Evaluation of image compression methods for aerial photos

Ville Hautamäki and Jussi Heino
villeh@cs.joensuu.fi, juhehe@utu.fi

24.12.2000

University of Joensuu
Computer Science
Bachelor's Thesis

Table of Contents

1 Introduction

Aerial images are photographs of terrain taken from air or, in the case of satellite images, from satellite. Conventional maps can be represented in digital form as vector graphics or as bi-level bitmap. Aerial photos, on the other hand, need more color information.

Aerial images are very large in size, so what is also needed from compression codec is some kind of support for random access. JPEG2000, for example, supports random access property up to subband block level, these code-blocks can be decompressed individually. Details in large aerial photos requires lot of space, our test images are 600 x 450 pixels sub-images. The uncompressed size of these detail images is 270 kB each. Normal GSM DATA link has the speed of 9600 bits/s. Thus, with normal GSM phone it would take 3 minutes and 45 seconds to download a sub-image of that size.

It can be easily seen, that some kind of compression is needed to effectively use digitized aerial images in different application ranging mobile cartography to Internet applications. Using lossy compression with ratio 8:1 we can compress an image to 33 kB. Lossless methods seldom achieve compression ratio of 2:1, corresponding to 135 kB.

In this report, we study different methods for reducing the amount of data required by the images. The original images are 8 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 and the forth-coming compression standards JPEG [21] and JPEG2000 [10], and a fractal compressor known as FIASCO [19].

It is expected that the devices using the images will not support 8 bpp but much fewer gray levels can be shown. It is therefore possible to get further reduction by quantizing the images, for example, to 2 bits per pixel, corresponding to 4 colors. Quantized images can be compressed more efficiently by lossless methods. Even universal dictionary-based compression methods, such as LZ77 [12] or LZ78 [13] work quite well for 2 bpp images.

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

2 Compression methods

Image compression methods can generally be divided into two main categories, lossy and lossless. Lossless compression means that after compression and decompression the image in question will be identical. Lossy methods are not identical pixel-by-pixel, but they try to achieve minimal degradation in the visual quality of the output image. The difference between various compression methods is compared by the compression ratio, i.e., how much we can compress the image.

In lossy methods, we compare compression rate (bitrate) to mean square error (MSE). This measure is very simple as it is calculated as the sum of the squared pixel-by-pixel differences. Resulting value is then divided by the total number of pixels in the image. More formally:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(A_i - B_i)^2 \tag{1}$$

Where N is the total number of pixels in the image, A_i is the value of the original pixel and B_i is value of the corresponding pixel. Thus, MSE approximates the average difference between input and output images. MSE values were calculated using authors' Linux *MSE-software*. Sources of this program can be found: (`http://cs.joensuu.fi/~villeh/papers.html`)

Peak-to-peak signal to noise ratio (PSNR) is defined as:

$$PSNR = 10 * log_{10}\left(\frac{255^2}{MSE}\right) \quad dB \tag{2}$$

The only way to make meaningful comparisons between lossy and lossless methods is to find a bitrate where the visual degradation is not visible to human eye. These rates can then be compared.

Table 1: Summary comparison of methods.

	Lossy	Lossless	Compression method
JPEG	X		DCT
JPEG2000	X	X	Wavelet + EBCOT
FIASCO	X		Fractals
ECW	X		Wavelet
GIF		X	LZ78
PNG		X	LZ77
JBIG		X	JBIG

Compression methods used in the evaluation are: JPEG, JPEG2000, FIASCO, ECW, PNG, GIF, TIFF and JBIG, see Table 1. JPEG2000 can be applied both in lossy and lossless mode. This is because it has both modes. In this evaluation we use only the lossy mode. Pointers to the sources are summarized in Table 2.

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

Table 2: Software used in this evaluation.

Method	Codec	Reference	Date
JPEG	libjpeg6b	http://www.ijg.org	2000-08
JPEG2000	JJ2000	http://jpeg2000.epfl.ch	2000-07
JPEG2000	JasPer	http://www.ece.ubc.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 8x8 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 it's 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

Image is first processed by Daubechies 7/9 biorthogonal *wavelet transform* [16]. The transform 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 1.

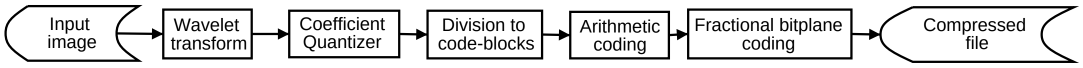

Figure 1: Genaral view of JPEG2000 encoding architecture.

Discrete Wavelet Transform (DWT)

Normal image has a high correlation between it's pixel values. This means that neighbouring pixels have usually similar pixel values. Edges in the image are an important exception to this. Wavelet transform has two filtering operators or *convolving functions*, low-pass and high-pass filters. Each filter is used in every other pixel one at a time. This results two *sub-sampled* coefficient bands, where low-pass band contains smooth version of the image and high-pass band contains edges. First wavelet found was the *Haar wavelet* [1], in which the result of low-pass filter is the average pf two pixels and the result of high-pass filter their difference.

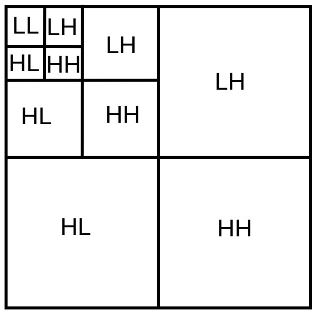

Figure 2: Subband decomposition structure in wavelet transform.

Figure 2 shows *mallat ordering* [17] of the transformed subbands, which 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. This results in that most of the total signal energy is contained in low-pass subband. After each decomposition step, resulting low-pass and high-pass non-zero coefficients magnitude increases compared to previous level. Actual increase depends on the wavelet transform function used; a good approximation is that the magnitude doubles after each decomposition step. Figure 3 shows that the non-zero coefficients are scarce in high-pass bands. The magnitude of the low-pass coefficients has also increased.

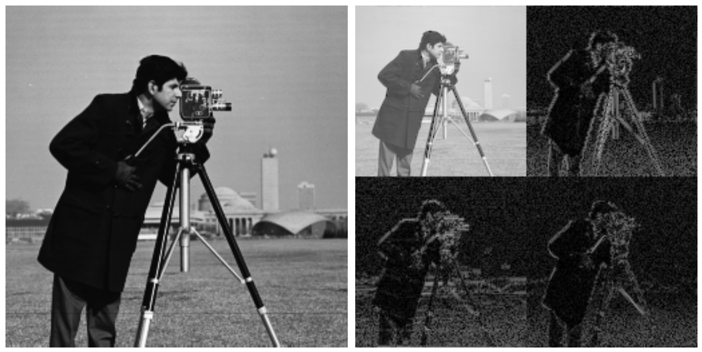

Figure 3: Original camera man test image and the subbands decomposed up to one level.

Wavelet image compression

Previous wavelet image coding algorithms suchs as *Embedded zerotree wavelet* (EZW) [11] and *Set Partitioning in hierarchical trees* (SPIHT) [3] create *SNR scalable* bitstream. SNR scalable feature in bitstream signifies that we can stop decoding the image at any user definable bitrate. EZW and SPIHT algorithms try to minimize distortion at every bitrate.

The Embedded block coding with optimized truncation (EBCOT) algorithm [6], which is 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. Next resolution level can be obtained from *inverse wavelet transform* (IWT) of subbands LL, LH, HL and HH at the same level. It has to be noted that resolution cannot be set to arbitrary size. User needs to exploit some form of interpolation technique to obtain image at the arbitrary size.

In EZW and SPIHT, the SNR scalability is achieved by ordering coefficients by magnitude and then coding them by bitplane. Thus output file size can be set in the accuracy of one bit. In EBCOT, each subband is divided into code-blocks, and these code-blocks are then coded individually, thus achieving resolution scalability. EBCOT set's finite number of truncation points in each code-block. Output bitstream is then a collection of different truncated code-blocks. As these truncation points are finite we cannot choose the exact bitrate as:

$$BPP_{output} \leq BPP_{target}. \tag{3}$$

Where BPP_{target} signifies user defined bitrate and BPP_{output} signifies real bitrate generated by the coder.

Arithmetic coder in EBCOT

EBCOT algorithm contains two different coding engines, *Tier 1* and *Tier 2*. Tier 1 engine operates transformed coefficients and places truncation points to code-blocks. These code-blocks are transmitted to Tier 2 engine, which places fragments of these code-blocks to different quality layers, which correspond to different bitrates. Tier 2 engine creates the actual compressed file. Figure 4 illustrates this process.

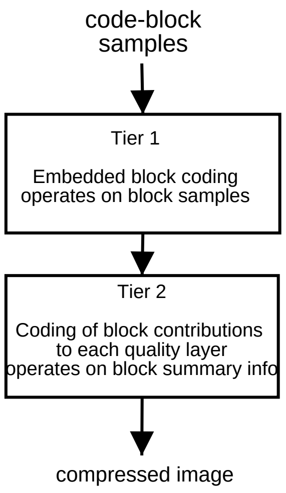
code-block samples
Tier 1
Embedded block coding operates on block samples
Tier 2
Coding of block contributions to each quality layer operates on block summary info
compressed image

Figure 4: Two-tiered coding system in EBCOT.

The code-blocks are processed by dividing them into bitplanes, which are further divided into smaller sub-blocks. In each bitplane, we first evaluate each sub-block whether it is significant in the current bitplane. If not it is by-passed. 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*.

Traditional bitplane coders code each bitplane in one pass. 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]. Simplified version of the relationship between fractional bitplane coding and the quality layers in the final compressed file can be seen in Figure 5.

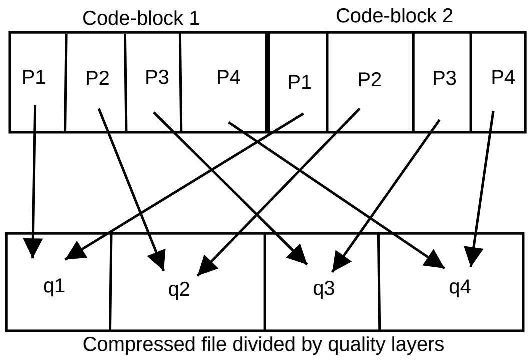

Figure 5: Relationship between the different coding passes and the quality layers in the compressed file.

Random access property in EBCOT is achieved up to the code-block level. EBCOT can seek individual code-blocks to be decompressed individually. Because of boundary conditions EBCOT has to retrieve larger area than just the requested number of code-blocks.

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 bitrates. For Turku2 image, even the bitrate 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 bitrates generate similar blocking artifacts as the baseline JPEG with similar bitrates. This is maybe acceptable for low-bitrate video coding but not for good quality still image coding.

2.4 GIF

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. The GIF method [4] was developed by Compuserve and patented by Unisys until 2004.

2.5 PNG

PNG is similar method to GIF. Gray-scalemages 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 lossless. 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. ECW plug-ins are available for various image and map processing software as well. With our test images, the ECW tool didn't always achieve the target ratio and produced images only at 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 images were downloaded from the web server of the city of Turku, Finland: (`http://www.turku.fi/mito/cgi-bin/kartta?x=1500&y=1125&z=0&i=1`). The pictures are 24-bit JPEG images. So those images were 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 pixel values around the image, and thus they are difficult to observe visually.

Representing 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 bits (256 tones) is considered enough for image to be perfect for human eye. However, perfect result is not needed for most applications and, e.g., hand-held devices might be able to display only 4 color tones (2 bits per pixel).

Color reduction for gray-scale images from 256 colors to 4 colors reduces storage space to 1/4 of the original image. Linear color reduction, without dithering, approximates the 256-tone color space by 4 tones and makes the color mapping to each pixel individually. Better visual quality can be obtained using dithering methods. The tradeoff is that the dithering methods affect also the compression methods worsening the compression performance. Moreover, most efficient compression by are lossy (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 also lossless compression methods designed for 1-bit images (JBIG). It is likely 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]. This approach is a serial process and the dithering of a single pixel affects the neighbouring pixels to the right and below. The error, that is the difference between the exact pixel value and the approximated value actually displayed, is spread to the color values of the four pixels below and right to the pixel in question.

Linux program *The Gimp* version 1.1.24 (`http://www.gimp.org`) was used to generate gray-scale images from the original color image. The 8 bpp $\rightarrow$ 2 bpp color reduction, and dithering was generated using the option "*Floyd-Steinberg*" with "*no-bleed*". The bit planes of the 2-bit images were separated using Pasi Fränti's *bitlevel* software (`http://cs.joensuu.fi/franti/softat/bitlevel.exe`).

Enlargements of the test images and their processed 2-bit versions are shown in Figures 6, 8, 8 and 9. The full-scale images can be found in Appendix A (8-bit images) and Appendix B (2-bit images and 2-bit images with dithering).

