JOURNAL OF VISUAL COMMUNICATION AND IMAGE REPRESENTATION
Vol. 6, No. 4, December, pp. 366–377, 1995

Compression of Binary Images by Composite Methods Based on Block Coding

PASI FRÄNTI AND OLLI NEVALAINEN

Department of Computer Science, University of Turku, Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland

Received March 21, 1994; accepted March 8, 1995

Composite methods for compressing binary images are studied. Hierarchical block coding is the main component in all of them. An attempt is made to increase the compression by augmenting the block coding by predictive coding and bit row reordering. The purpose is to increase the number of white pixels and all-white blocks. An error image is constructed from the differences between the predicted and original values of the pixels. The error image is then coded by hierarchical block coding, in which Huffman coding is used to encode the different bit patterns at the lowest level of the hierarchy. In the method, the global level dependencies are thus handled by block coding and the local pixel-to-pixel dependencies by Huffman coding. © 1995 Academic Press, Inc.

1. INTRODUCTION

Binary (or *black-and-white*) *images* represent the simplest and most space-economic form of raster images and are of great interest when colors or gray scales are not needed. Typical examples of binary images are facsimiles and technical drawings.

A binary image can be stored as a bit map consisting of Y rows, each containing X black or white points. The input alphabet for a compression method is thus {BLACK, WHITE} and each individual point, a *pixel*, is coded by one bit. Further, one can organize eight consecutive bits in a row to form a byte of data. In most textual images the number of white pixels greatly exceeds the number of black pixels and there is a strong correlation between the values of neighboring pixels. A compression technique thus has excellent chances of succeeding.

The simple visual interpretation of the data file has helped in the development of a large number of nice compression algorithms of varying performances [1–7]. Block coding is one of the interesting compression techniques in this class [8–10] due to its relatively simple implementation, speed, and reasonable compression ratio. In addition, one can augment the basic technique with preprocessing steps aiming at a better compression ratio at the cost of increased running time. Of these we consider both fixed and adaptive *prediction techniques* and introduce an *image reordering* preprocessing phase. These have been used earlier within run-length coding and other methods; however, only a very simple one-pixel prediction model has been considered within block coding [9]. The reordering technique is similar to that used by Netravali and Mounts [11] in conjunction with run-length coding.

We start in Section 2 by recalling the block coding technique and its hierarchical variant. At the lowest level of the hierarchy a small block of pixels can be coded with a suitable variable length code, such as Huffman coding. Prediction techniques are then discussed in Section 3. To keep the compression algorithm simple, we consider only relatively simple context models (of orders 4 and 8). In Section 4 *adaptive* prediction is briefly discussed in the case of *dithered* images. A new variant of block coding is introduced in Section 5. Here the image reordering is applied to the *error image* (or *difference image*) prior to the final block coding phase. A comparison to standard facsimile compression techniques is given in Section 6.

2. HIERARCHICAL BLOCK CODING

The natural idea in block coding is to divide the image into blocks of pixels. A totally white block (*all-white block*) is coded by a single 0-bit. All other blocks (*nonwhite blocks*) thus contain at least one black pixel. They are coded with a 1-bit as a prefix followed by the contents of the block, bit by bit in row-major order; see Fig. 1.

In a hierarchical variant of block coding one first selects the size of the largest pixel block by fixing a value b ($=2^i$). The bit map is then divided into square blocks of $b \times b$ pixels. If a particular block is all-white, it is coded by a single 0-bit. Otherwise the block is coded by a 1-bit and then divided into four subblocks of the same size which are recursively coded in the same manner. Block coding is carried on until the block reduces to a single pixel, which is stored as such, see Fig. 2.

We performed computer tests with hierarchical block

1047-3203/95 $12.00

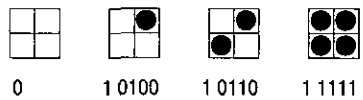


FIG. 1. Basic block coding technique. Coding of four sample blocks.

coding and noted that the initial block size 16 × 16 is a good choice. In the tests we used a set of typical white-dominant images including handwritten and typewritten text and a technical drawing (see the Appendix). The A4-format images were scanned with a resolution of 150 × 150 dpi, giving ca. two million pixels per image. The best and the worst case results of using the 32 × 32 block size can be estimated as follows.

A 32 × 32 block size could, in the worst case (when there are no 32 × 32 all-white blocks), increase the code length by 1 bit for each original 32^2 bits, giving an excess of ca. 0.1%, which is negligible. To estimate the optimal gain, let us make an idealistic assumption that all the 16 × 16 all-white blocks are oriented so that they form a maximal number of 32 × 32 all-white blocks. Then at most three bits for each 32 × 32 all-white block could be saved by using the greater block size. For the set of test images, the gain remains less than 1.5% of the code length compared to the code length with a block size of 16 × 16. Thus the benefits from a larger (32 × 32) block size would have been insignificant. On the other hand, one would have needed a buffer two times as large as that of the 16 × 16 blocks, and the processing would have been slower.

As pointed out by Kunt and Johnsen [9], the power of block coding can be improved by coding the bit patterns of the 2 × 2 blocks by *Huffman coding*. Because the frequency distributions of these patterns are quite similar for our set of test images, we can determine a static Huffman code from the joint distribution of the test images, see Fig. 5. It should be noted that the frequency distribution excludes the all-white blocks 4 × 4 or larger. The Huffman coding gives an improvement of ca. 10% in comparison to the basic hierarchical block coding.

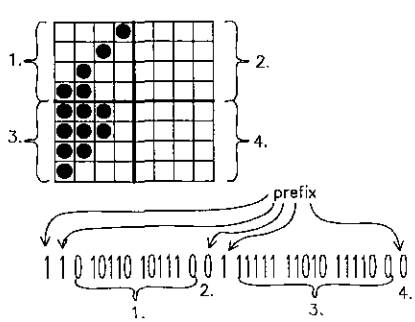


FIG. 2. Hierarchical block coding.

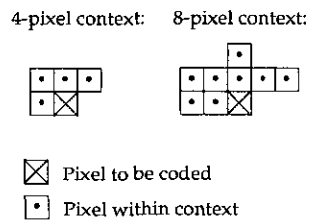


FIG. 3. The 4- and 8-pixel prediction contexts.

3. THE USE OF PREDICTION TECHNIQUES

Hierarchical block coding takes advantage of all-white blocks in the image. It works very well for typical facsimile images with a high proportion of white pixels but becomes inefficient when the proportion of black pixels is unusually high. Moreover, the Huffman coding for the 2 × 2 blocks is fixed, and thus strongly depends on the predefined probability distribution of the pixel patterns. Preprocessing of the image by a suitable prediction technique [11] gives a solution to these problems. Here we describe our implementation of the method.

