Data compression of scanned halftone images

Forchhammer, Søren; Jensen, Kim S.

Published in:
I E E E Transactions on Communications

Link to article, DOI:
1881-1893

Publication date:
1994

Document Version
Publisher's PDF, also known as Version of record

Data Compression of Scanned Halftone Images

Søren Forchhammer and Kim S. Jensen

Abstract — **A new method for coding scanned halftone images is proposed. It is information-lossy, but still preserving the image quality, compression rates of 16-35 have been achieved for a typical test image scanned on a high resolution scanner. The bi-level halftone images are filtered, in phase with the halftone grid, and converted to a gray level representation. A new digital description of (halftone) grids has been developed for this purpose. The gray level values are coded according to a scheme based on states derived from a segmentation of gray values. To enable real-time processing of high resolution scanner output, the coding has been parallelized and implemented on a transputer system. For comparison, the test image was coded using existing (lossless) methods giving compression rates of 2-7. The best of these, a combination of predictive and binary arithmetic coding was modified and optimized achieving a compression rate of 9.**

I. Introduction

Halftone images create an illusion of shades of gray by varying the area coverage of the halftone dots. The halftone dots are so closely spaced that the grid structure is not noticeable at normal viewing distance. To maintain high quality images, it is necessary to use high resolution scanners with resolutions from 1200-2500 dpi. In newspaper and magazine reproduction, pages with text, graphics, and halftones are often transmitted for production at remote printing plants. Therefore, we address the problem of high resolution fax coding of halftones for the graphic arts industries. This is of interest when only the halftone versions of the images are available. The aim is on-line processing for a 64 kb/s line with compression rates of about 20, but the work has more general applications , especially as higher resolution scanners make their way into the office environment.

We have developed a new coding method for halftones. Image analysis techniques are used for estimation of halftone screen parameters and locating and masking out halftone dots thereby descreening (i.e. low-pass filtering) the halftone image. The areas of the dots are coded. At the receiver the halftone image is rescreened by inserting dots having the size of the coded areas. Described this way, the coding is an information-lossy de- and rescreening of the halftone image. The descreening has a higher resolution than that of the original halftone screen.

Presently, the CCITT Facsimile Group IV MMR (Modified Modified READ) code [1] is used for high resolution fax coding. This code is optimized to compress documents with text and line art and does not perform well on halftones. Currently, a new standard for progressive transmission of fax images is in preparation [2]. It is based on a combination of predictive and arithmetic coding [3]-[5]. Usubuchi et al. [6] have implemented an adaptive predictive runlength coding of newspapers with halftone screen at 45°, achieving a compression rate of 5. Crosfield has presented a similar method [7]. These methods utilize both the redundancies of the gray value image and the halftone presentation, but not optimally. Chao [8] has presented a two-channel scheme. It operates directly on a computer generated halftone image and presumes knowledge of the exact screen structure and phase. In contrast, our method works on distorted (scanned) images and estimates the (non-linear) screen parameters. Chao achieved a compression rate of about 10 for lossy coding.

Section II gives an introduction to halftoning describing halftones as a source of data. The notation used when describing and processing halftones digitally using the new grid description is also given. In Section III, the data format of the coding scheme is presented, and algorithms for the coding are described. Section IV considers the data compression performance and gives the compression results for a test image. The performance is compared with that of other methods. One of these is modified and optimized with respect to high resolution halftone images. The resulting images for lossy coding and evaluation thereof is given in Section V. An implementation on a 17-processor transputer system is described in Section VI. The real-time performance of the system is investigated and aspects of the parallellization are discussed.

II. Description and Notation of Digital Halftone Images

In this section, we describe the generation of halftone images and give the notation of the new digital grid description used for halftone images.

A. Halftoning of images

In the halftoning process three entities are involved: The original gray value image (G) is combined with the halftoning screen (T) to create the resulting bi-level halftone image (B). Each of these entities may

Paper approved by Barry G. Haskell, the Editor for Image Communications Sytems of the IEEE Communications Society. Mauniscript received March 27, 1991; revised March 18, 1992 and October 12, 1992.

S. Forchhammer is with the Institute of Circuit Theory and Telecommunication, Technical University of Denmark, DK-2800 Lyngby, Denmark.

K. S. Jensen was with the Institute of Circuit Theory and Telecommunication, Technical University of Denmark, DK-2800 Lyngby, Denmark. He is now with Eskofot A/S, DK-2600 Glostrup, Denmark.

IEEE Log Number 9401590.

mathematically be described as a function of two variables (x,y) giving the position.

In conventional halftoning, the screen function is added to the gray value image and thresholded in a photomechanical process. In digital halftoning there are two basic approaches. Threshold or electronic halftoning [9] is a digital equivalent of conventional screening. The gray value image $G(x,y)$ is thresholded with the screen function $T(x,y)$ to create the bi-level image $B(x,y)$, i.e.

$$B(x,y) = \begin{cases} 0 & \text{for } G(x,y) > T(x,y) \\ 1 & \text{for } G(x,y) \le T(x,y) \end{cases} \quad (1)$$

where $(x,y) \in \mathbf{Z}^2$. $T(x,y)$ is periodic with the period of the halftone screen.

In look-up table or orthographic gray scale font halftoning [9], the bi-level image is built up of smaller 'characters'. Each 'character' has a number of dark pels (i.e. bi-level pixels) reflecting the gray value at the position of the 'character'.

The screen grid is described by the vectors $V_1 = (V_{1x}, V_{1y})$ and $V_2 = (V_{2x}, V_{2y})$. Normally, the vectors are orthogonal and of the same length. In this case, the screen ruling is the inverse of the length and the screen angle is $\theta = \tan^{-1}(V_{1y}/V_{1x})$. The halftone screen can be described in a continuous coordinate system (s,t) with integer values at the screen dot centers. The images can be described in another continuous coordinate system (x,y) with integer values at the scanner/plotter pel centers. There is a bijective relation between the two coordinate systems. A linear grid is described by the relation

$$x(s,t) = s \cdot V_{1x} - t \cdot V_{2x} + e \quad (2a)$$

$$y(s,t) = s \cdot V_{1y} + t \cdot V_{2y} + f \quad (2b)$$

where $(V_{1x}, V_{1y}, V_{2x}, V_{2y}) \in \mathbb{R}^4$ are the coordinates of the vector basis and $(e,f) \in \mathbb{R}^2$ gives the offset.

In color reproduction, four color separations with four different screens are used. The screens are conventionally angled at multiples of 15°. $\theta = 45°$ is used for the dominating color (and for monochrome images) because the eye is least sensitive to structures at this angle [10]. The three other color separations are often placed at $\theta = 15°$, 75°, and 60° to minimize the Moiré effects [10].

In digital processing of halftone images, the ratio in resolution between the bi-level representation and the halftone screen should be at least 8-10 [9] to maintain quality. Ratios of 12-16 are preferable. Kekolahti [11] advises the use of four gray value pixels per halftone dot.

Different common dot shapes as circular, diamond, and elliptical can be obtained by using different screen functions. We use a clustered dot shape described by cutting the function $D(s,t) = \cos(2\pi s) + \cos(2\pi t)$, which is periodic in the two dimensions. Before using $D(s,t)$ as the screen function $T(s,t)$ in (1), it should be normalized with respect to gray values and the gray value range of $G(x,y)$. The different dot shapes give different dot gain in the printing process. Therefore, compressing a halftone image, it could be desirable to maintain the dot shape.

B. Digital grids, cells, and triangles

This section introduces the new digital grid description for processing (scanned) halftones. In scanned bi-level images, the (s,t) coordinate system of the screen may vary non-linearly in relation to the scanner (x,y) coordinate system. We approximate this using a blockwise polynomial description. The image is partitioned into N by N pel blocks. In each block, the (s,t) values, at the corners (Fig. 1), with subscripts 1, 2, 3, and 4 are used as control points $((s,t)_i)$ to describe the grid. The relation for the polynomial grid in a block is

$$\begin{pmatrix} s(x,y) \\ t(x,y) \end{pmatrix} = \begin{pmatrix} s \\ t \end{pmatrix}_1 + \left[\begin{pmatrix} s \\ t \end{pmatrix}_2 - \begin{pmatrix} s \\ t \end{pmatrix}_1 \right] \frac{x}{N} + \left[\begin{pmatrix} s \\ t \end{pmatrix}_3 - \begin{pmatrix} s \\ t \end{pmatrix}_1 \right] \frac{y}{N}$$

$$+ \left[\begin{pmatrix} s \\ t \end{pmatrix}_4 - \begin{pmatrix} s \\ t \end{pmatrix}_3 - \begin{pmatrix} s \\ t \end{pmatrix}_2 + \begin{pmatrix} s \\ t \end{pmatrix}_1 \right] \frac{xy}{N^2} \quad (3)$$

Using common control points $((s,t)_i)$ for neighboring blocks, the (s,t) coordinate system will be continuous at the block borders ensuring phase continuity of the halftone screen. (The importance of this is illustrated in Section V.)