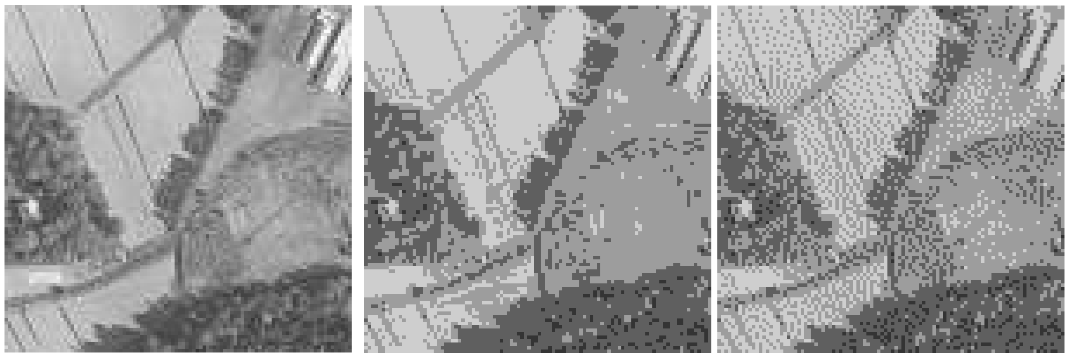

Figure 6: Turku1 zoomed from original, 2 bpp and 2bpp-dithered.

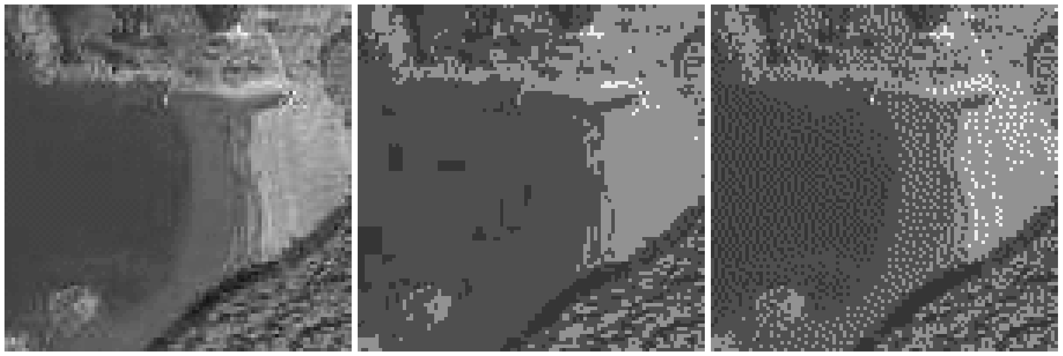

Figure 7: Turku2 zoomed from original, 2 bpp and 2bpp-dithered.

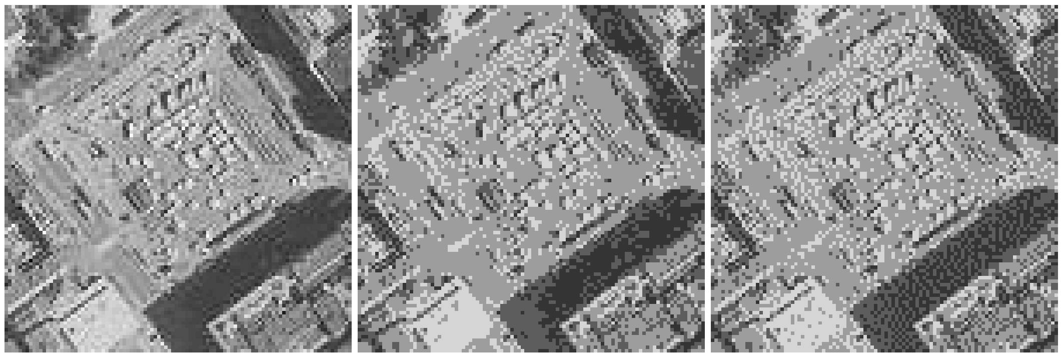

Figure 8: Turku3 zoomed from original, 2 bpp and 2bpp-dithered.

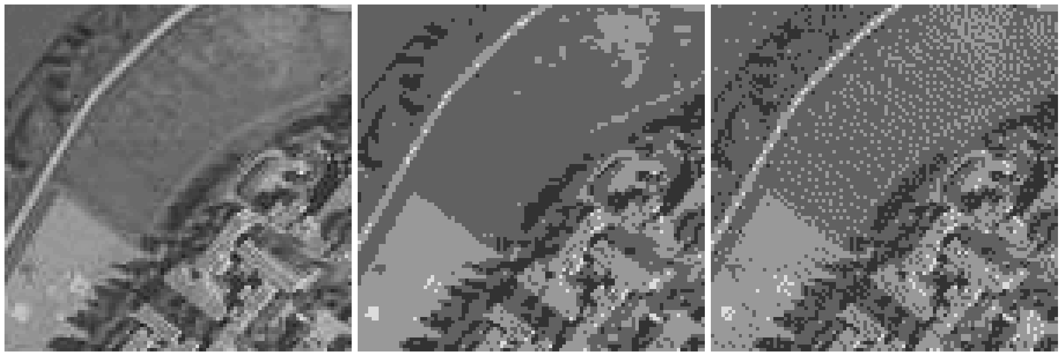

Figure 9: Turku4 zoomed from original, 2 bpp and 2bpp-dithered.

4 Compression results

4.1 Lossless results for 8-bit images

Table 3: Results of lossless compression of 8-bit images.

Method	Image	BPP
JBIG by bitplane	Turku1	6.8
PNG	Turku1	5.895
GIF	Turku1	8.685
JBIG by bitplane	Turku2	4.3
PNG	Turku2	3.706
GIF	Turku2	5.457
JBIG by bitplane	Turku3	7.6
PNG	Turku3	6.903
GIF	Turku3	9.997
JBIG by bitplane	Turku4	7.2
PNG	Turku4	6.465
GIF	Turku4	9.411

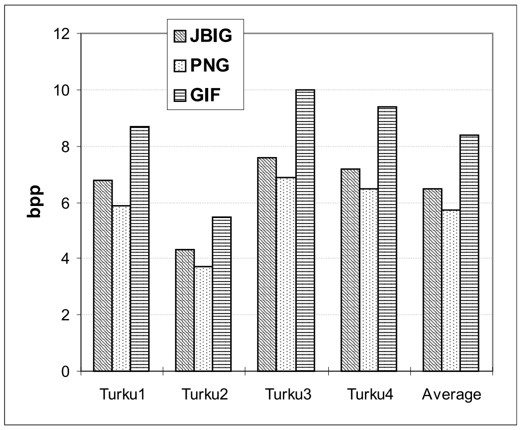

Figure 10: Results of lossless compression methods for 8-bit test images Turku1..4

Lossless results are summarized in Table 3. We can see that the dictionary-based lossless methods (GIF, TIFF) work quite poorly for 8-bit images. In most cases (Turku1, Turku3 and Turku4) these methods generated files *larger* than the original uncompressed file. The image Turku2 contains large areas of uniform color, which leads to good compression ratio. For example, PNG achieves 2:1 compression ratio. PNG outperforms the other methods and it achieves 2:1 compression ratio at best. It can be concluded that the compression performance depends a lot on the type of the image. At best lossless methods achieve 4 bpp (2:1 compression ratio). And with hardest images, which depicted center of the city, compression rate even with the best method (PNG) was 6 bpp.

4.2 Lossy results for 8-bit images

Lossy results are given in Table 4, and visually illustrated in Figure 15, which describes average PSNR-BPP ratio. Individual results for each picture can be seen in Figure 16. When comparing low quality scale, we find that, in some cases for same quality JPEG2000 method results in bitrate of half the size than that of JPEG. For example, image Turku1 compressed to quality 25dB in PSNR scale yields 0.25 bpp rate with JPEG2000 and 0.5 rate with JPEG. Having the same PSNR value does not mean, that pictures are identical, it only means, that the pixel-wise difference when compared to original picture is the same.

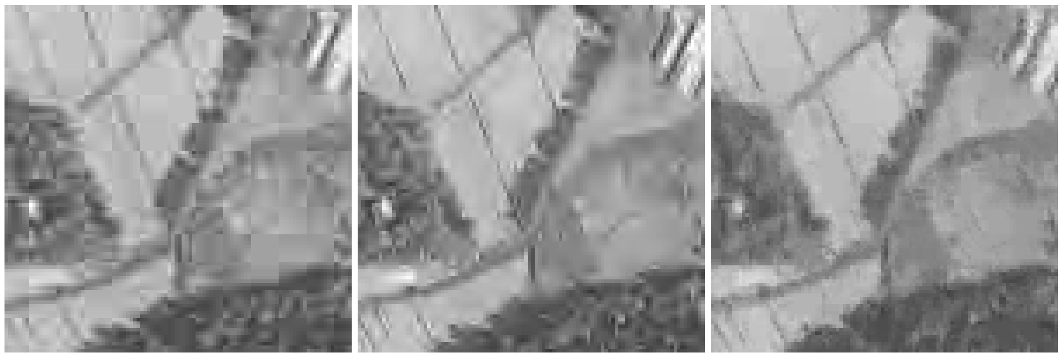

Figure 11: Turku1 compressed with JPEG, JPEG2000, and FIASCO at 0.5 bpp.

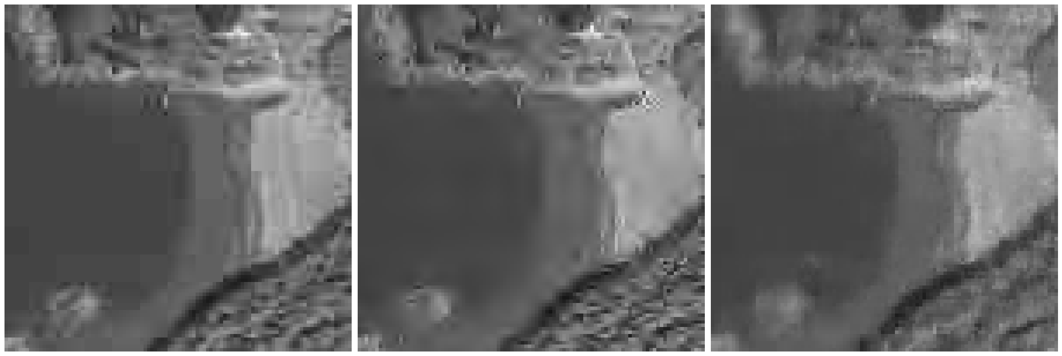

Figure 12: Turku2 compressed with JPEG, JPEG2000, and FIASCO at 0.5 bpp.

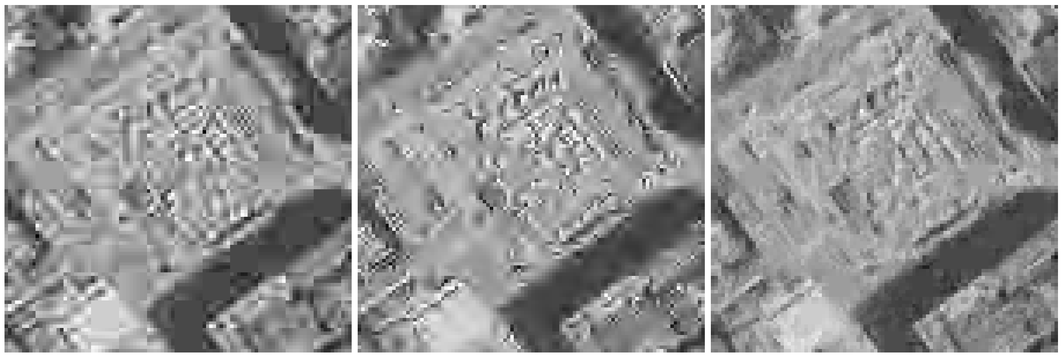

Figure 13: Turku3 compressed with JPEG, JPEG2000, and FIASCO at 0.5 bpp.

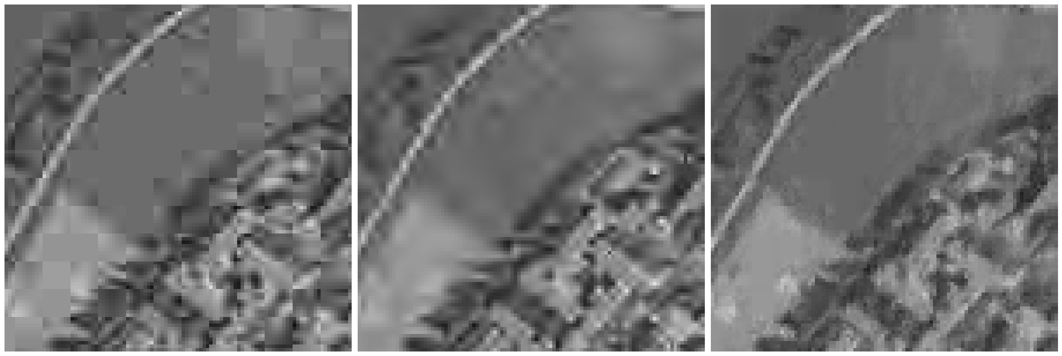

Figure 14: Turku4 compressed with JPEG, JPEG2000, and FIASCO at 0.5 bpp.

JPEG and JPEG2000 generate different visual artifacts when used in low bitrates. JPEG has well known blocking artifacts, which are due to fact, that JPEG method divides the image to 8x8 pixel blocks, which compressed independently from each other. The block boundaries become visible when compressing in low bitrates. A typical JPEG2000-artifact is *ringing* around the edges. The image is often blurred, too.

