BURROWS-WHEELER BASED JPEG

***Yair Wiseman*[1*]**

*[*1] Computer Science Department, Bar-Ilan University, Ramat-Gan 52900, Israel*
Email: wiseman@cs.huji.ac.il, http://www.cs.biu.ac.il/~wiseman

ABSTRACT

Recently, the use of the Burrows-Wheeler method for data compression has been expanded. A method of enhancing the compression efficiency of the common JPEG standard is presented in this paper, exploiting the Burrows-Wheeler compression technique. The paper suggests a replacement of the traditional Huffman compression used by JPEG by the Burrows-Wheeler compression. When using high quality images, this replacement will yield a better compression ratio. If the image is synthetic, even a poor quality image can be compressed better.

Keywords: BWT, JPEG, Data Compression

1 INTRODUCTION

One of the basic classical applications of imaging technology is representing images as a set of pixels. Each pixel is defined by a numerical value and is separately manipulated. There are many ways to store image data without degrading the original data. Some of these ways lose part of the data, but it is usually not noticeable. For example the BMP (Luse, 1984) standard is a well known format for image storage without data compression, whereas the GIF (Willard, Zempel, Liv, & Cohn, 1984) standard is a well known format for image storage which uses the LZW (Welch, 1984) algorithm for compressing the image's data.

JPEG (Wallace, 1991; Information Technology, 1993) encodes images using either a version of Huffman coding (Huffman, 1952) or a version of Arithmetic coding (Witten, Neal, & Cleary, 1987). Both of these codes are statistical codes. Usually, compressing texts by one of these techniques yields poor results. Texts of any language have certain rules of words' appearance. For example, in English the letter "q" is almost always followed by the letter "u," and the letter "z" is never followed by the letter "x". Even non-tightly connected languages like Hebrew have a few rules. Huffman coding and Arithmetic coding ignore the rules of words' appearance. Hence, their results are poorer. There are improved versions of Huffman coding and arithmetic coding that have some ability to take the context into consideration; however, the versions of Huffman coding and arithmetic coding that are used by JPEG do not use these versions. Images do not have strict rules like human languages. Thus, apparently the use of a statistical coding cannot be considered as a disadvantage.

This paper claims that even though the data sequence in images does not recur precisely in the same format, there are some characteristics of data sequence which recur. The temporal correlation which is captured by the DCT frequency coefficient in JPEG is not taken into account by current image compression methods implemented by JPEG. This correlation is left out; therefore valuable information is omitted. Hence, exploiting this information can improve the compression efficiency. Similar strategy is reinforced by the use of a dictionary compression method, e.g. by the GIF standard and by the PNG standard. GIF uses LZW, and PNG uses LZ77; both get some nice results. The idea of optimizing JPEG has been suggested by some researches e.g. (Westen, Lagendijk & Biemond); however, their ideas were different and did not take into account the temporal correlation.

2 THE JPEG STANDARD

JPEG is a well known standardized image compression technique. JPEG loses information, so the decompressed picture is not the same as the original one. By adjusting the compression parameters, the degree of loss can be adjusted. The wide use of JPEG stems from two fundamental reasons: it reduces the size of image files and stores full color information.

Reducing image files is an important procedure for transmitting files across networks or archiving libraries. Usually, JPEG can remove the less important data before the compression; hence JPEG is able to compress images meaningfully, which produces a huge difference in the transmission time and the disk space.

The second advantage of JPEG is its capability of storing full color information: 24 bits/pixel or 16 million colors, while for example, the GIF format can store only 8 bits/pixel or 256 colors.

Here is a brief overview of the JPEG algorithm:

At the first step, the algorithm transforms the image color into a suitable color space. There are several methods for transforming the image into a color space (Hearn & Baker, 1986; Jain, 1986) The most common methods are the split into RGB components (Hunt, 1995) and the split into YUV components (LaPlante & Stoenko, 1996). These components are interleaved together within the compressed data. The ratio between these components is usually not one to one. When YUV components are used, usually the Y component is weighted by a factor of four. The human eye is less sensitive to the frequency of chrominance information than to the frequency of the luminance information represented by the Y component in the YUV format. Hence, the Y component gets a higher weight (Awcock, 1996).

