Compression of aerial images for reduced-color devices

Pasi Fränti and Ville Hautamäki

Department of Computer Science, University of Joensuu
Box 111, FIN-80101 Joensuu, FINLAND

ABSTRACT

Mobile devices are more often capable for locating the user on the globe by using GPS or network-based systems. The location is given to the user in a meaningful context such as map or aerial image. We study compression methods for reducing the amount of data required by aerial images. We consider the two well known lossless graphics file formats GIF and PNG, the compression standards JPEG and JPEG2000, a fractal compressor known as FIASCO, and commercial wavelet-based method developed for aerial images. It is also expected that the devices support fewer than 256 gray levels. It is therefore possible to get further reduction by quantizing the images prior to compression, for example, to two-bit per pixel, and then apply lossless compression methods such as JBIG, GIF and PNG. For four color mobile displays error diffusion dithered images approximate the original 8-bit color images quite well. The trade-off in dithering is that the lossless compression ratios decrease. Solution to this might be to store the 8-bit images compressed by using a decent lossy compressor such as JPEG2000. Quantization and dithering should happen only at the moment when the image is displayed.

Keywords: Aerial images, image compression, mobile imaging, JBIG, JPEG, JPEG2000.

1. INTRODUCTION

Aerial images are photographs of terrain taken from air or from satellite. Conventional maps can be represented in digital form as vector graphics or as bi-level bitmap. Aerial images, on the other hand, need more color information. The mobile devices, however, have limited storage and reproducing capabilities. It is therefore important that the image can be compressed efficiently.

In this report, we study different compression methods for reducing the amount of data required by aerial images. The original images are 8-bit per pixel (bpp) gray-scale images, which give plenty of choices for the compression methods. We consider the two well known lossless graphics file formats GIF [4] and PNG [20], the current compression standards JPEG [21] and JPEG2000 [10], and a fractal compressor known as FIASCO [19], and commercial wavelet-based method ECW developed for aerial images [16].

Impact of the source data suitability for different compression methods can be seen especially from poor performance of lossless GIF. The PNG method is supposed to achieve always slightly better results than the popular GIF. In this case, GIF produced unacceptable larger-than-original compression ratio, while PNG files were about 2/3 of the size of GIF. In lossy category, wavelet-based JPEG2000 clearly outperformed JPEG, FIASCO and ECW by a wide margin.

For four color mobile displays error diffusion dithered images approximate the original 8-bit color images quite well. The trade-off in dithering is that the lossless compression ratios decrease. Solution to this might be to store the 8-bit images compressed by using a decent lossy compressor such as JPEG2000. Quantization to 2-bit and dithering should happen only at the moment when the image is displayed. This approach is demonstrated in Figure 1. The choice of the compression method always heavily depends of the details of source data, as is also seen in this case. If the source data would have been pre-compressed with different method, or not compressed at all, or if it had different noise factors than the present artifacts from JPEG method, compression results might be slightly different. Nevertheless, the JPEG2000 was still the best method for 8-bit images and possibly also for 2-bit images.

The rest of the report is organized as follows. In Section 2, we briefly recall the compression methods and give pointers where sources can be found. In section 3, the test image set is described. Compression results with 8-bit and with 2-bit images are summarized in Section 4. Conclusions are drawn in Section 5.

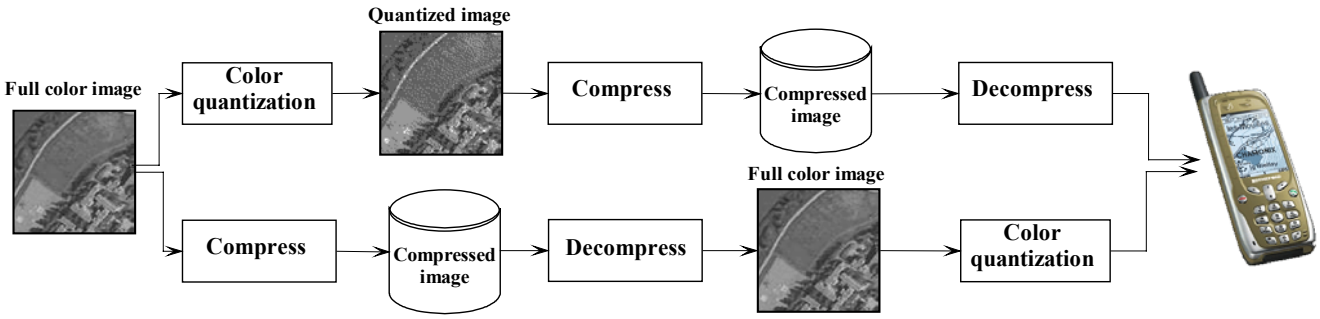

Figure 1: System diagram of using color quantization and compression in mobile device.

2. COMPRESSION METHODS

Image compression can generally be divided into two categories: lossy and lossless. Lossless compression means that the image after compression and decompression will be identical. The compression result can be evaluated simply by calculating the compression ratio, which is the number of-bit in the input image file divide by the number of-bit in the compressed image file.