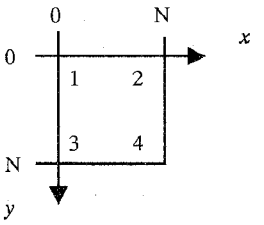

Fig. 1. The numbering of block corners.

In a halftone image, black dots are observed in highlight areas and white dots in shadow areas. The black and white dots are positioned in two offset grids. The black screen dot centers are defined by $(x(s,t); y(s,t))$, $(s,t) \in \mathbf{Z}^2$ and similarly $(x(s+\frac{1}{2}, t+\frac{1}{2}); y(s+\frac{1}{2}, t+\frac{1}{2}))$, $(s,t) \in \mathbf{Z}^2$ define the white screen dot centers. The scanner pel positions coincide with integer (x,y)-coordinates. Rounding of the dot center (x,y)-coordinate values gives the corresponding digital grid points [12] (Fig. 2). The prescript digital denotes a discrete representation in the scanner coordinate system.

Drawing digital straight lines [13] between 4-neighboring digital black grid points partitions the plane into digital white cells (Fig. 3). The digital black

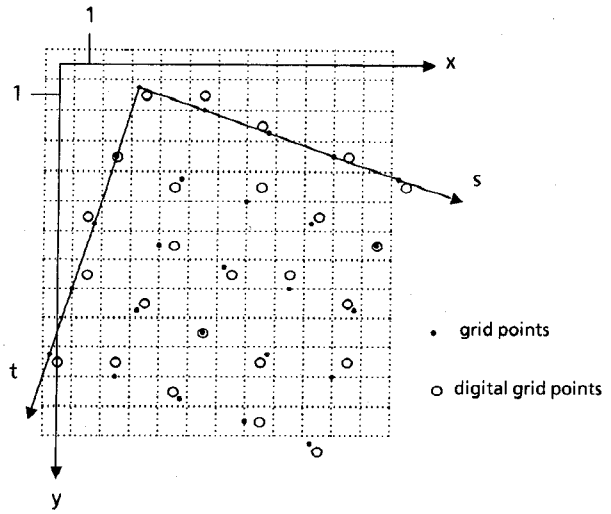

Fig. 2. Digital grid points.

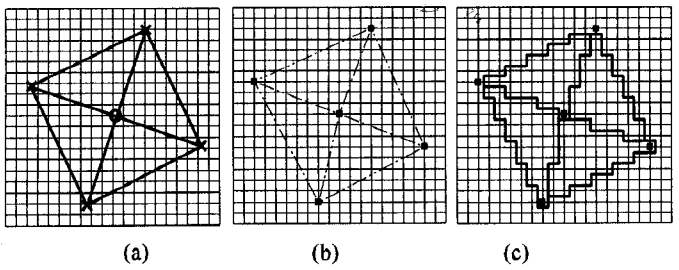

Fig. 3. a) A cell with triangles. b) The digital grid points. c) The digital cell and triangles.

cells are similarly defined. These (approximating) cells do not correspond to a digitization of the lines bounding the continuous screen cells nor do they have the uniformity in area usually required in digital halftoning (variations of the area limited to one pel).

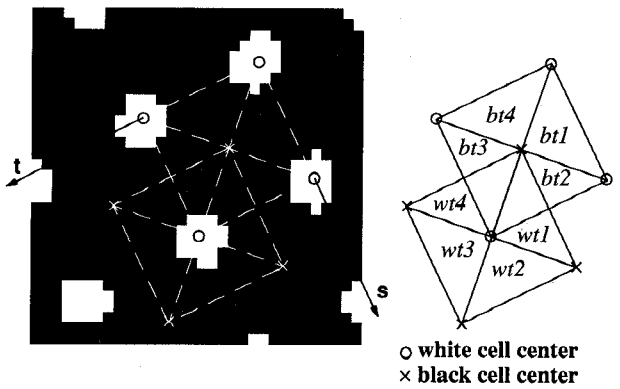

Fig. 4. Enlargement of bit-map showing halftone dots and corresponding black/white cells and triangles.

These problems are circumvented by restricting the use of white cells to dominantly dark areas and black cells to light areas. In this way, the quantized cell boundaries are generally located in regions of the dominant color, leaving the number of dots in the non-dominant color within a cell unaffected by the quantization. This reduction of (halftone cell) quantization error is at the cost of increased complexity. Drawing digital straight lines from the digital center of a cell to the four corners will partition the cell into four digital triangles (Figs. 3-4). These triangles will be used when changing the color of the cells. The structure above has the advantage of being able to handle a varying grid while still being fairly fast to generate.

III. Coding Scheme

This section describes the data format of the new coding scheme for scanned halftones and algorithms complying with this format. The bi-level images are divided into N by N blocks which are processed and coded blockwise. The coding of halftone blocks basically consists of determining the halftone grid, finding the areas of the halftone dots, and coding these. At the receiver the image is rescreened. If there are blocks with text and line art these could be segmented and coded with an information lossless code.

We shall distinguish between global operations and block level operations. For halftone parts, at global level, the halftone grid vector and (optionally) the dot shape are determined. At block level, using a polynomial grid, the control points, which define the grid (3) are determined. From this description, the digital cells and triangles (Sect. II) may be computed. As there are white dots in shadow areas and black dots in highlight areas, it is advantageous to change between black and white cells accordingly. This reduces the errors due to quantization of cell borders (Sect. II) and the low-pass filtering effect of determining dot areas.

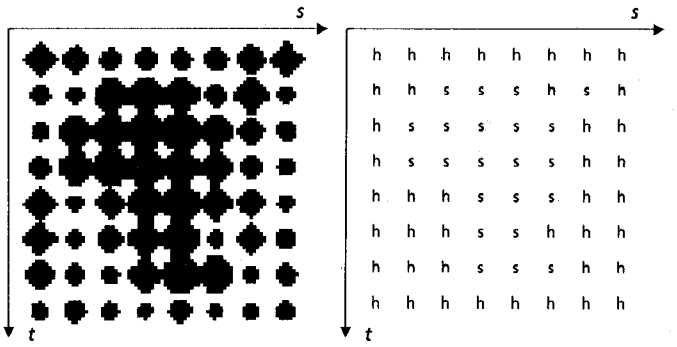

Fig. 5. Bit-map of halftone and corresponding states of black cells. h = highlight and s = shadow.

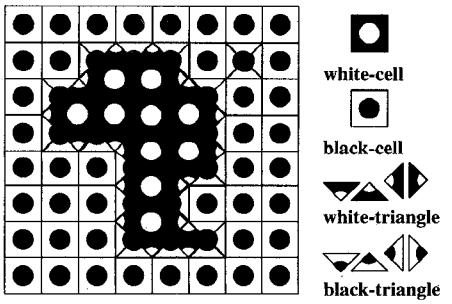

Fig. 6. Segmentation in cells and triangles of the image of Fig. 5.

(Consider the black and the white cell of Fig. 4. The white cell has white pels of only one dot. The black cell has white pels of four dots. Counting white pels within the black cell therefore averages over the four white dots.) The changing is obtained by assigning a highlight or shadow state to each black cell (Fig. 5). When changing, triangles are used in between the white and black cells (Fig. 6). For each cell and triangle the dot area is counted. These dot areas are encoded, thereby representing the halftone, by what we call a descreening coding.

The following list summarizes, in sequence, the main processes of the coding:

Global level operations:
- Grid vector estimation
- Dot shape estimation (optional)

Block level operations:
- Determine control point coordinates
- Compute digital grid and digital cell and triangles
- Count dot areas (i.e. descreening)
- Encode cell and triangle dot areas

Details of the coding are given below.