The rate-distortion curve of JPEG2000 is very linear for all images.

In very low bitrates FIASCO has better quality than JPEG, but FIASCO cannot achieve bitrates higher than 0.5 bitrate. ECW, on the other hand, cannot achieve bit rates lower than 2 bpp.

Interesting result is, that JPEG is better than JPEG2000 for bitrates higher than 2 bpp. Moreover, the pictures Turku1, Turku2 and Turku4 have strange peak around 1.5-2.0 bpp. This might be due to the fact that the pictures were originally compressed with JPEG before using in this test, probably using bitrates corresponding to the peaks. In this, JPEG quantization step has removed the same transform coefficients that frequencies being removed in the previous compression step. And we tried to quantize values, that were already quantized, resulting no new distortion in the image.

Table 4: Average PSNR-results of lossy compression methods.

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

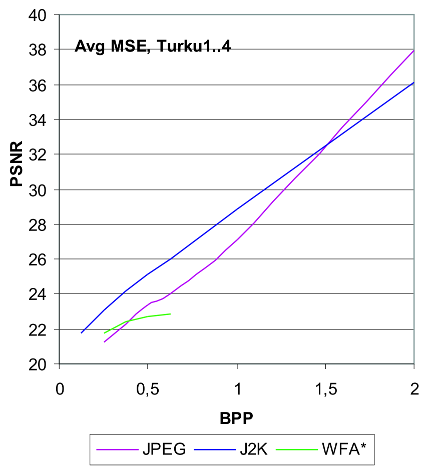

Figure 15: Results are average values for the test images Turku1..4.

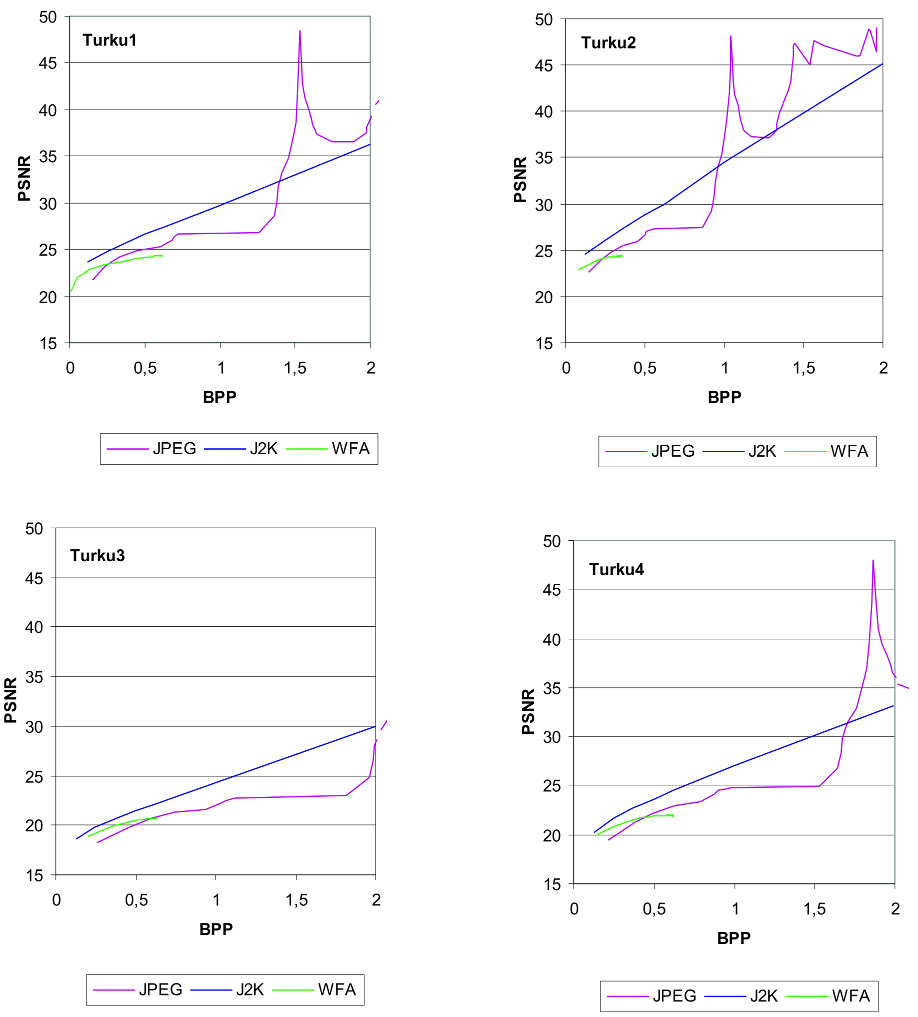

Figure 16: PSNR-BPP graph for Turku1..Turku4.

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 difference was seen. These results are summarized in Table 5. From these results, we conclude that the lossy methods are significantly more efficient for 8-bit images and are therefore better choice.

Table 5: Lossy and lossless methods. Values in parenthesis are PSNR values, rest are bpp values.

	JPEG	JPEG2000	GIF	PNG	JBIG
Turku1	1.5 (34.7)	1 (29.7)	8.7	5.9	6.8
Turku2	0.9 (27.5)	0.4 (27.5)	5.5	3.7	4.3
Turku3	1.2 (22.8)	1 (24.3)	10	6.9	7.6
Turku4	1.5 (25.0)	1 (26.9)	9.4	6.5	7.2
Average	1.3 (27.5)	0.8 (27.1)	8.4	5.8	6.5

4.4 Results for 2-bit images

Lossless compression methods used for the 2 bpp images test are GIF, PNG and JBIG. The GIF and PNG methods are applied to the images as such, whereas the JBIG is applied for the two bit planes separately. The results are summarized in Table 6 both for the non-dithered and the dithered images.

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

Method	Image	BPP	BPP (dithered)	Average
GIF by bitplane	Turku1	1.36	1.71	1.54
JBIG by bitplane	Turku1	1.00	1.34	1.17
PNG	Turku1	1.16	1.41	1.29
GIF	Turku1	1.10	1.38	1.24
GIF by bitplane	Turku2	1.21	1.54	1.38
JBIG by bitplane	Turku2	0.91	1.22	1.07
PNG	Turku2	0.86	1.24	1.05
GIF	Turku2	0.87	1.20	1.04
GIF by bitplane	Turku3	1.92	2.05	1.99
JBIG by bitplane	Turku3	1.43	1.58	1.51
PNG	Turku3	1.72	1.81	1.77
GIF	Turku3	1.67	1.79	1.73
GIF by bitplane	Turku4	1.70	1.96	1.83
JBIG by bitplane	Turku4	1.21	1.51	1.36
PNG	Turku4	1.42	1.62	1.52
GIF	Turku4	1.36	1.58	1.47
Average:				
GIF by bitplane	Turku[1...4]	1.55	1.82	1.69
JBIG by bitplane	Turku[1...4]	1.14	1.42	1.28
PNG	Turku[1...4]	1.29	1.52	1.41
GIF	Turku[1...4]	1.25	1.49	1.37

When we compare the non-dithered results in Table 6 to the dithered results, we can see that the non-dithered images have always smaller size. On the other hand, when we compare the visual quality between the dithered and non-dithered images, we can see that dithering achieves better quality. The use of dithering is therefore compromize between the quality and file size.

In the compression, JBIG gives the best results both in the case of non-dithered and dithered images. In the experiments, we also applied GIF to the bit planes but as we can see from the results, GIF gives better results when applied to the original 2 bpp images as such.

The results also show that the lossless compression of 2 bpp images are not any more efficient that the lossy methods for 8 bpp 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 existing JPEG2000 codecs did not support 2 compression of arbitrary color depths.

Because of the reasons explained above, it makes sense to study the following approach: we store the images as 8 bpp images and apply lossy compression. The images are quantized and dithered to 2 bpp only after the decompression at the moment 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.

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

Image	2-bit dithered JBIG	JPEG2000	JPEG2000
Turku1	1.34 bpp	1 bpp	0.25 bpp
Turku2	1.22 bpp	1 bpp	0.25 bpp
Turku3	1.58 bpp	1 bpp	0.25 bpp
Turku4	1.51 bpp	1 bpp	0.25 bpp

We compare the lossless 2 bpp results with lossy 8 bpp results with bit rates 1.00 and 0.25 bpp. The first one is slightly lower than the lossless results on average, whereas the second one is clearly superior. The compression results are compared in Tables 5, 6 and 7. We can see, that the images with 0.25 bpp are lower quality than the compressed 2 bpp images (with dithering) whereas the file size is less than 25% that of the JBIG compressed files. The 1.00 bpp results, on the other hand, have both good quality and slightly better bit rate. The complete set of the lossy compressed and quantized images are in Appendix D.

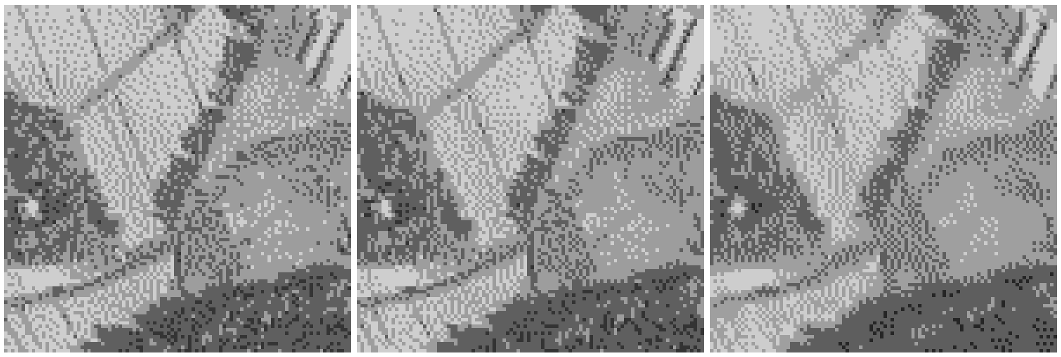

Figure 17: Turku1 lossless 2-bit dithered, JPEG2000 at 1 bpp and 0.25 bpp

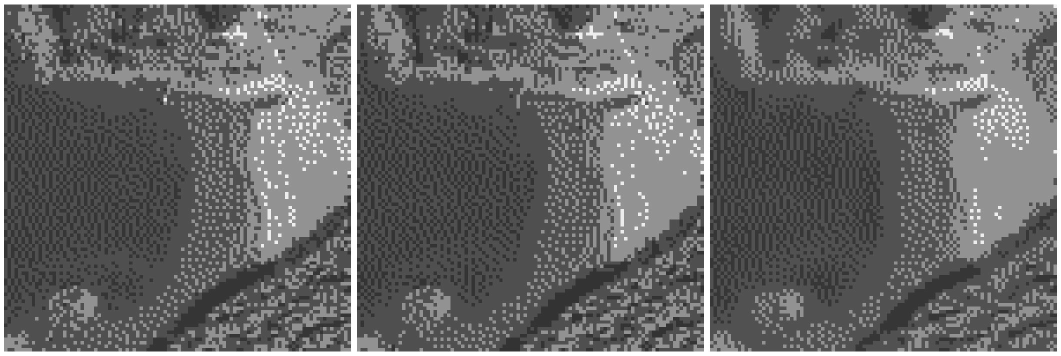

Figure 18: Turku2 lossless 2-bit dithered, JPEG2000 at 1 bpp and 0.25 bpp

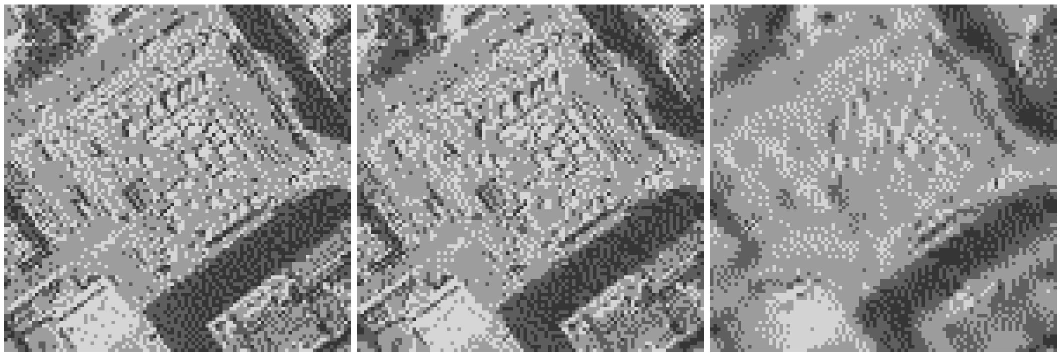

Figure 19: Turku3 lossless 2-bit dithered, JPEG2000 at 1 bpp and 0.25 bpp

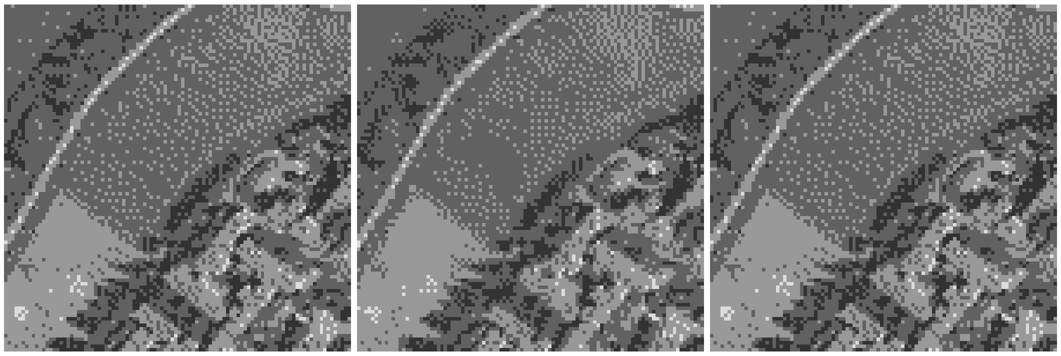

Figure 20: Turku4 lossless 2-bit dithered, JPEG2000 at 1 bpp and 0.25 bpp

5 Conclusions

Impact of source data suitability for different compression methods can be seen especially from poor performance of lossless GIF. PNG is supposed to achieve always slightly better results than 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 and TIFF.