At the second step the algorithm groups the pixels into blocks of 8X8 pixels. Then, it transforms each block through a Forward Discrete Cosine Transform (FDCT) (Rao & Yip, 1990). The DCT gives a frequency map, with 8X8 or 64 elements. The transformation keeps the low frequency information which a human eye is sensitive to. In each block the DCT coefficients are composed of:

- A single Direct Current (DC) coefficient number, which represents the average intensity level value in each block.
- The remaining 63 named Alternating Current (AC) coefficients, which reflect the frequency information of their row and column.

The next step is the *quantization*. The 63 AC coefficients are ordered into a zigzag sequence which arranges them into a one dimensional array. In each block, each of the 64 frequency components is divided by a separate "*quantization coefficient.*" The quantization coefficients are set according to the desired image quality. The results of the division are rounded to integers. This step loses some information because of the rounding. Furthermore, it can be noted that even if the quantization coefficient is 1, some information will be lost because typically the DCT coefficients are real numbers.

At the last step the algorithm encodes the reduced coefficients using either Huffman or Arithmetic coding. Usually a strong correlation appears between DC coefficients of adjacent 8X8 blocks. Therefore, JPEG encodes the difference between each pair of adjacent DC coefficients. The baseline JPEG model uses two different Huffman trees to encode the data, one for the DC coefficients' length and the other for the AC coefficients' length.

Finally, the compression parameters are written in the file header, so that the decoder module will be able to decode the compressed image. JPEG's procedure is summarized in Figure 1.

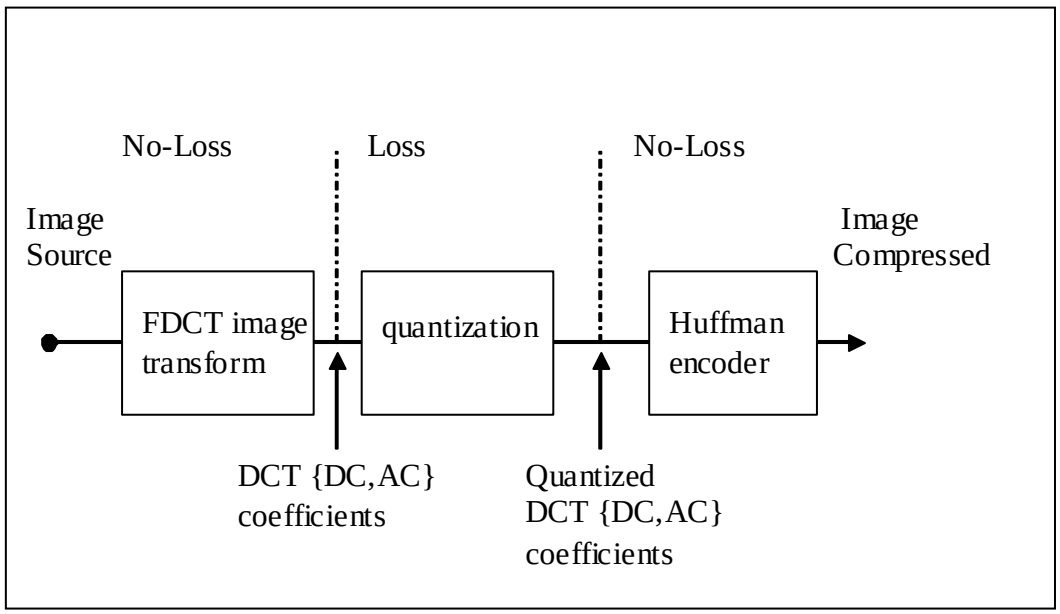

Figure 1: JPEG Model for a lossy image compression

The decompression process performs an inverse procedure. It decompresses the Huffman or the Arithmetic codes. Then, it makes the inversion of the Quantization step. In this stage, the decoder raises the small numbers by multiplying them by the quantization coefficients. The results are not accurate, but they are close to the original numbers of the DCT coefficients. Finally, an Inverse Discrete Cosine Transform (IDCT) is performed on the data received from the previous step.