A. Data format for coding halftone pictures

In the following, a blockwise representation of halftones based on coding of dot areas is presented at block level. The halftone dot shape, the block size (N) and other parameter settings are specified at a level higher than the block level [14]. Our representation of a *coded-halftone-block* is in accordance with the following BNF-rules (4-7). The rules are annotated with an informal description of the syntactic symbols.

$$<\textit{coded-halftone-block}> ::= <\textit{block-id}> \ \textit{halftone} \ <\textit{grid-description}> \ [<\textit{grid-element}>]^* \quad (4)$$

Each block starts with a block header with a unique *block-id* giving the position of the block and the type of coding (here *halftone* coding). The syntax for the *grid-description* of a *coded-halftone-block* with a *polynomial-grid* is

$$<\textit{grid-description}> ::= \textit{polynomial-grid} \ <\textit{grid-vectors}> \ <\textit{grid-control-points}> \quad (5)$$

stating that a *polynomial-* and not a *linear-grid* is used. In addition, the *grid-vectors* (V_1, V_2) and the four *grid-control-points* $((s,t)_1;(s,t)_2;(s,t)_3;(s,t)_4)$ of the polynomial grid in (3) are given.

The digital cells and triangles are obtained from the *grid-description*, as described in Section II.B. The black cells of a whole block are segmented into shadow and highlight state elements. Following the halftone grid within the block, these state elements are traversed back and forth, row by row. A *grid-element* is coded for each state element. The *grid-elements* represent state elements and dot areas of the cells and triangles covering the block. The syntax for *grid-element* is

$$<\textit{grid-element}> ::= [<\textit{change}>] \ [<\textit{black-cell}>] \ [<\textit{white-triangle}>]^* [<\textit{black-triangle}>]^* \ | \ <\textit{white-cell}> \quad (6)$$

The state elements are coded implicitly transmitting a *change* symbol when there is a change of state element value during the traversal. A coding state consists of four state elements numbered 1 to 4 (Fig. 7), number 4 being the current state element. Apart from the *change* symbol, the coding state specifies the *grid-element* of (6) (corresponding to the current state element) as described in Fig. 7. The figure defines the *black-* and *white-cell* and *black-* and *white-triangle* dot area values to be coded with possible triangles of the same color ordered according to the numbering of Fig. 4.

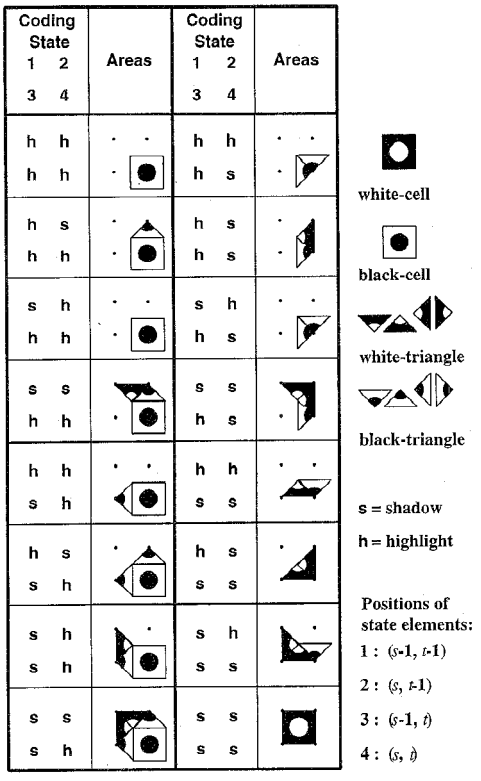

Fig. 7. State table specifying coding states and corresponding cells and triangles to be coded.

When segmenting into highlight and shadow state elements, a hysteresis is applied to reduce the number of state changes. For the *grid-elements*, the grid vector and the hysteresis values of the highlight/shadow segmentation will give the range of dot area values of *black-cell*, *white-cell*, *black-triangle*, and *white-triangle*, respectively. (We apply the hysteresis when traversing the grid.) These values may be coded directly with at least one value free for the *change* signs.

The *change* symbol of (6) is rewritten as

$$<\textit{change}> ::= \textit{simple-change} \mid \textit{border-change} \quad (7)$$

where *simple-change* is used within the block and *border-change* is used, when changing from one row to

the next, to represent a state change before the first coded dot area of the new row. In order to make the state decodable and at the same time avoid coding dot areas outside the block at the borders, the *border-change* also codes the relative position of the state change (Fig. 8).

A number of additional rules, listed below, supplement the syntax to ensure well-formedness, e.g. that the decoding states of the receiver corresponds exactly to the coding states of the sender. This is especially important as the data format does not support resynchronization if a fault occurs.

1) The traversal starts with the black cell center which has the lowest s-value of those with the lowest t-value. The initial direction is towards higher s-values. The traversal is then done row by row, changing the direction for each new row (Fig. 8).

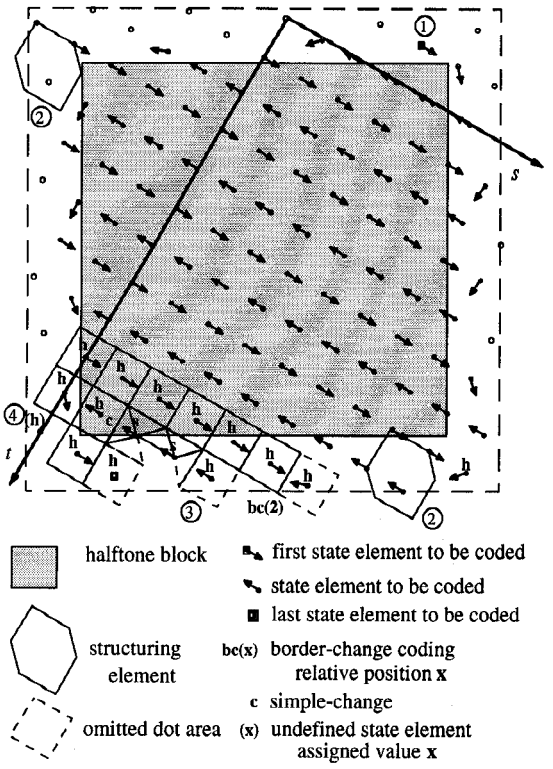

Fig. 8. A coding example including both *simple-change* and *border-change* and illustration of coding rules: ① Start state and traversal, ② state elements to be traversed (showing structuring elements both hitting and missing the block), ③ omitted dot areas, and ④ assignment of state value to undefined state element.

2) A state element is traversed and coded if it as current state element in any coding state may lead to insertion of at least one pel (Fig. 7). This is determined using the structuring element C, defined as the union of the pels of all digital cells and triangles associated with the current state element (no. 4) of any coding state in Fig. 7. If the structuring element, C, positioned at the current state element, hits the block, i.e. has a non-empty intersection with the block, the corresponding *grid-element* and thereby state is encoded (Fig. 8).

3) A cell/triangle dot area is omitted if the cell/triangle is completely outside the block boundary (Fig. 8), thereby reducing the number of values to be transmitted.

4) An undefined state element (corresponding to a state element which is not coded according to rule 2) is given the state value of the nearest defined state element in the same row (Fig. 8). If no values are present, a bias value (*highlight*) is used. This is also the case for the element before the first element.

5) All arithmetics of the grid description must be the same at the sender and the receiver, e.g. within the ANSI-IEEE 754 [15] floating point standard. This is to ensure that the grid calculation at the receiver leads to the same grid values as those found by the sender.

There are two additional options of coding, not previously specified, within the data format. One is to use DPCM (differential pulse code modulation) for the dot areas of the grid element cells. This is specified at a level, higher than the block level. The DPCM code specified is inspired by the work of Girod e.a. [16]. Within highlight and shadow areas one (1. order DPCM) or two (2. order DPCM) previously coded cell dot area values are used to predict the cell value to be coded. The error (e) of the prediction is non-linearly quantized (e') and coded. The quantizer used [16] is the coarsest quantizer with quantization errors $q(e) = e - e'$ bounded by

$$|q(e)| \leq b\sqrt{|e|} \tag{8}$$

where b is a parameter which uniquely specifies the coarsest quantizer satisfying (8) for given b. The quantized prediction errors (e') are Huffman coded [14].

The above coding is information-lossy. To achieve information lossless coding, the option of coding a (difference) bit-map to be XOR'ed with the reconstruction is given.