In lossy category wavelet-based JPEG2000 clearly outperformed JPEG and FIASCO by a wide margin of MSE-quality and consistent bitrate scalability. Apparenty ECW is optimized only for much larger images than our test images, or there is inefficiency in input file filter. Otherwise, the poor quality - file size ratio is a disappointment. Lossy methods with good visual quality produce file sizes less than half of the file sizes created with lossless methods.

Image size (and quality) can also be reduced by reducing the number or colors in the image. Traditional approach to compress an image that has been color quantized is to use lossless methods. These methods are the obvious choice for compressing 2-bit (4 color) images, as the lossy methods in this evaluation produce best 8-bit results. PNG outperformed GIF and TIFF.

For four color mobile displays error diffusion dithered images approximate original 8-bit color images quite well. Trade-off in dithering is that lossless compression ratio decreases. Solution to this might be to store the images as compressed 8-bit images using a decent lossy compressor such as JPEG2000. Quantization to 2-bits and dithering should happen only at the moment when the image is displayed.

Choosing best 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.

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.

Appendices

Appendix A Original images

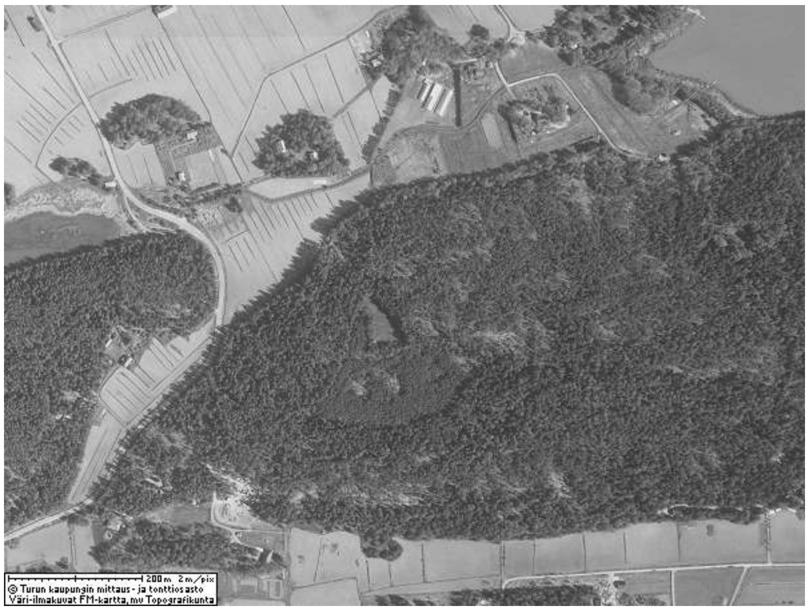
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 21: Turku1 (600x450 pixels, 8-bit).

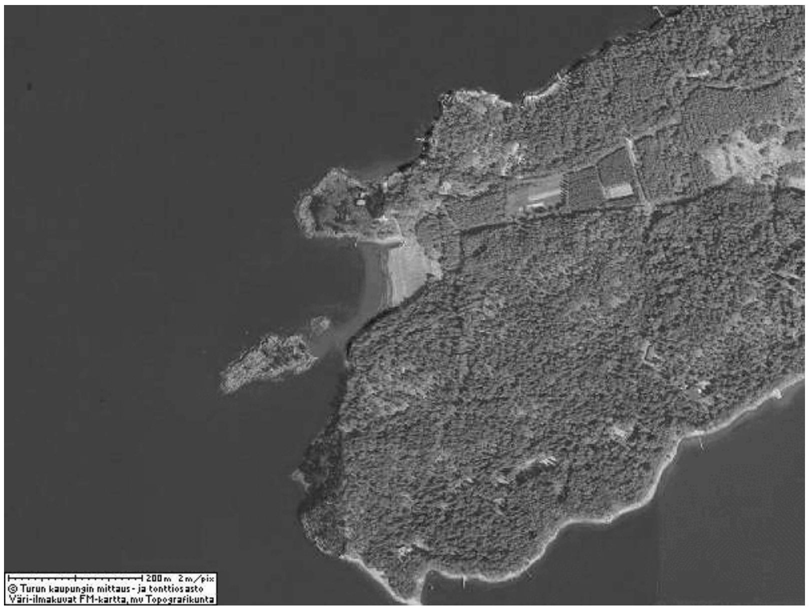
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 22: Turku2 (600x450 pixels, 8-bit).

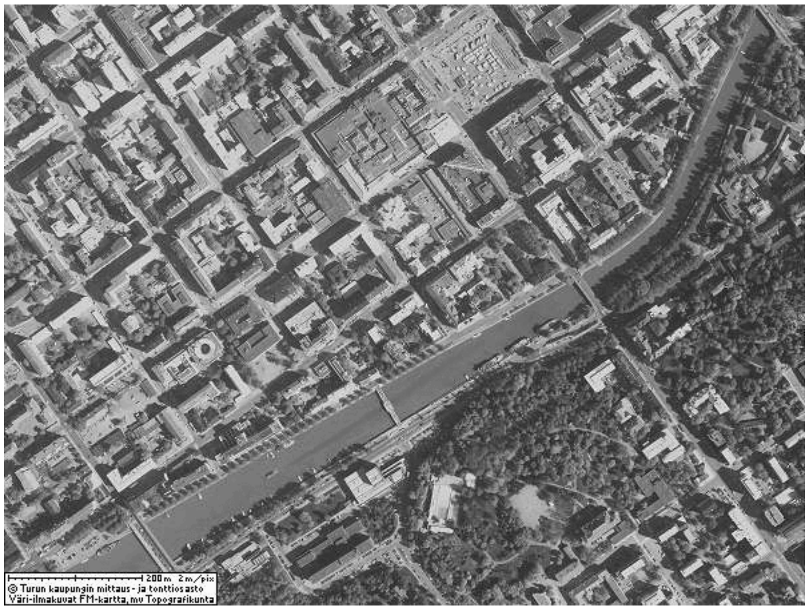
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 23: Turku3 (600x450 pixels, 8-bit).

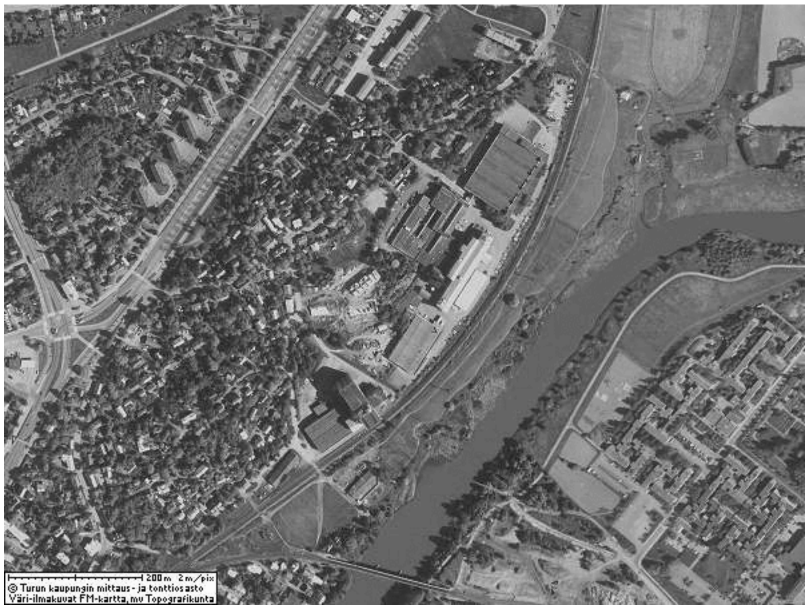
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 24: Turku4 (600x450 pixels, 8-bit).

Appendix B 2-bit images

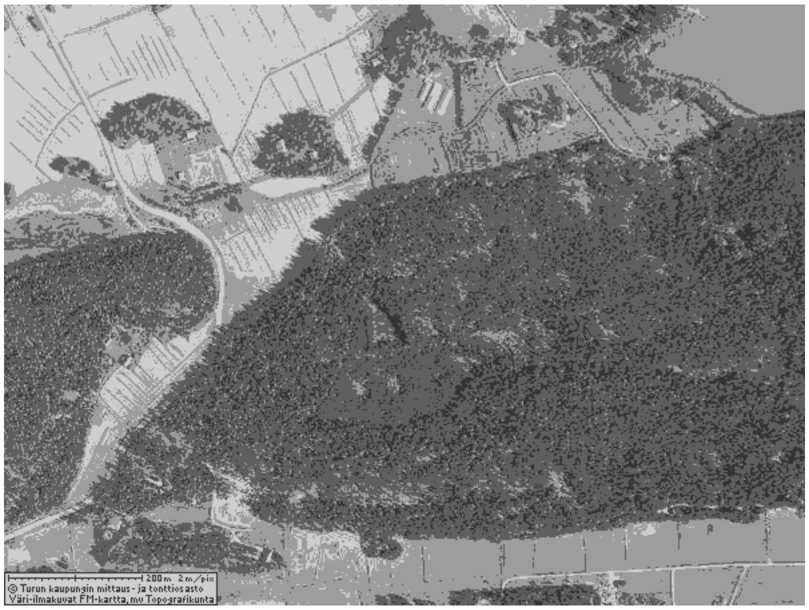
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 25: Turku1 (600x450 pixels, 2-bit).

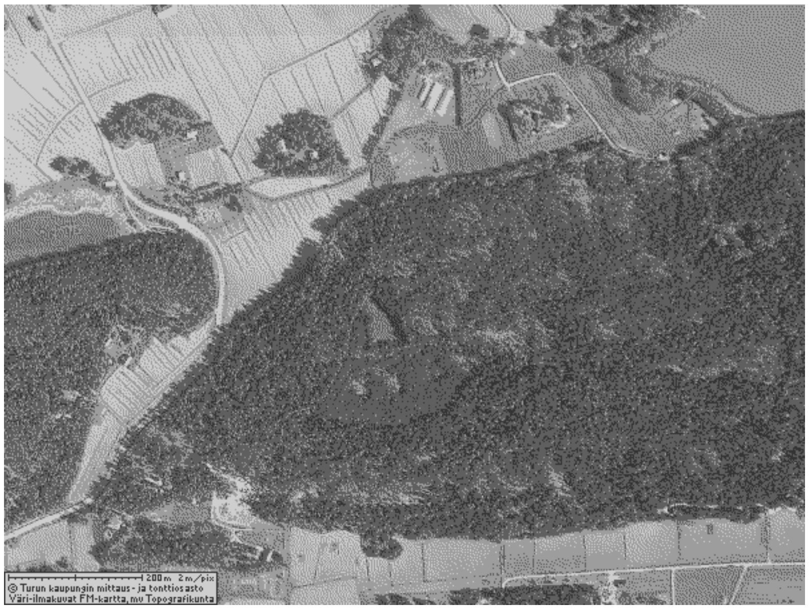
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 26: Turku1 (600x450 pixels, 2-bit) Floyd-Steinberg dithering.

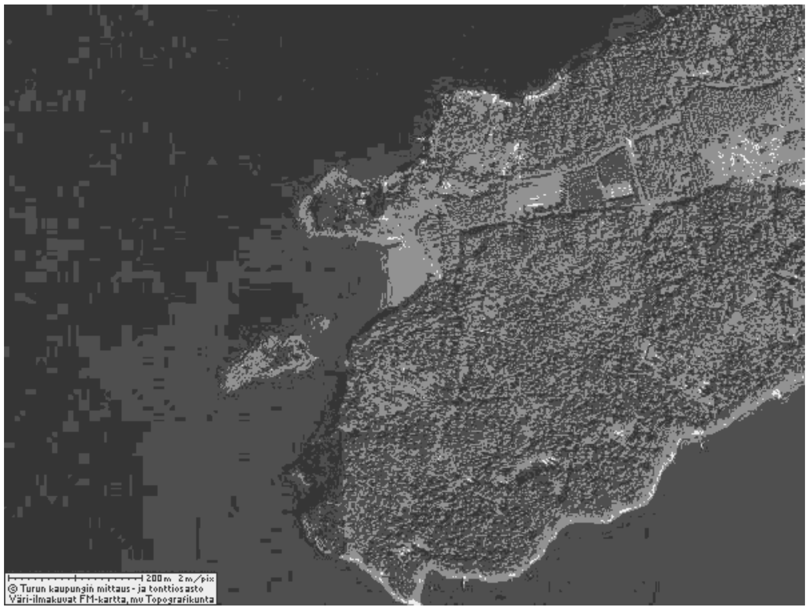
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 27: Turku2 (600x450 pixels, 2-bit).

