

Compressible Halftoning

Peter G. Anderson and Changmeng Liu

Rochester Institute of Technology, 102 Lomb Memorial Drive, Rochester, NY 14623-5608, USA

ABSTRACT

We present a technique for converting continuous gray-scale images to halftone (black and white) images that lend themselves to lossless data compression with compression factor of three or better.

Our method involves using novel halftone mask structures which consist of non-repeated threshold values. We have versions of both dispersed-dot and clustered-dot masks, which produce acceptable images for a variety of printers.

Using the masks as a sort key allows us to reversibly rearrange the image pixels and partition them into groups with a highly skewed distribution allowing Huffman compression coding techniques to be applied. This gives compression ratios in the range 3:1 to 10:1.

Keywords: Image compression, halftone masks, Huffman code, image file formats

1. BACKGROUND

Halftone images are black-and-white images formed using patterns whose detailed structures are nearly invisible. These images thus convey continuous gray-tone pictures to human eyes.

Halftone images' entropies tend to be very high hence unsuitable for typical statistical approaches to data compression. Many halftone compression approaches involve converting the image back to a gray-level continuous image, then applying JPEG or a similar method. This works, but the image may be severely degraded ("lossy compression").

The utility `xv` converts contone images to black-and-white using the Floyd-Steinberg algorithm. The Gnu or Unix utility, `compress`, can reduce the file sizes of the resulting images only slightly.

The mask-based halftoning we introduce below produces images that `compress` can reduce by ratio of around 4:1.

Other approaches to compression of halftoned images are reported by Denecker^{1,2} with compression ratios comparable to ours.

2. HALFTONE CONVERSIONS VIA MASKING

For the present discussion, we will use input black-and-white images with image pixels satisfying $0 \leq I_{pq} \leq 1$. Similarly, our halftone masks satisfy $0 \leq M_{pq} \leq 1$, with a subscript range identical to the image's. Creation of a bilevel image, B , via a mask follows the rule

$$\begin{aligned} I_{pq} < M_{pq} &\Rightarrow B_{pq} = 0 \\ I_{pq} \geq M_{pq} &\Rightarrow B_{pq} = 1 \end{aligned} \tag{1}$$

Generally, we expect the halftone image B to be such a high-entropy mixture of black and white pixels that compression attempts are doomed. However, the rule of Eq. (1) suggests that the values of B are not truly disordered, but are somewhat predictable, hence compressible; specifically:

$$M_{pq} = Pr[B_{pq} = 0] \tag{2}$$

Further author information: (Send correspondence to P.G.A.) E-mail: pga@cs.rit.edu, Internet: www.cs.rit.edu/~pga, Telephone: 1-585-475-2979, Address: Rochester Institute of Technology, 102 Lomb Memorial Drive, Rochester, NY 14623-5608, USA

that is, when M_{pq} is large, it is highly probable that $I_{pq} < M_{pq}$, making $B_{pq} = 0$.

If we use masks with no repeated values, we can sort the pixels of B using the values of M as a sort key, exploit the observed tendency of one end of the sorted list to be predominantly 0's and the other end predominantly 1's, as Eq. (2) suggests, and compress the rearranged pixel stream. After decompression, the pixels can be rearranged (unsorted) to recover the original B .

3. HALFTONE MASKS

We need large masks of unique values that will result in good bilevel images. Fortunately these two goals do not conflict—in fact, for small clustered-dot masks, the unique-values requirement helps.

We first define a computational rule, denoted by \star , allowing us to combine two matrices into a larger matrix. Specifically, if X and Y are matrices of dimensions $h_X \times w_X$ matrix and $h_Y \times w_Y$, respectively, then

$$Z = X \star Y \quad (3)$$

is a matrix with dimensions

$$\begin{aligned} h_Z &= h_X h_Y \\ w_Z &= w_X w_Y \end{aligned} \quad (4)$$

Z is given by the rule

$$Z_{ph_X+q, rw_X+s} = Y_{pr} + h_Y w_Y X_{qs} \quad (5)$$

An alternative formulation of Eq. (5) is

$$Z_{tu} = Y_{t/h_X, u/w_X} + h_Y w_Y X_{t\%h_X, u\%w_X} \quad (6)$$

in Eq. (6) division is integer division with no remainder, and $\%$ denotes remainder or modulo. Some straightforward but messy calculation establishes that \star is associative.

If X and Y are permutations of the non-negative integers less than $h_X w_X$ and $h_Y w_Y$, respectively, then it is easy to see that Z is a permutation of the non-negative integers less than $h_X w_X h_Y w_Y$. Visually, we can picture $Z = X \star Y$ as a tiling of $h_Y \times w_Y$ copies of the multiple $h_Y w_Y X$, where the (p, q) copy is offset by Y_{pr} .