B. Proof of coverage

The data format described above does not explicitly state anything about covering the area of the blocks or the efficiency of the coverage. We shall show that the state diagram of Fig. 7 gives a unique partitioning of the plane into cells and triangles.

Theorem: Given the state elements (S) of a square grid, the following rules give a unique partitioning of the plane into cells and triangles.

1) A *highlight* element results in a *black-cell* centered at the element grid point.
2a) Two 4-neighbor *shadow* elements give the two *white-triangles* with common long side connecting the two element grid points.
2b) Four *shadow* elements forming the corners of a unit grid cell give a *white-cell* instead of the

four *white-triangles* within the cell resulting from rule 2a).

3) A *shadow* element which is a 4-neighbor of a *highlight* element gives a *black-triangle* with a vertex in the *shadow* element grid point and the opposing long side coinciding with the nearest side of the black cell corresponding to the highlight element.

Proof: Consider two black cell centers (a and b), two white cell centers (c and d), and the sub-triangle, st which is the intersection of two overlapping black (bt) and white triangles (wt) (Fig. 9). st is covered exactly once as part of the black triangle $bt(acd)$ or the white triangle $wt(abd)$ as

$$st \in bt(acd) \Leftrightarrow S(a) = highlight \vee (S(a) = shadow \wedge S(b) = highlight) \quad \text{(by rule 1 and 3).}$$

$$st \in wt(abd) \Leftrightarrow S(a) = shadow \wedge S(b) = shadow \quad \text{(by rule 2).}$$

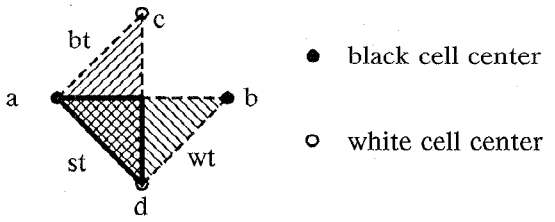

Fig. 9. Triangles st, $bt(acd)$, and $wt(abd)$. st is the intersection of $bt(acd)$ and $wt(abd)$.

The boundaries of all black and white triangles partition the plane into sub-triangles the size of st. The above argument is valid for all eight orientations of a and b and combinations of c and d and therefore for any sub-triangle. □

The state diagram of Fig. 7 follows the three rules of the Theorem and uniquely associates the cells and triangles of rules 1-3 with specific state elements. Cells and triangles of rules 1 and 3 involving state element 4 (and 2 and 3) are coded in connection with state element 4. Triangles (or the cell) of rule 2 inside the cell of the four grid points 1-4 are coded in connection with state element 4. This way the state diagram gives a unique partitioning of the plane associating each cell or triangle with exactly one element as current state element. All state elements covering part of the current block are coded and therefore the whole block is coded.

C. Algorithms

The digital cells and triangles are obtained from the digital grid points. These are found by rounding off the grid points of the grid description as described in Sect. II.B. The encoding of cell and triangle dot areas is described above. The following presents the basic ideas of the other main algorithms for computing and coding the cell and triangle dot areas: grid vector estimation, dot shape estimation, and control point estimation. The algorithms are based on image processing techniques. Other algorithms may be used as the coding is not information lossless.

The first halftone block(s) may be used to estimate the grid vectors using the Fourier transform. Allebach [17] has derived an analytical expression of the Fourier spectrum of digital halftone images. It shows that there are peaks at the screen frequencies corresponding to the grid vectors (Fig. 10). The two-dimensional discrete Fourier transform of $B(x,y)$ is given by

$$F(m,n) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} B(x,y) \exp(-i2\pi(mx+ny)/N) \quad (9)$$

where N^2 is the block size. The maximum likelihood estimator of the frequency of a single sinusoid in white Gaussian noise is the frequency where the two-dimensional (continuous) periodogram (and thereby $|F|$) attains its maximum [18]. The square block has the effect of a window on the data. Therefore, the Fourier transform of the halftone image is convolved with the transform, $\sin(\pi m)\cdot\sin(\pi n)/(\pi m\cdot\pi n)$ of the square. Assuming a peak at the screen frequencies, this may be used to interpolate an estimate from the discrete amplitude spectrum, $A = |F|$. The largest peak of the discrete spectrum, A_p, with coordinates (m_p,n_p) and the neighboring discrete frequencies with largest amplitudes in the two directions $A(m',n_p)$ and $A(m_p,n')$, respectively, are determined. Fitting the three points of the discrete spectrum to the convolved transform of the window, we get the estimate $(\hat{m}_s,\hat{n}_s)$ of the screen coordinates in the frequency domain,

$$\hat{m}_s = m_p + (m' - m_p) \cdot \frac{A(m',n_p)}{A_p + A(m',n_p)},$$

$$\hat{n}_s = n_p + (n' - n_p) \cdot \frac{A(m_p,n')}{A_p + A(m_p,n')} \quad (10)$$

From this, we can obtain estimates of the length of the grid vector $|\hat{V}| = (\hat{m}_s^2 + \hat{n}_s^2)^{-\frac{1}{2}}$ and the grid angle $\hat{\theta} = \tan^{-1}(\hat{n}_s/\hat{m}_s)$. If reliable 3. or 2. order harmonics of the screen frequency are found, these are used instead of the 1. order harmonic to obtain higher accuracy.

The halftone dot shape is described by the screen function T, which may be inferred (analyzing a screened gray-scale) based on the statistical occurrences of black pels as a function of the grid coordinates modulo a screen cell. This dot shape estimation is optional as a default dot shape could be used at the receiver.

To use a polynomial grid (3), four control points are needed. Dot centers near the corners are used to calculate the control points extrapolating the (s,t)

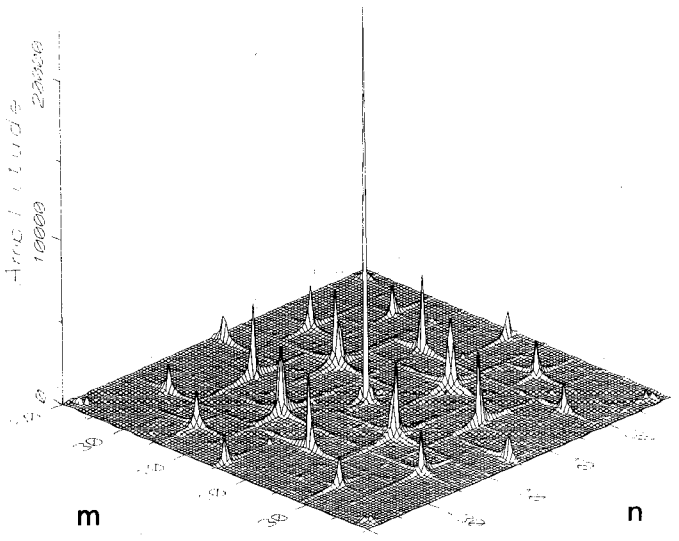

Fig. 10. Amplitudes of lower frequencies of 256 by 256 block of halftone image. The largest peak is at (0,0) while the other peaks reflect the screen frequencies.

coordinates of the dot centres to the block corners using the grid vectors V_1 and V_2. The dot centers (of clustered halftone dots) are found by a combination of two techniques. 1) As centers of maximal inscribed circles and 2) checked by the balance of the dot areas of opposing triangles of a cell. Dots of the non-dominant color are used as they, in general, give the best localization of the centers.

The centers of maximal inscribed circles are found using a distance transform over a small area including the region of interest e.g. the block corners. A distance transform gives the minimum distance from a pel of one color to a pel of the other color. The local maximum minimum distances give the desired centers. A weighted (or chamfer) distance transform [19] with a 3 by 3 mask is used. This method gives approximations of Euclidean distances. The result of the distance transform is vulnerable to bit errors. Therefore the dot centers are checked by a balancing method.

The balancing method tests dot areas of opposing triangles in a cell of the non-dominant color. Let G denote the dot area of a cell and G_i the corresponding dot area of triangle i (Fig. 4). The expression $2\cdot[(G_1-G_3)^2+(G_2-G_4)^2]^{\frac{1}{2}}/(3\sqrt{G})$ is used to estimate the distance of the error in positioning of the center. This error estimate is thresholded to determine whether the balance is adequate.