JPEG has some disadvantages. Unfortunately, even with a JPEG viewer, it takes a longer time to decode and view a JPEG image than to view an image of a simpler format such as GIF, BMP, etc. Another disadvantage is the compression method that does not take into account the temporal correlation of the coefficients.

3 THE BURROWS-WHEELER TRANSFORMATION

The Burrows-Wheeler TransformationBurrows & Wheeler, 1994; Nelson, 1996) is a context based method. The method utilizes repetitions of words' sequences in order to compress better, similar to dictionary methods. The Burrows-Wheeler Transformation does not lose any information in the compression procedure.

The traditional dictionary compression methods are the Lempel Ziv methods. WINZIP (Winzip, 1998) of Windows and (gzip, 1991) of UNIX use versions of the Lempel Ziv coding. Usually, the Burrows-Wheeler Transform gives better results than the versions of the Lempel Ziv coding. Hence, some Burrows-Wheeler compression utilities have been implemented for many environments (Manzini, 1999), and nowadays the new text compression utilities are based on the Burrows-Wheeler algorithm e.g. Zzip (Debin, 2002) of Windows and bzip2 of UNIX (SGI® IRIX®, 2003). The main deficiency of the Burrows-Wheeler Transform is the long execution time.

The Burrows-Wheeler Transform has these basic steps:

At the first step the algorithm puts pointers to the file at each character. Then, the pointers are sorted according to the characters that they are pointing to. The preceding characters of each of the pointers are sent to the next step according to the order of the sorted pointers. Actually, this sequence of characters in the output has the same characters as the original file, but the order of the characters is different.

At the second step the algorithm performs the "move to front." procedure (Elias, 1987). This procedure keeps all the 256 possible characters in a list. Each character in the sequence is read and its position in

the list is sent. After the character is 'sent', it will be moved from its current position in the list to the front of the list (i.e., to position 0).

At the next step the algorithm applies a run-length coding to the output of the previous step. The output of the run-length coding is compressed by Huffman coding or Arithmetic coding.

The Burrows-Wheeler compression is quite slow because of the sorting, so usually the file is split into some blocks in order to reduce the execution time. However, reducing the block size might cause a worse compression because there will be fewer repetitions of strings. The Burrows-Wheeler compression removes the temporal correlation, and when there are fewer strings in a block, it will be harder to detect the correlation.

4 BURROWS-WHEELER BASED JPEG

Even though the repetitions of items' sequences in images do not exactly recur, the characteristics of the sequences in images repeat on themselves. Particularly, in JPEG images the AC coefficients in the beginning of a block are usually higher than the AC coefficients in the end of a block. In fact, most of the last AC coefficients in a block are zero.

In (Baik, Ha, Yook, Shin, & Park, 1999; Baik et al., 1999), the authors suggest checking the "degree of repetition." If the degree of repetition is high enough, they suggest omitting the run-length encoding and the entropy encoding of JPEG. Instead, they suggest applying the Burrows- Wheeler Transform. However, their definition of "degree of repetition" is unclear. Moreover, the motivation for the omission of JPEG's run-length coding is unclear. JPEG's run-length coding is designed to deal with the significant number of zeros usually appearing in JPEG and their tendency to be located in the end of the block. Burrows-Wheeler's run-length coding does not have this advantage. Apparently, this replacement can damage the compression. Thus, we did not succeed in obtaining good results using their method.