We study the following compression methods: JPEG, JPEG2000, FIASCO, ECW, PNG, GIF and JBIG, see Table 1. JPEG2000 can be applied both in lossy and lossless mode. In this evaluation we use only the lossy mode. Pointers to the sources of the compression methods are summarized in Table 2.

Many lossy encoders do not allow user to set the output bit rate before the compression. This makes it difficult to generate truly objective test results. In this evaluation, JPEG and FIASCO represent such methods, in which it is not possible to set the exact bit rate where as in JPEG2000 this is possible. With FIASCO it is impossible to generate bit rates much higher than 0.6 as it is very greedy on memory. ECW, on the other hand, cannot produce bit rates smaller than 1.6.

Table 1: Summary of compression methods used in the comparison.

	Lossy	Lossless	Compression method
JPEG	×		DCT + Huffman
JPEG2000	×		Wavelet + EBCOT
FIASCO	×		Fractals
ECW	×		Wavelet
GIF		×	LZW
PNG		×	LZ77 + Huffman
JBIG		×	Context model + QM-coder

Table 2: Software used in the evaluation.

	Codec	Reference	Date
JPEG	libjpeg6b	http://www.ijg.org	2000-08
JPEG2000	JJ2000	http://JPEG2000.epfl.ch	2000-07
JPEG2000	JasPer	http://www.ece.uvic.ca/~mdadams/jasper/	2000-07
FIASCO	FIASCO	http://ulli.linuxave.net/fiasco	2000-07
ECW	ECW	http://www.ermapper.com	2000-07
GIF	Image Alchemy 1.11	http://www.handmadesw.com	2000-07
PNG	PaintShop Pro 6.02	http://www.jasc.com	2000-08
JBIG	jbigkit 1.2	http://www.jbig.org	2000-08

2.1 JPEG

JPEG divides the image into 8×8 pixel blocks, which are compressed separately. The blocks are first processed by *discrete cosine transform*, and the resulting coefficients are then quantized. The quantized coefficients are coded using run-length modeling and Huffman coding.

The images were coded using the *libjpeg6b* library of the Independent JPEG Group (IJG). Quantization tables are computed with IJG library a bit differently when compared to the strategy, where each quality setting has its own quantization table. IJG scheme works like this: quantization table is taken from JPEG specification (section K.1.) and the coefficients in this table have been scaled for producing tables for different quality factor. For example, quality factor 100 corresponds to scaling with factor 0, and quality factor 0 to scaling with factor 100.

2.2 JPEG2000

In JPEG2000, the image is first processed by *wavelet transform*, which is recursively applied to the low-pass coefficients until desired decomposition level is reached. Resulting coefficients are then divided into code blocks. These code blocks are quantized and coded using arithmetic coder. Binary code blocks produced by arithmetic coder are then reordered into quality layers by fractional bitplane coder. General view of the JPEG2000 architecture is illustrated in Figure 2.

The first wavelet found was the *Haar wavelet* [1] by JPEG2000 uses the Daubechies 7/9 biorthogonal wavelet filter. *The mallat ordering* [17] of the transformed subbands is the most common decomposition method. Other possible methods include the *Spacl method*, which also can be used with JPEG2000. The HH subband includes diagonal high-pass coefficients, LH subband includes vertical high-pass coefficients, LH includes horizontal high-pass coefficients, and LL subband includes low-pass coefficients. Most of the information content in the transformed image is in the low-pass subband.

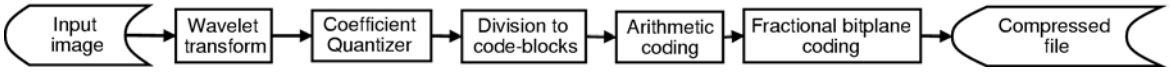

Figure 2: General view of JPEG2000 encoding architecture.

Wavelets-based image compression algorithms suchs as *Embedded zerotree wavelet* (EZW) [11] and *Set Partitioning in hierarchical trees* (SPIHT) [3] create *SNR scalable-bit*tream. The SNR scalable feature in-bittream signifies that we can stop decoding the image at any user definable bit rate. EZW and SPIHT algorithms try to minimize distortion at every bit rate.

The Embedded block coding with optimized truncation (EBCOT) algorithm [6] used in JPEG2000 is both resolution and SNR scalable. Resolution scalability in EBCOT uses the feature in mallat ordering, where low-pass subband coefficients actually represent image in some resolution. In EBCOT, each subband is divided into code blocks, and these code blocks are then coded individually. EBCOT also set's finite number of truncation points in each code block. Output-bittream is then a collection of different truncated code blocks, which are further divided into smaller sub-blocks. Symbols are coded with four different coding primitives: *Zero coding* (ZC), *Run-length coding* (RLC), *Sign coding* (SC) and *Magnitude refinement* (MR). Each primitive has it's own arithmetic coder contexts. Binary arithmetic coder is known as the *MQ-Coder*.

EBCOT codes each bitplane of the code block in four passes. This creates passes $P_1..P_4$ so that P_1 contains the first samples and P_4 the last samples of the code block on a given bitplane. Thus, it creates four more truncation points inside one code block. This four pass coding is identified as the *fractional bitplane coding* [8].

