JBIG for Printer Pipelines: A Compression Test

Daniele Ravì[1], Tony Meccio[1], Giuseppe Messina[1,2], and Mirko Guarnera[2]

[1] Università degli studi di Catania, D.M.I., Catania, Italy
[2] STMicroelectronics, Advanced System Technologies, Catania, Italy

Abstract. The proposed paper describes a compression test analysis of JBIG standard algorithm. The aim of such work is to proof the effectiveness of this standard for images acquired through scanners and processed into a printer pipeline. The main issue of printer pipelines is the necessity to use a memory buffer to store scanned images for multiple prints. This work demonstrates that for very large scales the buffer can be fixed using medium compression case, using multiple scans in case of uncommon random patterns.

1 Introduction

In the latest years there has been a growing demand for independent multifunctional printing and scanning devices. During standalone printing processes, there are cases where it is necessary to keep the whole image in memory, for example when it is requested to make multiple hard copies of a document acquired by the integrated scanner. While the first copy can be done "on the fly" during the scanning phase, the constraint to have identical multiple copies implies the storing of the whole scanned document. This can lead to excessive memory requirements, which is the reason why a compression method is normally used, especially for low-cost products, where it is important to embed as little physical memory as possible. Moreover, the compression must be lossless [1,2] to have identical copies of the same input media. The compression phase is normally placed just after the halftoning (see Fig.1) , because it already uses data with reduced bit planes [3]. A widely used algorithm to achieve this purpose is JBIG.

1.1 JBIG Compression Standard

JBIG is short for the 'Joint Bi-level Image experts Group'. This was a group of experts nominated by national standards bodies and major companies to work to produce standards for bi-level image coding.

JBIG developed IS 11544 (ITU-T T.82) for lossless compression of bi-level images [4]. It can also be used for coding grayscale and colour images with limited numbers of bits per pixel. It can be seen as a form of facsimile encoding, similar to Group 3 or Group 4 fax, offering between 20% and 80% improvement in compression over these methods (about 20:1 over the original uncompressed digital bit map).

A. Trémeau, R. Schettini, and S. Tominaga (Eds.): CCIW 2009, LNCS 5646, pp. 180–187, 2009.

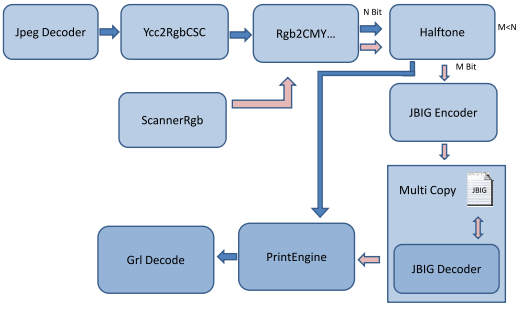

Fig. 1. A possible printing pipeline

Basically it models the redundancy in the image as the correlations of the pixel currently being coded with a set of nearby pixels called the template. An example template might be the two pixels preceding the current one on the same line, and the five pixels centred above the current one on the previous line. Note that this choice only involves pixels that have already been seen from a scanner.

The selected pixel is then arithmetically coded based on an eight-bit state so formed. So there are 256 possible contexts to be coded. The arithmetic coder and probability estimator for the contexts is IBM's (patented) Q-coder. The Q-coder uses low precision, rapidly adaptable (those two are related) probability estimation combined with a multiply-less arithmetic coder. The probability estimation is closely tied to the interval calculations necessary for the arithmetic coding.

To overcome this issue the JBIG uses adaptive templates. A description of the Q-coder as well as the prior version of JBIG can be found in the November 1988 issue of the IBM Journal of Research and Development [5,6,7,8].

JBIG can be used on both grey-scale and color images by simply applying the algorithm one bit-plane at a time. The JBIG works well up to about six bits per pixel, beyond which JPEG's lossless mode works better. The Q-coder must also be used with JPEG to get this performance. Actually no lossless mode works well beyond six bits per pixel, since those low bits tend to be noise, which doesn't compress at all. In any case the actual intent of JBIG is to replace the less effective group 3 and 4 fax algorithms.

The work described in this paper aims to evaluate the performance of the JBIG algorithm through a critical (synthetic) cases analysis. The tests will consider also some specific processing effects during scanning phase; in particular, we focused on CIS [9] (Contact Image Sensors) scanning methodologies.

