | - |
| EJO | | 22.81 | 5 | 9.53 | 12 | 41,404 | 0 |
| PRESCRN | S+Q | 16.28 | 32 | 2.86 | 73 | 41,730 | 0.8 |
| | S+H+C | 19.48 | 19 | 6.22 | 42 | 41,566 | 0.4 |
| | S+Q+H+C | 16.07 | 33 | 2.61 | 76 | 41,925 | 1.3 |
| ALL | | 16.04 | 33 | 2.60 | 76 | 41,925 | 1.3 |

**Table 3. Prescreening pass rates when different features are used.**

| | S | S+Q | S+H+C | S+Q+H+C |
|---|---|---|---|---|
| % passes | 19.7 | 4.7 | 11.8 | 4.0 |

features together with symbol size (S+Q+H+C), we can reduce the prescreening pass rate to 4%.

Over a channel of fixed bandwidth, transmission of a bigger file takes longer time. Some applications, e.g., sending an international fax, favor the shortest channel time possible. For such applications, achieving the best compression is the most important, even if it takes some extra encoding time. Other applications, especially real-time applications, can only tolerate a small delay between the sender and receiver. For these applications, the goal is to achieve the best compression within a short encoding time. However, better compression usually requires longer encoding time. Figure 2 shows the trade-off between coding time and coding efficiency by plotting compression achieved as a function of total encoding time spent. The SPM system is shown as an example. Similar observations are made for the PM&S system. From Figure 2 we clearly see the trade-off between these two figures of merit. The

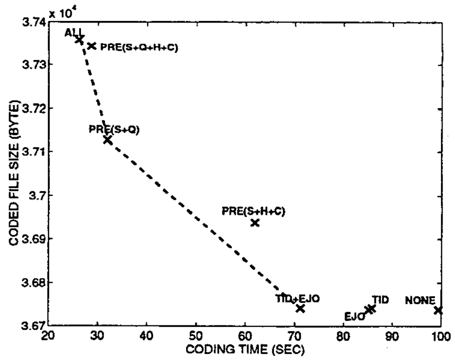
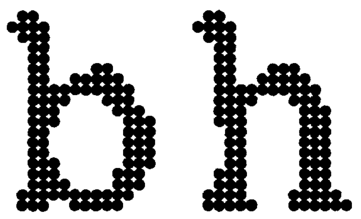
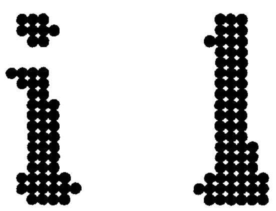


**Figure 2. Trade-offs between encoding time and coding efficiency using different speedup techniques, for the SPM system.**

lower boundary of the convex hull, as shown by the dashed line segments, represents the best trade-off that our techniques can achieve between coding time and efficiency. Although the H and C features are not as efficient as the Q feature in terms of this trade-off (the PRE(S+Q) marker lies on the lower boundary but the PRE(S+H+C) marker does not), in [17] we showed that the H and C features can help effectively control the reconstructed image quality in



(a) ''b'' has one internal hole; ''h'' has none



(b) ''i'' has two connected components; ''l'' has only one

**Figure 1. Examples of similar bitmaps that have different features. Bitmaps "b" and "h" differ in the number of holes; and bitmaps "i" and "l" differ in the number of connected components.**

standard CCITT images f01 and f04 (resolution 200 dpi), and ten images from the University of Washington Document Image Database I [15] (resolution 300 dpi). The computer system used is a Pentium Pro 200MHz, running Red Hat Linux 6.0, with 64MB physical memory. We measure encoding time (in sec) using the C library function "clock()." Our code was not specifically optimized for speed. All coding results given are averaged over all test images.

Tables 1 and 2 give the total encoding time, time spent on pattern matching, and coded file size for each individual technique and several combinations of them, for SPM- and PM&S-based JBIG2, respectively. In SPM (see Table 1), the encoding time spent on pattern matching accounts for up to 90% of the total encoding time. The rest of the encoding time (spent on symbol extraction, arithmetic bitmap and integer coding, etc.) is a fixed value of 8.6 seconds. For the PM&S mode (see Table 2), the time spent on pattern matching accounts for up to 45% of the total encoding time. The rest of the encoding time is a bigger fixed value of 13.3 seconds. Compared to the first row (NONE) where no speedup techniques are used and prescreening is only by symbol size, the TID row (limited dictionary search technique) in Table 1 saves about 15% of the pattern matching time, while causing almost no coding loss. Note that the TID technique is only applicable in the SPM mode. The early jump-out technique (EJO) saves 16% and 12% of the pattern matching time in SPM and PM&S, respectively, while incurring no bit rate penalty at all. In SPM, we can combine TID and EJO together to achieve a time gain of 31% with no coding loss. For the PRESCRN rows, adding the quadrant centroid distance (S+Q) to prescreening saves almost 3/4 of the pattern matching time, while incurring a bit rate penalty of around 1%. Using the numbers of holes and connected components together (S+H+C) saves 40% of the pattern matching time, which is less efficient than the Q feature. However, the bit rate penalty incurred is only half as big (0.5%). Combining all these speedup techniques together (the ALL rows in the two tables) saves 81% and 76% of the pattern matching time in SPM and PM&S, respectively. In terms of total encoding time, these numbers translate into savings of 74% and 33%, respectively. The bit rate penalty incurred is relatively small, 1.7% for SPM and 1.3% for PM&S.