If no center is found at a block corner, the procedure of finding dot centers is repeated along the adjacent block borders. The control points are communicated to (adjacent) blocks via a shared data structure (see Section VI.A). If a control point of a block is not found in the current or any adjacent block, the block is rejected as a halftone block and may be lossless coded instead.

For internal representation, the cells are always composed of four triangles. The shapes of the digital triangles may be stored and used to mask out the pels of a triangle before counting the dot area. The digital triangle is described by the two vectors between the digital end points at the two short sides of the triangle. With a linear grid, the coordinates of each vector may assume two different x-values and two different y-values. Assuming this to be true for the polynomial grid too, we get 16 different shapes for each of the four orientations of the triangles. This gives 64 shapes of triangular masks at the sender and 64 multiplied by the number of different triangle dot areas at the receiver. At the receiver, the grid has to be generated and the triangles and cells inserted with the coded dot areas. This demands a larger look-up table than at the sender. Various kinds of indirect addressing can be used to limit this table to an acceptable size [20].

IV. Data Compression

A descreening coding of a (scanned) halftone image is generally not an information lossless coding and there are several reasons for this: 1) Deformation of dots may convey information at higher resolution than the screen resolution. 2) Different dot shapes may be used. 3) Inaccuracies in the positioning of the dots. 4) Noise effects in the scanning and digitization process.

From a visual point of view, it is primarily the first factor which is important. The dot shape might give a slight difference in the printing process due to different dot gain. It is not important to code the third and fourth factors and it might even be desirable to filter them out. Therefore, lossless coding is not strictly necessary.

Assuming uniform distribution of values, the dot by dot entropy is $H_d = \log(r^2)/r^2$, where r^2 equals the average number of pels per screen cell. Coding dot areas of each screen cell directly, the round up number of bits $\lceil \log_2(r^2) \rceil$ has to be used. Within the compression scheme, the range of dot areas is reduced because of the shadow/highlight segmentation, but relatively more bits are used on coding the triangle values. In addition, dots across block borders will generally be coded twice.

A. Compression rates

The test image (Fig. 13) was compressed using various versions of descreening coding. The compression rates are given in Table I for direct coding of cell dot areas, lossless DPCM of cell dot areas, and DPCM coding with $b = 0.9$, 1.3, and 1.7 corresponding to different quantization coarseness (8). The mean square errors (mse) and mean numerical errors (me) of gray values (cell dot areas normalized to [0,255]) due to DPCM quantization are also given.

The test image was also coded using various

TABLE I
COMPRESSION RATES (C_R) AND ERROR STATICS FOR CODING OF 4864 BY 6144 PEL BI-LEVEL HALFTONE TEST IMAGE. INCREASING b VALUES GIVE COARSER DPCM QUANTIZATION

Method	C_R	mse	me
Dot areas	16.3	0	0
DPCM 1. order	21.7	0	0
DPCM 2. order	22.4	0	0
DPCM $b=0.9$	27.9	1.38	0.787
DPCM $b=1.3$	32.2	2.84	1.22
DPCM $b=1.7$	35.3	4.69	1.66

information lossless codes [14]. The results are given in Table II. The CCITT MMR code [1] codes the boundaries of the objects line by line. It is optimized for 8 typical telefax documents and does not perform well on halftones. The Liv, Zempel, Welch (LZW) [21] code is a general code for coding (binary) strings. The rest of the codes all use prediction of pel values combined with coding of prediction errors/(-probabilities). For these codes, the pel to be coded is predicted based on the values of a number of previous prediction pels (PP). The set of prediction pels is called the template. The codes differ with respect to the choice of template and the coding method of the prediction errors. Fig. 11 illustrates a number of the templates. Usubuchi, Omachi, and Iinima [6] use 12 PP. Some of the PPs are at the distance of the diagonal of one screen cell away for 45^0. We have modified this for arbitrary screen angles to one screen vector away (MUOI, see Fig. 11). Usubuchi e.a. use an ordering technique [22] to code prediction errors. The other predictive codes use different arithmetic coding [23] of predicted conditional probabilities of the current pel. The (7PP) Q-coder of [3]-[5] uses a fixed set of prediction pels (Fig. 11). In current standardization work (JBIG) [2], the template is enlarged to 9 fixed pels plus one movable pel which may be moved from the default position (▲) to another previous position ($M=(x,y)$) [2] (Fig. 11). This may give an increase in compression rate with an optimum of 7.2 for the test image when M is close to one screen vector away ($M=(11,-5)$) from the pel being coded.

Using the modified 12PP predictor of Usubuchi e.a. (MUOI) controlled by the grid vector as described above and 'JBIG' arithmetic coding of the prediction result, the compression rate was increased. The next idea was to find the 10 and 12 optimal prediction pels, choosing templates differing from those permitted in JBIG. Computing the compression rate for all combinations of 10 or 12 pels within a window including just a few halftone dots is very costly. Instead the (nearly) optimal pels ('Opt') were found one by one within a 661 pel region around the pel to be coded over a smaller part of the image. A last improvement

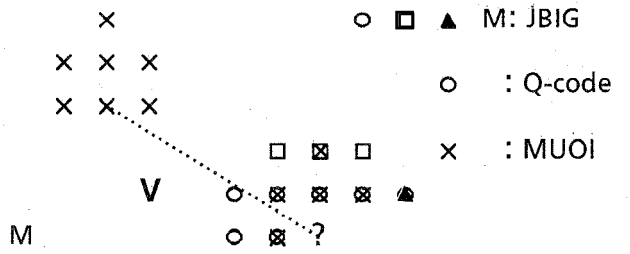

Fig. 11. Various templates used for prediction of the pel being coded (marked as ?). In the JBIG proposal the pel (▲) may be moved to another position (M). 7 of the MUOI pels (x) are placed one screen vector (V) away.

was achieved by altering the line by line traversal of the image. Instead the image was traversed line by line within column stripes of 1024 pels (=4 blocks) (Table II), resulting in the highest compression rate, 9.0 [14].

TABLE II
COMPRESSION RATES (C_R) FOR INFORMATION LOSSLESS CODING OF 4864 BY 6144 BILEVEL HALFTONE TEST IMAGE FOR DIFFERENT METHODS.

Method	Template	Traversal	C_R
LZW	-	Line	2.6
CCITT MMR	-	Line	3.4
Q Coder	7PP	Line	4.7
MUOI	12 PP MUOI	Line	4.9
JBIG	JBIG	Line	5.4
JBIG	JBIG, $M=(-27,0)$	Line	6.4
JBIG	JBIG, $M=(11,-5)$	Line	7.2
'JBIG'	12 PP MUOI	Line	7.8
'JBIG'	'Opt' 10 PP	Line	8.4
'JBIG'	'Opt' 12 PP	Line	8.8
'JBIG'	'Opt' 12 PP	Stripe	9.0

The difference map (Fig. 12) was also coded to achieve information lossless coding in combination with the descreen code. The difference map contained 7.6 % error pels for the direct coding as well as for DPCM coding with b = 0.9, 1.3, 1.7, and 2.1. Using Huffman coding of runlengths based on image statistics, a compression rate of 2.7 was achieved. The best result for a 12 PP template combined with arithmetic coding was as low as 3.9. Filtering the difference map at block and pel level before compression gave a rate of 21.5. Combining this with the 2. order DPCM descreening coding gave 11.0 in rate. The resulting image was an improvement compared to the image without filtering [14].

The halftone test image and 14 blocks of text scanned at high resolution were MMR coded blockwise (256 by 256). As may be seen (Table III), high compression rates for the text were achieved and there is a significant difference for the two types of data. This indicates that a crude segmentation of blocks in text and halftones based on a MMR coding of the blocks thresholding at a rate of 10 would work.

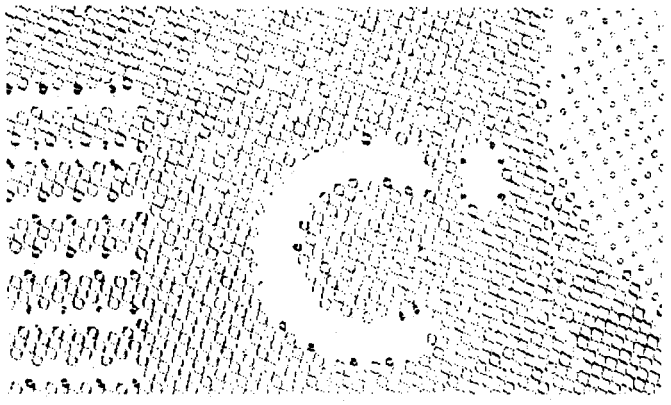

Fig. 12. Part of the difference map.