We would like to suggest another algorithm:

- Transform the image color into a suitable color space.
- Apply DCT.
- Perform the quantization procedure.
- Instead of assigning the Huffman codes to each DC value and each pair of zeros' length and an AC value, assign a number to each of them.
- Process the above numbers by the Burrows-Wheeler transform.
- Process the output by Move-to-front procedure and Run-length coding.
- Finally, compress by Arithmetic coding.

The concept of this technique is performing the DCT and the quantization as they are done in the original JPEG algorithm. Then, we take the run-length scheme of JPEG including the zigzag order, but instead of giving variable length codes as in Huffman coding, we give fixed length codes. These fixed length codes are the input of the Burrow-Wheeler compression. The suggested algorithm is depicted in Figure 2.

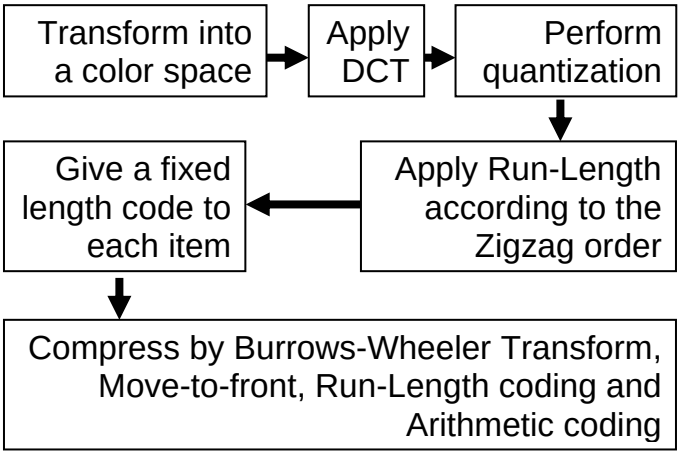

Figure 2: The suggested algorithm

5 WHY CAN BURROWS-WHEELER BE APPLIED?

In JPEG format the source image samples are grouped into 8X8 blocks and shifted from unsigned integer to signed integer. Then Forward DCT is applied to them. The DCT-based compression views the FDCT as a harmonic analyzer and the IDCT as a harmonic synthesizer. In JPEG format, each 8X8 block of the source image samples is effectively a 64-point discrete signal, which is a function of the 2D dimensional space x and y. The FDCT takes such a signal as its input and decomposes it into 64 orthogonal basis signals. The output of the FDCT is the set of unique 64 basis signal amplitudes, which can be regarded as the relative amount of the 2D spatial frequency contained in the 64-point input signal.

The results of these mathematic calculations are treated as integers. Moreover, usually, some number divides these results, and the quotients are accommodated as integers. This dividing and casting into integers is called "quantization".

Obviously, rounding floating-point numbers to integers causes some loss of data, so the values of JPEG's coefficients are inaccurate; however, objects in reality tend to have low frequencies values. Therefore, long strings with low values may recur. Since the values are inaccurate, small differences will be omitted, and as a result, many strings will be totally equal.

The output of the quantization goes into the Burrows-Wheeler Transform. The Burrows-Wheeler algorithm will achieve a better compression ratio because of the equal strings, so the inaccuracy of the data becomes an advantage.

In (Guo & Burrus, 1997) the authors suggest applying Burrows-Wheeler to a wavelet transformation similar to the DCT of JPEG but without a quantization. As explained above this yields worse results.

It should be noted that unlike Huffman coding or arithmetic coding, Burrows-Wheeler Transform has the ability to consider the context inside the processed block. The sorting stage makes Burrows-Wheeler Transform a context-aware compression, whereas Huffman coding treats each value separately.

6 EXPERIMENTAL RESULTS

This method has been tested on the images seen in Figure 3. The original images are very large e.g. the upper left image is 5MB in BMP format. The images were compressed by the original JPEG method and by the Burrows-Wheeler based JPEG.