The idea is to form a so-called "*error image*" from the original one by comparing the value of each original pixel to the value given by a *prediction function*. If these two are equal, the pixel of the error image is white; otherwise it is black. Benefit is gained from the increased number of white pixels and all-white blocks. Moreover, the prediction technique transforms the image toward an uncorrelated pattern of pixels. This decreases the local dependencies in the 2 × 2 blocks, and thus makes the static Huffman coding more robust on different types of images.

The prediction is based on the values of certain (fixed) neighboring pixels. These pixels have already been encoded and are therefore known to the decoder. The prediction is thus identical in the encoding and decoding phases. The number of the neighboring pixels gives the *degree* of the prediction function. We consider the 4- and 8-pixel *prediction contexts* of Fig. 3. The 4-pixel context is preferred, because the 8-pixel context is too excessive to gain any remarkable improvement, and is slower to operate with. The image is scanned in row-major order and the value of each pixel is predicted from the particular observed combination of the four neighboring pixels.

Takagi and Tsuda [6, p. 834; 12] have proposed a 4-pixel prediction function which is based on Boolean algebra and has the same prediction context as our implementation. Kunt and Johnsen [9, p. 778] applied a simple 1-pixel prediction model where the difference between the current pixel and the pixel above is determined; pixel $p'_{x,y}$ of the error image is the modulo-2 sum of the original pixels $p_{x,y}$ and $p_{x,y+1}$. This prediction scheme can be considered a Markov model of degree 1 (whereas ours and Takagi's are

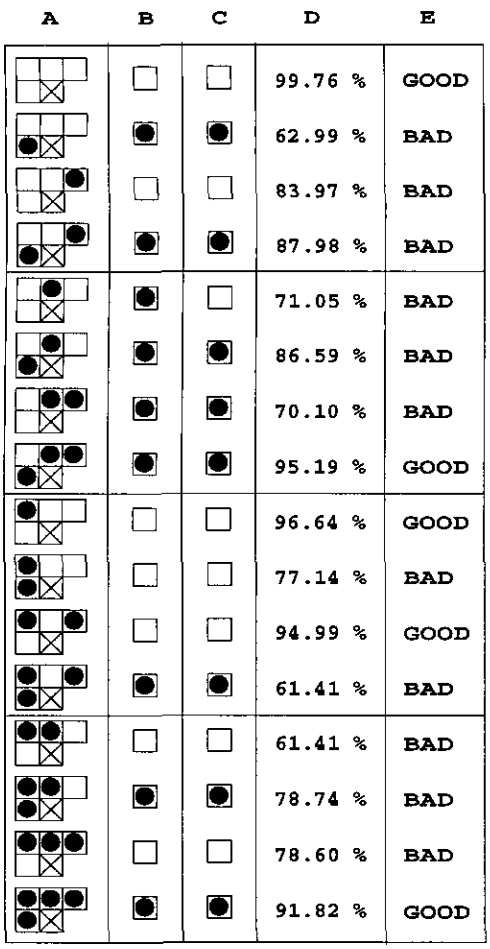


FIG. 4. Prediction function from test images. (A) Prediction context: X = the point to be predicted. (B) Predicted pixel value. (C) Takagi and Tsuda's prediction function. (D) The proportional frequency of the occurrence of the predicted value. (E) Quality of the prediction (see Section 5).

of degree 4). The advantages of the 1-pixel model are speed and simplicity.

One could use a higher order prediction model, cf. Netravali and Mount's [11] 7-pixel context and Moffat's [3] experiments with the arithmetic coding of up to 22-pixel contexts along with two different alternative models. However, the increase in the degree of the model brings extra benefit only up to a point. A large prediction context indeed gives a more accurate prediction, but in contrast to arithmetic coding, the exact probability of the value of a pixel is not important here; it only matters whether the estimate is above or below 50%.

We determined a prediction function from the set of test images using the 4-pixel context of Fig. 3. The function and the pattern frequencies are shown in Fig. 4. The prediction function is identical (with only one exception) to that given by Takagi and Tsuda [6, 12]. The frequency of a correct prediction varies from 61.4 to 99.8%, depending on the context; the completely white context predicts a white pixel with a very high probability, and a context with two white and two black pixels usually gives only an uncertain prediction.

It is observed that the predictive coding has the desired effect of increasing the proportion of white pixels, see Table 1a. The proportion of black pixels has decreased to one third compared to that of the original images. This in turn gives an advantage in the block coding in the form of an increase in the number of all-white blocks, see Table 1b.

The Huffman codes for the 2×2 blocks are given in Fig. 5 for both the original and the error images. The entropy of the 2×2 blocks due to prediction technique has decreased to 2.50 compared to 2.93 for the original image blocks. This originates from the increase in white pixels in the images. The average code lengths of the Huffman codes are very close to the entropy.

A closer examination of the 2×2 patterns of the error images shows that the correlation between individual pixels has significantly decreased. The differences between the probabilities of the patterns are mainly induced by the number of white pixels in each pattern; the more white pixels there are the higher the probability of that particular pattern. This can be verified by comparing the entropies of the 0-degree and the 4-degree Markov models in the images. For random pixel patterns these two numbers are equal, whereas for correlated patterns the 4-degree model gives lower entropies.

The entropies of the 0-degree (memoryless) and the 4-degree Markov models for the original and the error images are shown in Table 2. In the original images the correlation of the pixels is clearly reflected in the 4-degree entropy, which is much lower than the entropy of the 0-degree model. For the error images, the difference in the entropies has reduced significantly (indicating less correlated pixel patterns). At the same time, a minor increase in the entropy of the 4-degree patterns is observed.

4. ADAPTIVE PREDICTION

In the previous section the prediction function was fixed; the same function was used for whatever images are to be

TABLE 1
The Effect of the Prediction Technique

(a) Proportion of white pixels

	Image 1	Image 2	Image 3	Image 4	Average
Original image	98.0%	96.4%	94.8%	87.1%	94.1%
Error image	99.0%	98.2%	98.9%	96.0%	98.0%

(b) Proportion of all-white blocks at the different levels of block size

	16×16	8×8	4×4	2×2
Original images	66.3%	35.4%	40.3%	40.7%
Error images	67.4%	38.5%	45.9%	54.4%

	Original images:		Error images:	
	40.67	0	54.36	0
	2.90	10111	8.22	100
	2.87	10110	7.81	1111
	2.58	111101	7.53	1110
	2.50	111100	7.51	1101
	5.23	11111	2.51	10111
	5.27	1000	2.04	10101
	5.74	1010	1.79	110011
	5.67	1001	1.43	110001
	0.16	11100101	2.05	10110
	0.12	11100100	0.77	101000
	1.81	111000	1.24	101001
	2.21	111011	0.35	11001000
	1.75	1110011	1.26	110000
	2.18	111010	0.77	1100101
	18.29	110	0.38	11001001
	Entropy:	Average code length:	Entropy:	Average code length:
	2.93	2.99	2.50	2.54

FIG. 5. Proportional frequencies (%) and Huffman codes of the 2×2 blocks for original and error images.

coded. One fact speaking for a fixed mode is that for all four test images the prediction function was identical when determined separately. There are, however, binary images that differ significantly from typical facsimile images. One such group consists of *dithered images*; see test image Dundee in Fig. 6.