Instead of determinining the dot areas within the special data format of Section III, the halftone could be converted into a normal gray value image organized in a regular (sampling) grid. Coding this image using an Adaptive Discrete Cosine Transform [24], compression rates as high as 100 would be within reach. This would be at the cost of reduced image quality and higher complexity.

TABLE III
COMPRESSION RATES (C_R) USING CCITT MMR CODING BLOCKWISE FOR THE HALFTONE TEST IMAGE AND FOR 14 BLOCKS OF TEXT.

	Max. C_R	Min. C_R	Overall C_R
Halftone	5.7	2.5	3.4
Text	158	14.8	26.2

V. IMAGE RESULTS

The test image was scanned at 640 dots per cm (ca. 1600 dpi) on a Scitex Raystar scanner and plotted at 500 dots per cm (1270 dpi) for reproduction. The screen ruling of the halftone is 52 lines per cm at 640 dots per cm. Figs. 13-15 show (24 of 30 Mb of) the scanned image and two reconstructions after coding. Fig. 14 is the reconstruction from the basic code, which corresponds to 'Dot area', 'DPCM 1. order', and 'DPCM 2. order' in Table I. In the reconstruction of Fig. 15 lossy DPCM coding ($b=0.9$) was applied to achieve higher compression (Table I).

Subjective evaluation was carried out at a normal viewing distance of 30-35 cm [9]. Following Stoffel and Moreland [9], we shall use three qualitative metrics for evaluation of halftone images:

Low frequency rendition (i.e. the ability to reproduce the gray scale values of the original image.): The images have the same tone gradation. The descreening based coding gives the possibility of correcting the change of tonal reproduction in the scanning process (see Fig. 16). Lossless coding obviously does not have this possibility.

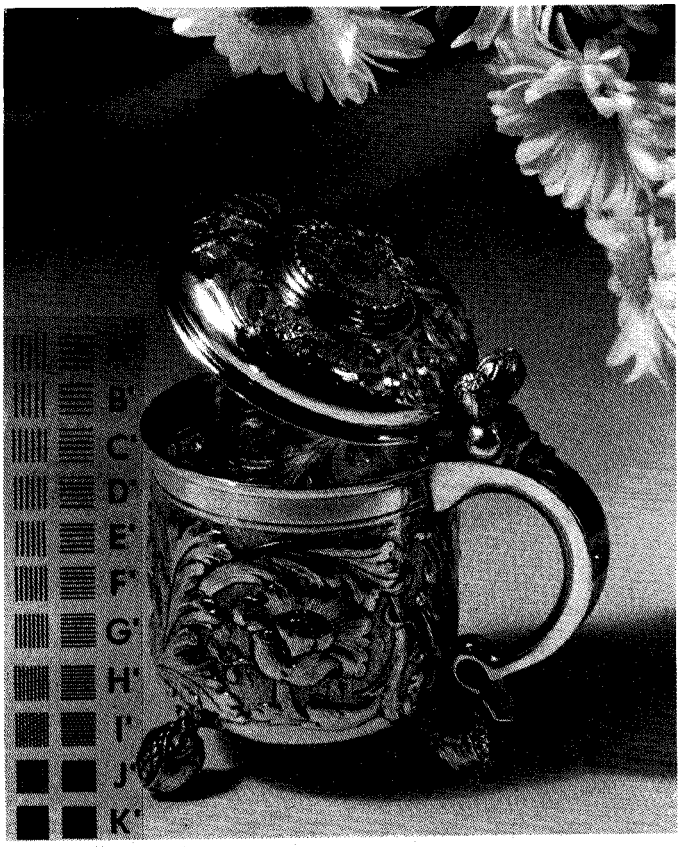

Fig. 13. The bi-level scanned test image plotted at 500 dots per cm i.e.1270 dpi (24 of 30Mb).

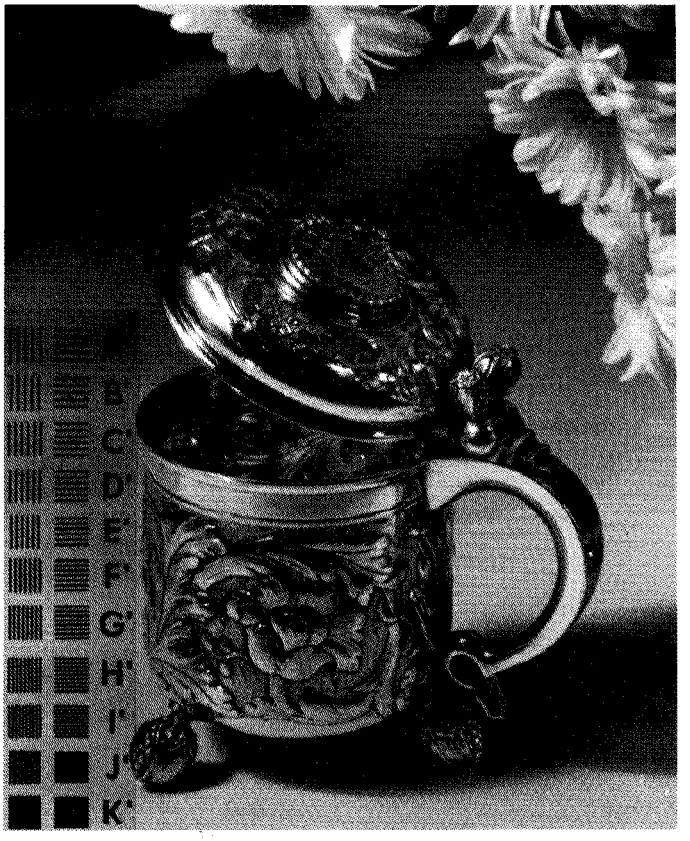

Fig. 14. The result of the described descreen coding scheme. The compression rate is 22.4 (using 2. order lossless DPCM).

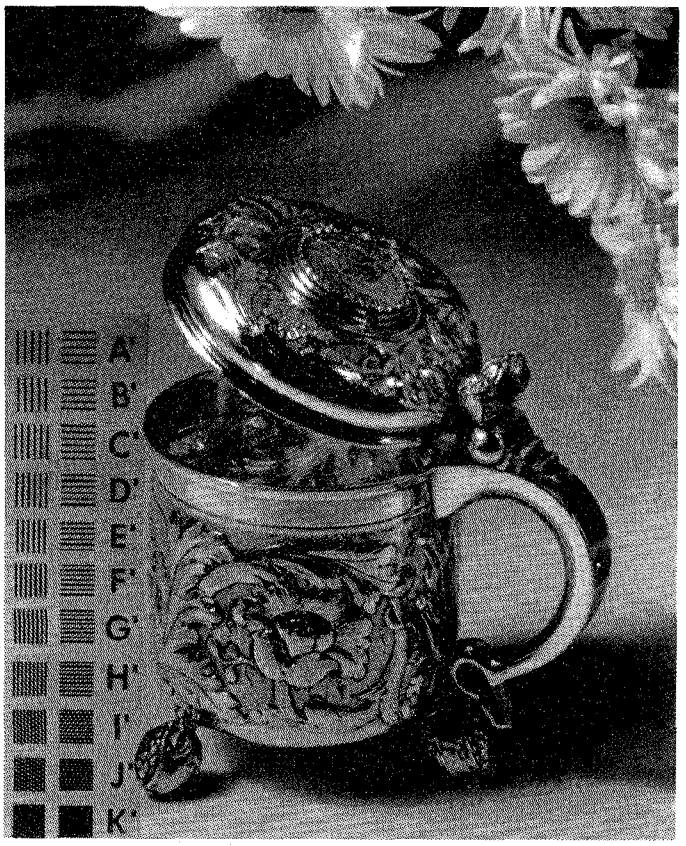

Fig. 15. As for Fig. 14 but using lossy DPCM coding yielding a compression rate of 27.9.

High frequency rendition (i.e. the ability to reproduce sharpness and fine detail): The images have a good high frequency rendition. The descreening code rendering is not quite as sharp as the original. This is mainly seen in the area with letters and bar strip code and in the detailed areas. The bar strip code is resolved equally well in the coded images.