General introduction to JPEG2000 standard can be found in [2]. Analytical study of the JPEG2000 features compared to similar features in other still image compression standards is available in [5]. In the evaluation, we use two different encoders: *JJ2000* and *Jasper*. The JJ2000 codec is written by The JPEG Organization, and The Jasper codec by Michael D. Adams. We refer JJ2000 as J2K, and the Jasper codec as JP2. The compression results between these two codecs were approximately equal, and therefore we report only the results of JJ2000.

2.3 FIASCO

FIASCO (Fractal Image And Sequence COdec) codec is a fractal compressor based on *weighted finite automata* (WFA) originally developed by K. Culik and J. Kari [14]. In WFA, the input image is first split into non-overlapping blocks. Each block is then encoded using sub-images from *domain pool*. Blocks are represented as approximations of weighted linear combination of selected images from the domain. This splitting process is recursively applied until a decent approximation is found. The weights are quantized to integer values and then coded using arithmetic coder. The resulting approximation is then added to the domain pool.

With FIASCO codec, it is nearly impossible to generate high bit rates. For *Turku2* image, even the bit rate 0.5 was too high for FIASCO. This limitation is due to repeated split process, which results to larger and larger domain pool with the increased accuracy. This results in an overwhelming increase in memory and cpu usage. Lower bit rates generate similar blocking artifacts as the baseline JPEG with similar bit rates. This is maybe acceptable for low-bit rate video coding but not for good quality still image coding.

2.4 GIF

The GIF method [4] has been developed by Compuserve and it is patented by Unisys until 2004. In GIF, color images are converted into 256 color palette images. As we use 8-bit gray-scale images, GIF can be directly applied without any conversion The image is then coded using a variant of the LZ78 algorithm [13], which is known as LZW [18]. This compression method is lossless. The images were compressed to GIF using *Image Alchemy* version 1.11 with the -g option.

2.5 PNG

PNG is similar method to GIF. Gray-scale images are converted into 256 color palette images. This indexed image is then coded using a variant of the LZ77 algorithm [12]. The PNG method is also lossless. The images were compressed to PNG using *Paint Shop Pro* version 6.02.

2.6 JBIG

JBIG, is a lossless bi-level (1-bit, black and white) image coding standard by the *Joint Bi-Level Image Experts Group* (JBIG). JBIG format is formally defined in the *ITU-T Recommendation T.82*, International Standard ISO/IEC11544. JBIG algorithm uses pixel-by-pixel context-based statistical modeling and arithmetic coding using the coder known as the *QM-Coder*.

Gray-scale images can be compressed with JBIG, by converting them first to eight 0/1-bit planes (in case of 8 bpp images), or two 0/1 bit planes (in case of 2-bit images). Individual bit planes were compressed using pbmtojbg software from jbigkit package.

2.7 ECW

ER Mapper Compressed Wavelet (ECW) images in this test were created using free command line compressor/decompressor (Copyright Earth Resource Mapping Pty Ltd) from (http://www.ermapper.com) with -g -e best option using different target ratio values ranging from 10:1 to 400:1 and decompressed for PSNR-comparison using PaintShop Pro ECW plug-in. The ECW plug-ins are available for various image and map processing software as well. With our test images, the ECW tool did not always achieve the target ratio and produced images only at the rate 1.6 ... 3.3 bpp (5:1 ... 2:1). Larger bit rates were not interesting for this study, and therefore not examined. Lower bit rates were not possible.

ECW format is a wavelet-based [16] lossy compressor, and has been developed for storing large map images with georeference and locational information. ECW pictures are 8-bit gray scale images or 24-bit color images with RGB-channels YUV-coded. ECW is able to compress also 1- or 2-bit images but as wavelet-based it probably is not at it's best in it. The method does not suite well for small images (< 2 MB) but it supports images larger than 2 GB and level-of-detail decompression [7].

3. TEST IMAGE SET

Four test images were downloaded from the web server of the city of Turku, Finland: (http://opaskartta.turku.fi/). The pictures are 24-bit JPEG images shown in Figure 3. The images were therefore actually compressed already once, which *might* have some effect on the compression results as the baseline JPEG creates small artifacts into the image. These artifacts are small changes in the pixel values around the image, and thus they are difficult to observe visually.

Turku1 *Turku2* *Turku3* *Turku4*

Figure 3: The set of test images. Each of the images are of size 600×450 pixels represented by 8-bit per pixel.

Representing the colors by computer is limited by storage space and display device. Typical desktop monitor has 24-bit (16,777,216) color space, which is considered enough to show smooth transition of colors. For gray-scale images 8-bit (256 tones) is usually considered enough for human eye. However, perfect result is not needed for most applications and hand-held devices might be able to display less colors. We consider 4-color display (2-bit per pixel).

Color reduction from 256 to 4 gray scales reduces storage space to 1/4 of the original image. Linear color reduction makes the color mapping for each pixel individually. Better visual quality can be obtained using *dithering methods*. The tradeoff is that the dithering affects also the compression methods worsening the compression performance. Moreover, most efficient compression are achieved by lossy methods (JPEG, JPEG2000), and they work efficiently only with images that have large color space (at least 64 tones per color).