"Dithering" stands for a preprocessing method where the number of colors or gray scales is decreased without too much loss in the quality of the image. Here gray-scale images have been transformed to binary images by maintaining the average gray value of a region. This is done by mixing black and white pixels so that the original grayness seems to remain, see [13–15].

Now the fixed prediction model gives weak results since the local statistics in the image differ from the typical facsimile images. Moreover, the statistics of the local pixel patterns strongly depend on the chosen dithering method. The application of the fixed compression method may even cause an increase in the storage size.

TABLE 2
Entropy of the 0-Degree and 4-Degree Markov Model for Test Images

	Image 1	Image 2	Image 3	Image 4	Average
Original image:					
0-degree entropy	14.1%	22.4%	29.5%	55.5%	30.4%
4-degree entropy	3.8%	6.9%	5.3%	15.5%	7.9%
Error image:					
0-degree entropy	8.0%	13.2%	8.7%	24.2%	13.4%
4-degree entropy	5.6%	9.5%	6.4%	18.9%	10.1%

In the adaptive modeling the frequencies of black and white pixels in each context are updated while encoding the image. Each prediction value is determined on the basis of the current frequencies. The method significantly increases the proportion of white pixels and the all-white blocks, see Table 3.

The fact that adaptive modeling brings effective compression does not mean that its application is necessarily reasonable in practice. It gives remarkable improvement in the case of dithered images, whereas for the other test images the advantage is only marginal; see Section 6. The disadvantage of adaptive modeling is the increased running time. The fixed prediction takes only half the time needed for the adaptive prediction.

5. ORDERING TECHNIQUES

In this section we introduce a new block coding technique in which a special reordering of the image is performed after the prediction phase. Our compression method consists of the following steps:

(1) Construct an error image.

(2) Reorder the pixels of each row in the error image.

(3) Compress the reordered image by hierarchical block coding.

The idea is from Netravali and Mounts [11] with the difference that instead of block coding they applied run-length coding at step 3. *Image reordering* is a natural follower of the predictive coding. The goal is to rearrange the pixels so that the number of all-white blocks would increase. Here each prediction context is classified as either GOOD or BAD according to the estimated frequency of a successful prediction. Thus one can fix a threshold frequency in the prediction model and refer to all predictions with a higher frequency as GOOD and to the remaining ones as BAD. With a threshold of 90%, which is due to [11], we have the classification shown in Fig. 4.

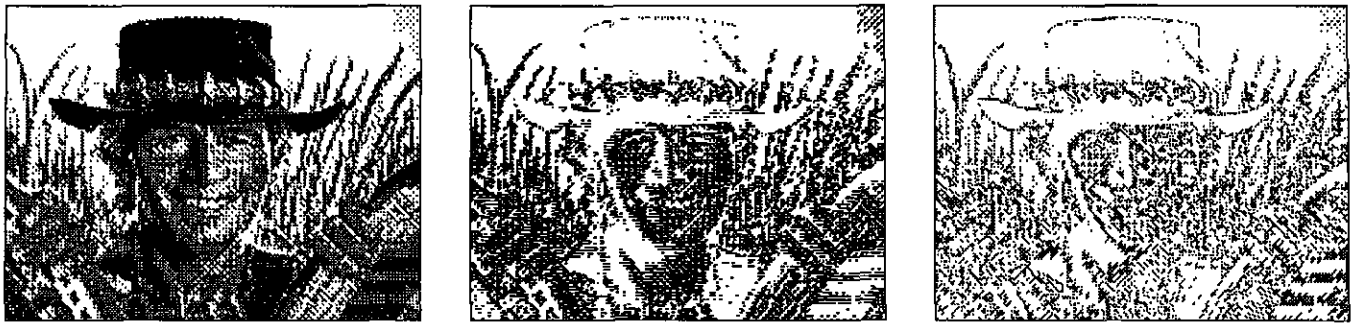


FIG. 6. Test image Dundee (256 × 192 pixels) (left), error image of Dundee according to fixed prediction (middle), and error image of Dundee according to adaptive prediction (right). Reprinted with permission from Paramount Pictures.

The pixels of an error image are reordered so that all the predicted pixels for which the prediction is classified as GOOD are stacked by starting from the left border of the row, and those in the BAD category are stacked by starting from the right border toward the beginning of the row. Note that both black and white pixels appear in the beginning and end of a row, but it is expected that the white pixels are prevailing in the beginning of a row. Although the pixels are mixed in the reordering phase, it is still possible to decode each row correctly. This is because the coding context is known (cf. Fig. 3) while a particular pixel is being decoded and thus the decoder knows whether the next pixel is to be picked from the left or the right stack (corresponding to a GOOD or a BAD prediction context).

The number of all-white blocks increased in the test images but the increase was less than expected; see Table 4. The number of 8 × 8-blocks increased but the number of 2 × 2-blocks, on the other hand, decreased. The all-white blocks required overall 6.9% fewer bits, yielding a reduction of 1.8% in the total size of the compressed file. The reordering, however, changes the distribution of the nonwhite 2 × 2 blocks so that the Huffman coding becomes more effective. This gives an extra gain of 1.5% in the compression ratio, leading to a 3.3% overall improvement.

TABLE 3
The Proportion of White Blocks in the Test Image Dundee

	16 × 16	8 × 8	4 × 4	2 × 2	Proportion of white pixels
Original image	5.7%	4.1%	7.6%	15.7%	54.2%
After fixed prediction	7.8%	7.5%	15.5%	24.8%	62.9%
After adaptive prediction	7.8%	7.2%	15.5%	36.7%	82.6%

TABLE 4
Proportion of All-White Blocks at the Different Levels of the Block Hierarchy

	16 × 16	8 × 8	4 × 4	2 × 2	Proportion of white pixels
Error images	67.4%	38.5%	45.9%	54.4%	98.0%
Reordered error images	66.4%	43.9%	46.7%	51.5%	98.0%

6. COMPARISON TO STANDARD COMPRESSION METHODS

The composite block coding methods (see Table 5) are compared to the CCITT (Consultative Committee for International Telegraphy and Telephone) Group 3 and Group 4 standard compression algorithms (G3, G4), and to the QM coder [16] with an 8-pixel context model (QM8). The latter algorithm is the main component of the JBIG (Joint Bi-level Image Group) [2; 17, Section 20.3], which is a draft of international standards by the ISO/IEC (International Organization for Standardization/International Electrotechnical Committee) and CCITT.