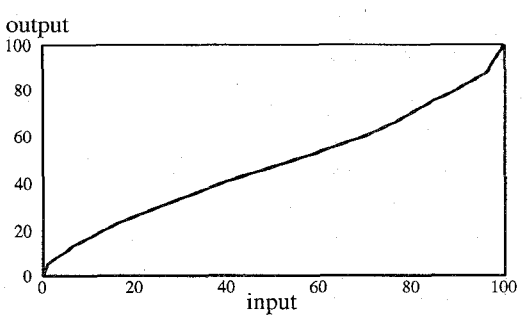

Fig. 16. Tone gradation for input and output of the scanner, showing a non linear change.

Processing artifacts: The descreening gives a barely noticeable false contour between the knobs of the lid and the flowers. The DPCM coding introduces a little disturbance above the letter C'.

For the halftone coding, we used the polynomial grid description as it is more robust towards variations in the grid due to inaccuracies in the input material and the scanning process. As shown in Fig. 17 no block effects occur between halftone coded blocks or between a halftone coded block and an information lossless coded block with halftone parts. So lossless coding of e.g. a part of the boundary of the halftone image would cause no problem. The area of the test image with the letters illustrates the quality of text coded as halftone. This shows that it is possible to halftone code text but it is still better to lossless code across the boundary between a halftone and a text region.

Fig. 17. (a) Part of scanned halftone image (512 by 512 pels). (b-d) Reconstructed images processed in four 256 by 256 pel blocks. (b) A visually disturbing block effect appears because a linear grid description not allowing common control points was used. (c-d) Reconstructions using common control points in a polynomial grid description, eliminating the block effect. In (c) three of the blocks were lossless coded.

VI. Transputer Implementation

To investigate the possibility of using the proposed coding system for real-time processing, e.g. coding the output from a scanner for a 64 kb/s channel, the coding system was implemented on a multiprocessor system. The multiprocessor consists of 16 INMOS-T800 transputers with 1 Mbyte of local RAM each and a single T800 transputer (the host transputer) placed on a PC plug-in card with 8 Mbyte of RAM. The T800 transputer is a RISC-based microprocessor with an on-chip floating point unit and hardware support for process scheduling and high speed interprocessor communication. The T800 version used runs at an internal clock of 17 MHz providing a maximal computing power of 1.3 MFLOPS. The multiprocessor

is designed as a general purpose research system and contains no specialized hardware for image processing.

For the implementation, the network is configured as a 4-dimensional hypercube with the host transputer added on one of the connections. This topology is used because it provides a high degree of connectability by utilizing all of the 16 by 4 communication links of the network transputers. The links are DMA supported serial communication lines which provide full duplex communication at a speed of 20 Mb/s. The maximal effective bandwidth exceeds 1600 Kbytes/sek in one direction. All programming is done in the transputers 'natural' language OCCAM which provides parallelization and a synchronized interprocess/interprocessor communication as language primitives.

For the experimental set-up, the algorithms are divided into two groups. Those that have to be performed as part of a (global) set-up and those that are used for every block. The algorithms belonging to the first group, i.e. (global) screen parameter estimation and generation of tables with digital shapes to be used in the screening and rescreening algorithms, are excluded from the calculations of the real-time performance of the coding algorithm. Depending on the actual system, buffer capacity has to be available at both sender and receiver to buffer data during set-up.

As proposed in Section IV, the MMR coding algorithm is used for segmentation of the blocks in halftone and text/linework. Blocks resulting in a compression lower than a threshold (here 10) are labelled *potential halftone block* and possibly coded as halftone (see Section VI.A).

A. Parallelization

For parallelization of the coding algorithm, we use a simplified problem heap (processor farm) approach [25]. A number of calculating processes are placed on each processor and connected with a general network to a subproblem distributing and a result collecting process. The latter two are placed on the processor connected to the environment (the host). The distribution of data is done one image line at a time thereby minimizing the buffer capacity needed by the distributing process.

The load balancing algorithm is simple: before assigning a processor for a given problem, the number of subproblems (blocks) under evaluation on each processor are compared by the distributing process and the processor with the fewest subproblems is chosen. The load information is maintained in the distributing process with messages sent from the calculating processes to signal the termination of a subproblem. To minimize the control communication, the allocation of calculating processes is done locally. The network allocates a process when the first line of a new block is received at the target processor and maintains a mapping of block identifiers onto local process numbers.

Dividing the complete page into blocks and coding these on different processors requires some care if a time efficient parallelization is to be obtained. This is especially the case for the sender while the receiver is simpler because the blocks can be treated independently. The following discussion is therefore related to the sender:

1) Because halftone dots on block borders have to be used in more than one block, neighboring blocks are made overlapping. Although this leads to increased communication it is preferable to a distribution of dot area values as there is no delaying synchronization involved.

2) To ensure continuity of the polynomial grid description, it is necessary for adjacent blocks to use the same control points. This is administrated by a centralized data structure which receives locally calculated control points and returns an updated set of control points including those received from adjacent blocks. The data structure is kept consistent by disallowing already recorded points to be overwritten.

3) A final matter of interest to the parallelization is the block segmentation. To reduce the time spent on aborted halftone codings, only blocks which are potential halftone blocks and have potential halftone blocks as 4-neighbors are coded as halftones. This introduces a synchronization problem in the sense that great care has to be taken to avoid computing power to be wasted on waiting. A solution is to have a number of calculating processes on each processor. The information exchange itself is implemented as a monitor-like data structure which includes a wait and wakeup mechanism. This kind of block segmentation dictates a lower limit of the total number of calculating processes in the network to ensure that no wait deadlock can occur.

B. Processing results

The quantitative results of implementing the proposed compression algorithm on a general multiprocessor system are divided into two major parts. First the sequential algorithms have been tested to get a typical processing time for a halftone block ($T_{s,h}$) as well as for a text block ($T_{s,t}$). Secondly, the load balancing algorithm has been tested against two important parameters: the number of processors (N_P) and the sequential processing time per block (T_s). The test image shown in Section V is used to encompass a realistic amount of block synchronization as well as varying block processing time for a typical page.

Table IV contains the sequential times for a typical halftone block. It is seen that the descreening, i.e. counting of dot areas is the bottle-neck with the MMR coding of halftones in second place and that the sender demands more computational resources than the receiver.

TABLE IV
MEAN SEQUENTIAL BLOCK PROCESSING TIMES (T_S)

SENDER

HALFTONE	ms	TEXT	ms
Data preparation	77	Data preparation	77
MMR coding	419	MMR coding	67
Control points	48		
Grid calculation	90		
Descreening	1092		
Encoding	48		
Total	1775	Total	144

RECEIVER

HALFTONE	ms	TEXT	ms
Grid calculation	90	MMR decoding	80
Decoding	47		
Rescreening	515		
Total	652	Total	80

Fig. 18 shows the relative speed-up, $S_r = T_S/T_p$ (i.e. the proportion between the sequential processing time for each block, T_s and the total time consumed on each block, T_p) as a function of T_s and the number of processors, N_p used for calculating processes. The page size used for these measurements is 572 (22·26) blocks corresponding to the size of the test image. The processing times are obtained by replacing the coding algorithm with a simple loop that runs for the specified time when executed sequentially.

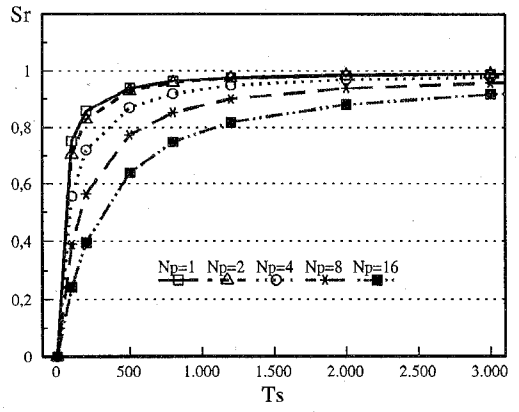

Fig. 18. The relative speed-up (S_r) versus the sequential processing time (T_s).

The mean sequential processing time for a block in the test image has been measured to be $T_s = 1639$ ms. Together with the results in Fig. 18 this leads to an estimated time of $T_p = 1907$ ms. Comparing this with a measured result of $T_p = 2117$ ms indicates the time spent on information exchange and block synchronization. This type of loss is only noticable in halftone parts of a page as text parts can be MMR coded without delay.