On the other hand, 2-bit images can be divided into two 1-bit images and apply lossless image compression (such as JBIG) that are designed for 1-bit images. It is expected that this produces better compression result. Other methods designed to work with all images with 1-8 bit color space (GIF, PNG), should also considered for compressing the bit planes.

The dithering method used in this test is the *error diffusion*, method known as the *Floyd-Steinberg dithering* [15]. Linux program called as *The Gimp* version 1.1.24 (http://www.gimp.org) was used to generate the gray-scale images from the original color image using the option "*Floyd-Steinberg*" with "*nobleed*". Enlargements of the test images and their processed 2-bit versions are shown in Figures 4, 5, 6 and 7.

4. COMPRESSION RESULTS

4.1 Lossless results for 8-bit images

Lossless compression results are summarized in Table 3. We can see that the dictionary-based lossless methods (GIF, PNG) work quite poorly for 8-bit images. In most cases (*Turku1*, *Turku3* and *Turku4*) the GIF generated files *larger* than the original uncompressed file. The image *Turku2*, on the other hand, contains large uniform areas, which leads to better compression result. For example, PNG achieves 2:1 compression ratio. We can conclude that the compression performance depends on the image type. In the best case, the lossless methods achieve about 2:1 compression ratio. In the worst case, which depicted the center of the city (*Turku3*), the compression rate even with the best method (PNG) is 6.90-bit per pixel.

Figure 4: *Turku1* zoomed from the original 8-bit image, 2-bit image, and dithered 2-bit image.

Figure 5: *Turku2* zoomed from the original 8-bit image, 2-bit image, and dithered 2-bit image.

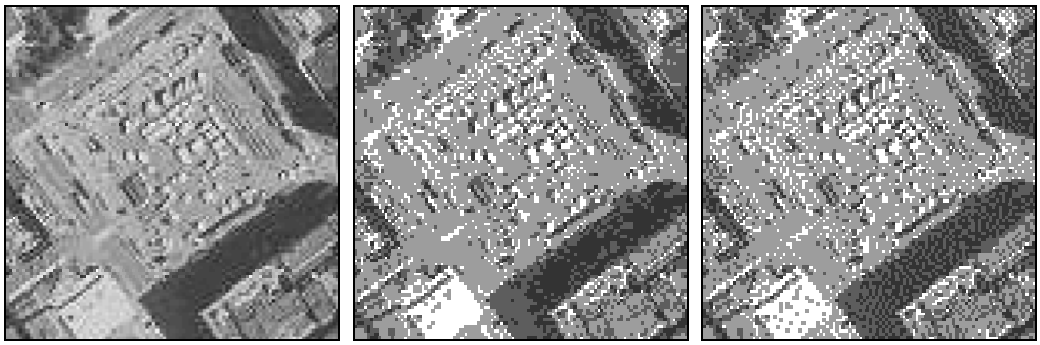

Figure 6: *Turku3* zoomed from the original 8-bit image, 2-bit image, and dithered 2-bit image.

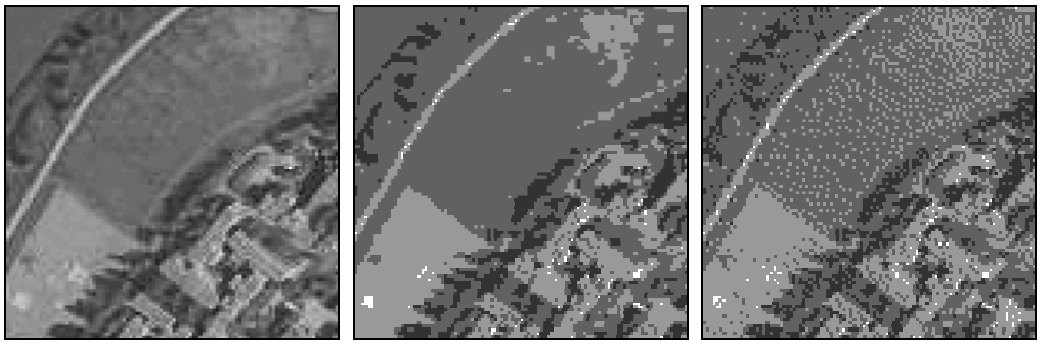

Figure 7: *Turku4* zoomed from the original 8-bit image, 2-bit image, and dithered 2-bit image.

Table 3: Lossless compression results for 8-bit images.

Image	JBIG (bitplanes)	PNG	GIF
Turku1	6.80	5.90	8.69
Turku2	4.30	3.71	5.46
Turku3	7.60	6.90	9.99
Turku4	7.20	6.47	9.41
Average	6.23	5.50	8.05

4.2 Lossy results for 8-bit images

Lossy results are summarized in Table 4, and the rate-distortion curves are shown in Figure 8 for each image separately. At the lower bit rates, we find that the JPEG2000 outperforms JPEG clearly resulting in bit rates half of that of the JPEG with the same quality (PSNR value). For example, *Turku1* is compressed to quality 25dB in PSNR with 0.25-bit per pixel by JPEG2000, and with 0.5-bit per by JPEG.