CCITT Group 3 and Group 4 Standards

Run-length coding (*RLE*) [1] is probably the best known compression method for binary images. The coding is extremely simple due to the binary alphabet. It is sufficient only to code the lengths of each run; no color information is needed. The method efficiently codes large uniform areas in the images, even though the two-dimensional correlations are ignored. Instead of the lengths of the runs, one can code the location of the boundaries of the runs (the black/white transitions) relative to the boundaries of the previous row. This is the

TABLE 5
Abbreviations of the Compression Algorithms

Abbrev.	Description of the method
B	Two-dimensional hierarchical block coding
PB1	Fixed 1-pixel prediction technique + block coding
PB4	Fixed 4-pixel prediction technique + block coding
PB8	Fixed 8-pixel prediction technique + block coding
OB4	Fixed 4-pixel prediction technique + ordering + block coding
OB8	Fixed 8-pixel prediction technique + ordering + block coding
DOB8	Adaptive 8-pixel prediction technique + ordering + block coding
G3	CCITT Group 3 standard compression algorithm
G4	CCITT Group 4 standard compression algorithm
QM8	QM coder with 8-pixel context model

basic idea in the method called *relative element address designate* (*READ*) coding [1].

The RLE and READ algorithms are included in two image compression standards, known as CCITT Group 3 (G3) and Group 4 (G4). They are nowadays widely used in FAX machines. In the G3 standard every kth line of the image is coded by the 1-dimensional RLE method (also referred as *modified Huffman*) and the 2-dimensional READ code (more accurately referred as *modified READ*) is applied for the rest of the lines. The G3 and G4 standards include two optional resolutions of the image (low resolution, 200 × 100 dpi; high resolution, 200 × 200 dpi). (In G3, the k-parameter is set to 2 for low resolution images, and to 4 for high resolution images. In G4, k is set to infinite so that every line of the image is coded by READ code.)

Context-Based Compression with QM Coder (Baseline JBIG)

In the baseline JBIG (sequential mode) the image is processed in row-major order from left to right. A local context model is applied for each pixel of the image. A probability is assigned for the pixel, which is then encoded by arithmetic coding according to its probability. The probabilities are adaptively determined on the basis of the pixels that have already been encoded. For consistency, we apply here the same 8-pixel context model (see Fig. 3) that was used in the predictive block coding scheme in Section 3. The difference between this method and our predictive coding scheme in Section 3 lies in the coding phase. No error image is constructed here, but arithmetic coding is applied directly to the pixels of the original image.

A QM coder is used as the arithmetic coding component because it is fast, efficient, and tailored for a binary source. For example, all multiplication operations have been replaced by fast approximations or by shift-left operations. The coding algorithm of a QM-coder is suboptimal, but the overall performance is usually better when compared to a straightforward count-based probability estimation with an optimal arithmetic coder like [18]. This originates from the sophisticated probability estimation of QM coder.

Test Results

The composite methods of the block coding and the QM8 algorithm were implemented by using C language and a TurboC 2.0 compiler. Good writing style but no special optimizations was performed for the codes. Test runs with the different compression algorithms of Table 5 were performed in a 486/33 PC-compatible. It is noted that the compression and decompression times for these algorithms are of the same order of magnitude and therefore only compression times are given. The results of the G3 and G4 were obtained by compressing the images into TIFF format (options -*t*3, -*t*4) using shareware software called *Image Alchemy* (v. 1.7). We want to emphasize that the running times of the G3 and G4 should be examined with caution in comparison to the other methods. The compression efficiency is measured by compression ratio:

$$\textit{compression ratio} = \frac{\textit{number of input bits}}{\textit{number of output bits}}. \qquad (1)$$

The results of the test runs are summarized in Fig. 7 for the set of test images (see Appendix) and for test image Dundee. Note that a compression ratio below 1.0 indicates expansion of the file size.

For test images 1 to 4, the block coding (*B*) is clearly the fastest but the compression ratio is much lower than that of the others. The performance of the composite block coding gets closer to the CCITT Group 4 algorithm, but so does the running time. The 4-pixel prediction context gives much better compression than the 1-pixel prediction. On the other hand, the 8-pixel context is too excessive to gain any remarkable improvement. The compression ratio of the QM coder is still superior to the others.

For the test image Dundee, the results are significantly different. The block coding methods are able to achieve hardly any compression at all. The ordering technique does not improve the performance either. The G3 and G4 algorithms even yield an expansion of the file size instead of compression. Only the adaptive compression schemes (*DOB8*, *QM8*) achieve reasonable compression ratios for an image of this type.

We finally consider random images containing only

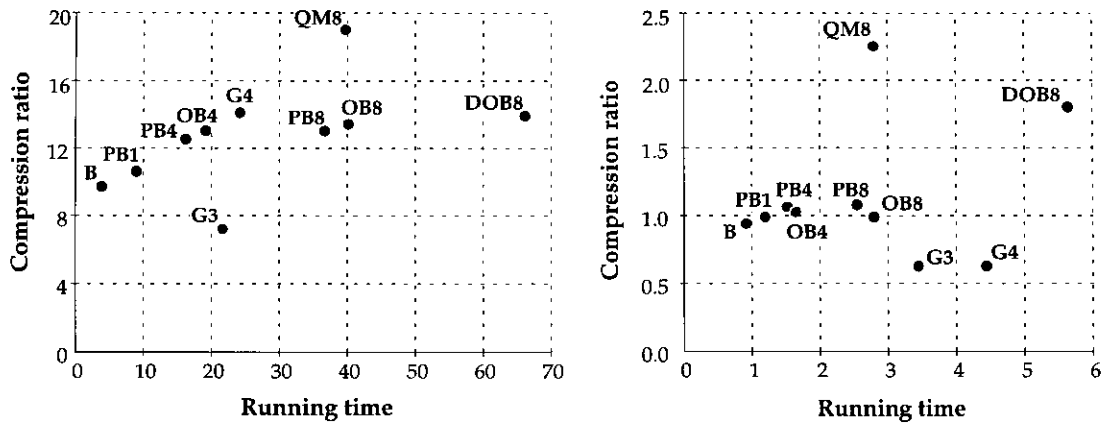


FIG. 7. A comparison of the various compression algorithms. (left) test images 1 to 4; (right) test image Dundee.

noise. In such an image there is no correlation between neighboring pixels and the only parameter is the proportion of white pixels. Though random images will hardly be faced in real life, they are a good test to demonstrate the performance of the various compression methods in extreme situations, and to expose possible weaknesses in the algorithms.

The compression ratios for random images are given in Table 6. Here H refers to the entropy of a *memoryless* source and gives an upper limit for the compression ratio. The results of the QM coder are nearly optimal, as expected. The gap between the entropy and the observed compression ratios of the block coding schemes originates from the use of block coding, which is a suboptimal coding method. The prediction techniques cannot improve the compression ratios for the random images, since there is no correlation to be taken advantage of. An exception to this is the black-dominant images, for which the original block coding fails since it takes advantage of all-white blocks only.