The result of Table IV can be combined with Fig. 18 to obtain estimates of the average processing time for a halftone ($T_{p,h}$) as well as a text block ($T_{p,t}$). From these figures, it is possible to calculate the average processing time for a block of a typical (monochrome) page containing a fraction, a_h halftone images and $a_t = 1 - a_h$ text or linework. For $N_p = 16$ processors, and $a_h = 0.2$ ($a_t = 0.8$) we get

$$T_p = (a_h/T_{p,h} + a_t/T_{p,t})^{-1} = (0.2/2023 + 0.8/452)^{-1} = 535 \text{ ms/block/processor} \quad (11)$$

The corresponding compression rate (C_R) can be calculated from the results listed in Table I (without DPCM) and Table III (overall for text):

$$C_R = (a_h/C_{R,h} + a_t/C_{R,t})^{-1} = (0.2/16.3 + 0.8/26.2)^{-1} = 23.4 \quad (12)$$

where $C_{R,h}$ and $C_{R,t}$ are the compression rates for halftone and text, respectively.

The processing time per block and average compression rate can be used to calculate the rate of output data by $N_p \cdot N^2/(C_R \cdot T_p)$, where N^2 is the block size. For the typical page ($a_h = 0.2$, $a_t = 0.8$) we get 82 kb/s. In halftone areas the corresponding figure is 32 kb/s. These figures indicate, that a real-time output rate of 64 kb/s is realistic when coding typical pages.

It should be mentioned that very little has been done to optimize the performance of the coding algorithm. In addition, the communication capacity can undoubtedly be increased with the size of the communication packets e.g. by distributing several lines or a whole block at a time. Another major improvement would be to exclude the MMR coding from the transputer hardware by using specialized hardware for this purpose.

VII. CONCLUSION

A new method of compressing scanned bi-level halftone images has been developed. The halftone image is filtered and the resulting gray values coded. The filtering is in phase with the scanned halftone in both shadow and highlight areas. This is achieved using a new description of halftone grids that facilitates a change in phase between shadow and highlight areas. This gives a better result than simple filtering would do, leading to improved sharpness and reduced risk of Moiré disturbance. Tests have been carried out on a representative halftone test image achieving compression rates of 16-22. The image quality is preserved at normal viewing distance, though close examination reveals a slight degradation in sharpness and one slightly visible artifact. In contrast to information lossless coding, it is possible to improve the tone gradation which is altered in the scanning process.

For comparison existing information lossless coding methods have been implemented and applied to the test image, giving up to 7.2 in compression rate. The best of these codes uses prediction based on previous pels followed by an arithmetic code. A different choice

of prediction pels and a block by block instead of line by line traversal of the image has been used to optimize this coding for halftone data. The prediction pels one vector (or an integer number of vectors) away are generally good. All in all compression rates of 2-9 were achieved using lossless codes. The information-lossy codes of this paper resulted in compression rates of 16-35.

Implementation of the coding method on a transputer system shows that real time coding for a 64 kb/s line is feasible. Parallelization is obtained using overlapping blocks. Some communication between processors processing neighboring blocks is needed at the sender. Otherwise the blocks may be coded independently. Based on the experimental implementation, estimates of output rate for a typical page with 20% halftone and 80% text were calculated to be 82 kb/s overall. In the halftone areas, the estimated output rate was 32 kb/s.

REFERENCES

[1] CCITT, Recommendation T.6, "Facsimile Coding Schemes and Coding Control Function for Group 4 Facsimile Apparatus," *CCITT, Recommendation*, vol. VII, ITU, Geneva, Switzerland, 1984.

[2] "Progressive Bi-level Image Compression Standard," Rev. 4.1, CCITT Draft Rec. T.82, ISO/IEC Comittee Draft 11544, Sept. 1991.

[3] G.G. Langdon and J.J. Rissanen, "Compression of black-white images with arithmetic coding," *IEEE Trans. Commun.* vol. COM-29, pp. 858-867, June 1981.

[4] W.B. Pennebaker and J.L. Mitchell, "Probability estimation for the Q-coder," *IBM J. Res. Develop.* vol. 32, pp. 737-752, Nov. 1988.

[5] J.L. Mitchell and W.B. Pennebaker, "Software implementations of the Q-coder," *IBM J. Res. Develop.* vol. 32, pp. 753-774, Nov. 1988.

[6] T. Usubuchi, T. Omachi, and K. Iinima, "Adaptive predictive coding for newspaper facsimile," *Proc. IEEE*, pp. 807-812, July 1980.

[7] B.J. Jordan, "An update on Crosfield data compression technology," *Taga Proc.* New York, pp. 96-107, 1985.

[8] Y. Chao, *An Investigation into the Coding of Halftone Pictures*, Ph.D. thesis, Massachusetts Institute of Technology, 1982.

[9] J.C Stoffel and J.F. Moreland, "A survey of electronic techniques for pictorial image reproduction," *IEEE Trans. Commun.* vol. COM-29, pp. 1898-1925.

[10] J. Wesner, "Screen patterns used in reproduction of continuous-tone graphics," *Applied Optics*, vol. 13, pp. 1703-1710, July 1974.

[11] P. Kekolahti, "The relationship between digital screening methods, number of pixels per picture and newspaper picture quality," *Lasers in Graphics: Publishing in the 80s, Conf. Proc.*, Florida, pp. 275-284, 1982.

[12] S. Forchhammer, "Digital plane and grid point segments," *Comput. Vision, Graphics Image Process*, vol. 47, pp. 373-384, 1989.

[13] L. Dorst and A.W.M. Smeulders, "Discrete representation of straight lines," *IEEE Trans. Pattern Anal. Machine Intell.* vol. PAMI-6, pp. 450-463, 1984.

[14] S. Forchhammer and K.S. Jensen, "Data Compression of Scanned Halftone Images," IT-TR-127, Technical University of Denmark, 1991.

[15] "IEEE Standard for Binary Floating-Point Arithmetic," ANSI-IEEE Std 754-1985.

[16] B. Girod, H. Almer, L. Bengtsson, B. Christensen, and P. Weiss, "A subjective evaluation of noise-shaping quantization for adaptive intra-/interframe DPCM coding of color television signals," *IEEE Trans. Commun.*, vol. COM-36, pp. 332-346, 1988.

[17] J.P. Allebach, "Aliasing and quantization in the efficient display of images," *J. Opt. Soc. Am.*, vol. 69, No. 6, pp. 869-877, June 1979.

[18] S. Kay, *Modern Spectral Estimation: Theory and Application*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[19] G. Borgefors, "Distance transformations in digital images," *Comput. Vision, Graphics Image Process*, vol. 34, pp. 344-371, 1986.

[20] S. Forchhammer and K.S. Jensen, "Electronic screening at arbitrary angles and rulings," *TAGA Proc.* pp. 20-35, 1991.

[21] TIFF 5.0, *An Aldus/Microsoft Technical Memorandum*, Appendix F, "Data compression scheme 5, LZW compression", 1988.

[22] A.N. Netravali and F.W. Mounts, "Ordering techniques for facsimile coding: A review," *Proc. IEEE*, vol. 68, pp. 796-806, July 1980.

[23] G.G. Langdon, "An introduction to arithmetic coding," *IBM J. Res. Develop.* vol. 28, pp.135-149, March 1984.

[24] "JPEG Technical Specification," Rev. 5, ISO/IEC, Jan. 1990.

[25] P. Møller-Nielsen and J. Staunstrup, "Problem heap: A paradigm for multiprocessor algorithms," *Parallel Comput.* 4, pp. 63-74, 1987.

Søren Forchhammer was born in Copenhagen, Denmark, in 1959. He received the M.S. degree in engineering and the Ph.D degree from the Technical University of Denmark, Lyngby, in 1984 and 1988, respectively. He is currently an Associate Professor at the Institute of Circuit Theory and Telecommunication at the Technical University of Denmark where he has been employed since 1988. His main interests include data compression, image communications, and source coding.

Kim Skovgård Jensen was born in Middelfart, Denmark, in 1963. He received the M.S. degree in electrical engineering from the Technical University of Denmark, Lyngby, in 1989. From 1989 to 1992 he worked at the Institute of Circuit Theory and Telecommunication at the Technical University of Denmark. Since 1992, he has been employed at Eskofot A/S in Denmark working with image processing for graphical applications.