Matrices X which are permutations of non-negative integers less than $h_X w_X$ can be used as halftone masks in the sense above: simply divide the elements by $h_X w_X$.

Our simplest family of halftone masks is a dispersed-dot mask, a variant of the Bayer mask.^{3,5} Start with a 1×1 matrix

$$M^{(0)} = [0] \quad (7)$$

and recursively develop a $2^k \times 2^k$ matrix

$$M^{(k)} = M^{(k-1)} \star \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix} \quad (8)$$

A similar construction yields a family of $3^k \times 3^k$ matrices:

$$N^{(0)} = [0] \quad (9)$$

and

$$N^{(k)} = N^{(k-1)} \star \begin{bmatrix} 0 & 3 & 6 \\ 2 & 5 & 7 \\ 4 & 7 & 1 \end{bmatrix} \quad (10)$$

The pattern artifacts in images halftoned using $N^{(k)}$ are slightly different from those of $M^{(k)}$.

Modern, high-addressability, electrophotographic printers generally do not produce acceptable images using dispersed dot halftoning schemes. As with their aged, analogue ancestors, they produce better (more reliable, less noisy) images using clustered dot schemes—albeit with dot clusters at least as fine as 150dpi.

To create a clustered dot halftone matrix suitable for our purposes, we start with a clustered dot halftone cell, $C_x^{(0)}$, such as one of the following ($x = 1$ or 2):

$$C_1^{(0)} = \left[\begin{array}{cc|cc} 15 & 11 & 9 & 13 \\ 7 & 3 & 1 & 5 \\ \hline 6 & 2 & 0 & 4 \\ 14 & 10 & 8 & 12 \end{array} \right] \quad (11)$$

$$C_2^{(0)} = \left[\begin{array}{cccc|cccc} 27 & 19 & 23 & 31 & 37 & 45 & 41 & 33 \\ 11 & 3 & 7 & 15 & 53 & 61 & 57 & 49 \\ 9 & 1 & 5 & 13 & 55 & 63 & 59 & 51 \\ \hline 25 & 17 & 21 & 29 & 39 & 47 & 43 & 35 \\ 32 & 40 & 44 & 36 & 30 & 22 & 18 & 26 \\ 48 & 56 & 60 & 52 & 14 & 6 & 2 & 10 \\ 50 & 58 & 62 & 54 & 12 & 4 & 0 & 8 \\ 34 & 42 & 46 & 38 & 28 & 20 & 16 & 24 \end{array} \right] \quad (12)$$

Recursively create ever larger masks by

$$C_x^{(k)} = C_x^{(k-1)} \star \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix} \quad (13)$$

The cell $C_1^{(0)}$ defined in Eq. (11) is a simple clustered dot with 16 threshold levels, growing out from the center of the cell. This 4×4 mask is constructed so that the four quadrants are translations of reflections of each other. This structure, as well as that of Eq. (12), gives us patterns of similar mask thresholds which will be repeated throughout an image at similar gray levels. This similarity and repetition allows for better compression ratios.

The cell $C_2^{(0)}$ defined in Eq. (12) is based on the cell $C_1^{(0)}$ of Eq. (11): its lower right quadrant is twice the pattern of Eq. (11), and the other quadrants follow similar rules:

$$C_2^{(0)} = \begin{bmatrix} 1 + \mathcal{R}_r D_1 & 63 - \mathcal{R}_r D_1 \\ 62 - D_1 & D_1 \end{bmatrix} \quad (14)$$

where $D_1 = 2C_1^{(0)}$ and \mathcal{R}_r denotes row reversal. Eq. (12) gives $C_2^{(0)}$ with 64 threshold levels and an equivalent pattern when black and white are interchanged. Its effect is that of a clustered-dot halftone screen rotated 45° (also known as the “classical screen”).

In case the starter-cell $C_x^{(0)}$ is a halftone cell with very few thresholds (for instance $C_1^{(0)}$), simply tiling an image with unaltered copies of $C_x^{(0)}$ would produce unacceptable halftone images—they would have striping artifacts and look like “paint by number.” The modification of simple tiling given by Eq. (13) yields a huge number of thresholds: if the starting cell $C_x^{(0)}$ has θ_0 distinct thresholds, then $C_x^{(k)}$ has $\theta_k = 4^k \theta_0$ thresholds. Different starting cells can be used for different color planes to effect screen rotations.