TABLE 6
Compression Ratios of the Various Methods for Random Images

	Proportion of white pixels				
	100%	99%	50%	1%	0%
B	253.82	7.72	0.72	1.16	1.20
PB1	253.82	6.34	0.67	6.44	181.42
PB4	253.82	5.78	0.72	7.14	253.59
PB8	253.82	5.80	0.72	5.90	253.59
OB4	253.82	6.21	0.74	7.12	253.59
OB8	253.82	5.90	0.72	5.63	253.59
DOB8	253.82	8.87	0.72	9.07	252.44
G3	37.71	6.85	0.51	4.19	25.60
G4	414.47	5.68	0.47	4.16	275.08
QM8	139600	11.97	0.97	12.29	39886
H	∞	12.38	1.00	12.38	∞

7. CONCLUDING REMARKS

A new composite compression algorithm based on block coding was proposed. In the algorithm the power of block coding is increased by the application of predictive coding and bit row reordering. The purpose of predictive coding and reordering is here to increase the global dependencies by reducing the local ones. Block coding takes advantage of the global-level dependencies, while the local dependencies are exploited by Huffman coding. The performance of the composite block coding gets closer to the CCITT Group 4 algorithm, but so does the running time. The case of dithered images indicates that an adaptive prediction model may in certain applications be useful.

The net effect of the ordering technique was ca. 3.3% on the compression ratio. The advantage in the compression ratio, however, is paid for by an increase in the running time. Moreover, the overall algorithm is more complex. The main reason for the modest gain in the reordering is that it was originally designed for run-length coding. Thus the pixels are ordered within each row, not blockwise as in the block coding method. The potential benefits of such a blockwise reordering scheme, however, are not studied in this paper.

The compression ratio of the QM coder with a local context model is still superior to all others. The compression performance of the block coding methods, on the other hand, can be improved by coding the all-white/non-white decisions by arithmetic coding, instead of using 1 bit per decision. In fact, this is the main idea of the method introduced in [19, 20].

APPENDIX

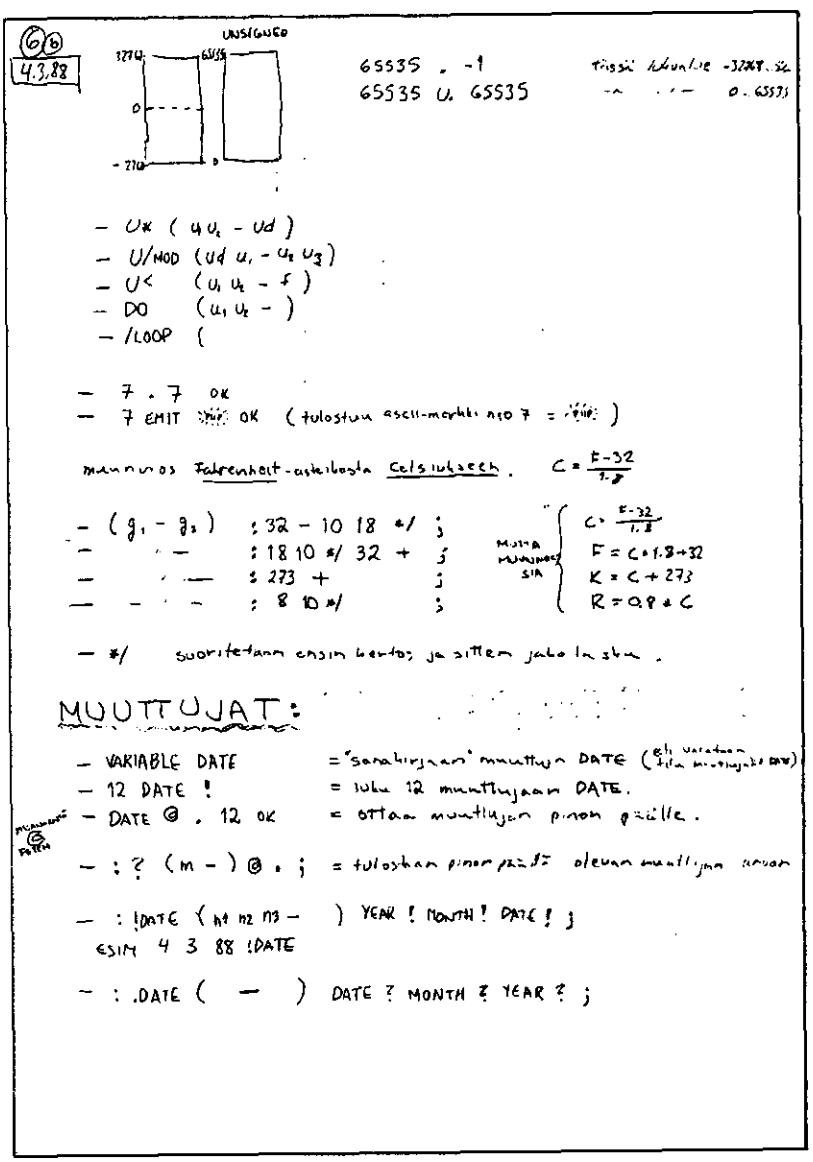
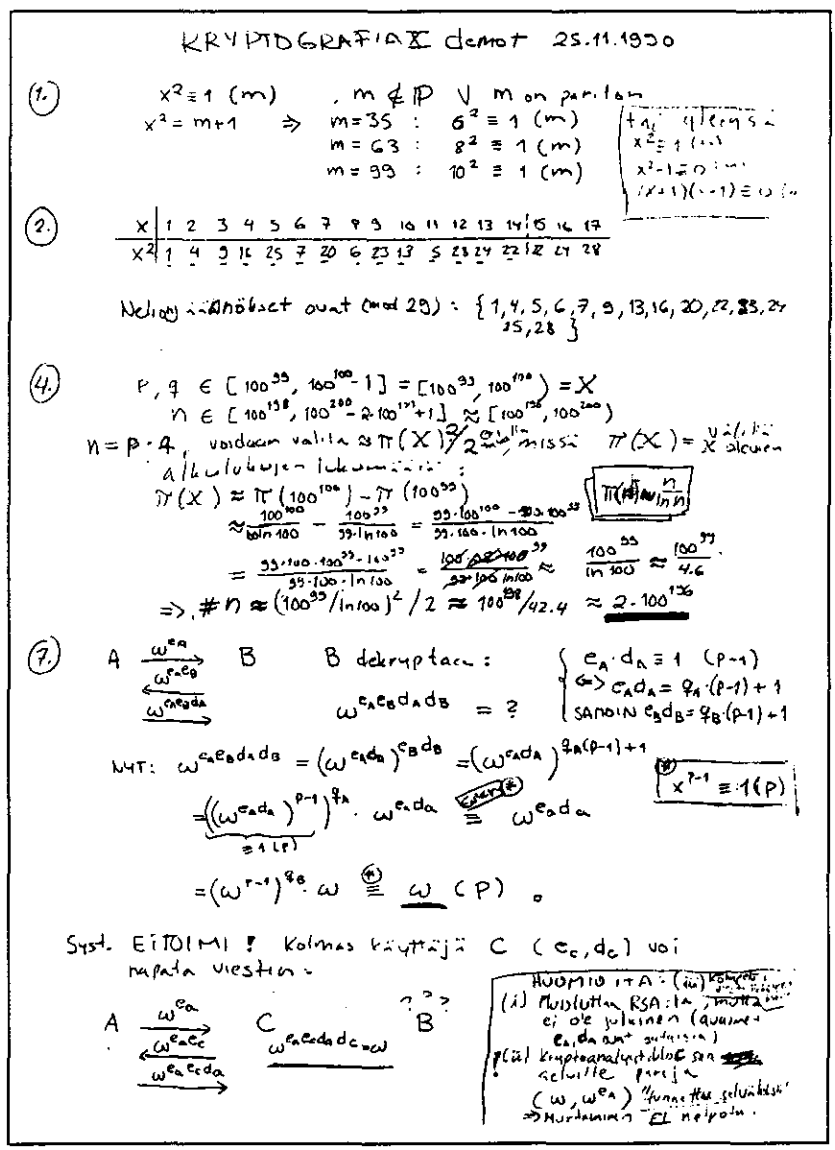