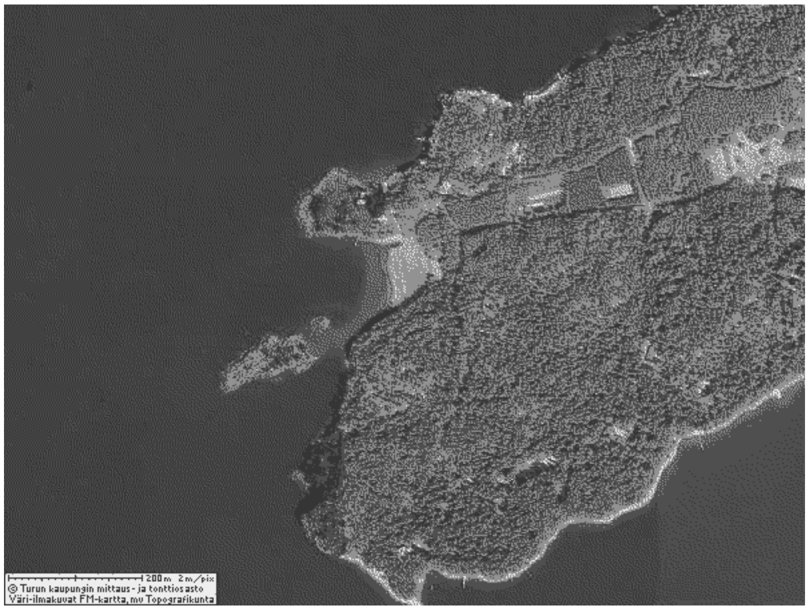
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 28: Turku2 (600x450 pixels, 2-bit) Floyd-Steinberg dithering.

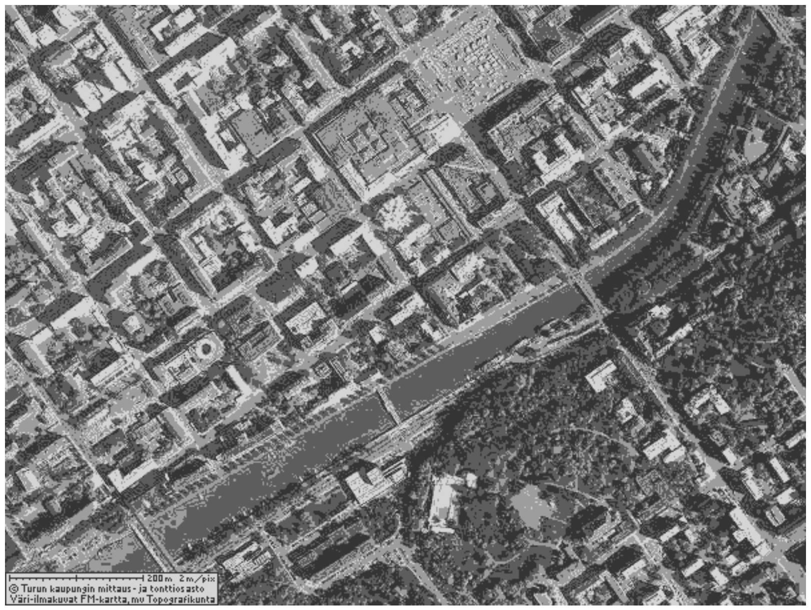
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 29: Turku3 (600x450 pixels, 2-bit).

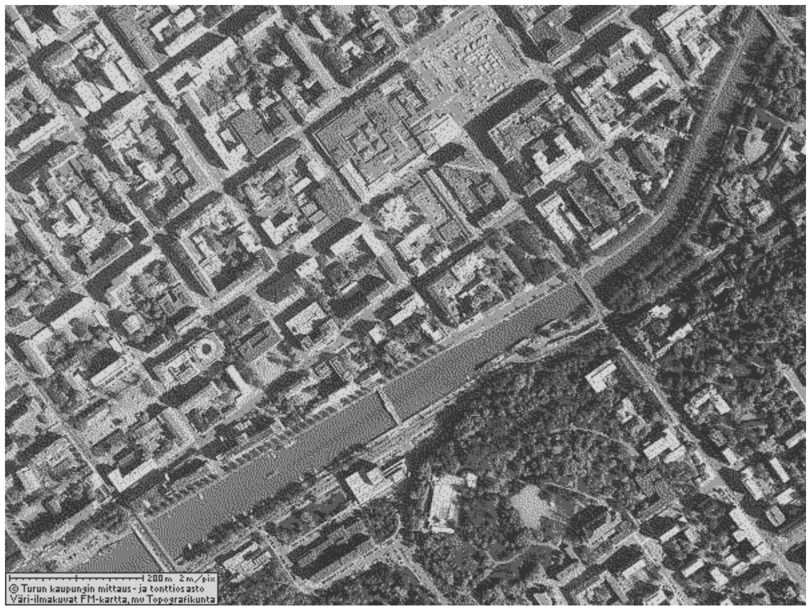
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 30: Turku3 (600x450 pixels, 2-bit) Floyd-Steinberg dithering.

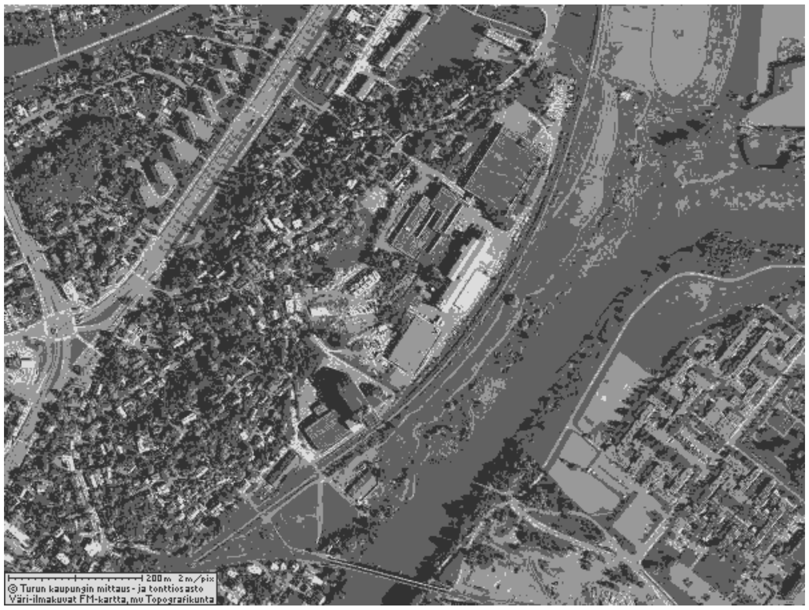
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 31: Turku4 (600x450 pixels, 2-bit).

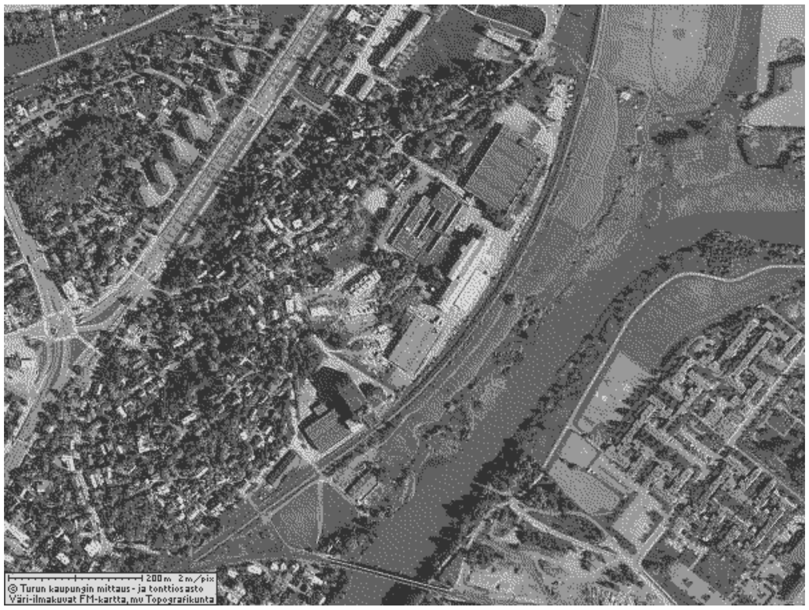
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 32: Turku4 (600x450 pixels, 2-bit) Floyd-Steinberg dithering.

Appendix C Compressed images 8-bit images

Turku1

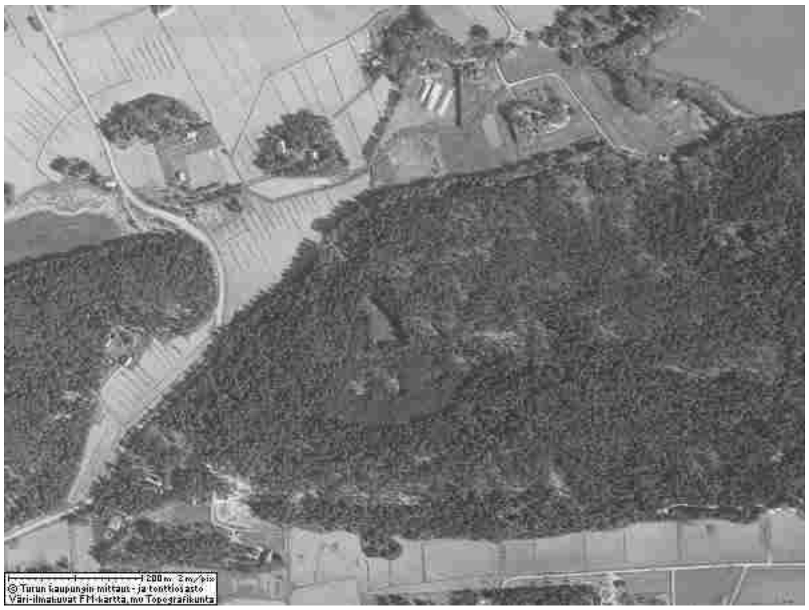
200 m 2 m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 33: JPEG at 0.5 bpp, MSE: 195.

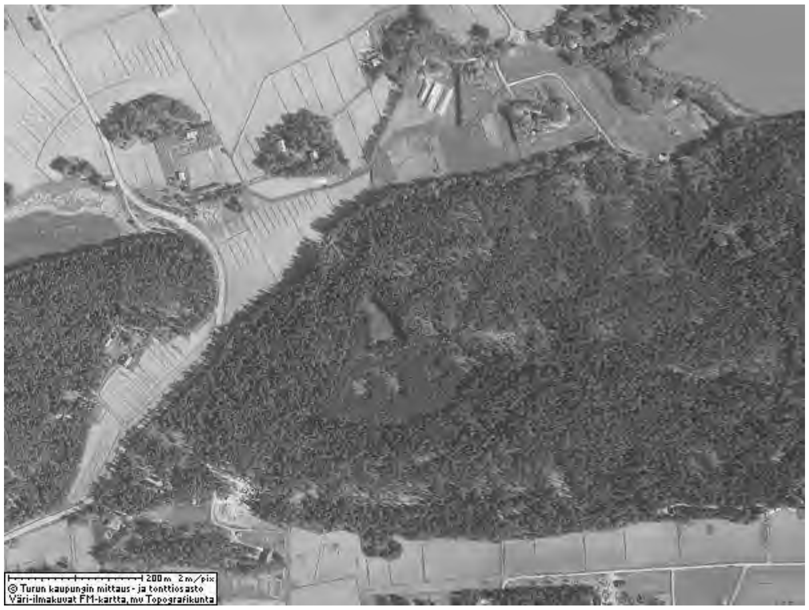
200 m 2 m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 34: JPEG2000 at 0.5 bpp, MSE: 142.

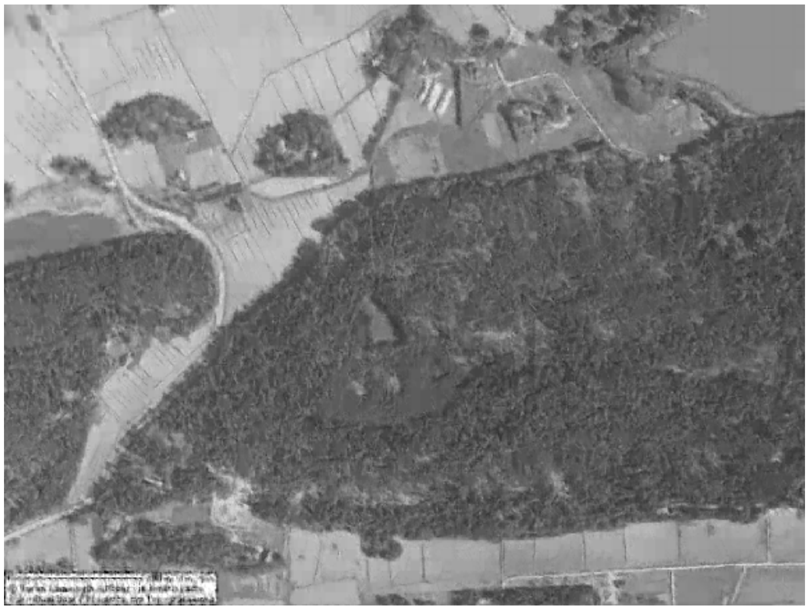

Figure 35: FIASCO at 0.5 bpp, MSE: 254.