The rate-distortion curve of JPEG2000 is linear for all images whereas the JPEG results are somewhat surprising. The interesting result is that the JPEG is better than JPEG2000 for bit rates higher than 2 bpp. Moreover, the pictures *Turku1*, *Turku2* and *Turku4* have peaks around 1.5-2.0 bpp. This is due to the fact that the pictures were originally compressed with JPEG before using in this test, most likely using bit rates corresponding to the peaks. In this, JPEG quantization step has removed the same transform coefficients that frequencies being removed in the previous compression step. In other words, the compression has quantized values that have already been quantized resulting not so much new distortion in the image.

At very low bit rates, FIASCO has better quality than JPEG. The FIASCO, on the other hand, cannot achieve bit rates higher than 0.5 bit rate due to overwhelming memory consumption. ECW, on the other hand, cannot achieve bit rates lower than 2 bpp. Thus, only the JPEG and JPEG2000 can operate with all tested bit rates.

The visual quality of JPEG, JPEG2000 and FIASCO are compared in Figures 9-12. It can be seen that JPEG and JPEG2000 generate different visual artifacts. JPEG has well known blocking artifacts because the image is divided into 8×8 pixels blocks, which are compressed independently. The block boundaries become therefore visible when compressing at the lower bit rates. Typical JPEG2000-artifacts are *ringing* around the edges, and the image is often blurred.

Table 4: Average compression results as the quality (PSNR) with various bit rates.

Target bit rate	JPEG	JPEG2000	FIASCO	ECW
0.125	×	21.28	×	×
0.250	21.23	23.07	21.77	×
0.375	22.29	24.17	22.39	×
0.500	23.39	25.15	22.71	×
0.625	24.06	25.99	22.83	×
1.000	27.12	28.86	×	×
2.000	37.93	36.14	×	25.89

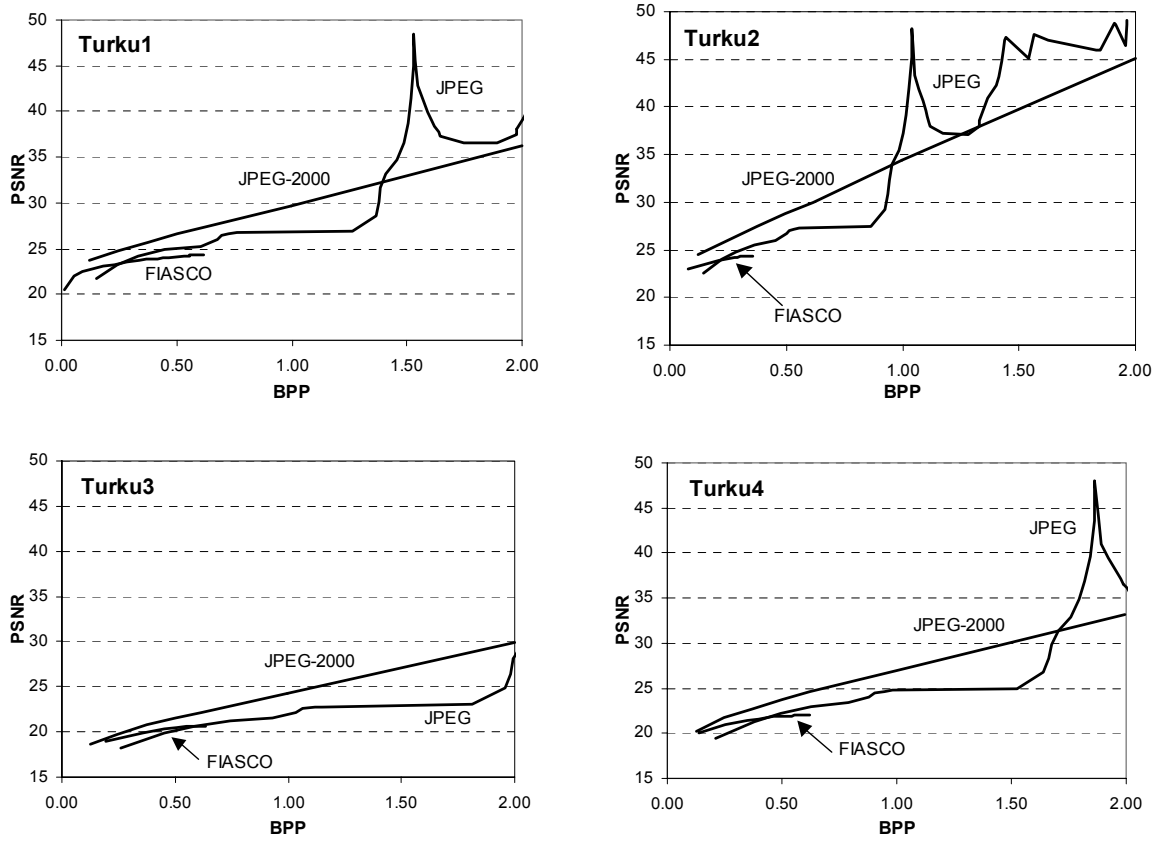

Figure 8: Rate-distortion curves graph for the set of test images.