FIG. A1. Test image 1.



KRYPTOGRAFIAX demot 25.11.1990

(1.) $x^2 \equiv 1\ (m)$, $m \notin \mathbb{P} \vee m$ on parillen

$x^2 = m+1 \Rightarrow$ $m = 35$: $6^2 \equiv 1\ (m)$

$m = 63$: $8^2 \equiv 1\ (m)$

$m = 99$: $10^2 \equiv 1\ (m)$

(2.)

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
x^2	1	4	9	16	25	7	20	6	23	13	5	23	24	22	12	27	28

Neliönäistä ovat (mod 29) : $\{1, 4, 5, 6, 7, 9, 13, 16, 20, 22, 23, 24, 25, 28\}$

(4.) $p, q \in [100^{99}, 100^{100} - 1] = [100^{99}, 100^{100}) = X$

$n \in [100^{198}, 100^{200} - 2 \cdot 100^{199} + 1] \approx [100^{198}, 100^{200})$

$n = p \cdot q$, voidaan valita $\approx \pi(X)^2/2$ eri tavalla, missä $\pi(X) =$ X:n alkuluvun

alkulukujen lukumäärä :

$\pi(X) \approx \pi(100^{100}) - \pi(100^{99})$

$\approx \frac{100^{100}}{\ln 100} - \frac{100^{99}}{99 \cdot \ln 100} = \frac{99 \cdot 100^{100} - 100^{99}}{99 \cdot 100 \cdot \ln 100}$

$= \frac{99 \cdot 100 \cdot 100^{99} - 100^{99}}{99 \cdot 100 \cdot \ln 100} = \frac{100^{99}}{\ln 100} \approx \frac{100^{99}}{4.6}$

$\Rightarrow \# n \approx (100^{99}/\ln 100)^2 / 2 \approx 100^{198}/42.4 \approx 2 \cdot 100^{196}$

$\pi(n) \approx \frac{n}{\ln n}$

(7.) A $\xrightarrow{\omega^{e_A}}$ B

B $\xleftarrow{\omega^{e_A e_B}}$

$\xrightarrow{\omega^{e_A e_B d_A}}$

B dekryptaa : $\omega^{e_A e_B d_A d_B} = ?$

$e_A \cdot d_A \equiv 1\ (p-1)$

$\Leftrightarrow e_A d_A = q_A \cdot (p-1) + 1$

SAMOIN $e_B d_B = q_B \cdot (p-1) + 1$

MYT: $\omega^{e_A e_B d_A d_B} = (\omega^{e_A d_A})^{e_B d_B} = (\omega^{e_A d_A})^{q_B(p-1)+1}$

$= \underbrace{((\omega^{e_A d_A})^{p-1})}_{\equiv 1\ (p)}{}^{q_A} \cdot \omega^{e_A d_A} \overset{(*)}{\equiv} \omega^{e_A d_A}$

$x^{p-1} \equiv 1\ (p)$

$= (\omega^{p-1})^{q_B} \cdot \omega \overset{(*)}{\equiv} \omega\ (p)$.

Syst. EITOIMI ! Kolmas käyttäjä C (e_C, d_C) voi napata viestin :

A $\xrightarrow{\omega^{e_A}}$

$\xleftarrow{\omega^{e_A e_C}}$

$\xrightarrow{\omega^{e_A e_C d_A}}$

C $\omega^{e_A e_C d_A d_C = \omega}$??? B

HUOMIO ITA : (ii) Kryptoanalyytikolle on [illegible] (i) Muistuttaa RSA:ta, mutta ei ole julkinen (avaimet e_A, d_A ovat salaisia) (ii) Kryptoanalyytikolle on [illegible] selville paria (ω, ω^{e_A}) "tunnettu selväksi" ⇒ Murtaminen EL hetkellä.

FIG. A2. Test image 2.