Turku2

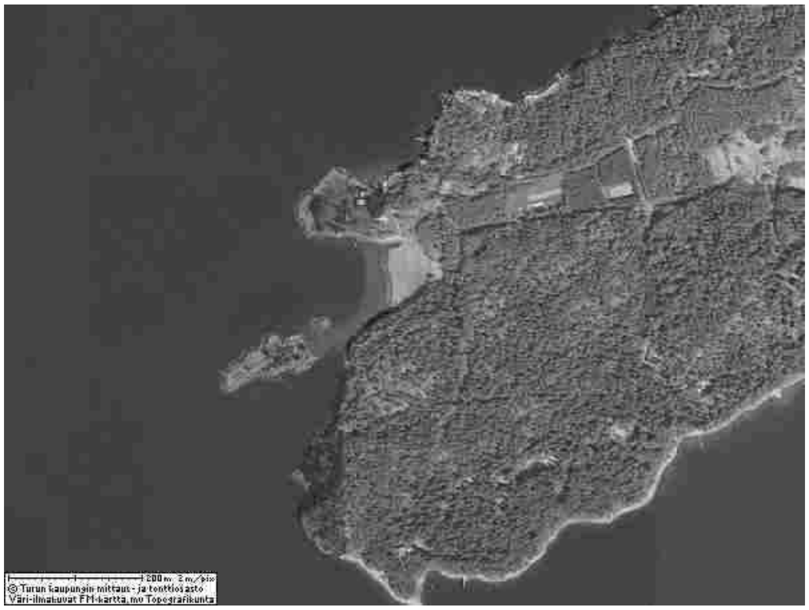
200 m, 2 m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 36: JPEG at 0.375 bpp, MSE: 172.

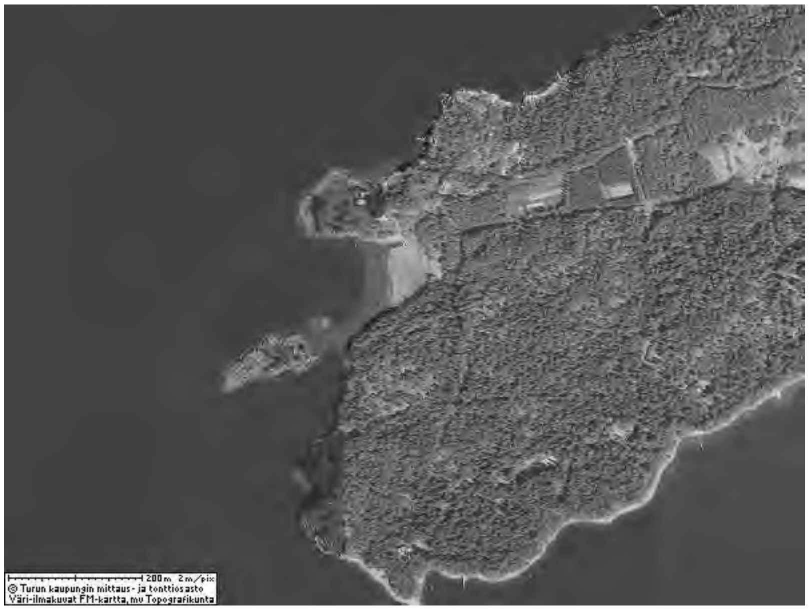
200 m, 2 m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 37: JPEG2000 at 0.375 bpp, MSE: 122.

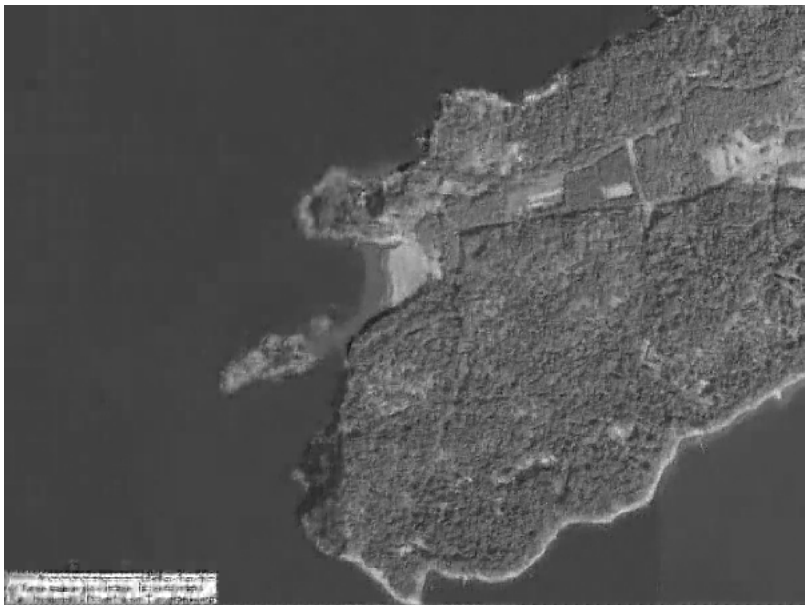
Figure 38: FIASCO at 0.375 bpp, MSE: 239.

Turku3

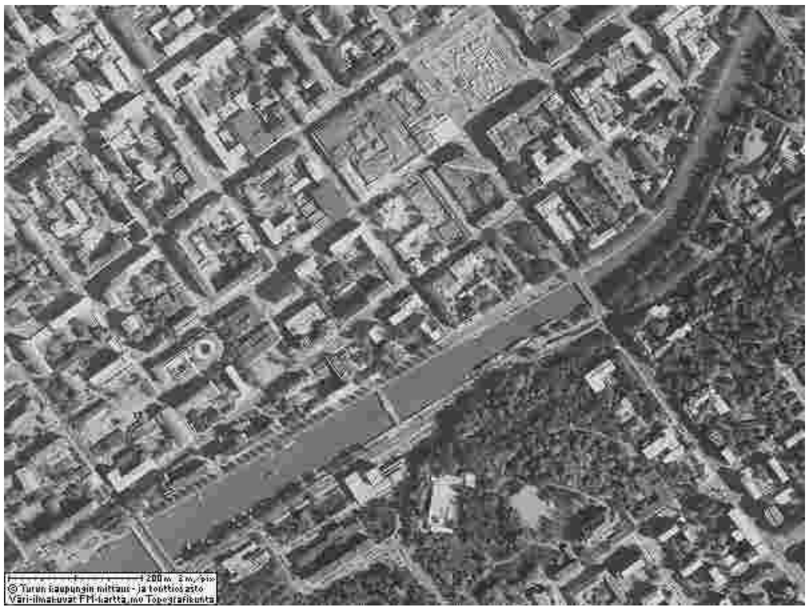
200 m 2 m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 39: JPEG at 0.5 bpp, MSE: 564.

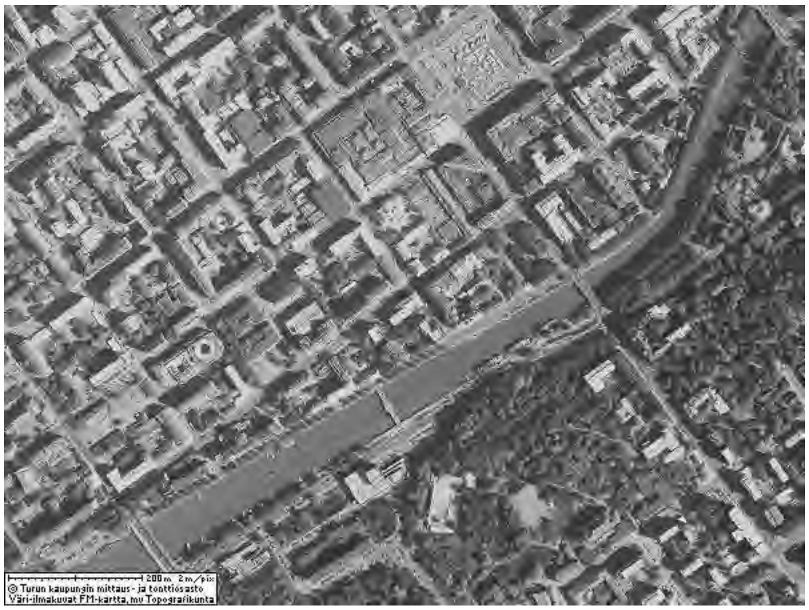
200 m 2 m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 40: JPEG2000 at 0.5 bpp, MSE: 466.

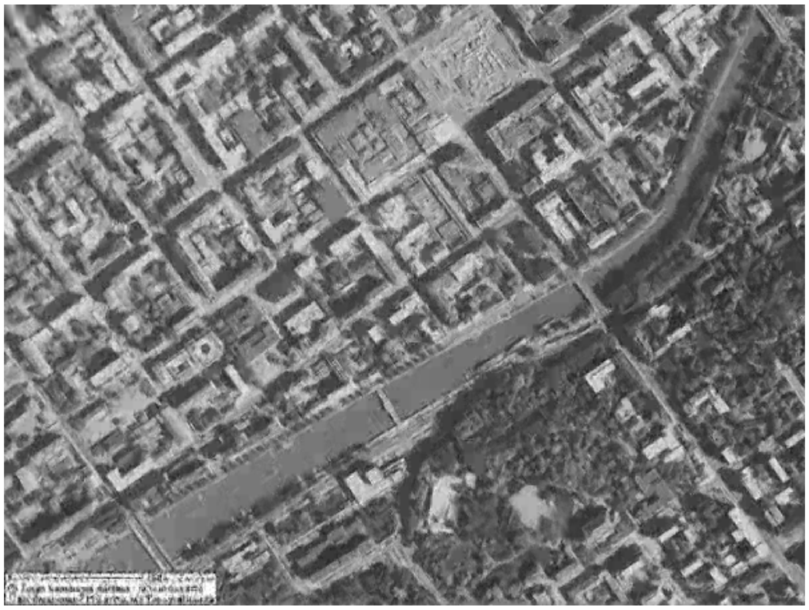

Figure 41: FIASCO at 0.5 bpp, MSE: 579.

Turku4

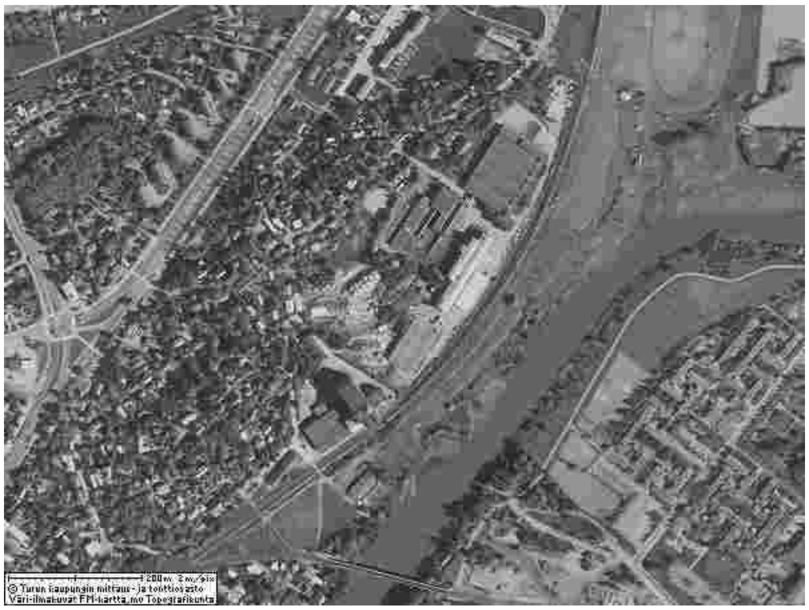
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 42: JPEG at 0.5 bpp, MSE: 374.

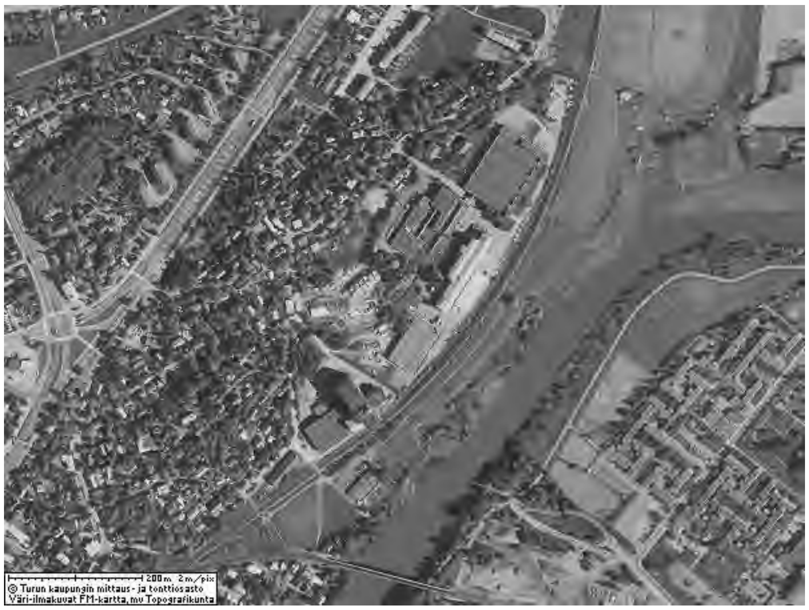
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 43: JPEG2000 at 0.5 bpp, MSE: 281.