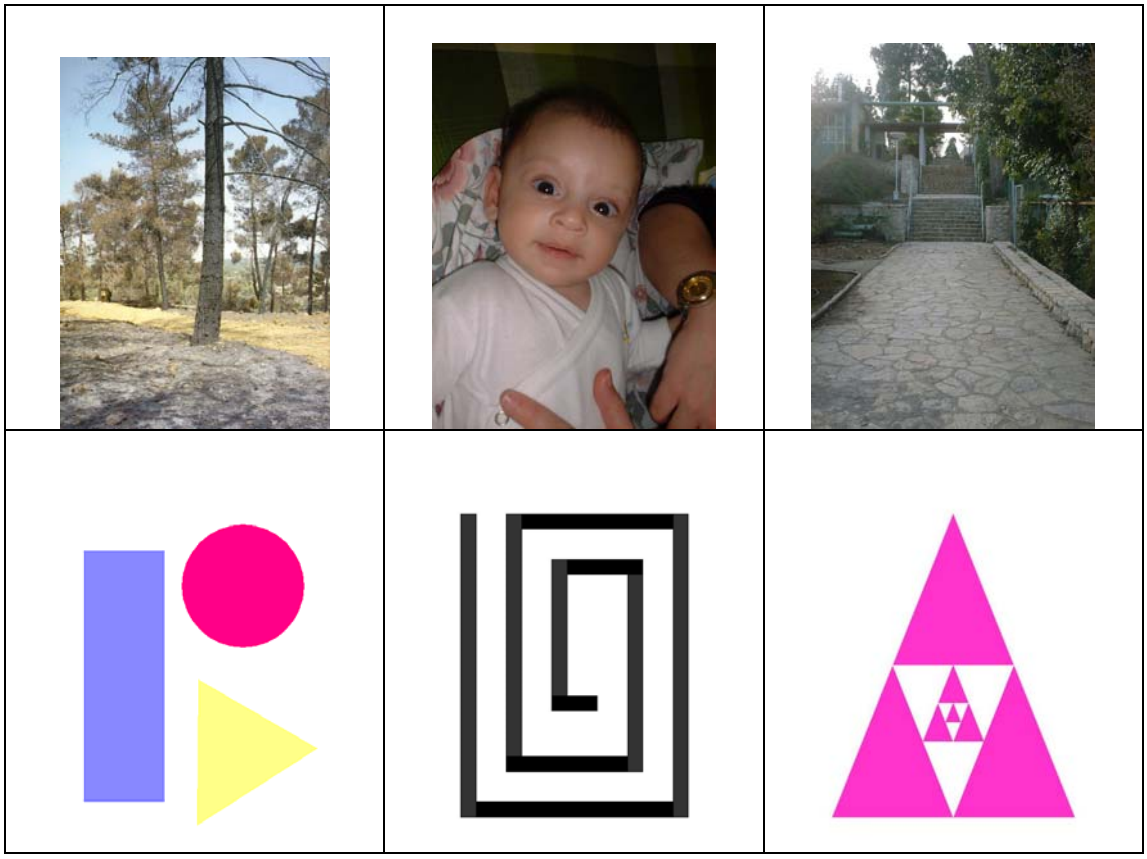

Figure 3: The tested images

This version of the Burrows-Wheeler Transform was taken from (SGI® IRIX®, 2003). The results of the compression efficiency are given in Figure 4. The results are of the upper images in Figure 3, but almost the same results have been obtained when using other real life images. The chunks of data, which have been used by the Burrows-Wheeler Transformation, were from 100KB to 900KB. The size of chunks has no influence on Huffman coding, so the original JPEG compression is not influenced by the different sizes. However, Burrows-Wheeler is influenced by the size of chunks, and the influence can be seen in the results. 90, 95, and 100 are the JPEG parameters of the image quality. Obviously, reducing the quality of an image will diminish the image size.