Without the TID technique, each symbol searches among all dictionary symbols for its best match. For our test image set, on average the dictionary contains 638 symbols. If prescreening uses only symbol size, then about 20% of dictionary symbols or 128 symbols will pass prescreening and be handed to the pattern matcher (see Table 3). With the TID method, however, experiments show that a symbol needs to search among an average of only 34 dictionary symbols, most of which will pass prescreening as they are very similar to the current symbol. Comparing 34 symbols with 128 symbols, we see a 75% reduction. In terms of the time spent on finding dictionary matches for all symbols, this translates to a reduction from 19.66 seconds without TID to 5.30 seconds with TID.

Without the EJO technique, the pattern matcher will examine in full every pair of symbols passed on to it, i.e., it will go over 100% of the bitmap area before making a decision. With EJO, however, experiments show that on average only 44% of the bitmap area will be examined. Furthermore, on average 89% of all the pattern matching calls result in early termination. Although EJO has to spend extra time comparing integer mismatch scores, it still reduces the average number of CPU clocks used to match two symbols from 68 to 60. An important advantage of the TID and EJO techniques is that they save encoding time almost "for free," meaning without bit rate penalty (see Tables 1 and 2).

To see how enhanced prescreening helps effectively rule out unlikely matches, we list the percentages of prescreening passed in Table 3. Using the symbol size (S) feature alone is not efficient enough; around 20% of the symbol pairs will still be given to the pattern matching process. Adding the number of holes and number of connected components (S+H+C) reduces the pass rate to 12%; adding the quadrant centroid distance (S+Q) only 5% of the symbol pairs can pass prescreening. By combining all three lossy coding by reducing the number of character substitutions.

## 4 Conclusion

In this paper we propose three techniques to speed up the pattern matching process in text image compression with JBIG2. Experiments show that the limited dictionary symbol search technique and the early jump-out technique can each bring about 15% of savings in encoding time without loss in coding efficiency. Depending on the specific features used, the enhanced prescreening technique can save up to 75% of encoding time while only suffering a small bit rate penalty of at most 1.6%. These speedup techniques are effective for both SPM-based and PM&S-based JBIG2.

### Acknowledgements

This research was supported by the Center for Wireless Communications at UCSD and by the CoRe program of the State of California.

## References

[1] ISO/IEC JTC1/SC29/WG1 N1545. *JBIG2 Final Draft International Standard*, Dec. 1999.

[2] P. Howard, F. Kossentini, B. Martins, S. Forchhammer, W. Rucklidge, F. Ono. The Emerging JBIG2 Standard. *IEEE Trans. on Circuits and Systems for Video Technology*, pp. 838–48, Vol. 8, No. 5, Sept. 1998.

[3] F. Ono, W. Rucklidge, R. Arps, C. Constantinescu. JBIG2 – the Ultimate Bi-level Image Coding Standard. *Proc. of the 2000 IEEE Intl. Conf. on Image Processing (ICIP)*, Vancouver, Canada, Sept. 2000.

[4] D. Tompkins and F. Kossentini. A Fast Segmentation Algorithm for Bi-level Image Compression Using JBIG2. *Proc. of the 1999 IEEE International Conference on Image Processing (ICIP)*, pp. 224–228, Kobe, Japan, Oct. 1999.

[5] R.N. Ascher and G. Nagy. Means for Achieving a High Degree of Compaction on Scan-digitized Printed Text. *IEEE Trans. on Computers*, pp. 1174–79, Nov. 1974.

[6] P. Howard. Lossless and Lossy Compression of Text Images by Soft Pattern Matching. In J.A. Storer and M. Cohn, editors, *Proc. of the 1996 IEEE Data Compression Conference (DCC)*, pp. 210–19, Snowbird, Utah, March 1996.

[7] I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes.* Morgan Kaufmann, 1999.

[8] K. Mohiuddin, J. Rissanen, and R. Arps. Lossless Binary Image Compression Based on Pattern Matching. *International Conference on Computers, Systems and Signal Processing*, pp. 447–51, Bangalore, India, Dec. 1984.

[9] CCITT. Progressive Bi-level Image Compression. *CCITT Recommendations T.82*, 1993.

[10] CCITT. Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus. *CCITT Recommendations T.6*, 1984.

[11] C. Constantinescu and R. Arps. Fast residue coding for lossless textual image compression. *Proc. 1997 IEEE Data Compression Conference (DCC)*, pp. 397–406, Snowbird, Utah, March 1997.

[12] Y. Ye and P. Cosman. Dictionary design for text image compression with JBIG2. *IEEE Trans. on Image Processing*, Vol. 10, No. 6, pp. 818–828, June 2001.

[13] Y. Ye and P. Cosman. Fast and memory efficient JBIG2 encoder. *Proc. 2001 IEEE Intl. Conf. on Acoustics, Sound, and Signal Processing (ICASSP)*, Salt Lake City, Utah, May 2001.

[14] W. K. Pratt, P. J. Capitant, W. Chen, E. R. Hamilton and R. H. Wallis. Combined symbol matching facsimile data compression system. *Proc. of the IEEE*, pp. 786–796, Vol. 68, No. 7, July 1980.

[15] E. S. Askilsrud, R. M. Haralick and I. T. Phillips. A quick guide to UW English Document Image Database I, version 1.0. CD-ROM. Intelligent Systems Lab, University of Washington. August 1993.

[16] B. Horn. Robot Vision, Chapter 3. MIT Press. 1986.

[17] Y. Ye and P. Cosman. Feature monitored shape unifying for lossy SPM-JBIG2. *Proc. of the Sixth IEEE Intl. Symposium on Signal Processing and its Applications (ISSPA 2001)*, Kuala Lampur, Malaysia, August 2001.