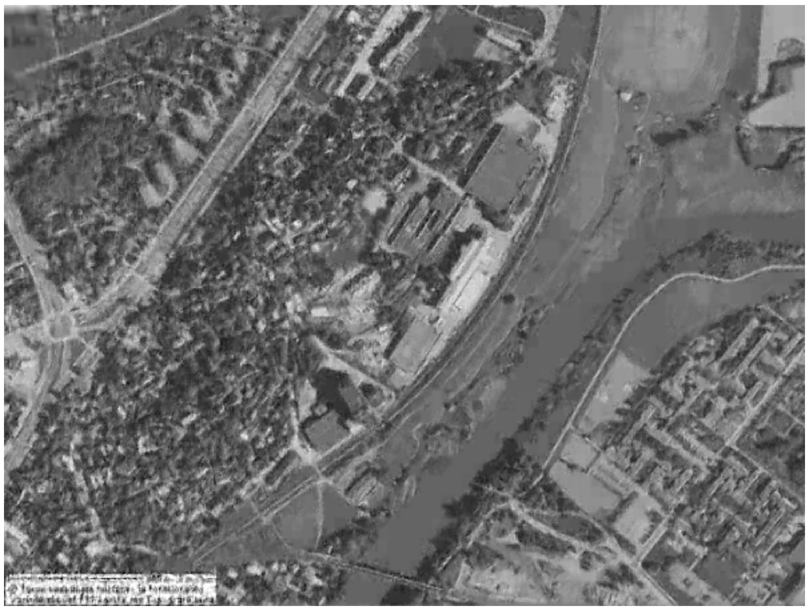

Figure 44: FIASCO at 0.5 bpp, MSE: 420.

Appendix D Compressed 2-bit dithered images

Turku1

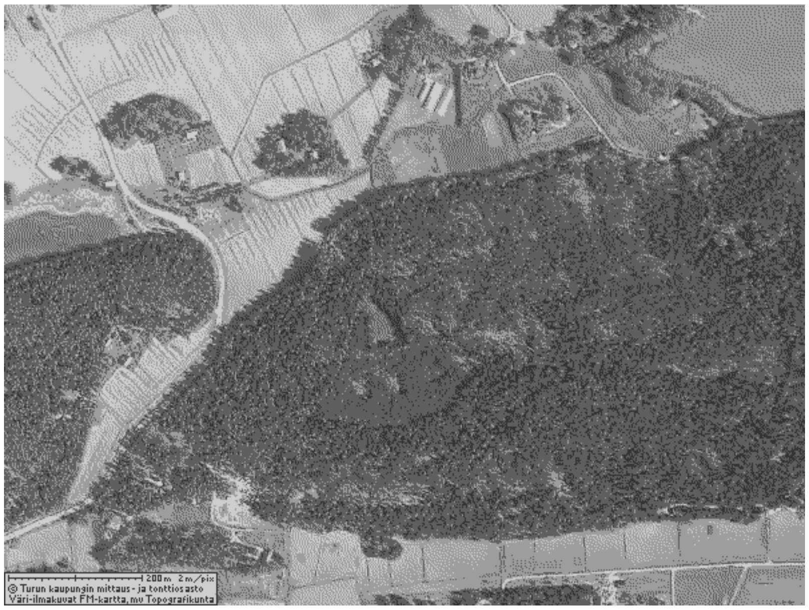
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 45: Compressed JPEG2000 at 1 bpp decompressed and Floyd-Steinberg error diffusion dithered to 2-bit image.

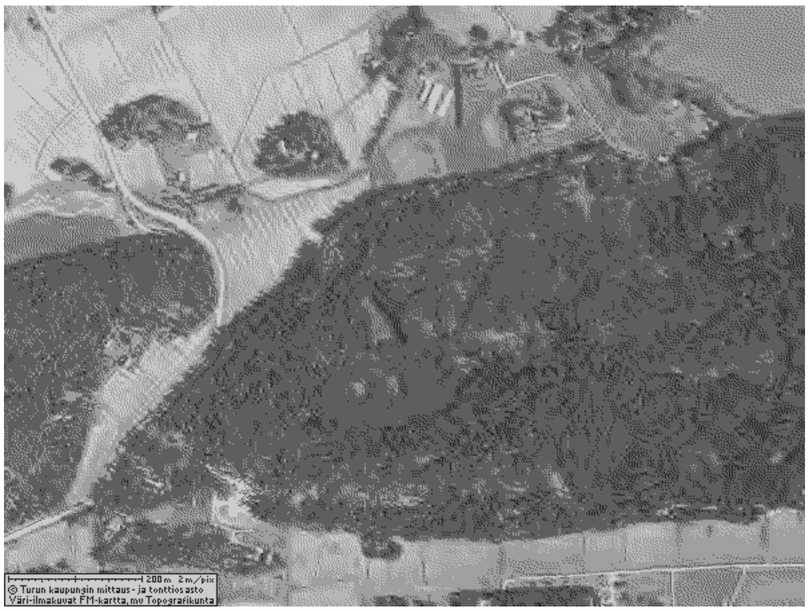
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 46: Compressed JPEG2000 at 0.25 bpp decompressed and Floyd-Steinberg error diffusion dithered to 2-bit image.

Turku2

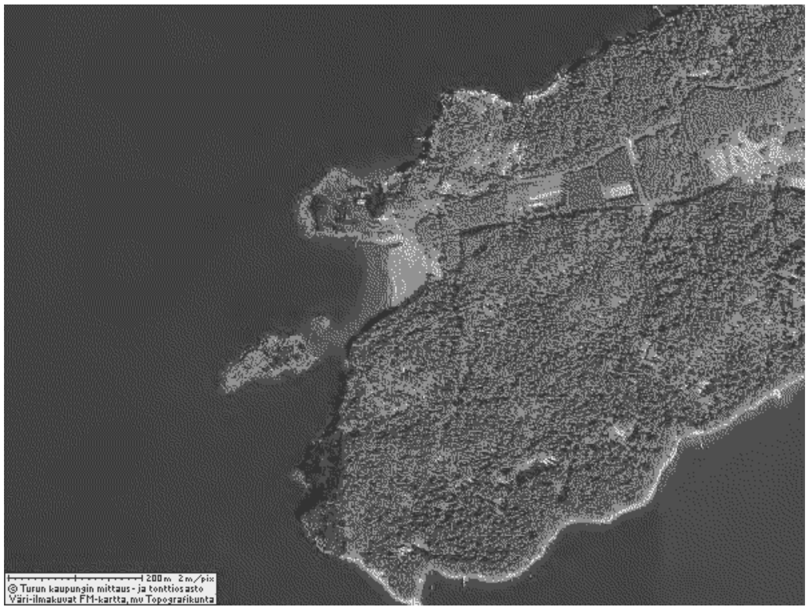
200 m 2 m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 47: Compressed JPEG2000 at 1 bpp decompressed and Floyd-Steinberg error diffusion dithered to 2-bit image.

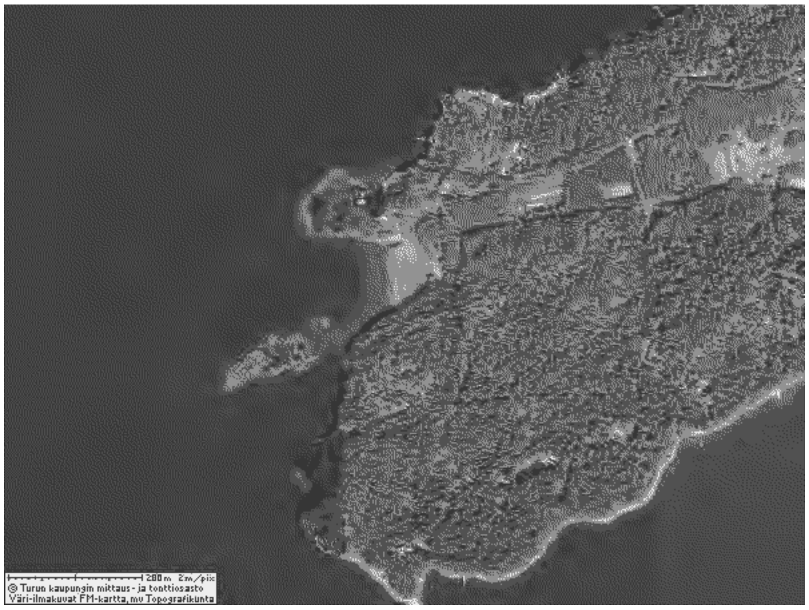
200 m 2 m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 48: Compressed JPEG2000 at 0.25 bpp decompressed and Floyd-Steinberg error diffusion dithered to 2-bit image.

Turku3

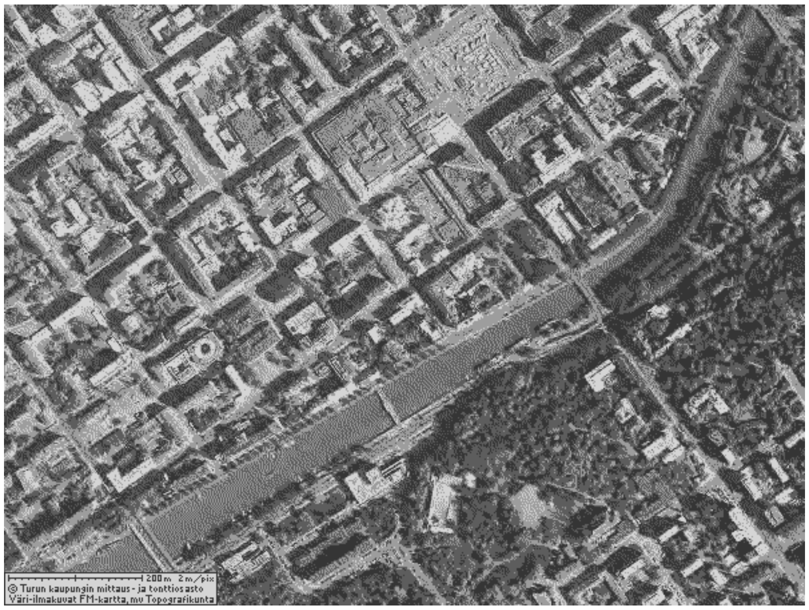
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 49: Compressed JPEG2000 at 1 bpp decompressed and Floyd-Steinberg error diffusion dithered to 2-bit image.

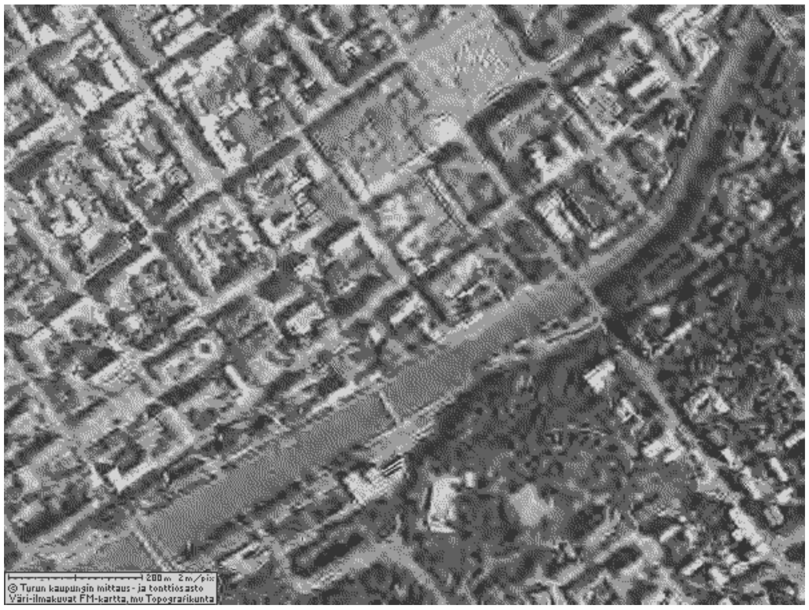
200m 2m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 50: Compressed JPEG2000 at 0.25 bpp decompressed and Floyd-Steinberg error diffusion dithered to 2-bit image.