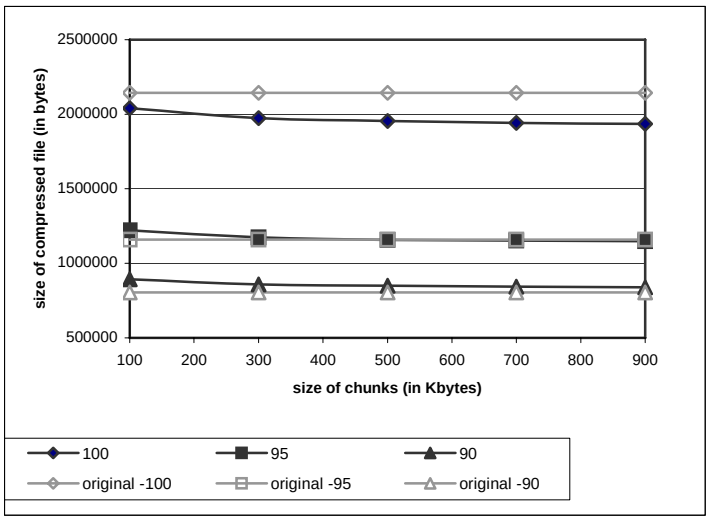

Figure 4: Compression efficiency of real life images

When using larger chunks, the compression obviously will become better, whereas using smaller chunks will yield poorer compression. When the image quality is below 90, the size of the compressed blocks will become too small. Most of the data will be long lists of zeros that will be represented by a few bits. This leads to very short strings of data. Hence fewer repetitions occur, and thus the compression becomes poorer. In such cases the original Huffman should be applied in order to get better results. The Burrows-Wheeler based JPEG should be applied only for high quality real life images.

It should be noted that real life images usually yield poor compression ratio if high quality is needed. Original JPEG with the highest parameter of quality – 100, reduces the image size to 41.86% of the original size, whereas Burrows-Wheeler based JPEG reduces the image size to 37.79% of the original size. Both of them are a lossy compression method. Comparing these results to lossless compression methods like PNG (Duce et al., 2004) shows that the loss of data is still worthy. PNG (Deflate) reduces the image size to only 94.20% of the original size.

JPEG-2000 (Information Technology, 2004) is a newly published standard. However, as shown in (Ebrahimi, Chamik, & Winkler, 2004), high-quality images produce better compression ratios using the traditional JPEG. JPEG-2000 is used for fair quality and poor quality images. Since our tool is aimed for high quality images, we use the traditional JPEG. Moreover, the traditional JPEG is still widely used.

In (Lane, 1999) the author claims that arithmetic coding based JPEG creates 5 to 10 percent smaller files than Huffman coding based JPEG. These results have been obtained using the Q-Coder (Michael & Pennebacker, 1988)] – a specific variant of arithmetic coding employed by JPEG. Unfortunately, this variant is subject to a patent of IBM, AT&T, and Mitsubishi. In addition, the arithmetic coding can save 5 to 10 percent when the picture quality is not high because low quality makes the variety of AC elements smaller, and in such a case the arithmetic coding can almost reach entropy, whereas Huffman coding can reach entropy only when there exists a big variety of AC elements as explained in (Bookstein & Klein, 1993). Our paper aims to improve the compression of high quality pictures where there is no big difference between Huffman coding and arithmetic coding.

Unlike real life images, synthetic images have many more repetitions; hence the results are significantly better even for poor quality images. The lower images results in Figure 3 can be seen in Figure 5. The results are better, even when using small chunks for Burrows-Wheeler, and there seems to be a near constant improve in the compression. This suggests that there are strong temporal correlations in the zigzag scans of the synthetic data. The results are of the traditional JPEG; however, JPEG-2000 will compress almost the same way when the compressed image contains no fine details and each shape within the image is of one color (Information Technology, 2004).