4. IMAGE REARRANGEMENT AND COMPRESSION

As with a variety of compression schemes, the data to be compressed is transformed so that some low-entropy aspect is evident. Such transformations must be reversible. In the present application, we consider both the mask and the bilevel image to be one-dimensional lists of length $L = hw$, then sort the bilevel list using the mask list as a sort key. As we suggested above, this transformation puts mostly 0’s at the beginning of the list and mostly 1’s at the end. The transformation is, of course, reversible.

Our approach to compression is now to rearrange the sorted L pixels into a rectangular array of dimension $K \times L/K$; that is, L/K columns of length K . These columns tend to begin with 0's and end with 1's and have few transitions of value within themselves. Furthermore—this is the most important observation—the columns patterns are in a very skewed distribution. The values are very far from being equally likely. For an illustration of this, we present an image halftoned using our $2^k \times 2^k$ dispersed dot mask and clustered dot mask (Figs. 1 and 2, resp.). These images are printed with very coarse fat-pixels to show the underlying structure. Halftone choices must be tuned to their target printing processes, using choices not under our control for conference proceedings.

Table 1 shows the compressions that we obtain for various column sizes K (the data in this table correspond to the dispersed dot halftone of Fig. 1, but the data for Fig. 2 was almost identical). The “entropy” is the number of bits needed to encode the average K -bit column assuming an optimal coding scheme. If the probability of the n -th column pattern is P_n , this notion of entropy is evaluated by:

$$-\sum_n P_n \log_2 P_n \quad (15)$$

Observe the compression ratios associate with various values of K in Table 1: the sweet spots are at the powers of 2. This is an artifact of the halftone masks which grow at powers of 2. The sorting and grouping operation puts together in single columns the pixels associated with a tiling of a component matrix X in the mask formation $X \star Y$. (We noticed similar sweet spots at powers of 3 with the halftone masks constructed using 3×3 building blocks.) More generally, if the starter-cell, such as our $C_x^{(0)}$ cells, has θ_0 thresholds, and the large cells $C^{(k)}$ have $\theta_k = M^k \theta_0$ thresholds, then we would expect the sweet spots for block sizes to be of the form $K = M^k \theta_0$ (in our examples, we have $M = 4$ or 9). Of course we would use quite small values of M . If $C_x^{(0)}$ had an internal structure (as $C_2^{(0)}$ does), then some $M^k \theta_0$ may be divided by an appropriate integer to find good vales for K . Image sorting using these masks as sort keys followed by the rearrangement into blocks with appropriate sizes, K , creates blocks of pixels corresponding to what we would usually consider to be halftone cells (or fractions thereof) in the images.

Table 2 shows the 16-bit column patterns associated with the clustered dot image Fig. 2. The bulk—over 97%—of these patterns are 0's followed by 1's (shorthand: $0^m 1^{16-m}$). This would correspond to blocks of 4×4 pixels in the original image that are relatively constant. But that is expected of images. The patterns not of the form $0^m 1^{16-m}$ would correspond to image regions with large gray-scale gradients or discontinuities.

In Table 2 we have 46 different patterns. We could assign each of them a 6-bit code, yielding an immediate compression ratio of almost 3:1. However, the frequencies of the patterns are so skewed, we can use Huffman coding to achieve a compression ratio of 5:1. An accompanying table to permit decompression will add about 10% overhead to the compressed file.

5. HUFFMAN CODING OF THE COLUMN PATTERNS

The problem to be addressed now is how to assign short codes to the most frequent patterns and longer codes to the less frequent one (cf. Morse code: the most frequent letters got the shortest codes). This is achieved by Huffman code.⁴

Huffman code converts a sequence of “messages” (in our case, that sequence is the list of column patterns) into bit strings of varying lengths. It does this in such a way as to obviate any separator between codes for subsequent messages. A rough explanation of the encoding mechanism is the following. The collection of messages with their associated frequencies is successively reduced by combining the two messages with the lowest frequencies into a single message, but discriminating between them using the last bit, 0 for one, 1 for the other. This continues until all messages have been combined into a single one. The Huffman codes for the 16-bit patterns are shown in Table 2 with their frequencies.

The three most frequent patterns, making up 63% of the patterns, use 2- or 3-bit codes.