Turku4

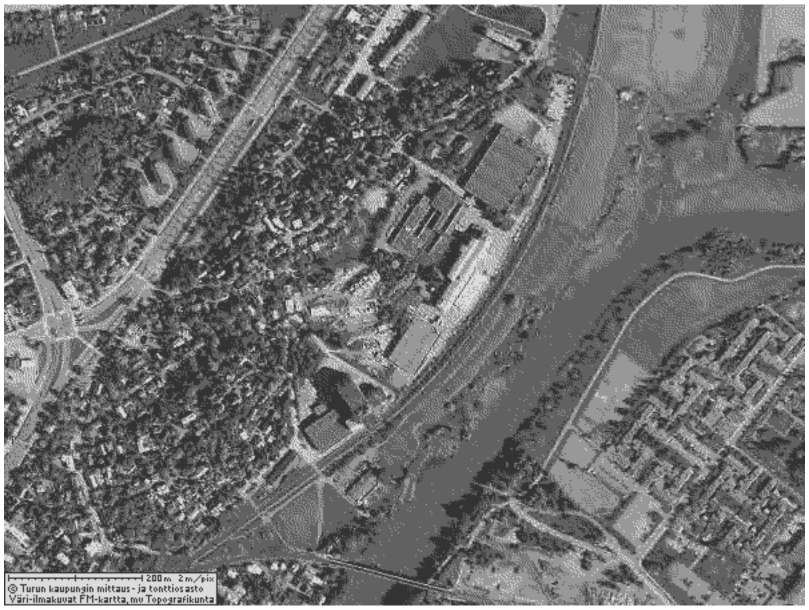
200 m 2 m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 51: Compressed JPEG2000 at 1 bpp decompressed and Floyd-Steinberg error diffusion dithered to 2-bit image.

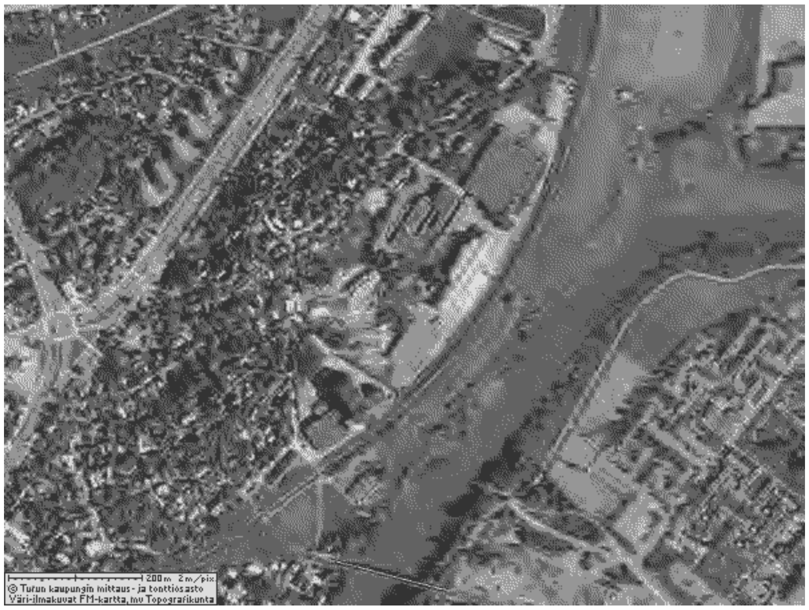
200 m 2 m/pix
© Turun kaupungin mittaus- ja tonttiosasto
Väri-ilmakuvat FM-kartta, mv Topografikunta

Figure 52: Compressed JPEG2000 at 0.25 bpp decompressed and Floyd-Steinberg error diffusion dithered to 2-bit image.

Appendix E Complete compression results

Table 8: Lossy compression results for image Turku1.

Target bpp	Method	Filesize	Actual bpp	MSE	PSNR
0.125	J2K	4182	0.123	278.49	23.68
0.125	JP2	4323	0.128	276.84	23.71
0.25	J2K	8419	0.249	216.32	24.78
0.25	JP2	8546	0.253	216.47	24.78
0.25	JPEG	8161	0.241	306.50	23.27
0.25	WFA	7776	0.230	303.53	23.31
0.375	J2K	12589	0.373	173.62	25.73
0.375	JP2	12922	0.382	172.36	25.77
0.375	JPEG	11153	0.330	250.55	24.14
0.375	WFA	12503	0.370	269.60	23.82
0.5	J2K	16875	0.500	141.37	26.63
0.5	JP2	16885	0.500	143.04	26.58
0.5	JPEG	15099	0.447	212.13	24.86
0.5	WFA	15713	0.466	254.66	24.07
0.625	J2K	21090	0.624	120.35	27.33
0.625	JP2	20765	0.615	123.45	27.22
0.625	JPEG	20331	0.602	194.36	25.24
0.625	WFA	20806	0.616	242.96	24.28
1	J2K	33696	0.998	69.82	29.69
1	JP2	33840	1.002	70.89	29.63
1	JPEG	25644	0.759	138.94	26.70
2	J2K	67471	1.999	15.29	36.29
2	JP2	67561	2.001	16.52	35.95
2	JPEG	66772	1.978	10.08	38.10
2	ECW	68522	2.03	114.402	27.55

Table 9: Lossy compression results for image Turku2.

Target bpp	Method	Filesize	Actual bpp	MSE	PSNR
0.125	J2K	4168	0.123	228.87	24.53
0.125	JP2	4322	0.128	227.54	24.56
0.25	J2K	8430	0.249	162.48	26.02
0.25	JP2	8509	0.252	163.30	26.00
0.25	JPEG	7559	0.223	259.82	23.98
0.25	WFA	7212	0.214	260.56	23.97
0.375	J2K	12652	0.374	115.61	27.50
0.375	JP2	12748	0.377	116.47	27.47
0.375	JPEG	12305	0.364	180.88	25.56
0.375	WFA	12106	0.359	236.79	24.39
0.5	J2K	16803	0.497	85.41	28.82
0.5	JP2	16924	0.501	86.28	28.77
0.5	JPEG	16895	0.500	140.08	26.67
0.5	WFA	12100	0.359	236.74	24.39
0.625	J2K	20885	0.618	66.26	29.92
0.625	JP2	20777	0.615	68.22	29.79
0.625	JPEG	18755	0.555	119.62	27.35
0.625	WFA	11995	0.355	239.35	24.34
1	J2K	33705	0.998	23.40	34.44
1	JP2	33836	1.002	23.99	34.33
1	JPEG	33275	0.985	18.75	35.40
2	J2K	67470	1.999	1.98	45.16
2	JP2	67430	1.997	2.50	44.16
2	JPEG	66195	1.961	0.80	49.10
2	ECW	64992	1.925	38.677	32.26

Table 10: Lossy compression results for image Turku3.

Target bpp	Method	Filesize	Actual bpp	MSE	PSNR
0.125	J2K	4196	0.124	898.29	18.60
0.125	JP2	4327	0.128	892.94	18.62
0.25	J2K	8225	0.243	684.53	19.78
0.25	JP2	8523	0.252	676.93	19.83
0.25	JPEG	8714	0.258	979.88	18.22
0.25	WFA	6620	0.196	833.95	18.92
0.375	J2K	12561	0.372	550.24	20.73
0.375	JP2	12903	0.382	545.97	20.76
0.375	JPEG	8714	0.258	979.88	18.22
0.375	WFA	11974	0.355	659.94	19.94
0.5	J2K	16867	0.499	462.42	21.48
0.5	JP2	16976	0.502	464.98	21.46
0.5	JPEG	15308	0.453	680.18	19.80
0.5	WFA	17150	0.508	580.44	20.49
0.625	J2K	20910	0.619	395.88	22.16
0.625	JP2	20499	0.607	407.08	22.03
0.625	JPEG	19845	0.588	557.86	20.67
0.625	WFA	21085	0.625	553.79	20.70
1	J2K	33745	0.999	240.68	24.32
1	JP2	33767	1.000	243.88	24.26
1	JPEG	31481	0.932	448.85	21.61
2	J2K	67449	1.998	65.54	29.97
2	JP2	67521	2.000	67.71	29.82
2	JPEG	67362	1.995	102.52	28.02
2	ECW	66977	1.984	423.109	21.87

Table 11: Lossy compression results for image Turku4.

Target bpp	Method	Filesize	Actual bpp	MSE	PSNR
0.125	J2K	4199	0.124	612.92	20.26
0.125	JP2	4254	0.126	612.85	20.26
0.25	J2K	8338	0.247	438.78	21.71
0.25	JP2	8433	0.249	438.88	21.71
0.25	JPEG	7201	0.213	741.46	19.43
0.25	WFA	8590	0.255	529.35	20.89
0.375	J2K	12470	0.369	349.39	22.70
0.375	JP2	12901	0.382	345.14	22.75
0.375	JPEG	12806	0.379	491.21	21.22
0.375	WFA	11717	0.347	469.51	21.41
0.5	J2K	16866	0.499	279.74	23.66
0.5	JP2	16927	0.501	281.69	23.63
0.5	JPEG	16615	0.492	389.23	22.23
0.5	WFA	16706	0.495	421.98	21.88
0.625	J2K	21010	0.622	227.58	24.56
0.625	JP2	20864	0.618	232.09	24.47
0.625	JPEG	21250	0.629	327.75	22.98
0.625	WFA	20866	0.618	412.11	21.98
1	J2K	33604	0.995	130.18	26.99
1	JP2	33814	1.001	131.26	26.95
1	JPEG	33227	0.984	216.92	24.77
2	J2K	67389	1.996	31.59	33.14
2	JP2	67584	2.002	32.67	32.99
2	JPEG	67094	1.987	14.44	36.53
2	ECW	66977	1.984	423.109	21.87

Table 12: Lossless compression results by bitplanes for 2bit images.

Format	Image	Layer 1	bpp	Layer 2	bpp	Layers total	total bpp	Image type
GIF	Turku1	36.315	1.08	21.318	0.63	57.633	1.71	2b dither
JBIG	Turku1	29.837	0.88	15.439	0.46	45.276	1.34	2b dither
TIFF	Turku1	56.112	1.66	30.814	0.91	86.926	2.58	2b dither
GIF	Turku2	32.831	0.97	19.051	0.56	51.882	1.54	2b dither
JBIG	Turku2	26.096	0.77	15.091	0.45	41.187	1.22	2b dither
TIFF	Turku2	90.402	2.68	27.556	0.82	117.958	3.50	2b dither
GIF	Turku3	39.483	1.17	29.861	0.88	69.344	2.05	2b dither
JBIG	Turku3	33.456	0.99	19.878	0.59	53.334	1.58	2b dither
TIFF	Turku3	63.426	1.88	37.480	1.11	100.906	2.99	2b dither
GIF	Turku4	37.715	1.12	28.383	0.84	66.098	1.96	2b dither
JBIG	Turku4	30.936	0.92	20.003	0.59	50.939	1.51	2b dither
TIFF	Turku4	55.370	1.64	37.930	1.12	93.300	2.76	2b dither
GIF	Turku1	29.992	0.89	15.848	0.47	45.840	1.36	2b nodither
JBIG	Turku1	23.017	0.68	10.610	0.31	33.627	1.00	2b nodither
TIFF	Turku1	34.176	1.01	18.776	0.56	52.952	1.57	2b nodither
GIF	Turku2	23.792	0.70	16.909	0.50	40.701	1.21	2b nodither
JBIG	Turku2	17.549	0.52	13.080	0.39	30.629	0.91	2b nodither
TIFF	Turku2	28.548	0.85	20.516	0.61	49.064	1.45	2b nodither
GIF	Turku3	37.745	1.12	26.983	0.80	64.728	1.92	2b nodither
JBIG	Turku3	30.803	0.91	17.343	0.51	48.146	1.43	2b nodither
TIFF	Turku3	50.984	1.51	29.668	0.88	80.652	2.39	2b nodither
GIF	Turku4	33.366	0.99	23.935	0.71	57.301	1.70	2b nodither
JBIG	Turku4	25.468	0.75	15.343	0.45	40.811	1.21	2b nodither
TIFF	Turku4	40.246	1.19	27.412	0.81	67.658	2.00	2b nodither

Table 13: Results of lossless 2bit compression.

Format	Image	Compressed Size	bpp	Image type
PNG	Turku1	47635	1.411	2bit dither
PNG	Turku2	41826	1.239	2bit dither
PNG	Turku3	61238	1.814	2bit dither
PNG	Turku4	54606	1.617	2bit dither
PNG	Turku1	39016	1.156	2bit
PNG	Turku2	28933	0.857	2bit
PNG	Turku3	57894	1.715	2bit
PNG	Turku4	47773	1.415	2bit
GIF	Turku1	46443	1.376	2bit dither
GIF	Turku2	40643	1.204	2bit dither
GIF	Turku3	60578	1.794	2bit dither
GIF	Turku4	53208	1.576	2bit dither
GIF	Turku1	37222	1.102	2bit
GIF	Turku2	29465	0.873	2bit
GIF	Turku3	56499	1.674	2bit
GIF	Turku4	46021	1.363	2bit
TIFF	Turku1	49306	1.460	2bit dither
TIFF	Turku2	43476	1.288	2bit dither
TIFF	Turku3	65604	1.943	2bit dither
TIFF	Turku4	57310	1.698	2bit dither
TIFF	Turku1	39962	1.184	2bit
TIFF	Turku2	32182	0.953	2bit
TIFF	Turku3	61282	1.815	2bit
TIFF	Turku4	49692	1.472	2bit