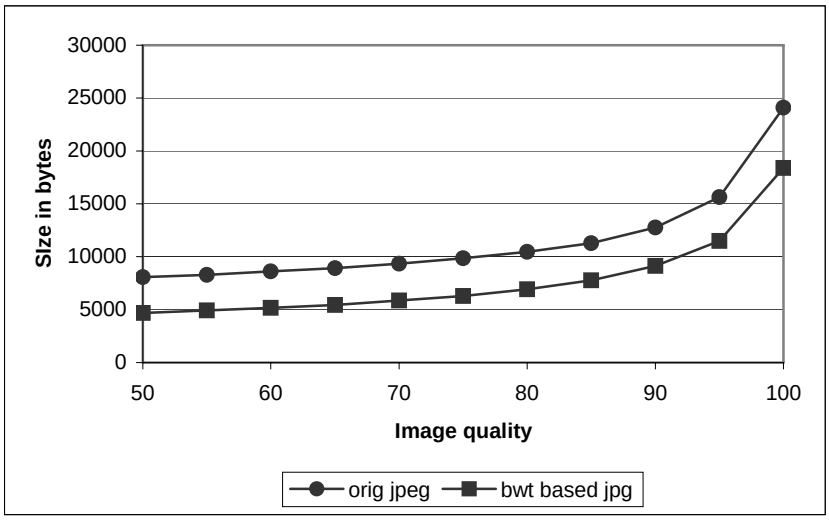

Figure 5: Compression efficiency of synthetic images

The lossless PNG compression technique (Duce et al., 2003) is well suited for synthetic images, and PNG indeed outperforms both the original JPEG and Burrows-Wheeler based JPEG if JPEG specifies

high quality images. In point of fact, the original JPEG has to specify at most 78 as its parameter of quality, and Burrows-Wheeler based JPEG has to specify at most 92 as its parameter of quality in order to get better results than PNG.

The main deficiency of Burrow-Wheeler based JPEG is the execution time. Burrow-Wheeler is much more time consuming than Huffman. The real life images in Figure 3 can be compressed in about 3 seconds by the original JPEG on a SUN Ultra-4 Sparc machine. Burrows-Wheeler based JPEG will compress the same data in about double time. The decompression, however, consumes roughly just one second, which is expedient, since usually an image is created just one time but is seen several times.

7 CONCLUSIONS

The results of the experiments are promising. High quality images are very important for some implementations such as astronomic data or military data. In such cases Burrows-Wheeler based JPEG can yield better results. The synthetic images do not require high quality. Synthetic images are very common e.g. in companies' logos. The Burrow-Wheeler based JPEG could be used to reduce the size of Internet sites with synthetic images. The main deficiency of the Burrows-Wheeler based JPEG is the long compression time. This problem can be addressed by faster implementations of the Burrows-Wheeler transformation.

8 REFERENCES

Awcock, G. J. (1996) *Applied Image Processing*, McGraw-Hill Book Company, pp. 260-261.

Baik, H. K., Ha D. S., Yook, H. G., Shin, S. C., & Park, M. S. (1999) Selective Application of Burrows-Wheeler Transformation for Enhancement of JPEG Entropy Coding, *Proceedings of International Conference on Information, Communications & Signal Processing, PN.230.*

Baik, H. K., Yook, H. G., Shin, S. C., Park, M. S., & Ha, D. S (1999) A New Method to Improve the Performance of JPEG Entropy Coding Using Burrows-Wheeler Transformation, *International Symposium on Computer and Information Sciences*, pp. 502-509, Turkey, October 1999.

Bookstein, A. & Klein, S. T. (1993) Is Huffman coding dead? *Computing 50* pp. 279-296.

Burrows, M. & Wheeler, D. (1994) Block sorting Lossless Data Compression Algorithm, *System research center, research report 124, Digital System Research Center, Palo Alto, CA.*

Debin, D. (2002) Retrieved February 25, 2007 from the World Wide Web:
http://debin.net/zzip/index.php

Duce, D. et al (2003) Portable Network Graphics (PNG) Specification (Second Edition), *Information technology ISO/IEC* 15948:2003, Oxford Brookes University.