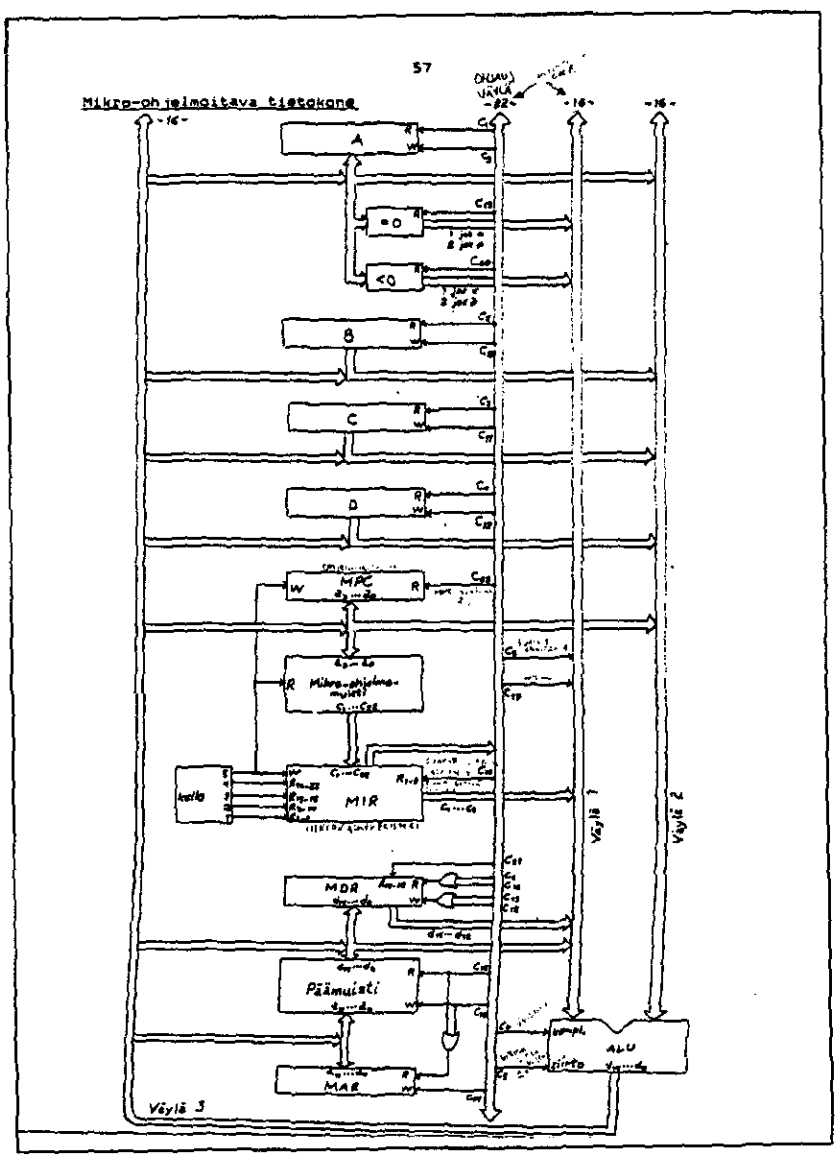
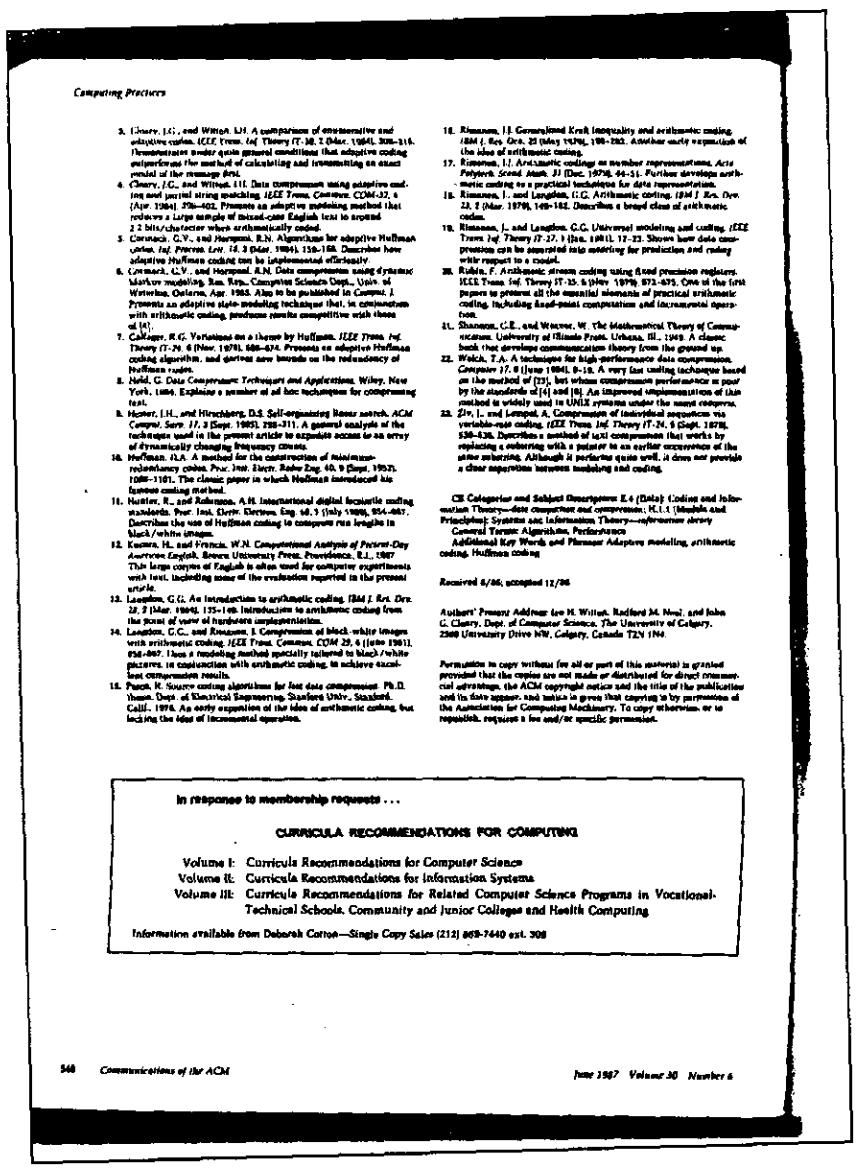


FIG. A3. Test image 3.