4.3 Comparison of lossy and lossless results

We make tentative comparison of the lossy and lossless methods by taking the lowest possible quality where no visual distortion was seen in the image. The results are summarized in Table 5. From these results, we conclude that the lossy methods are significantly more efficient for 8-bit images and lossy compression is therefore a better choice.

Table 5: Compression comparison for the lossless and the lossy methods with high PSNR values.

	Lossy methods		Lossless methods		
	JPEG	JPEG2000	GIF	PNG	JBIG
Turku1	1.5 (34.7 dB)	1.0 (29.7 dB)	8.7	5.9	6.8
Turku2	0.9 (27.5 dB)	0.4 (27.5 dB)	5.5	3.7	4.3
Turku3	1.2 (22.8 dB)	1.0 (24.3 dB)	10.0	6.9	7.6
Turku4	1.5 (25.0 dB)	1.0 (26.9 dB)	9.4	6.5	7.2
Average	1.3 (27.5 dB)	0.8 (27.1 dB)	8.4	5.8	6.5

Figure 9: *Turku1* compressed with JPEG, JPEG2000 and FIASCO at 0.5 bpp.

Figure 10: *Turku2* compressed with JPEG, JPEG2000 and FIASCO at 0.5 bpp.

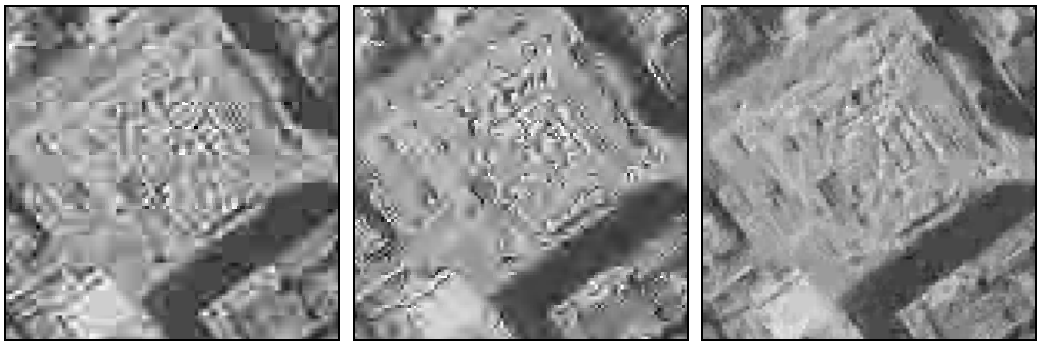

Figure 11: *Turku3* compressed with JPEG, JPEG2000 and FIASCO at 0.5 bpp.

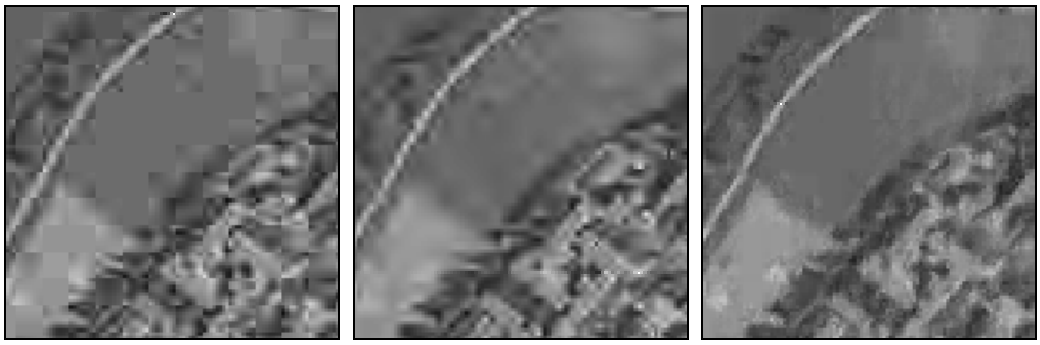

Figure 12: *Turku4* compressed with JPEG, JPEG2000 and FIASCO at 0.5 bpp.

4.4 Results for 2-bit images

Lossless compression methods for 2-bit images are summarized in Table 6 both for the non-dithered and the dithered images. GIF and PNG are applied as such, and JBIG is applied for the two bit planes separately. When we compare the non-dithered results to the dithered results, we can see that the dithered images have always higher file size but better visual quality. The use of dithering is therefore a compromise between the quality and bit rate. In the compression, JBIG gives the best results both in the case of non-dithered and dithered images.

Table 6: Summary of lossless compression methods for 2-bit and 2-bit dithered images.