1.2 Contact Image Sensors

Contact Image Sensors, abbreviated as CIS, is a type of optical flatbed scanner that does not use the traditional CCD arrays that rely on a system of mirrors and lenses to project the scanned image onto the arrays. CIS scanners gather

light from red, green and blue LEDs (which combine to create white light) and direct the light at the original document being scanned (see fig.2). The light that is reflected from the original is gathered by a lens and directed at an image sensor array that rests just under the document being scanned. The sensor then records the images according to the intensity of incident light. The sensors of the CIS systems cannot directly measure color hues. Instead, color is obtained as a linear combination of the three base colors (Red, Green and Blue). Each line of the image is illuminated with a light beam of one of the three colors, and reflected light is captured by the sensors which measure the intensity of the corresponding light component; the other two components are interpolated from adjacent lines, in a way similar to color demosaicing in single-sensor cameras.

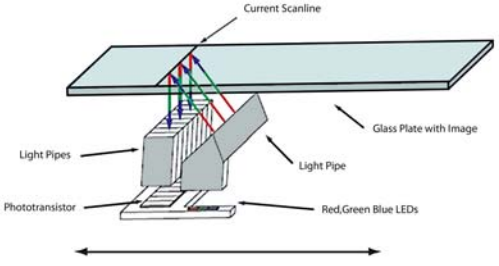

Fig. 2. Contact Image Sensor schematic plan

The paper is organised as follows: in section 2 a JBIG compression experiment in a standalone system is described; section 3 shows the same experiments in a printing pipeline. Finally a conclusion is presented taking into account the analysis of the tests.

2 JBIG Compression Experiments in a Standalone System

To evaluate the performance of JBIG it is important to use a set of images whose patterns can not be predicted via predetermined templates: random-generated images are thus used. The synthetic image generation algorithm produced has been parameterized in order to adjust the probability to generate each color, so that images with different degrees of randomness can be produced. For black-and-white images (1 bit per pixel), the only parameter used is the probability to generate a black pixel, varying in the range [0.5 - 1] (the behaviour of the algorithm in the range [0 - 0.5] is specular). If the probability is $p_0 = 0.5$ the output image is most random and less predictable, then hardest to compress. On the other hand, if the probability is nearly 1 then the output image is more uniform, thus more predictable and easier to compress.

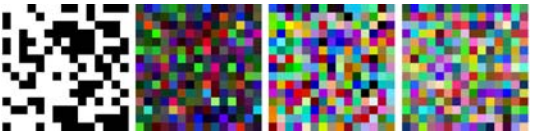

Fig. 3. Randomly generated patterns using (a) 1 bit per pixel with probability $p_i = 0.5$ for white/black value, (b) 8 bits per pixel with probability $p_i = 0.001$ for RGB values between 101 and 255 and $p_i = 0.0085$ otherwise, (c) 8 bits per pixel with probability $p_i = 0.0039$ for each value, and (d) 8 bits per pixel with probability $p_i = 0.001$ for RGB values between 0 and 100 and $p_i = 0.0057$ otherwise.

For images with n bits per pixel, there are 2^n possible colors, each with an associated probability to be randomly generated, given the constraint that the sum of all probabilities must be 1.

$$\sum_{i=0}^{2^n} p_i = 1 \tag{1}$$

Fig.3 shows some examples of randomly generated images, with different number of bit planes, and different probability associated to each color.

The index used to evaluate the experiments is the Compression Percentage, defined as:

$$CompPer = \left(1 - \frac{SizeCompressedImage}{SizeOriginalImage}\right) * 100 \tag{2}$$

The main goal of the experiments is to demonstrate that there are cases where the compressed images are bigger than the original ones. In particular it will be

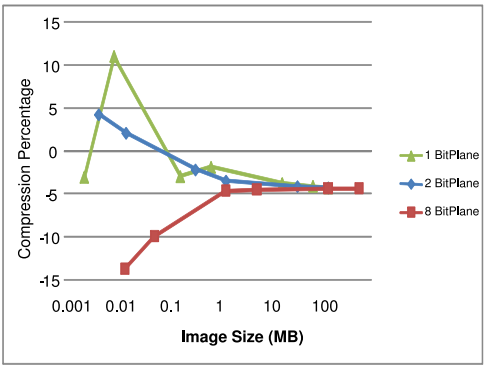

Fig. 4. Standalone test: Compression percentage over randomly generated pattern images using 1, 2 and 8 Bitplanes. The size of the images varying from 100x100 to 5000x5000 pixels.