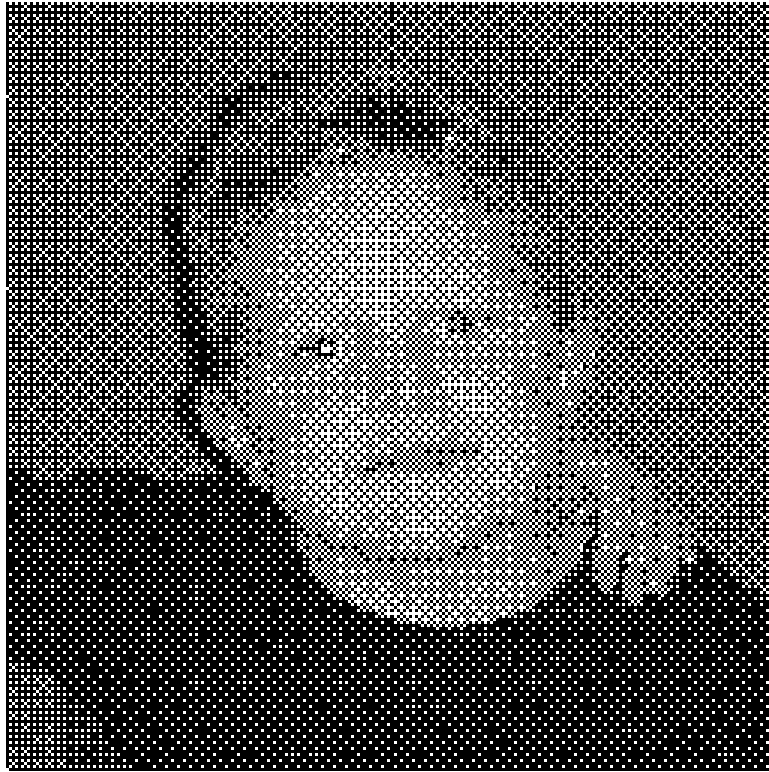


Figure 1. A photograph halftoned using a dispersed dot scheme.

K	patterns	entropy	bits	ratio
2	4	1.14	37340	1.76
3	6	1.41	30715	2.13
4	6	1.71	28002	2.34
5	12	2.16	28263	2.32
6	22	2.56	28011	2.34
7	21	2.99	28000	2.34
8	16	2.36	19363	3.38
9	43	3.58	26037	2.52
10	47	4.07	26641	2.46
11	75	4.31	25647	2.56
12	89	4.67	25493	2.57
13	109	5.04	25388	2.58
14	123	5.34	24978	2.62
15	169	5.57	24340	2.69
16	46	3.19	13069	5.01
32	115	4.48	9182	7.14
64	275	6.07	6211	10.55

Table 1. Compression ratios for the halftoned image of Fig. 1 using various block sizes. The column labeled *patterns* is the distinct pattern count for size= K . The column labeled *bits* is the estimate of the compressed file size (not counting the accompanying table).

Table 2. Frequencies of the 46 different (of a total of 4096) 16-bit column patterns of the clustered dot halftone. In this example, the most frequent 97% of the patterns are of the form $0^m 1^{16-m}$. The horizontal line occurs just above the first example not of that form.

count	pattern	code
916	0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1
857	0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1	1 0
819	0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1	0 1 1
252	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1	0 1 0 1
249	0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1	0 0 1 1
239	0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1	0 0 0 1
151	0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1	0 1 0 0 1
145	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	0 1 0 0 0
133	0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1	0 0 1 0 1
131	0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1	0 0 1 0 0
84	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1	0 0 0 0 1
18	0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 1 1
12	0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 1 0 1
12	0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1	0 0 0 0 0 1 1 1
10	0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1	0 0 0 0 0 0 1 0 1
7	0 0 0 1 0 0 0 1 1 1 1 1 1 1 1 1	0 0 0 0 0 1 1 0 1
7	0 0 0 0 0 0 1 0 0 1 1 1 1 1 1 1	0 0 0 0 0 1 1 0 0
6	0 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 1 0 0 1
5	0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1	0 0 0 0 0 0 1 0 0 0
3	0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0
3	0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 1	0 0 0 0 0 1 0 0 1 0
3	0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1	0 0 0 0 0 1 0 0 1 1
3	0 0 0 0 0 0 1 1 0 1 1 1 1 1 1 1	0 0 0 0 0 1 0 0 0 0
3	0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1	0 0 0 0 0 1 0 0 0 1
2	0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 0
2	0 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1
2	0 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 0 0
2	0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 0 1
2	0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 1	0 0 0 0 0 0 0 1 0 1 0
2	0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1	0 0 0 0 0 0 0 1 0 1 1
1	0 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1	0 0 0 0 0 0 0 0 0 0 0 0
1	0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 1
1	0 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 1 0
1	0 0 1 0 0 0 1 0 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 1 1
1	0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 1 0 0
1	0 0 0 0 1 1 1 1 1 0 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 1 0 1
1	0 0 0 0 1 1 1 1 0 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 1 1 0
1	0 0 0 0 1 0 1 0 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 1 1 1
1	0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 1 0 0 0
1	0 0 0 0 0 1 1 1 1 0 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 1 0 0 1
1	0 0 0 0 0 1 1 1 0 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 1 0 1 0
1	0 0 0 0 0 1 0 1 0 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 1 0 1 1
1	0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 1 1 0 0
1	0 0 0 0 0 0 0 1 0 1 0 1 1 1 1 1	0 0 0 0 0 0 0 0 0 1 1 0 1
1	0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1	0 0 0 0 0 0 0 0 0 1 1 1 0
1	0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1	0 0 0 0 0 0 0 0 0 1 1 1 1