Image	2-bit				2-bit dithered			
	GIF (bitplanes)	JBIG (bitplanes)	PNG	GIF	GIF (bitplanes)	JBIG (bitplanes)	PNG	GIF
Turku1	1.36	1.00	1.16	1.10	1.71	1.34	1.41	1.38
Turku2	1.21	0.91	0.86	0.87	1.54	1.22	1.24	1.20
Turku3	1.92	1.43	1.72	1.67	2.05	1.58	1.81	1.79
Turku4	1.70	1.21	1.42	1.36	1.96	1.51	1.62	1.58
Average	1.55	1.14	1.29	1.25	1.82	1.42	1.52	1.49

The results also show that the lossless compression of 2-bit images are not much more efficient than the lossy methods for 8-bit image. This indicates that lossy methods could be used. Unfortunately the lossy methods such as JPEG and JPEG2000 are not well applicable to images with reduced number of colors. JPEG, for example, performs poorly because of the cosine transform. Also, the JPEG2000 codecs applied here did not support compression of arbitrary color depths. Because of the reasons, we study the following approach: we store the images as 8-bit images and apply lossy compression. The images are then quantized and dithered 2-bit only after the decompression when output on the display. These post- processing steps are not computational expensive in comparison to the decompression. Therefore, this approach would be realistic in practice if it turns out to provide better results both in file size and in quality.

The lossless 2-bit results with lossy 8-bit results with bit rates 1.00 and 0.25 bpp are summarized in Table 7, and the visual quality shown in Figures 13-16. The first bit rate is slightly lower than the lossless results on average, whereas the second one is clearly superior in compression. We can see, that the images with 0.25 bpp are of lower quality than the compressed 2-bit images (with dithering) whereas the file size is less than 25% that of the JBIG compressed files. The 1.00-bit results, on the other hand, have both good quality and better bit rate.

Table 7: Summary of compression results for 2-bit dithered images.

Image	2-bit dithered JBIG	JPEG2000	JPEG2000
Turku1	1.34 bpp	1.00 bpp	0.25 bpp
Turku2	1.22 bpp	1.00 bpp	0.25 bpp
Turku3	1.58 bpp	1.00 bpp	0.25 bpp
Turku4	1.51 bpp	1.00 bpp	0.25 bpp

Figure 13: *Turku1* lossless 2-bit dithered, JPEG2000 at 1 bpp and 0.25 bpp.

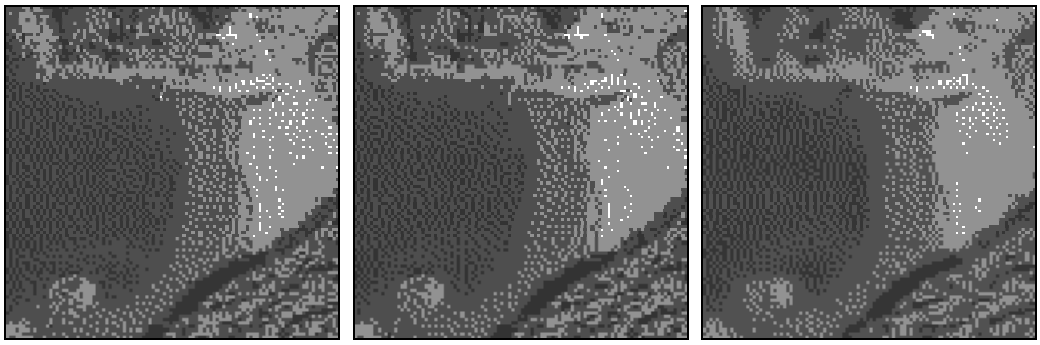

Figure 14: *Turku2* lossless 2-bit dithered, JPEG2000 at 1 bpp and 0.25 bpp.

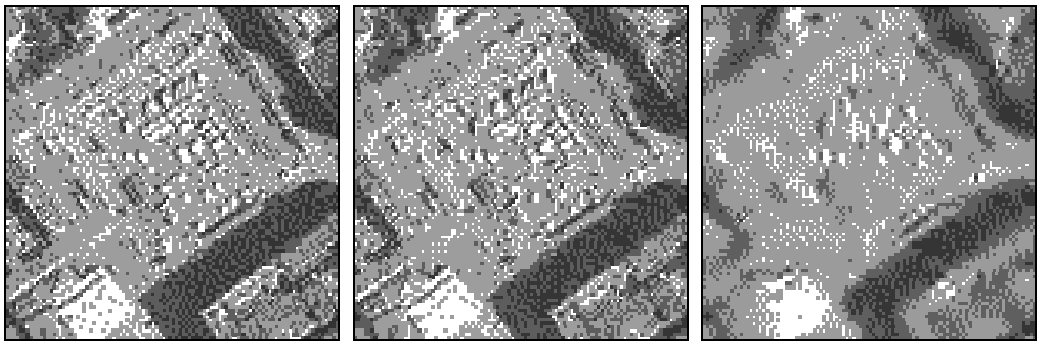

Figure 15: *Turku3* lossless 2-bit dithered, JPEG2000 at 1 bpp and 0.25 bpp.