Ebrahimi, F., Chamik, M., Winkler, S. (2004) JPEG vs. JPEG2000: An objective comparison of image encoding quality, *Proceedings of SPIE Applications of Digital Image Processing, vol. 5558, pp. 300-308*, Denver, CO, August 2-6.

Elias, P. (1987) Interval and recency rank source coding: two on-line adaptive variable-length schemes, *IEEE Transactions on Information Theory, Vol. 33* pp. 3-10.

Guo, H. & Burrus, C. S. (1997) Waveform and Image Compression Using the Burrows Wheeler Transform and the Wavelet Transform, *Proceedings 1997 International Conference on Image Processing (ICIP '97)*, pp. 65-68, Washington, DC.

Gzip (1991) *Free Software Foundation, Inc*., 675 Mass Ave, Cambridge, MA, USA.

Hearn, D. & Baker, M. P. (1986) *Computer Graphic* Prentice Hall, Englewood Cliffs, NJ, pp. 295-307.

Huffman, D. (1952) A method for the Construction of Minimum Redundancy Codes *Proc. of the IRE 40, pp. 1098-1101*.

Hunt, R. W. G. (1995) *The Reproduction of Colour* Fountain Press England, pp. 507-511.

Information Technology (1993) Digital Compression and Coding of Continuous-Tone Still Images Requirements and Guidelines *International Standard ISO/IEC 10918-1*.

Information Technology (2004) *JPEG 2000 Image Coding System*, International Standard ISO/IEC 15444-1:2004.

Jain, A. K. (1986) *Fundamental of Digital Image Processing* Prentice Hall Information and Sytem Sciences Series, pp. 553-557.

Lane, T. (1999) JPEG image compression FAQ, Retrieved February 25, 2007 from the World Wide Web: http://www.faqs.org/faqs/jpeg-faq/part1/

Laplante, P. A. & Stoyenko, A. D. (1996) *Real Time Imaging, Theory, Techniques and Applications* IEEE Press Inc. NY, pp. 122-124..

Luse, M. (1994) The BMP File Format *Dr. Dobb's Journal, Vol 9, Issue 10, pp. 18-22*.

Manzini G. (1999) The Burrows-Wheeler Transform: Theory and Practice, *Lecture Notes in Computer Science, Springer Verlag, Volume 1672, pp. 34-47*.

Michel, J. L., & Pennebaker, W. B**.** (1988) Software Implementation of the Q-Coder, *IBM J. Research and Development, vol. 32*, pp. 753-774.

Nelson, M. R. (1996) Data Compression with the Burrows Wheeler Transformation, *Dr. Dobb's Journal, pp. 46-50.*

Rao, K. R. & Yip P. (1990) Discrete Cosine Transform Algorithms, Advantages, *Applications Academic Press Inc.*, London 1990.

SGI® IRIX® (2003) Freeware distribution, 1600 Amphitheatre Pkwy. Mountain View, CA, USA, Edition of February 2003.

Wallace, G. K. (1991) The JPEG Still Picture Compression Standard *Communication of the ACM 34, pp. 3-44*.

Welch, T. A. (1984) A Technique for High-Performance Data Compression *IEEE Computer Society* 17 pp. 8-19.

Westen, S. J. P., Lagendijk, R. L., and Biemond, J. (1996) Optimization of JPEG color image coding using a human visual system model. Proceedings of the SPIE, Vol. 2657, pp. 370-381.

Willard, L., Lempel, A., Ziv, J., & Cohn, M, (1984) Apparatus and method for compressing data signals and restoring the compressed data signals *US patent - US4464650.*

WinZip (1998) *Nico Mak Computing, Inc.*, Mansfield, CT, USA.

Witten, I. H., Neal, R. M., & Cleary, J. G. (1987) Arithmetic Coding for Data Compression, *Communication of the ACM 30, pp. 520-540.*

.