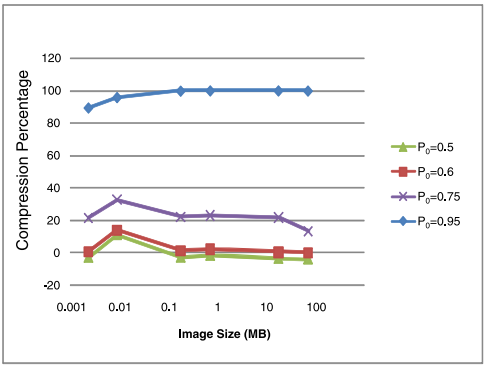

Fig. 5. Compression percentage vs. size of the input image, for black/white images

shown that for large enough images, in the worst case, the output images are 5% bigger than the original ones. This result is independent form the number of bits used and is always true for images not too small, since in such cases the performance is variable and depends on the initial guesses made by the JBIG arithmetic encoder. An example of compression percentage over randomly generated pattern is depicted in Fig.4.

Fig.4 shows that, for large enough images, compression percentage is always a negative number. This is explained by the lack of correlation between pixels in purely random ($p_i = 1/n$ for i in [1..n]) images, and is a well-known behavior of every lossless compression algorithm.

Fig.5 plots the compression percentage against the size of the input image, for bi-level images with different probabilities of a black pixel being generated.

3 JBIG Compression Experiments in a Printing Pipeline

During printing processes, there are cases where it is necessary to keep the whole image in memory, for example when it is requested to make multiple hard copies of a document acquired by the integrated scanner. This can lead to excessive memory requirements, which is why a compression method is adopted. Fig.1 shows a possible printing pipeline. Data received by the JBIG decoder is copied into a dedicated memory area, then decoded and sent to the subsequent pipeline step. As soon as the copy has been completely processed, the JBIG decoder can restart printing the document, which is stored in memory as compressed data.

In a printing pipeline, data received by the final block, controlling printer heads, have very different characteristics than original data: they show much greater correlation and more repeated patterns, introduced (respectively) by the color interpolation of the CIS scanning system and by the halftoning algorithm. This means that the results shown in the previous section 2, where a stand-alone system is taken into consideration, are no longer valid for a printing pipeline (a

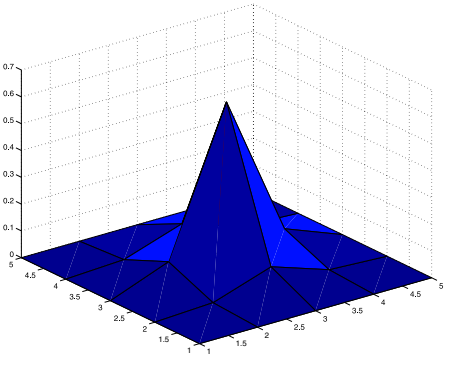

Fig. 6. A 5x5 Gaussian low-pass filter used to simulate the color interpolation of the CIS scanning system

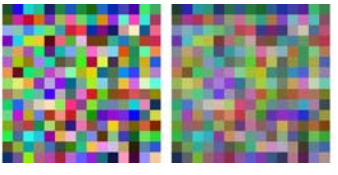

Fig. 7. Difference between pure randomly generated image and blurred randomly generated image

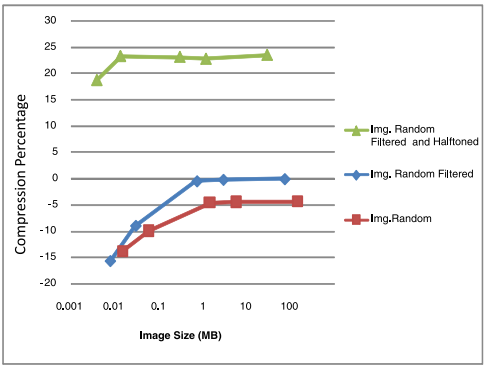

Fig. 8. JBIG compression tests with/without low-pass filter and halftoning over different randomly generated images

random image is not a realistic input in this case). To evaluate the compression system in a printing pipeline, new experiments have been performed: randomly generated images are low-pass filtered (Fig.6), in order to simulate the acquisition

system, and then processed by the pipeline algorithms. Fig.7 shows an example of how the image being processed is blurred compared to the original one.