Computing Practices

3. Cleary, J.G., and Witten, I.H. A comparison of enumerative and adaptive codes. *IEEE Trans. Inf. Theory IT-30*, 2 (Mar. 1984), 306–315. Demonstrates under quite general conditions that adaptive coding outperforms the method of calculating and transmitting an exact model of the message first.
4. Cleary, J.G., and Witten, I.H. Data compression using adaptive coding and partial string matching. *IEEE Trans. Commun. COM-32*, 4 (Apr. 1984), 396–402. Presents an adaptive modeling method that reduces a large sample of mixed-case English text to around 2.2 bits/character when arithmetically coded.
5. Cormack, G.V., and Horspool, R.N. Algorithms for adaptive Huffman codes. *Inf. Process. Lett. 18*, 3 (Mar. 1984), 159–165. Describes how adaptive Huffman coding can be implemented efficiently.
6. Cormack, G.V., and Horspool, R.N. Data compression using dynamic Markov modeling. Res. Rep., Computer Science Dept., Univ. of Waterloo, Ontario, Apr. 1985. Also to be published in *Comput. J.* Presents an adaptive state-modeling technique that, in combination with arithmetic coding, produces results competitive with those of [4].
7. Gallager, R.G. Variations on a theme by Huffman. *IEEE Trans. Inf. Theory IT-24*, 6 (Nov. 1978), 668–674. Presents an adaptive Huffman coding algorithm, and derives new bounds on the redundancy of Huffman codes.
8. Held, G. *Data Compression: Techniques and Applications*. Wiley, New York, 1984. Explains a number of ad hoc techniques for compressing text.
9. Hester, J.H., and Hirschberg, D.S. Self-organizing linear search. *ACM Comput. Surv. 17*, 3 (Sept. 1985), 295–311. A general analysis of the techniques used in the present article to expedite access to an array of dynamically changing frequency counts.
10. Huffman, D.A. A method for the construction of minimum-redundancy codes. *Proc. Inst. Electr. Radio Eng. 40*, 9 (Sept. 1952), 1098–1101. The classic paper in which Huffman introduced his famous coding method.
11. Hunter, R., and Robinson, A.H. International digital facsimile coding standards. *Proc. Inst. Elect. Electron. Eng. 68*, 7 (July 1980), 854–867. Describes the use of Huffman coding to compress run lengths in black/white images.
12. Kucera, H., and Francis, W.N. *Computational Analysis of Present-Day American English*. Brown University Press, Providence, R.I., 1967. This large corpus of English is often used for computer experiments with text, including some of the evaluation reported in the present article.
13. Langdon, G.G. An introduction to arithmetic coding. *IBM J. Res. Dev. 28*, 2 (Mar. 1984), 135–149. Introduction to arithmetic coding from the point of view of hardware implementation.
14. Langdon, G.G., and Rissanen, J. Compression of black-white images with arithmetic coding. *IEEE Trans. Commun. COM 29*, 6 (June 1981), 858–867. Uses a modeling method specially tailored to black/white pictures, in conjunction with arithmetic coding, to achieve excellent compression results.
15. Pasco, R. Source coding algorithms for fast data compression. Ph.D. thesis, Dept. of Electrical Engineering, Stanford Univ., Stanford, Calif., 1976. An early exposition of the idea of arithmetic coding, but lacking the idea of incremental operation.
16. Rissanen, J.J. Generalized Kraft inequality and arithmetic coding. *IBM J. Res. Dev. 20*, 3 (May 1976), 198–203. Another early exposition of the idea of arithmetic coding.
17. Rissanen, J.J. Arithmetic codings as number representations. *Acta Polytech. Scand. Math. 31* (Dec. 1979), 44–51. Further develops arithmetic coding as a practical technique for data representation.
18. Rissanen, J., and Langdon, G.G. Arithmetic coding. *IBM J. Res. Dev. 23*, 2 (Mar. 1979), 149–162. Describes a broad class of arithmetic codes.
19. Rissanen, J., and Langdon, G.G. Universal modeling and coding. *IEEE Trans. Inf. Theory IT-27*, 1 (Jan. 1981), 12–23. Shows how data compression can be separated into modeling for prediction and coding with respect to a model.
20. Rubin, F. Arithmetic stream coding using fixed precision registers. *IEEE Trans. Inf. Theory IT-25*, 6 (Nov. 1979), 672–675. One of the first papers to present all the essential elements of practical arithmetic coding, including fixed-point computation and incremental operation.
21. Shannon, C.E., and Weaver, W. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, Ill., 1949. A classic book that develops communication theory from the ground up.
22. Welch, T.A. A technique for high-performance data compression. *Computer 17*, 6 (June 1984), 8–19. A very fast coding technique based on the method of [23], but whose compression performance is poor by the standards of [4] and [6]. An improved implementation of this method is widely used in UNIX systems under the name compress.
23. Ziv, J., and Lempel, A. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inf. Theory IT-24*, 5 (Sept. 1978), 530–536. Describes a method of text compression that works by replacing a substring with a pointer to an earlier occurrence of the same substring. Although it performs quite well, it does not provide a clear separation between modeling and coding.

CR Categories and Subject Descriptors: E.4 [Data]: Coding and Information Theory—*data compaction and compression*; H.1.1 [Models and Principles]: Systems and Information Theory—*information theory*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Adaptive modeling, arithmetic coding, Huffman coding

Received 6/86; accepted 12/86

Authors' Present Address: Ian H. Witten, Radford M. Neal, and John G. Cleary, Dept. of Computer Science, The University of Calgary, 2500 University Drive NW, Calgary, Canada T2N 1N4

FIG. A4. Test image 4. Reprinted with permission from the ACM.

REFERENCES

1. R. Hunter and A. H. Robinson, International digital facsimile coding standards, *Proc. IEEE* **68,** 1980, 854–867.
2. ISO/IEC Committee Draft 11544, *Coded Representation of Picture and Audio Information—Progressive Bi-level Image Compression,* April 1992.
3. A. Moffat, Two-level Context Based Compression of Binary Images, in *IEEE Proceedings, Data Compression Conference, Snowbird, Utah, 1991* pp 382–391.
4. A. N. Netravali and B. G. Haskell, *Digital Pictures,* Plenum, New York, 1988.
5. I. Witten and J. Cleary, Picture Coding and Transmission using Adaptive Modelling of Quad Trees, in *Proceedings International Electrical, Electronics Conference I, Toronto,* 1983, pp. 222–225.
6. Y. Yasuda, Overview of digital facsimile coding techniques in japan, *Proc. IEEE* **68,** 1980, 830–845.
7. Y. Yasuda, Y. Yamazaki, T. Kamae, and B. Kobayashi, Advanced in FAX, *Proc. IEEE* **73,** 1985, 706–730.
8. B. Y. Kavalerchik, Generalized block coding of black and white images, *IEEE Trans. Image Process.* **1,** 1992, 518–520.
9. M. Kunt and O. Johnsen, Block coding: A tutorial review, *Proc. IEEE* **68,** 1980, 770–786.
10. G. Zeng and N. Ahmed, A block coding technique for encoding sparse binary patterns, *IEEE Trans. Acoust. Speech Signal Process.* **37,** 1989, 778–780.
11. A. N. Netravali and F. W. Mounts, Ordering techniques for facsimile coding: a review, *Proc. IEEE* **68,** 1980, 796–807.
12. M. Takagi and T. Tsuda, Facsimile bandwidth compression using two-dimensional prediction, *IECE Japan Trans.* **65-D,** March 1973. [In Japanese]
13. D. Bodson, R. A. Schaphorst, and S. J. Urban, Gray scale pictures via group 3 facsimile, *IEEE Comm. Mag.* Sept. 1989, 42–49.
14. D. E. Knuth, Digital halftones by dot diffusion, *ACM Trans. Graphics* **6,** 1987, 245–273.
15. R. Ulichney, *Digital Halftoning,* MIT Press, Cambridge, MA, 1987.
16. W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, and R. B. Arps, An overview of the basic principles of the Q-coder adaptive binary arithmetic coder, *IBM J. Res. Develop.* **32,** 1988, 717–726.
17. W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard,* Van Nostrand–Reinhold, New York, 1993.
18. I. Witten, R. Neal, and J. Cleary, Arithmetic coding for data compression, *Comm. ACM* **30,** 1987, 520–539.
19. P. Fränti, A fast and efficient compression method for binary images, *Signal Process. Image Comm.* **6,** 1994, 69–76.
20. P. Fränti, O. Nevalainen, A two-stage modelling method for compressing binary images by arithmetic coding, *Comput. J.* **36,** 1993, 615–622.

PASI FRÄNTI was born in Vaasa, Finland, in November 1967. He received the M.Sc. and Ph.D. degrees in computer science from the University of Turku, Finland, in 1991 and 1994, respectively. Since 1992 he has been with the University of Turku, where he is involved in research on image processing and compression.

OLLI NEVALAINEN was born in Kankaanpää, Finland, in April 1945. He received the M.Sc. and Ph.D. degrees in 1969 and 1976, respectively. From 1972 to 1979, he was a lecturer in the Department of Computer Science, University of Turku, Finland. Since 1976 he has been an associate professor in the same department. He lectures in the areas of data structures, algorithm design and analysis, compiler construction, and operating systems. His research interests are in the broad field of algorithm design, including image compression, scheduling algorithms, and production planning.