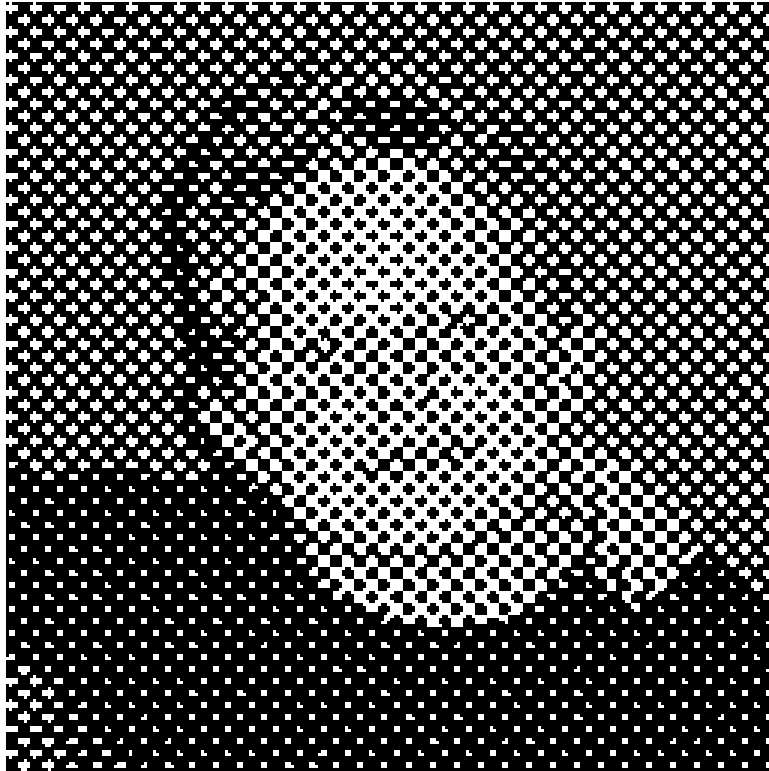


Figure 2. A photograph halftoned using the clustered dot scheme based on the classical screen.

6. FILE FORMATS FOR COMPRESSED IMAGES

The pattern frequencies will be particular to a given image, so enough information must be stored with the compressed image to allow for decompression. There are several approaches to this.

The simplest approach is to include a version of the frequency table (the first two columns of Table 2). We can store this table using 16 bits for each pattern and 12 bits for frequencies at a cost of 1,288 bits. The compressed image in this case required 13,319 bits, so this overhead is acceptable. Of course, a small amount of additional overhead must be used to encode the table length and any other image parameters, but those will not add appreciable overhead. For this worked-out example, the compression ratio is approximately 5:1.

7. ACKNOWLEDGMENTS

The ideas developed in this paper grew from seeds first planted in discussions with Prof. Charles A. Bouman of Purdue University.

We gratefully acknowledge the financial support of Hewlett-Packard for this research.

REFERENCES

1. Koen Denecker and Peter De Neve. A comparative study of lossless coding techniques for screened continuous-tone images. In B. Werner, editor, *Proceedings of the international conference on acoustics, speech, and signal processing (Munich Germany)*, volume 4, 1997.
2. Koen Denecker, Dimitrie Van De Ville, Frederik Habils, Ignace Lemahieu, and Adrian Munteanu. Software and hardware implementation of an improved lossless halftone image compression algorithm. In *International conference on imaging science, electronic imaging (Antwerp, Belgium)*, volume 2, 1998.

3. Henry Kang. *Digital Color Halftoning*. SPIE: The International Society for Optical Engineering, Bellingham, WA, 1999.
4. Khalid Sayood. *Introduction to Data Compression*. Morgan Kaufmann, 1996.
5. Robert Ulichney. *Digital Halftoning*. The MIT Press, 1987.