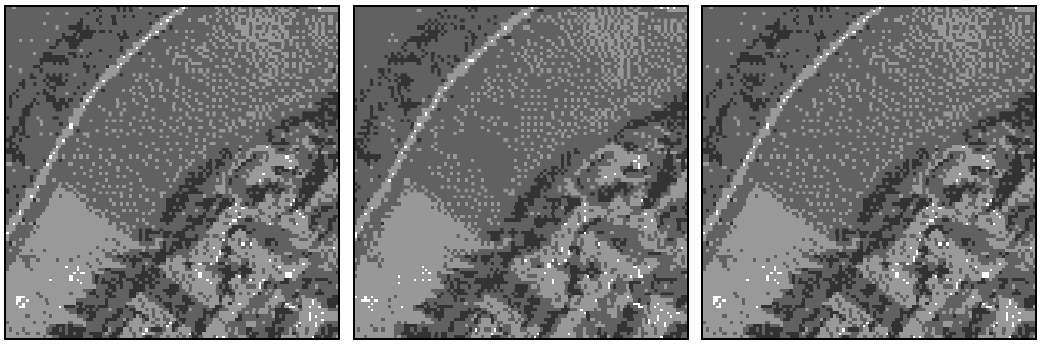

Figure 16: *Turku4* lossless 2-bit dithered, JPEG2000 at 1 bpp and 0.25 bpp.

5. CONCLUSIONS

In the case of lossy compression of 8-bit images, the wavelet-based JPEG2000 clearly outperformed JPEG and FIASCO by a wide margin of MSE-quality and consistent bit rate scalability. In the lossless compression of the 2-bit images, the JBIG gives the best compression for the 2-bit images. A better approach, on the other hand, turned out to be the use of lossy compression for the 8-bit images, and to apply quantization and dithering to at the moment of display. The JPEG2000 achieved good visual quality with file sizes less than that of the lossless compression.

REFERENCES

[1] A. Haar, "Zur Theorie der Orthogonalen Funktionensysteme", *Math. Annal.*, no. 69, pp. 331-371, 1910.

[2] A.N. Skodras, C.A. Christopoulos and T. Ebrahimi, "JPEG2000: The upcoming still image compression standard", *Proc. 11th Portugese Conference on Pattern Recognition* , Porto, Portugal, pp. 359-366, May 2000.

[3] A. Said and W.A. Pearlman, "A New and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, June 1996.

[4] Compuserve Incorporated, *Graphics Interchange Format* (version 89a), Columbus, Ohio, 1989, (http://www.w3.org/Graphics/GIF/spec-gif89a.txt).

[5] D. Santa-Cruz and T. Ebrahimi, "An Analytical Study of JPEG2000 Functionalities," *Proc. of the International Conference on Image Processing (ICIP)*, Vancouver, Canada, September 10-13, 2000.

[6] D. Taubman, "High Performance Scalable Image Compression with EBCOT", *IEEE Transactions on Image Processing*, vol. 9, no. 7, July 2000.

[7] Earth Resource Mapping Pty Ltd, *Compression White Paper Version 2.0*, June 2000, (http://www.ermapper.com/product/ermapper6/new/compression_white_paper1.pdf).

[8] E. Ordentlich, M. Weinberger and G. Seroussi, "A Low-Complexity Modeling Approach for Embedded Coding of Wavelet Coefficients", *Proc. IEEE Data Compression Conference*, Snowbird, UT, pp. 408-417, March 1998.

[9] ISO/IEC, *ISO/IEC 11544:1993 Information Technology - Coded Representation of Picture and Audio Information - Progressive Bi-level Image Compression*, March 1993.

[10] ISO/IEC JTC 1/SC 29/WG 1, *ISO/IEC FCD 15444-1: Information Technology - JPEG2000 Image Coding System: Core Coding System [WG 1 N 1646]*, March 2000, (http://www.jpeg.org/FCD15444-1.htm).

[11] J.M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelets Coefficients," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3445-3462, December 1993.

[12] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, vol. IT-23, no. 3, pp. 337-343, 1978.

[13] J. Ziv and A. Lempel, "Compression of Individual Sequences Via Variable-Rate Coding," *IEEE Transactions on Information Theory*, vol. IT-24, no. 5, pp. 530-536, 1978.

[14] K. Culic and J. Kari, "Image Compression Using Weighted Finite Automata", *Computers and Graphics*, Vol. 17, no. 3, pp. 305-313, 1993.

[15] R.W. Floyd and L. Steinberg "An Adaptive Algorithm for Spatial Gray Scale," *SID International Symposium Digest of Technical Papers*, pp. 36-37, 1975.

[16] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1999.

[17] S. Mallat, "An Efficient Image Representation for Multiscale Analysis," *Proc. of Machine Vision Conference*, Lake Taho, February 1987.

[18] T.A. Welch, "A Technique for High-Performance Data Compression", *IEEE Computer*, pp. 8-19, June 1984.

[19] U. Hafner, J. Albert, S. Frank and M. Unger, "Weighted Finite Automata for Video Compression", *IEEE Journal on Selected Areas in Communications* , Vol. 16, no. 1 , pp. 108-191, January 1998.

[20] W3C, *PNG (Portable Network Graphics) Specification*, October 1996, (http://www.w3.org/TR/REC-png).

[21] W.B. Pennebaker and J.L. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, 1993.