Results in Fig.8 show how in such cases, even for random images, the JBIG compression percentage is above 20%. This result guarantees that no worst case is encountered: therefore, using the JBIG compression algorithm inside a printing pipeline always allows to save memory. Lastly, Fig.9 shows the compression ratio for randomly generated images processed by low-pass filters with different strengths and apertures. It is easily seen how stronger low-pass filters generate greater correlation, which in turn translates to better compression ratios.

Experiments performed on real use cases showed that JBIG obtained an average 46% compression percentage over a set of 100 scanned photographs and an average 60% compression percentage over a set of 100 text-based documents.

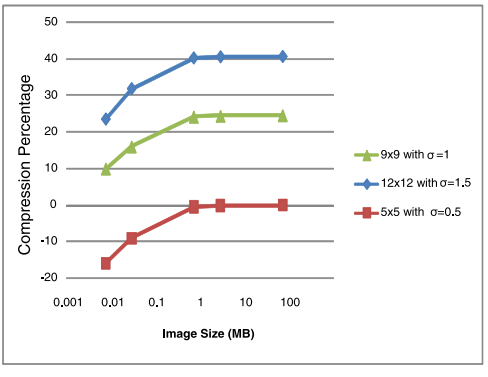

Fig. 9. The compression ratio for randomly generated images processed by low-pass filters with different strengths and apertures

4 Conclusion

The JBIG Encoder/decoder has been implemented into a real pipeline. In particular to be embedded into the original pipeline the following modifications were needed due to the constraints enforced by the pipeline steps:

- Memory dynamically allocated by each step must occupy a contiguous, well-delimited area, due to the sequential processing of the pipeline;
- Each step must run in a separate thread, to be able to perform a realtime scan-print processing.
- Data transmission between steps must be performed via shared buffers, due to the architecture of the workflow.
- Lastly the function standard JBIG_Split_Plane, which only works with 8 bits per pixel images, was rewritten into a new function, which works with any number of bits per pixel.

The optimization of memory buffer size, to store scanned images for multiple local copies, is a critical point in printer pipelines. This work demonstrates that using compressed randomly generated images the memory necessary to achieve such purpose is always greater than original size (worst case: for A4 images at 600 ppi, 5% more). Thus for such cases it is necessary to use stripe processing and multiple scan (without buffering images). On the other hand the statistics demonstrated that in the medium case the compression is always well performed and multiple print could be achieved with a reasonable buffer size.

References

1. Denecker, K., de Neve, P.: A comparative study of lossless coding techniques for screened continuous-tone images. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, April 21-24, 1997, vol. 4, pp. 2941–2944 (1997)
2. Savakis, A.E.: Evaluation of lossless compression methods for gray scale document images. In: International Conference on Image Processing, September 10-13, vol. 1, pp. 136–139 (2000)
3. Yovanof, G.S.: Compression in a printer pipeline. In: Proceedings of the 29th Asilomar Conference on Signals, Systems and Computers, vol. 2, p. 219 (1995)
4. ITU-T T.82 Information technology Coded representation of picture and audio information Progressive Bi-Level Image compression (March 1993)
5. Pennebaker, W.B., Mitchell, J.L., Langdon, G.G., Arps, R.B.: An overview of the basic principles of the Q-coder adaptive binary arithmetic coder. IBM Journal of research and development 32(6), 717–726 (1988)
6. Mitchell, J.L., Pennebaker, W.B.: Software Implementations of the Q-Coder. IBM Journal of Research and Development 32(6), 753–774 (1988)
7. Pennebaker, W.B., Mitchell, J.L.: Probability Estimation for the Q-Coder. IBM Journal of Research and Development 32(6), 737–752 (1988)
8. Mitchell, J.L., Pennebaker, W.B.: Optimal Hardware and Software Arithmetic Coding Procedures for the Q-Coder. IBM Journal of Research and Development 32(6), 727–736 (1988)
9. Anderson, E.E., Wang, W.-L.: Novel contact image sensor (CIS) module for compact and lightweight full-page scanner applications. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, May 1993, vol. 1901, pp. 173–181 (1993)